2012

User-Centric Quality of Service Provisioning in IP Networks

Culverhouse, Mark

http://hdl.handle.net/10026.1/1233

http://dx.doi.org/10.24382/4749
University of Plymouth

User-Centric Quality of Service Provisioning in IP Networks

by

Mark Culverhouse

A thesis submitted to Plymouth University
in partial fulfilment for the degree of

DOCTOR OF PHILOSOPHY

School of Computing and Mathematics
Faculty of Technology

In collaboration with
France Telecom Orange Group.

July 2012

Abstract

User-Centric Quality of Service Provisioning in IP Networks

Mark Culverhouse

The Internet has become the preferred transport medium for almost every type of communication, continuing to grow, both in terms of the number of users and delivered services. Efforts have been made to ensure that time sensitive applications receive sufficient resources and subsequently receive an acceptable Quality of Service (QoS). However, typical Internet users no longer use a single service at a given point in time, as they are instead engaged in a multimedia-rich experience, comprising of many different concurrent services. Given the scalability problems raised by the diversity of the users and traffic, in conjunction with their increasing expectations, the task of QoS provisioning can no longer be approached from the perspective of providing priority to specific traffic types over coexisting services; either through explicit resource reservation, or traffic classification using static policies, as is the case with the current approach to QoS provisioning, Differentiated Services (Diffserv). This current use of static resource allocation and traffic shaping methods reveals a distinct lack of synergy between current QoS practices and user activities, thus highlighting a need for a QoS solution reflecting the user services.

The aim of this thesis is to investigate and propose a novel QoS architecture, which considers the activities of the user and manages resources from a user-centric perspective. The research begins with a comprehensive examination of existing QoS technologies and mechanisms, arguing that current QoS practises are too static in their configuration and typically give priority to specific individual services rather than considering the user experience. The analysis also reveals the potential threat that unresponsive application traffic presents to coexisting Internet services and QoS efforts, and introduces the requirement for a balance between application QoS and fairness.

This thesis proposes a novel architecture, the Congestion Aware Packet Scheduler (CAPS), which manages and controls traffic at the point of service aggregation, in order to optimise the overall QoS of the user experience. The CAPS architecture, in contrast to traditional QoS alternatives, places no predetermined precedence on a specific traffic; instead, it adapts QoS policies to each individual's Internet traffic profile and dynamically controls the ratio of user services to maintain an optimised QoS experience. The rationale behind this approach was to enable a QoS optimised experience to each Internet user and not just those using preferred services. Furthermore, unresponsive bandwidth intensive applications, such as Peer-to-Peer, are managed fairly while minimising their impact on coexisting services.

The CAPS architecture has been validated through extensive simulations with the topologies used replicating the complexity and scale of real-network ISP infrastructures. The results show that for a number of different user-traffic profiles, the proposed approach achieves an improved aggregate QoS for each user when compared with Best effort Internet, Traditional Diffserv and Weighted-RED configurations. Furthermore, the results demonstrate that the proposed architecture not only provides an optimised QoS to the user, irrespective of their traffic profile, but through the avoidance of static resource allocation, can adapt with the Internet user as their use of services change.

Table of Contents

Table of Figures

List of Tables

Acknowledgements

This research was made possible by the funding awarded to me by the France Telecom Orange Group. I wish to extend my gratitude to the organisation for their support and giving me the opportunity to conduct the work.

I would like to thank Dr. Bogdan Ghita, my Director of Studies, for first making me aware of the project funding and for his unconditional support and guidance during the research programme, without his dedication to me as his student this work would not have been possible, and for this, I am indebted to him.

I also wish to extend my profound thanks to Professor Paul Reynolds, my supervisor, whose passion and willingness to help has been truly inspirational. I am sincerely grateful to have been under his supervision.

I thank my colleagues at the Centre for Security, Communications and Network Research for their encouragement and friendship throughout the duration of the research.

Finally, I owe my deepest gratitude to my family and Tracey, my girlfriend, for their support and patience over the past five years. I am sure they thought my life as a student would never end.

Glossary

AF	Assured Forwarding
AQM	Active Queue Management
AS	Autonomous System
CAPS	Congestion Aware Packet Scheduler
CBR	Constant Bit Rate
CIR	Committed Information Rate
CWND	TCP Congestion Window
DCCP	Datagram Congestion Control Protocol
Diffserv	Differentiated Servics
DNS	Domain Name Service
DSCP	Diffserv Code Point
DSL	Digital Subscriber Line
ECN	Explicity Congestion Notification
EF	Expedited Forwarding
FIFO	First In First Out
FTP	File Transfer Protocol
HTTP	Hyper Text Transfer Protocol
IETF	Internet Engineering Task Force
Intserv	Integrated Services
IP	Internet Protocol
ISP	Internet Service Provider
ITU	International Telecommunications Union
MOS	Mean Opinion Score
MPLS	Multi-Protocol Label Switching
MSS	Maximum Segment Size

NAT	Network Address Translation
P2P	Peer-to-Peer
PHB	Per Hop Behaviour
PIR	Peak Information Rate
QoS	Quality of Service
RED	Random Early Detection
RIO	Red with In and Out
RSVP	Resource Reservation Protocol
RTCP	Real Time Control Protocol
RTO	Retransmission Timeout
RTP	Real Time Protocol
RTT	Round Trip Time
SLA	Service Level Agreement
TCP	Transmission Control Protocol
TFRC	TCP Friendly Rate Control
TOS	Type of Service
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VoD	Video on Demand
VoIP	Voice over IP
VTC	Video Teleconferencing
WAN	Wide Area Network
WRED	Weighted Random Early Detection
WWW	World Wide Web

AUTHOR'S DECLARATION

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award without prior agreement of the Graduate Committee.

This study was financed with funding from the France Telecom Group.

Relevant scientific seminars and conferences were regularly attended at which work was often presented. Contacts from France Telecom provided technical advice and feedback during the early stages of the project. Details of the published papers are listed in the Appendices.

Word count of main body of thesis: 50,248

Signed …………………………………………….

Date ……………………………………………….

1. Introduction

The Internet today provides a worldwide delivery network to an ever expanding array of multimedia enriched services for an estimated 2.2 Billion users (Miniwatts-Marketing-Group 2012). Over the past decade many services have migrated from dedicated delivery infrastructures towards the Internet. Services such as Voice over IP (VoIP), Video on Demand (VoD), Video-Teleconferencing (VTC) and online gaming are now highly integrated in the modern lifestyle. This integration has been further facilitated by the advent of Internet-enabled mobile phones and mobile broadband for laptops. Historically, users purchased connectivity, a link between their premises and a service provider's network, but the ubiquity of the Internet and reliance on Internet services have changed the requirement from one of solely connectivity to one with an expectation of the quality of the delivered media.

The Internet Protocol (IP) (ARPA 1981), which was designed over 30 years ago, provides a best-effort delivery system, simply concerned with forwarding packets from their source to destination, and any requirement for additional functionality such as end-to-end reliability, prioritisation, sequencing or flow control must be provided by accompanying protocols. Traditional applications were not excessively affected by adverse network conditions, packet loss could be compensated for at a layer higher, which offered a reliable delivery service, for example the Transport Control Protocol (TCP), and therefore allowed for a best-effort delivery system from IP. An email could be delayed by a few seconds or the throughput of a file transfer may have been reduced causing an increase in the download time, but both would have little impact on the user experience. However, real-time applications demand more from a delivery network. Real-time applications, traditionally implemented using circuit switched networks, introduced many challenges when transported by packet switched technologies, with finite buffer space, multiplexing and queuing. Excessive delay or packet loss during a VoIP conversation would severely impair its Quality of Service (QoS) and therefore, efforts were needed to improve the best-effort nature of IP to accommodate these new traffic types.

The task of deploying an effective QoS implementation has been argued as a non-issue by some, who consider that the answer lies in ensuring a network has sufficient capacity to handle all the traffic that transits it. Indeed, upgrading capacity can be seen as a quick and easy solution to overcoming a congested network. However, increasing network bandwidth is expensive in terms of time and money, and should not be seen as a long term strategy to avoid successfully implementing QoS. , Although core bandwidth capacities are relatively easy to increase, access technologies such as xDSL and mobile/wireless networks face greater challenges to overcome, and in the end, a network connection is as fast as the slowest link. From a strictly philosophical point of view The Tragedy of The Commons (Hardin G. 1968) describes a dilemma whereby a shared resource is exhausted by a number of individuals who each act in their own short-term interest. Despite it not being in any of their individual interests or long term aim to deplete the resource, unmanaged use of the common resource will ultimately lead to its exhaustion. It is therefore analogised in this thesis that the utilisation of a network will always reach its capacity and therefore introduce QoS challenges unless managed in an appropriate manner.

In contrast to simply increasing bandwidth, many enhancements to the best-effort performance of IP for QoS provisioning purposes have been extensively researched over the past 20 years. While a Type of Service (TOS) field was defined in the IPv4 specification, it has remained largely unused on the Internet, and as such IPv4 treated each packet to be of equal importance. Two main philosophies were proposed by the Internet Engineering Task Force (IETF): Integrated Services (Intserv) (Braden R., Clark D. et al. 1994) and Differentiated Services (Diffserv) (Blake, Black et al. 1998). The Integrated Services model provides an architecture that enables network level guarantees in order to provide a requested level of service for a specific application. However, Intserv requires that each node in the forwarding plane is Intserv enabled, and for each of these nodes to maintain states for each application requiring guaranteed delivery. These limitations restricted the scalability of the architecture and motivated the development of Differentiated Services. Diffserv, which classifies packets into levels of precedence, is currently the most widely accepted approach of providing QoS to IP networks. Despite the success of Diffserv, may be too static in its configuration to accurately reflect the changing usage dynamics of the Internet today. Furthermore, the very structure of the Internet (interconnected Autonomous Systems) inhibits the effectiveness of Intserv and Diffserv beyond operating within a single Autonomous Systems. In summary, the problem is that while Diffserv has worked well in environments where traffic profiles can be determined and subsequently used to tailor the configuration, the rapidly changing usage of ISP customers makes profiling and configuring QoS policies far more challenging, particularly when following a 'one-size-fits-all' approach. This challenge is further complicated when attempting to satisfy the diverse requirements of ISP customers while also striving to balance QoS provisioning with fairness among Internet flows, applications and customers.

It is proposed that an ideal solution would be an architecture that requires no reservation of resources, but instead be able to dynamically evaluate current traffic flow and adjust resource allocation as a real-time response to changing network conditions. This would involve a move away from the static, configuration model of Diffserv, enabling a more user-centric approach to QoS. The following section expands on this idea and defines the aim and objectives for this research.

1.1. Aims and Objectives

The aim of this research was to propose and investigate a novel QoS architecture that can evaluate network conditions and manage application traffic in response to network congestion from a user-centric perspective (rather than the traditional focus on a single service).

In order to achieve this aim the following objectives should be met:

- To review the state of the art for QoS mechanisms, and identify the shortfalls of current solutions that have limited their success.
- To examine and characterise the properties of a selection of modern Internet services that together represent the multimedia enriched Internet.
- To propose a novel user-centric QoS architecture aware of the requirements and behaviour of Internet services, and can make informed decisions to provide an optimum QoS for all the services in use by an Internet user.
- To investigate the impact of the proposed architecture on traffic and application performance
- To benchmark the performance of the proposed architecture against the identified state of the art.

1.2. Thesis Contents

Chapter 2 presents a detailed review of the state-of-the-art of QoS mechanisms. The chapter begins by considering the need for QoS provisioning, highlighting the impact network impairments have from both the protocol and end-user perspective, and also the issue of applications operating in a selfish manner. Having firmly established a need for QoS provisioning the chapter provides a review of related research and current QoS technologies operating at the network, transport and application layers of the TCP/IP model. The chapter concludes by bringing together the merits and shortfalls of these methods, summarising the motivations for a novel approach to QoS provisioning.

Chapter 3 presents a detailed study into modern day Internet services, consisting of a number of the most popular Internet services, HTTP, streaming video, streaming audio, P2P services, and VoIP. Each service is analysed in detail from a packet, flow and application perspective, identifying how applications today adapt and respond to network conditions. This investigation reveals not only the characteristics of each service from an operational perspective, but also highlights the different methods used by content providers to deliver their media.

Chapter 4 builds upon the findings of the previous two chapters to reflect and affirm the motivations for a novel QoS architecture. These motivations are used to produce a detailed specification of an ideal solution to user-centric QoS, which is then realised in to a pragmatic specification, which is described in detail and contextualised in a network. A discussion of the design trade-offs that were made is also provided, highlighting a need for a balance between the level of control over Internet traffic and the complexity of the overall system. The chapter continues with a description of the proposed management techniques for VoIP, web, streaming video, FTP and P2P traffic, using the findings from chapter 3 to ensure synergy between the operation of a service and its management. The final part of Chapter 4 describes the integrating and enforcement of the proposed traffic management policies into the Diffserv framework, which together provides the Congestion Aware Packet Scheduler (CAPS). This part of the chapter also includes a discussion on the optimal queue design for the architecture, detailed a number of queuing alternatives that could be used for policy enforcement. Finally, the chapter describes the approach used for queue scheduling by CAPS, proposing a move away from priority queuing on the grounds of fairness.

Chapter 5 presents the validation of the CAPS architecture using the *ns*2 simulation environment. The chapter begins by detailing the simulation methodology and describes how the findings from chapter 3 were used to define realistic traffic sources. The validation was divided into two phases, the first focusing on demonstrating the functionality of the CAPS architecture for each of the considered traffic types. The performance of CAPS for each traffic type was then benchmarked against three alternative network configurations, and the results interpreted and presented. The second validation phase focussed on benchmarking the performance of CAPS against the three alternative configurations, for a medium-large scale topology. The chapter concludes with a discussion of the results from both validation phases, which provided encouraging results that successfully demonstrated the dynamic management of network resources for user traffic.

Chapter 6 presents the key achievements and contributions from this research in the area of QoS provisioning, and dynamic traffic management. The chapter continues to present areas of future research that were identified during the project. Finally, a number of appendices are provided as supported material to the main content, including simulation source code, simulation topology scripts, analysis script files and published papers arising from the project.

2. Quality of Service – The State of the Art

2.1. Introduction

Efforts to improve the best-effort nature of the Internet have approached the problem from many different perspectives, encompassing every layer of the TCP/IP model (IETF 1989). This literature survey covers four main areas. Firstly, the discussion focuses on the need for QoS in IP based networks, which also goes on to provide typical QoS metrics for a range of Internet based applications, giving insight into how QoS is measured. The section also investigates changes in application behaviour and expectation are considered, particularly focusing on the rise of so-called misbehaving applications, such as software that uses peer-to-peer architectures, which present threats towards network stability.

Secondly, a review of current methods to provide QoS at the link and network layer is provided, describing many of the difficulties encountered by QoS provisioning solutions.

Following the review of network layer solutions, the chapter leads onto evaluating the transport layer mechanisms that aim to maintain network stability, through congestion avoidance and fairness algorithms.

The fourth main section of this chapter describes a number of application layer technologies that enable services to optimise the QoS delivered to the user in spite of an absence of QoS mechanisms within the underlying network.

Throughout this thesis network functionality and technologies are referred to as operating at specific layers. For continuity purposes the layer name and number are clarified in Figure 2.1, which includes a mapping of QoS mechanisms discussed in this thesis to their respective layer. The layer naming and numbering conforms with the IETF document defining the communication layers required for Internet hosts (IETF 1989).

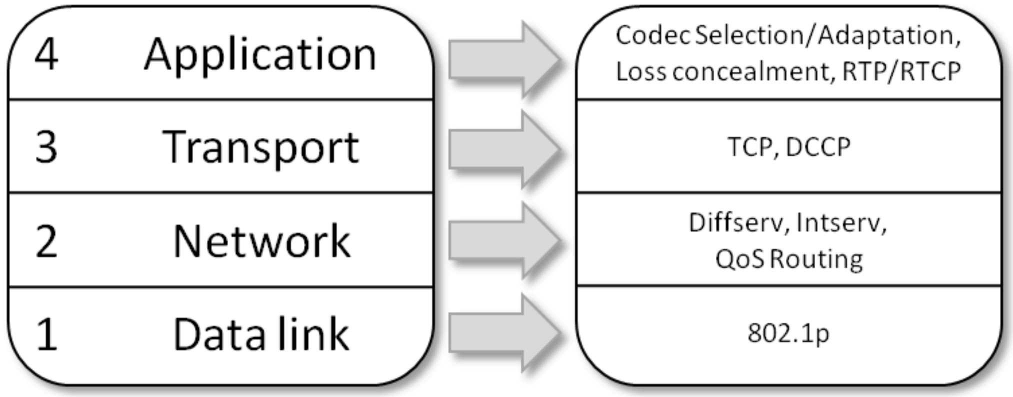

4 Application
3 Transport
2 Network
1 Data link
Codec Selection/Adaptation, Loss concealment, RTP/RTCP
TCP, DCCP
Diffserv, Intserv, QoS Routing
802.1p

Figure 2.1: IETF Internetwork Layers and corresponding QoS mechanisms

2.2. Defining Quality of Service

2.2.1. Quality of Service from the user or service perspective

The resulting application performance of QoS mechanisms is largely a subjective measure of an individual's satisfaction for a particular service. For example, the performance of voice services will primarily be based upon the clarity of the call, requiring a timely delivery of the voice data packets, which can reproduce clear, comprehensible speech to the receiver. One common approach to subjectively assessing telephony quality is using the Mean Opinion Score (MOS), which requires a test subject to rate the quality of a call from 1 (bad) to 5 (perfect). Similarly, the performance of a video conferencing service relies on a level of synchronicity between the audio and video streams. For non real-time services, such as web browsing or file downloading, the overall performance is likely to be judged on how quickly the web-page loads, or the length of time taken for a file to download.

However, subjectively assessing the quality of a service using either real life subjects or semi-automated perceptual evaluation techniques, for example the Perceptual Evaluation of Speech Quality methodology (ITU-T. 2001) is often neither practical nor desirable, and typically expensive to conduct for the majority of situations. Therefore, a quantitative approach based upon network parameters that can reflect or describe the subjectively experienced quality is often more attractive. For example, the performance of a voice service is influenced by not only codec choice but network parameters such as, bit-rate, delay, jitter (variance of inter-packet arrival times) and packet loss. The performance of video conferencing services rely on the audio and video streams experiencing similar end-to-end delays, with low jitter, allowing for synchronicity to be upheld during reassembly at the destination. In contrast, for non real-time services such as Web browsing or file downloading, the overall performance is the achieved throughput, which subsequently is a function of packet loss, end-to-end delay, and endpoint-related parameters.

Closely comparable with the MOS system of subjectively rating a voice call, the ITU-T G.107 E-Model (ITU-T. 2000) offers the ability to rate the performance of a voice call as a function of quantitative metrics, including packet loss and delay, a reduction of which, suitable for VoIP calls is given by (Cole and Rosenbluth 2001). The E-Model (and Cole's reduction) provides an R-Factor that scores the quality of a voice call between 0 and 100. Table 2.1 provides the relationship between the R-Factor, Mean Opinion Scores, the experience call quality and the expected level of user satisfaction.

R-Factor	MOS	VoIP Call Quality	User Satisfaction at lower limit
90-100	4.3-5	Best	Very satisfied
80-90	4.0-4.3	High	Satisfied
70-80	3.6-4.0	Medium	Some users dissatisfied
60-70	3.1-3.6	Low	Many users dissatisfied
50-60	2.3-3.1	Poor	Nearly all users dissatisfied

Table 2.1: MOS / R-Factor mappings

Equation 1 provides an estimated conversion from R-factor values to the corresponding MOS values (Reguera, Álvarez Paliza et al. 2008).

$$MOS = 1 + 0.035R + R(R - 60)x(100 - R)7 \times 10^{-6} \tag{1}$$

Table 2.2 provides a summary of typical performance metrics for a number of Internet services. While this list of services is not exhaustive, it covers a significant proportion of the most popular Internet services in use today, accounting for over 90% of all Internet traffic (Schulze and Mochalski 2009).

Service	Subjective Performance Metrics	Quantitative Performance Metrics
Voice	call clarity, acceptable delay, unbroken speech	bit-rate, end-to-end delay, jitter, packet loss
Video conferencing	voice clarity, smooth video, acceptable delay, synchronicity between audio and video	bit-rate, steady throughput (video smoothness), end-to-end delay, jitter, packet loss, low variance between audio and video stream arrivals
File downloading	time taken to fully download a file	packet loss, achieved throughput
Web browsing	fast loading of web pages	packet loss, end-to-end delay, response time
Peer-to-Peer	time taken to fully download a file	packet loss, combined throughput of incoming flows
Streaming Video	smooth playback, no waiting	maintained minimum throughput
Gaming	smooth in-game motion, low lag	delay, jitter, packet loss

Table 2.2: Performance Metrics for Internet Services

Table 2.2 describes the quantitative metrics that impact services, highlighting packet loss and delay as the two main contributing factors. However, it does not consider sensitivity of the application to these respective factors. Recommendation G.1010 (ITU-T. 2001) provides a model for End-user QoS categories Table 2.3, which classifies user activities into four groups, based upon their sensitivity to delay and whether they are error tolerant or intolerant, which describes if the service requires 100% of the transmitted data to be received, or if there is tolerance for data loss.

	Interactive (delay << 1s)	**Responsive** (delay ~2s)	**Timely** (delay ~ 10s)	**Non-critical** (delay >>10s)
Error Tolerant	Conversational voice and video	Voice/Video messaging	Streaming audio/video	
Error Intolerant	Command/control (e.g. Telnet, Messaging, interactive games)	Transaction based (e.g. E-commerce, WWW browsing, Email access)	Downloads – FTP, HTTP	Background (e.g. Peer-to-peer)

Table 2.3: ITU-T End-user QoS categories

2.2.2. Quality of Service from the protocol perspective

The previous sub-section introduced delay and packet loss as the two main factors that affect the performance of a service and its perceived QoS. This sub-section considers the impact that packet loss and delay have on performance from the perspective of the transport protocol.

2.2.2.1. Impact of network loss and delay on the UDP-based Applications

The simplicity of UDP means that packet loss and delay have no effect on the behaviour of the protocol itself; rather any effect will only be noticed at a higher level. For example, in the case of a UDP-based voice service, the loss or delaying of a packet will not impact the manner in which UDP delivers subsequent packets. However, from the application perspective missing or excessively delayed packets will impact the playback of the speech to the user, as described in section 2.2.1. Application layer mechanisms designed to minimise the impact of packet loss for UDP-based traffic, such as payload redundancy are reviewed later in this thesis in section 2.5.

2.2.2.2. Impact of network loss and delay on the TCP protocol

In direct contrast to UDP, the behaviour of TCP is fully dependant on the delay and packet loss experienced by the connection. TCP uses the additive increase multiplicative decrease (AIMD) algorithm (Allman, Paxson et al. 1999) to reach and maintain the maximum throughput possible, given the available bandwidth. Frequent packet drops or excessive delays inhibit the growth of the TCP congestion window (*cwnd* - the number of bytes that are in flight without acknowledgement), limiting the throughput that can be achieved. Over the years a number of modifications and improvements to TCP have been proposed and implemented, both for the congestion control and loss recovery components. Namely, AIMD (Jacobson 1995), CUBIC (Ha, Rhee et al. 2008) and Compound TCP (Tan, Song et al. 2006) for congestion control, and Reno (Allman, Paxson et al. 2009), New Reno (Floyd, Henderson et al. 2004) and SACK (Mathis, Mahdavi et al. 1996) for loss recovery. Little research is known of that describes the popularity of these components across modern Internet servers, and to this end the description of TCP behaviour that follows is based upon a TCP Reno implementation with AIMD congestion control, which can be considered a *base* implementation.

The TCP sender begins with the *slow start* algorithm, transmitting one segment and then waiting for its acknowledgement (ACK). Upon receiving the ACK (which describes the next expected segment. For example, having received the first segment, the receiver would send an ACK requesting the second segment) for the first segment, *cwnd* is increased from one to two, and the next two segments of data are sent. When the sender receives ACKs for these two segments *cwnd* is increased to four, and four segments are sent. This behaviour during *slow start* provides an exponential growth of *cwnd*, and continues until *cwnd* reaches the pre-defined threshold of *ssthresh,* or the advertised receiver window. If loss is inferred before *cwnd* reaches either of these thresholds then *ssthresh* is set to half of *cwnd* at the time loss was detected.

The sender infers that loss has occurred if three duplicate acknowledgements are received or if the retransmission timeout (RTO) expires. A duplicate acknowledgement occurs when the sender receives an acknowledgement for a segment, numbered lower than the last transmitted segment. Figure 2.2 illustrates this process.

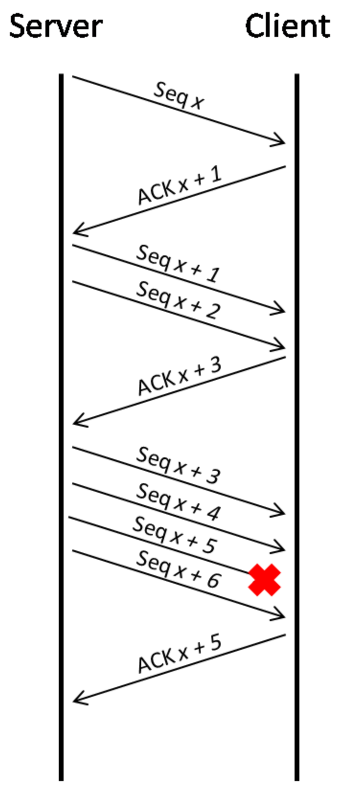


Figure 2.2: TCP Duplicate Acknowledgements following packet loss

The second method that a TCP sender may use to infer that loss has occurred is using a retransmission timer (RTO), which is estimated by the sender using the Round Trip Time (RTT) – the time taken for a segment to be sent and for its associated ACK to return to the sender. At the start of a TCP connection, (prior to the sending being able to estimate the RTT), the RTO is set at 3 seconds. Upon the sender receiving an ACK the RTT measurement *R* can be made, from which a value for RTO is calculated using Equations 2, 3 and 4, where SRTT is the smoothed RTT, G is the minimum value for RTO and *k* is set to 4, (Paxson and Allman 2000).

$$SRTT \leftarrow R \qquad \mathbf{2}$$

$$RTT_{var} \leftarrow \frac{R}{2} \qquad \mathbf{3}$$

$$RTO \leftarrow SRTT + \max(G, k.RTT_{var}) \qquad \mathbf{4}$$

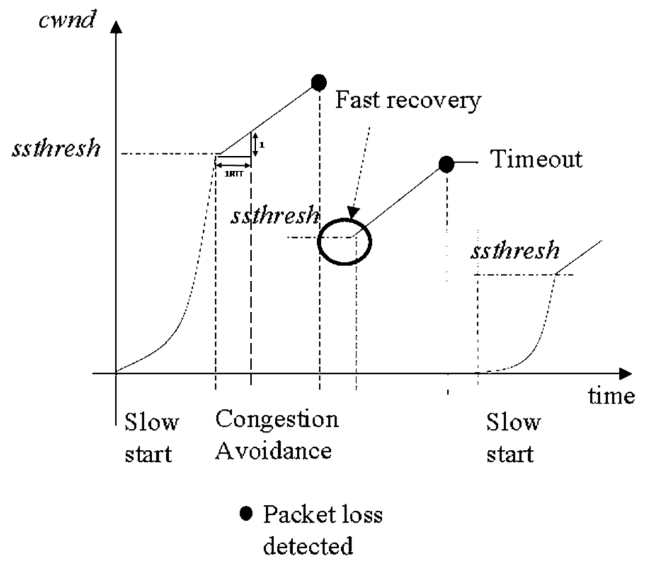
Following the subsequent RTT measurement R′, the sender uses the following formula to calculate RTO, where α = ⅛, β = ¼ , G is the minimum value for RTO and *k* remains at 4.

$$RTT \leftarrow (1-\alpha).SRTT + (\alpha.R') \qquad \mathbf{5}$$

$$RTT_{var} \leftarrow (1-\beta).RTT_{var} + \ (\beta.|SRTT - R'|) \qquad \mathbf{6}$$

$$RTO \ \leftarrow SRTT + \max(G, k.RTT_{var}) \qquad \mathbf{7}$$

Using these two methods of loss inference the slow start algorithm (shown in Figure 2.3), increments *cwnd* for each received acknowledgement, until either packet loss is inferred or *cwnd* reaches the advertised receiver window. If packet loss is inferred before *cwnd* reaches the advertised receiver window then the current value of *cwnd* is recorded as *ssthresh* and TCP will restart the slow start algorithm until *ssthresh* is reached again, after which it will enter *congestion avoidance* and *cwnd* is increased each RTT (providing a linear growth, rather than exponential). There are many variants of TCP, but TCP Reno is the most widely deployed, therefore the following description refers to the behaviour of TCP Reno. If three (or more) duplicate ACKs are received by the sender while in *congestion avoidance*, the sender employs *Fast Recovery* setting *cwnd* to half of *ssthresh* and retransmitting the lost packet without waiting for RTO to expire (*Fast Retransmit*). The connection will remain within *congestion avoidance*, increasing *cwnd* each RTT, and avoiding resetting *cwnd* to the initial value. However, if the sender fails to receive an ACK for a segment prior to RTO expiring (i.e. if the ACK is lost), *cwnd* is reset to the initial small value and the connection must restart from *slow start* again. This behaviour is illustrated in Figure 2.3-a, where the impact of RTO expiring can be seen to be far greater than the sender response to receiving three duplicate ACKs.



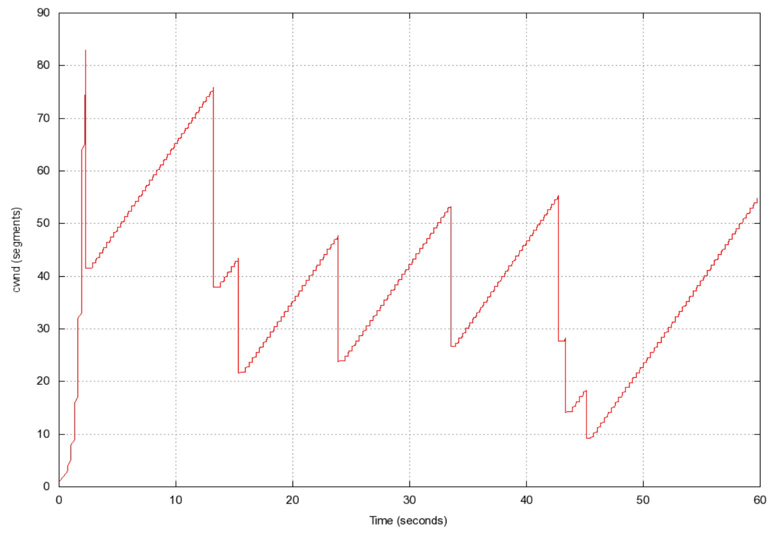


Figure 2.3: TCP Reno Congestion Window Evolution. Theoretical (top) and Simulated (bottom)

From Figure 2.3-b[1] the congestion window is shown to increase exponentially at the start of the connection, increasing until packet loss occurs. Following this loss *cwnd* is reduced to half of the maximum value before loss occurred and the connection enters *congestion avoidance*,

[1] The simulated representation of the TCP congestion window was produced using *ns*2 (ns2, 2010). A simple topology was used, including a traffic source and destination, coupled with an intermediate node, which was configured with a loss-module, enabling controlled packet loss to model the TCP *cwnd*.

continuing to increase *cwnd* but linearly as opposed to exponentially, until just after 10 seconds another loss occurs. At this point *cwnd* is again halved as TCP enters *Fast Recovery*, still increasing *cwnd* linearly by 1 MSS for every ACK received. Over the course of the transmission a number of subsequent packet drops can be seen, which consequently result in *cwnd* being reduced by half each occurrence.

Furthermore, it is noted that during the *slow start* the performance of the connection can be severely degraded if loss occurs. As discussed, the loss of a data packet during *slow start* will cause *cwnd* to be reset back to the initial value; however, even more detrimental to the connection performance during *slow start* is the loss of an ACK. In such a case, the sender would be required to wait for the expiration of RTO, which during *slow start* is unlikely to have been accurately smoothed towards the RTT, and may be as great as 3 seconds. For an exceptionally lossy link, repeated losses of ACKs may introduce substantial delay, and drastically impair the user experience.

The steady-state behaviour of TCP Reno has been described mathematically by, (Padhye, Firoiu et al. 2000). Provided in equation 8, Padhye et al. show the relationship between the sending rate of the connection *T*, given a loss rate *p*, a TCP retransmit timeout value of T_{RTO} and a packet size of *s*.

$$T = \frac{s}{RTT\sqrt{\frac{2p}{3}} + T_{RTO}\left(3\sqrt{\frac{3p}{8}}\right)p(1+32p^2)} \qquad \textbf{8}$$

2.2.2.3. The Issue of Fairness

It is widely accepted that self-limiting sources, such as TCP, will achieve a far lower throughput when competing against high-throughput unresponsive flows, such as UDP. This effect is illustrated in Figure 2.4, which shows the result of a simulation (as per section2.2.2.2, these simulations were also conducted using the *ns*2 simulator (ns2 Network Simulator 2010)) of two homogeneous TCP flows obtaining an equal share of bandwidth (graph on the left), and conversely the graph on the right shows that a high-throughput UDP flow maintains a disproportionate share of the bandwidth due to the lack of a mechanism to throttle back the transmission rate when congestion is detected within the network. Recently proposed congestion controlled alternatives to UDP at the transport layer are reviewed later in this study (section 2.4.1).

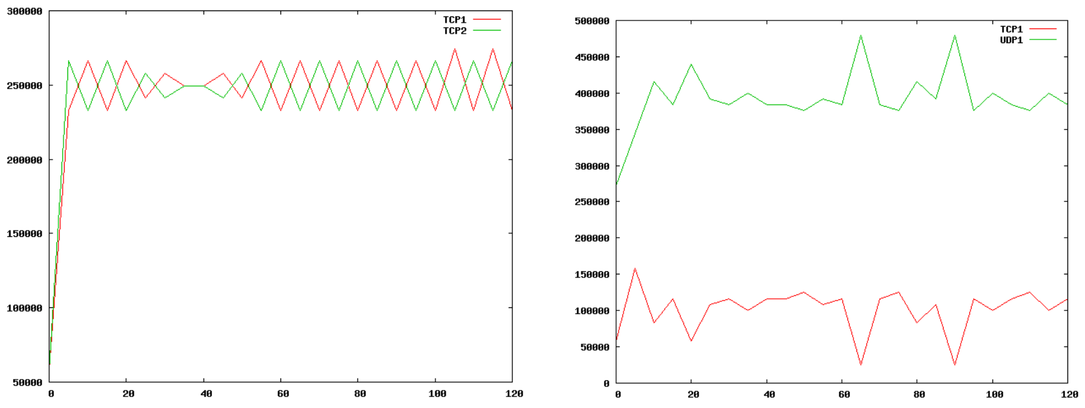


Figure 2.4: Illustration of bandwidth obtained by TCP and UDP flows

However, the problem of unfairness is not limited to between responsive and unresponsive flows. From the description of TCP behaviour in the previous section it can be appreciated that two flows with different RTTs will achieve different throughputs, since the flow with a smaller RTT will increase *cwnd* more rapidly, given the potentially higher rate ACKs will be received. Furthermore, two TCP clients with different advertised windows will also achieve dissimilar throughputs. Figure 2.5 and Figure 2.6 illustrate this unfairness between heterogeneous TCP connections.

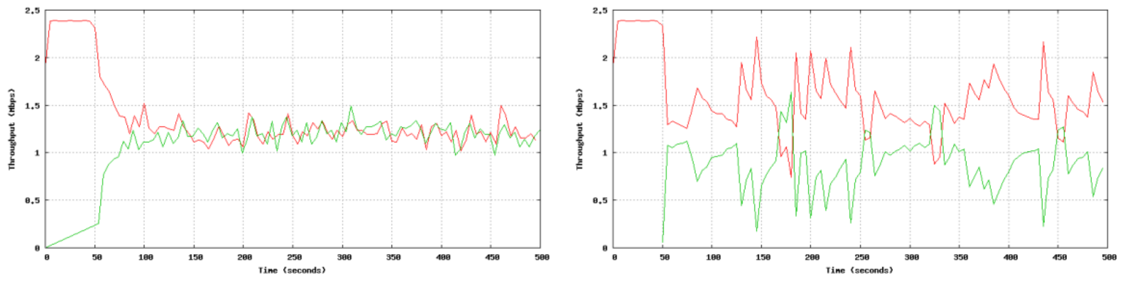


Figure 2.5: Throughput achieved by two concurrent TCP connections.

a). Homogeneous flows b). Heterogeneous flows (different RTTs)

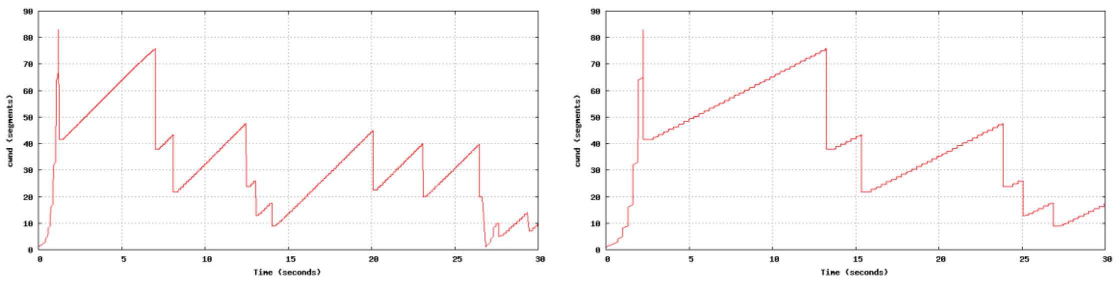


Figure 2.6: Observation of cwnd for two TCP flows, a). Short RTT ~ 150ms b). Long RTT ~ 300ms

Recognising the impact that heterogeneous path characteristics have on protocol performance is discussed throughout this thesis, specifically when attempting to control Internet flows proportionally to their achieved throughputs.

2.2.2.4. Changing Traffic Dynamics

The increasing threat posed by unresponsive transport protocols was addressed in the previous section. However, the popularity of Peer-to-Peer (P2P) distributed architectures has placed an additional demand on networks. P2P technologies have been primarily promoted by file sharing applications and assisted by the increased availability of residential broadband Internet access. As a result, in 2009 P2P was reported to account for almost 70% of all Internet traffic (Schulze and Mochalski 2009). In recent years the dominance of P2P has drastically declined to around 20% of Internet traffic, making way for Video on Demand (VoD) services, which overtook P2P as the largest contributor of Internet traffic in 2010 and represented 25-40% of all traffic in 2011 (Sandvine 2011). This shift in trend is possibly due to the instantaneous nature of VoD services, whereas media acquisition via P2P allows for storage and replay, VoD requires a download-per-view. However, despite this decline, P2P still represents approximately 20% of all Internet traffic, and due to its aggressive nature is still considered highly relevant when addressing QoS for the Internet.

Within P2P architectures, all participants (peers) act as both providers and consumers of resources (these resources include but are not restricted to, processing power, bandwidth, available memory or data), the P2P model contrasts with traditional client-server architectures.

In the context of P2P-based file sharing, peers take advantage of large numbers of participants having an identical copy (or portion) of a file. Using a P2P architecture peers can exploit these multiple copies of a file in multiple locations to potentially achieve a greater throughput than would be possible under a traditional client-server model. P2P-based file sharing applications open numerous TCP connections with other peers and begin downloading pieces of the file from these peers. As a user obtains more and more pieces of the file being downloaded, it too will begin to upload these pieces to other peers, thus creating a swarm of peers exchanging pieces of a file. The behaviour of P2P applications is looked at more closely in section 3.4 with a detailed study of the most popular P2P file sharing protocol, BitTorrent and also P2P properties of other P2P-based applications.

The behaviour of P2P-based applications raises significant concern over the assumption that TCP-based applications operate fairly with coexisting traffic. Even if each TCP flow shares similar parameters (RTT, RTO, W_{max} and packet loss), and hence should behave relatively fairly to each other, at the application level, P2P-based systems obtain a far greater share of the network bandwidth compared with applications that only establish single TCP connections, Figure 2.7 illustrates this scenario.

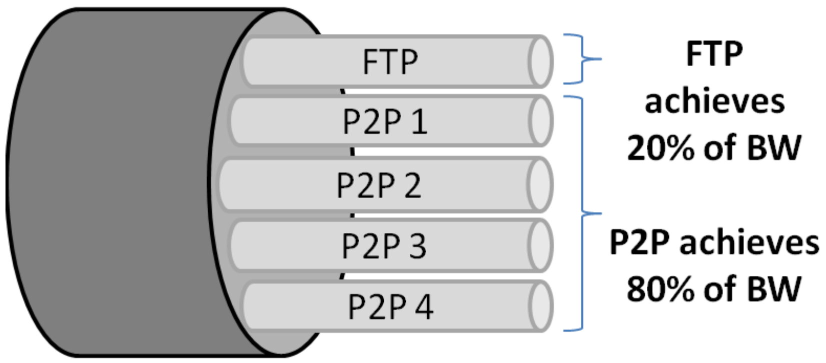


Figure 2.7: Illustration of the distribution of bandwidth between FTP and P2P applications

To further threaten network stability, P2P-based applications are also claimed to use UDP for signalling purposes(John, Tafvelin et al. 2008), which, while are typically very short (less than 3 packets), still contribute to the rising volume of unresponsive Internet traffic.

In addition to obtaining an unfair share of the bandwidth over other applications through multiple TCP connections, the rise of P2P has changed the direction of bandwidth demand from heavily asymmetrical to a more symmetrical model, however, consumer connections have remained largely unchanged in design.

2.2.3. Summary of Quality of Service Introduction

This section has introduced the performance requirements of a selection of Internet services, and how these can be mapped to network parameters. It is highlighted that the dissimilar requirements of Internet services conflict with the uniform forwarding of IPv4, and therefore the need for an enhanced service that can take into account an applications needs is clear.

This section has described in detail the relationship between the throughput of a TCP connection and the experienced delay and packet loss. This relationship was further examined to demonstrate how the performance of TCP flows with different properties (RTT, RTO, p and duration) varies greatly, highlighting the need to consider the application as well as the protocol requirements when provisioning for QoS.

The following sections of this chapter review the current methods of QoS provisioning at the network, transport and application layer, in order to satisfy the QoS requirements of Internet services, as described in this section.

2.3. Current methods of QoS Provisioning at the Network Layer

This evaluation of QoS provisioning at the network layer covers four main areas of interest: router queuing models, end-to-end QoS solutions such as Integrated and Differentiated services, QoS aware packet routing, and traffic management techniques, such as traffic shaping and policing.

2.3.1. Queuing Models

Packet loss in under provisioned packet switched networks (where under provisioned is defined as total throughput of flows being greater than bottleneck capacity) is unavoidable given the finite nature of buffers (queues) within network nodes. The impact of packet drops on application and protocol performance were introduced earlier, however, the nature of the drops, which are ultimately determined by the queuing models used within the network, were not considered. This section of the thesis provides an overview of queuing models that are commonly used within networks. Within this section each queuing model is simply described, a discussion of the best queuing model for the novel architecture is given later in the thesis (section 4.9.1).

The most simplistic queuing model that can be implemented is the First In First Out (FIFO) queue, also known as a taildrop queue. As the name suggests, arriving packets enter the queue sequentially, dequeuing in the order of their arrival. As the simplest queuing model available, taildrop is often cited as the most widely implemented queuing model in Internet routers (however, no definitive study has been identified to confirm this assumption). Although simple in its design and operation, in the event of persistent congestion a taildrop queue can result in higher delays and prolonged periods of congestion, furthermore when the queue reaches capacity packet drops can occur in bursts. This bursty packet loss can be detrimental to protocol performance, in particular causing TCP global synchronisation. This term describes a scenario when a number of TCP connections sharing a mutual congested link experience simultaneous packet loss and all suspend transmission together, and then restart transmission simultaneously, causing the network to oscillate between under-utilised and congested. Furthermore, taildrop queues do not implement any form of traffic precedence, which may lead to low priority packets remaining in the queue, while packets from higher priority services are dropped due to congestion. In response to these shortfalls of using taildrop queues a number of alternative queuing models have been developed over the past 20 years.

Random Early Detection (RED) (Floyd S. and Jacobson V. 1993), is an Active Queue Management (AQM) algorithm that was proposed to provide early warning to end hosts that the network is congested and that they should reduce their transmission rates. The theory behind RED suggests that early notification of congestion can prevent the queue from reaching its maximum capacity and hence addresses the bursty loss effect of a congested Taildrop queue, which would also prevent the global synchronisation of TCP connections (given that TCP connections would share the same *cwnd* evolutionary state at the same time). A RED queue is configured with two thresholds, *MinThres* and *MaxThres*, where *MaxThres* is less than the size of the queue. The RED algorithm computes the moving average queue size *avg*. If *avg* is less than *MinThres* then packets are forwarded as per usual, if *avg* lies between *MinThres* and *MaxThres* the router calculates the P_b , which varies linearly between 0 and Max_p where Max_p is the maximum value for P_b. Equation 9 summarises the packet drop probabilities for RED, and Figure 2.8 illustrates the drop functions for the Taildrop and RED queuing algorithms.

$$d(avg) = \begin{cases} 0 & if\ avg < MinThres \\ \dfrac{avg - MinThres}{MaxThres - MinThres} Max_p = P_b & if\ MinThres < avg < MaxThres \\ 1 & if\ avg > MaxThres \end{cases} \qquad \mathbf{9}$$

If the average queue size is greater than *MaxThres* then all arriving packets are dropped. The probability of dropping a packet is described as being approximately proportional to the flow's share of bandwidth, and is more evenly spaced compared with Taildrop, which prevents global synchronisation.

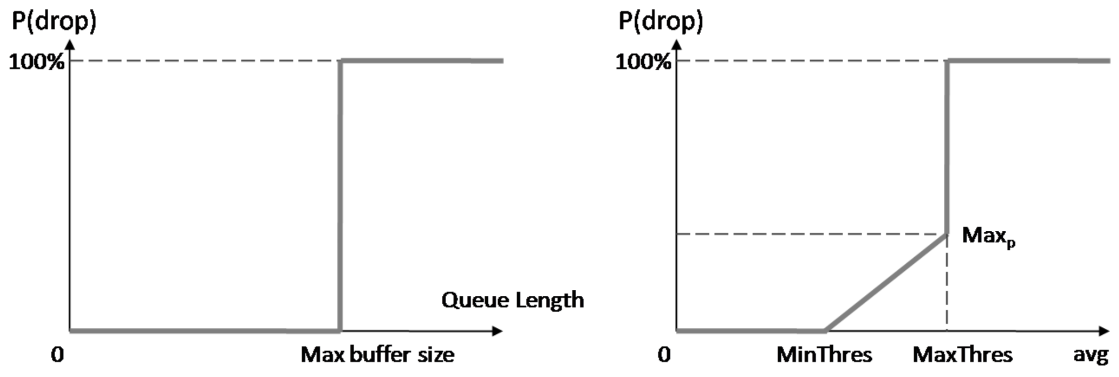


Figure 2.8: Drop Function for Taildrop and Random Early Detection Queues

A well documented challenge when using RED queues is how to choose the optimal parameters for the traffic in question (Lin and Morris 1997; Firoiu and Borden 2000; Floyd S., Gunmamadi R. et al. 2001). Tuning the values of $min_{th,}$ max_{th} and max_p is a task left to the network operator, but whether or not typical queue lengths are known is questionable; setting min_{th} too low will inadvertently trigger premature packet drops; setting it too high may render the RED implementation ineffective. Furthermore, Floyd and Gummadi discuss that setting max_p too low can results in reduced utilisation as fewer flows packets are dropped as a result of early detection and many drops occur once max_{th} has been reached. Floyd and Gummadi also discuss in their 2001 paper that the queue averaging coefficient needs to be considered when trying optimise the averaging function; if calculated too often then the *avg* will reflect the queue for a fraction of a RTT and not be fully representative of TCP behaviour. In light of these challenges, Floyd and Gummadi proposed Adaptive RED, which adjusts the value of max_p dynamically in order to maintain the *avg* halfway between min_{th} and max_{th}. Adaptive RED also includes a mechanism to smooth packet drops when *avg* exceeds $max_{th};$ Gentle RED (Floyd S. 2000) varies the drop probability linearly between max_p and 1 when $max_{th} < avg < 2 \times max_{th},$ rather than immediately being set to 1.

As an extension to the RED queuing algorithm, Explicit Congestion Notification (ECN) was proposed to inform end hosts of congestion rather than simply dropping packets and allowing the end host to infer congestion (Ramakrishnan, Floyd et al. 2001). ECN requires both end-points to recognise when the ECN flag has been set in the TOS field within the IPv4 header. Despite the potential gain in network stability, ECN has failed to be deployed on a large scale. A paper by (Matowidzki 2003) discusses some of the reasons for the poor deployment of ECN, which in 2004 was employed by just 2.4% of servers worldwide, (Medina, Allman et al. 2005). Matowidzki describes one limiting factor as the possibility for misbehaving end-points to ignore ECN flags and not respond to congestion notification, which, without the ability to enforce ECN response, raises concerns over whether ECN is worth the deployment challenges. A second remark from Martowidzki's article suggests a change in the Internet pricing model, from a best-effort model to a QoS orientated model, whereby service level agreements (SLA's) are the basis upon pricing schemes. If such a change in Internet model were to be fully implemented then difficulties may arise as to how services should be charged for in the event of ECN flags being set. For example, with ECN flags having been set it can be assumed the network is facing congestion, therefore there is a highly probable possibility for degradation in service quality, and therefore the question of whether a reduction in services charges is applicable is raised. A major shortfall which can be attributed to the poor widespread deployment of enhancements of established protocols such as ECN proposed to IP is highlighted by (Kuzmanovic 2005), suggesting that if ECN was deployed incrementally across the Internet how should ECN flagged packets should be treated when multiplexed with packets from sources not supporting ECN to ensure a fairness between ECN-capable clients/servers and those not supporting the feature.

RED addresses the problem of bursty loss and global synchronisation of TCP flows however, it does not offer the ability to provide precedence or QoS to specific traffic types, Weighted-RED can provide such functionality. In principle Weighted-RED operates in exactly the same manner as RED, but allows the use of the precedence field (TOS) within the IP header to influence the drop probabilities of the queuing packets. Using the TOS field a router can offer a higher drop probability to less favoured services while protecting preferred packets. Weighted-RED is available on commercial routers from Cisco Systems.

2.3.2. Integrated Services (Intserv)

As introduced briefly in the previous chapter, Intserv (Braden R., Clark D. et al. 1994) was one of two key philosophies for providing QoS to IP networks proposed by the IETF. Intserv, focuses on the explicit reservation of network resources between a source and destination. Every router between source and destination is required to implement Intserv, and each application requiring a level of QoS must request its own resource reservation. This resource reservation is performed using a flow specification (Flow Spec), which is distributed throughout the network using the Resource Reservation Protocol (RSVP). Given that the resource reservation is requested by the application, the information within the Flow Spec very accurately describes the nature of the traffic. A Flow Spec consists of two elements, a TSPEC and an RSPEC. The former describe the traffic characteristics from a token bucket perspective, for example the expected packet rate of the service (token rate) and its burstiness (token bucket depth). The latter, the RSPEC, describes the type of reservation required, for which three options exist; Best-effort, Controlled Load and Guaranteed. Best-effort, as the name suggests provides no explicit resource reservation, and the flow is subject to the same handling as it would receive in a non-QoS network. Controlled load ensures the flow receives a better-than-best-effort service, analogous of a lightly loaded network, likely to experience low delays and infrequent packet loss. The Guaranteed service provides bounded delays and zero packet loss, designed primarily for sensitive real-time services such as VoIP and Video conferencing.

While Intserv proved feasible on small-scale networks, maintaining flow states within routers restricted its deployment on the Internet at a time when the growth of the Internet was rapidly increasing. The explicit end-to-end reservation of resources introduces the need for each intermediate node to support Intserv, and further still maintain a state for each application requiring a guaranteed service. This design is cumbersome at a medium scale, but in the case of the Internet, the end-to-end path will almost certainly cross multiple Autonomous Systems; being able to maintain Inter-AS states on every node from source to destination is simply not implementable. Various modifications to aggregate resource reservations at the network edge were proposed, including (Bernet, Ford et al. 2000; Baker, Iturralde et al. 2001), however, the focus of providing QoS had shifted away from per-flow handling, towards per-aggregate.

2.3.3. Differentiated Services (Diffserv)

In contrast to the high granularity of the Intserv model, the IETF proposed Differentiated services (Diffserv) (Blake, Black et al. 1998), which featured a lower granularity than Intserv, adopting a class-based traffic management mechanism for a per-aggregate handling. Diffserv is typically configured to provide two Per-Hop Behaviour groups traffic, Expedited Forwarding (EF) (Davie, Charny et al. 2002) and Assured Forwarding (AF) (Heinanen, Baker et al. June 1999), in addition to the standard best-effort delivery of IP. Expedited Forwarding provides guarantees on delay, jitter and loss parameters and also queuing priority, while AF provides guarantees on packet loss and throughput, providing the traffic source stays within a committed information rate (CIR).

The Diffserv architecture adheres to the original design principle of the Internet, maintaining intelligence at the edge of the network, and simplicity within the core. Diffserv edge routers are responsible for four primary functions: classifying, metering, marking and shaping incoming traffic flows.

- Packet classification is performed by examining information within the protocol headers. Attributes used for classification include port numbers (application type), protocol type, source/destination addresses and layer 7 application signatures, identified through the use of Deep Packet Inspection (DPI). A combination of the use of port numbering and DPI recognition is the most common technique used.
- Flows classified as belonging to the AF PHB are also metered by the edge router to determine if they are conforming to the CIR, describing those packets within the CIR as *in-profile* (IN), and those beyond the CIR as *out-of-profile* (OUT).
- The marker component marks each packet with an appropriate Diffserv Code Point (DSCP) based on its classification and whether it is IN or OUT. The DSCP is stored in the Type of Service (TOS) field of the IP header, describing the PHB that the packet belongs to and its required precedence level.
- For flows not conforming to their CIR, packets may be passed through a traffic conditioner to *shape* the transmission rate of the flow, or in the event of congestion, drop OUT packets.

Figure 2.9 illustrates an example Diffserv topology, where an edge router connects a Diffserv domain to the core network. The operations of an edge router are expanded for clarity. It should be noted that DSCP markings of incoming packets are untrusted and are subject to re-marking by the edge router.

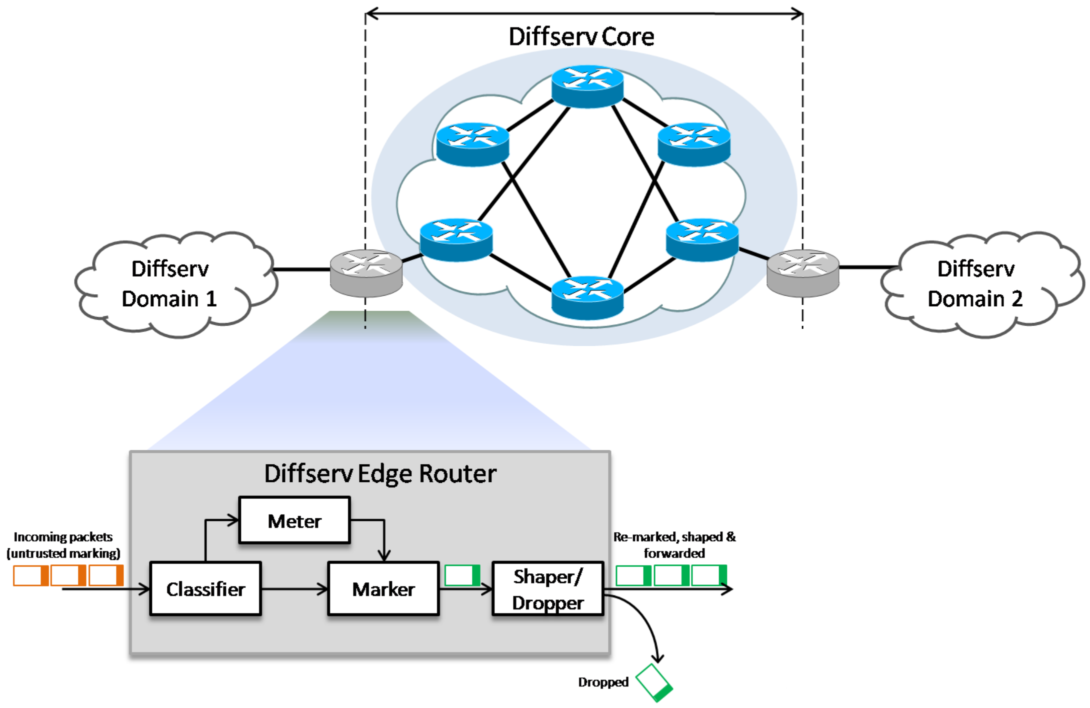


Figure 2.9: Example Diffserv Topology with an expanded view of edge router operations

The core routers are responsible for forwarding packets to their destination based upon the PHB described by the DSCP.

Diffserv allows network administrators to define their own traffic classification and forwarding policies for use within their own administrative systems, giving priority to the traffic that they deem to be the most important. However, it is normally recommended that the EF PHB is used for real-time services (such as VoIP) in order to meet the stringent QoS requirements of such services.

The AF PHB was designed for elastic traffic, providing a guaranteed throughput to a flow, provided it remains within its target throughput. AF can be configured for individual application flows, or to manage flow aggregates. Examples of applications that may use this PHB are streaming video and predictable periods of prolonged file transfer, i.e. overnight backups. The AF PHB uses RED queues with IN and OUT (RIO) (Clark and Fang 1998), to provide protection to IN packets and if necessary drop OUT packets. The AF PHB supports a maximum of four traffic classes, with each class featuring a maximum of three levels of drop precedence. It is common practice to use the three-colour system to describe IN packets as *green*, and OUT packets as *yellow* and *red*, providing two degrees of dropping for OUT packets (thus utilising three levels of drop precedence).

While the configuration of a Diffserv network is left to the discretion of the network administrator, guidelines have been provided based upon the ITU-T recommendation G.1010 (ITU-T. 2001). Using these ITU-T recommendations, (Babiarz J., Chan K. et al. 2006) present a suggested order of precedence for IP traffic within a Diffserv network, in descending order of precedence; Telephony, Real-time interactive, Broadcast video, Streaming multimedia, High-throughput data and Standard (all other traffic). This recommendation for QoS is considered flawed in a number of aspects, firstly, for those users not using premium services their traffic may not receive a fair allocation of network resources. Secondly, providing QoS from solely an application / class perspective does not consider a user's concurrent activities and therefore the quality of the overall user experience is not considered, only that of the favoured services.

2.3.3.1. Adaptive Diffserv markers

Despite not considering the overall user experience the EF PHB satisfies the requirements of real-time services well, however, the suitability of the AF PHB for TCP traffic has been the topic of much controversy over the past decade (Seddigh, Nandy et al. 1999; Lochin and Anelli 2009).

This concern stems from two main issues, firstly a lack of awareness by TCP as to whether a packet is IN or OUT. If a TCP connection is exploiting excess bandwidth that is available on the network, all packets beyond the CIR of a flow are marked as OUT. If the network then becomes congested, these OUT packets have a high probability of being dropped, and hence TCP will consequently reduce its throughput. However, TCP has no knowledge that these packets were opportunistically exploiting excess bandwidth, and therefore simply responds as normal by reducing its throughput, possibly below the CIR and hence incurring a delay while *cwnd* increases again and the throughput is allowed to return to the CIR.

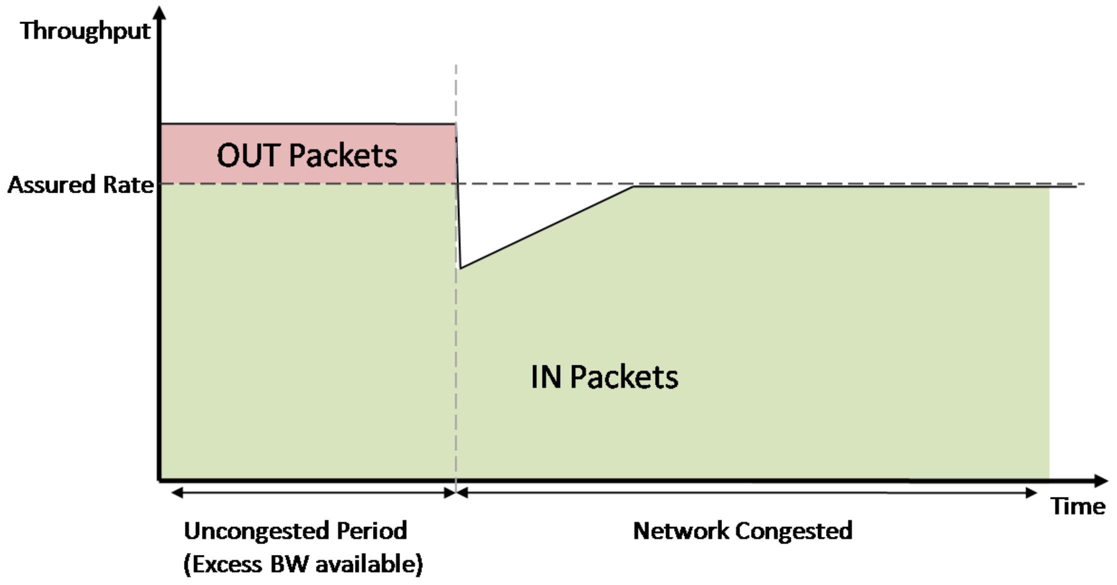


Figure 2.10: TCP responding to AF packet dropping

The second issue that is claimed to limit the effectiveness of AF provisioning for TCP is the marking process, a topic that has been the subject of much research over recent years. The standard DSCP marking process is argued to operate in a way that does not reflect the dynamics of TCP, raising concerns of fairness, which has led to efforts focussing on how the marking rate should be proportional to the throughput, target rate, packet loss, RTT and W_{max}.

Motivated by these shortfalls, many enhancements to the marking algorithm have been proposed, aiming to control the number of losses of OUT packets proportionally to the throughput of the flow, by adjusting the probability of marking a packet as OUT. The majority of this work has enhanced the Two Rate Three Colour Marker (TRTCM) and the Time Sliding Window Three Colour Marker (TSW3CM) – the two standard Diffserv marking schemes. TRTCM is based on the token bucket, determining if a packet is IN or OUT based on four service parameters; CIR, Committed Burst Size (CBS), Peak Information Rate (PIR) and Peak

Burst Size (PBS). The TSW3CM marks packets probabilistically as a function of the estimated throughput of the parent flow and the agreed CIR and PIR.

An early improvement to an enhancement of both TRTCM and TSW3CM was given by (Yeom and Reddy 2001), who propose a marking algorithm that dynamically adjusts the target rate of a flow given the current network conditions. The algorithm reduces or increases the target rate of the flow when the network is respectively over or under subscribed, managing the proportion of IN and OUT packets given the current network conditions. A similar proposal was the Enhanced Time Sliding Window Three Colour Marker (ItswTCM) (Su and Atiquzzaman 2003), which marks packets in proportion to the CIR of the flow. Similar to the proposal by Yeom and Reddy this algorithm improves on the sharing of excess bandwidth compared with the standard TSW3TCM algorithm. A number of modified versions of ItswTCM have also been recently proposed (Elshaikh, Othman et al. 2008; Sudha, Maddipati et al. 2008) each improving upon the fairness achieved among TCP aggregates, and the latter considering fairness between TCP and UDP flows, which is achieved through a more aggressive marking probability for UDP traffic. However, none of these proposals consider the effect heterogeneous TCP flows have on the respective marking algorithms, for example flows with different RTTs, and therefore the effectiveness of these algorithms in real networks with highly varied flows is questionable.

Extensively building upon the work outlined above, is a proposal from (El-Gendy and Shin 2003). The Equation Based Marker (EBM) uses the TCP throughput equation presented by (Padhye, Firoiu et al. 2000), which was presented in 2.2.2.2. Gendy et al. Use the inverse of Padhye's TCP equation to provide their algorithm with the loss probability needed to control a TCP flow at a specific throughput. The EBM algorithm requires an estimate of the RTT and T_0 for each flow, which is determined using TCP timestamps within the TCP header. Also estimated by EBM is the current loss probability of the network, measured by counting loss events within a period of time. Observing a loss event is relatively easy in the case of loss being inferred by three duplicate ACKs, but the authors fail to describe a method to identify packet loss that triggers a timeout. Gendy also suggests that TCP flows and UDP flows should not experience the same management techniques, given that TCP is responsive and UDP is unresponsive. It is proposed that two queues, one for TCP traffic and a second for UDP, combined with a weighted round-robin scheduler to service each queue proportionally based on the CIRs of the flows in each queue. Despite vastly improving on the fairness and accuracy of alternative markers, the algorithm remains an incomplete solution. Padhye's TCP equation models only TCP connections in steady-state - that is those which have exited slow start and are in congestion avoidance. Furthermore, Padhye's equation models the behaviour of TCP Reno, which, while among the most popular TCP variants implemented today, would require the algorithm to either be able to detect alternative TCP implementations and adapt its marking accordingly, or provide sub-optimal marking to non Reno flows.

Another proposed improvement to the Assured Forwarding PHB from recent years focuses less on attempting to integrate a TCP-like model into the marking algorithm, but instead aimed to protect the packets of a TCP flow during so-called vulnerable periods, i.e. during slow-start and fast recovery. (Mellia, Stoica et al. 2003) present two main implementations of the TCP-aware algorithm, firstly modifying the TCP stack of the host, and secondly at the ingress of an edge router. Acknowledging the difficulties in deploying changes to the TCP stack the router-based implementation is considered the most feasible in reality. However, assumptions are made by the router-based implementation as to the value of ssthresh and cwnd. While the results presented show a notable improvement in the completion times for short-flows, and a reported 20% increase in the throughput of long-flows, the entire study focused heavily of web-like traffic, with the longest flow being 90kB (158 packets). A similar proposal to achieve QoS for HTTP traffic within a Diffserv network is given by (Alcaraz, Gilly et al. 2007). A three colour marking algorithm is described that provides preferential treatment to short flows over medium and long flows, which reduces the probability of packet loss to vulnerable short flows.

2.3.3.2. Dynamic Diffserv Proposals

In contrast to these recommended guidelines there have also been dynamic, user-oriented Diffserv configurations proposed within research. (Kung and Kuo 2006) and (Molnar and Vrba 2008) present similar concepts that introduce a user interface to the Diffserv architecture. The framework presented by Kung et al. describes the configuration of traditional Diffserv from a very similar perspective given by this thesis, largely too static and detached from the end-user's perception. To address this problem, Kung et al. propose a hybrid framework that combines Intserv Local Area Networks (LANs) with a Diffserv Wide Area Network (WAN). The framework also provides a user interface that allows the user to request a specific level of service for each of their individual applications. The granularity of this request can vary depending upon the technical knowledge of the user, ranging from a basic four colour traffic classifier (Premium, Gold, Silver and Bronze), to specifying high level information such as audio sampling rate. The framework then negotiates the required resources using Intserv/RSVP within the network in order to guarantee the QoS for the user's application, and mapping the specified application requirements to the "most appropriate" Diffserv traffic class. Molnar et al. again argue that traditional Diffserv is too static in its configuration, and that the edge router may fail to correctly classify user traffic to the desired traffic class. To address this problem, a framework is presented that utilises the Simple Network Management Protocol (SNMP) to retrieve the Diffserv traffic classes from the network, and presents them to the user, outlining the DSCP value, minimum guaranteed throughput, maximum tolerated throughput, packet treatment for out-of-profile packets and the relative priority compared with alternative classes. It is proposed that using (some or all of) these parameters a user can have more control over their traffic, ensuring it receives the treatment they consider most appropriate. A similar proposal is made by (Schumacher, Dobler et al. 2010), whereby a mobile end system estimates the perceived quality of experience for the current services using the current network connection, and then the user is requested to confirm this estimate by rating their experience. This rating is then used to determine if an alternative network connection (i.e. 3G, UMTS, WiFi etc) could provide an improved QoE.

However, whilst these concepts do attempt to move away from the traditional static Diffserv configurations, a number of limiting factors can be identified. The first limiting factor is that both proposals require a degree of user interaction, which on one hand does allow for a more user-centric approach to be adopted but also may be intrusive for the user while potentially introducing an element of confusion for inexperienced users if asked to configure their QoS settings. Aside from requiring explicit user interaction both of these proposed frameworks consider Intra-domain networks, and do not consider the additional complexities of provisioning for QoS on the wider Internet where end-to-end control of the network is not possible. Furthermore, both proposals are realised in networks that are able to accommodate user traffic without considering the additional complexities of an over-subscribed network or a scenario when the requested level of service could not be guaranteed by the network.

A recent proposal (Cruvinel and Vazao 2011) describes an adaptive Diffserv architecture, in which core Diffserv routers periodically inform edge Diffserv routers of their queues utilisation. If congestion is detected by an edge router, and it [the edge router] is responsible for forwarding a large volume of traffic towards the core then the edge router may adjust its Diffserv policies to reduce the allocation of resources to non-priority traffic. Conversely, once the periodic reports from the core indicate utilisation levels have subsided, the edge router may once again increase the resource allocation for non-priority traffic. Although this research advocates a more dynamic Diffserv configuration, it does so at an aggregate coarse granularity, and with no regard for user fairness.

2.3.4. Multi-Protocol Label Switching & Diffserv Integration

While not strictly a network layer technology, the Multi-Protocol Label Switching (MPLS) standard, a label based core-network switching technology boasts QoS and traffic engineering capabilities. The marking of IP packets performed by Diffserv edge node (using the DSCP field) can be upheld within an MPLS core, as defined by (Heinanen J., Le Faucheur et al. 2002). Two proposals are described to aggregate the marked Diffserv packets into appropriate MPLS tunnels. Firstly, Label inferred LSP (L-LSP) suggests a predefined 1:1 mapping from DSCP values to MPLS labels. The label value is the only factor in defining the scheduling class for the packet when aggregated into a MPLS tunnel. The use of L-LSP theoretically allows for 64 different L-LSPs to be defined (bound by the DSCP field size), although the likelihood of having 64 distinct QoS classes is doubtful. An alternative approach of mapping Diffserv QoS provisioning into MPLS LSPs makes use of the experimentation (EXP) field within the MPLS header. EXP-Inferred-PSC LSPs (E-LSPs) provides eight distinct labels to mark packets according to the DSCP value. The EXP value is used by transit nodes to determine the per-hop-behaviour of packets within the LSP. The latter alternative has in recent years proved to be the widely implemented option, offering the ability for devices at the edge of an MPLS network to map the DSCP value of incoming IP packets into one of eight EXP values, for use over the MPLS core. This coarse granularity of QoS across the core of a network further advocates the practise of class-based QoS, and promotes a move away from a user-centric approach.

2.3.5. QoS Routing

In addition to the above solutions that provide QoS to IP networks, a smaller number of alternative efforts have been made to enhance the routing of IP packets. One such method proposed is to include QoS constraints within the parameters used during routing decisions (Crawley, Nair et al. 1998). Extensions to the interior routing protocol OSPF were suggested which facilitated QoS-informed routing decisions, (Apostolopoulos, Williams et al. 1999). For example, if an application has a low delay budget, then a router aware of the delay on each connected link could make an informed decision on the path packets should be sent along. However, the scalability of monitoring path properties such as bandwidth, delay and utilisation coupled with propagating these properties amongst neighbouring routers has limited such proposals from achieving widespread success. This method of QoS provisioning is further restricted when considered for an ISP network, where much of the traffic will be requested from devices external to the ISP's Autonomous System, and therefore from locations which are unlikely to be operating within the same IGP domain, thus making IGP cost parameters unavailable for external traffic sources.

2.3.6. Traffic shaping

Due to the complexity of Internet peering arrangements many Internet Service Providers (ISPs) choose not to implement dedicated QoS infrastructures on their networks such as Intserv or Diffserv. Instead, they opt to implement traffic management techniques that control the traffic entering their network. Based on summaries of UK ISPs (Kitz.co.uk 2009; Vuze 2010), it is apparent that many implement a form of traffic shaping to control the traffic on their network. For those offering "Quality of Service" or prioritised services a form of traffic shaping is performed using Deep Packet Inspection (DPI) coupled with class-based forwarding. Although DPI operates beyond the network layer (typically focussing at the transport and application layer parameters) the device and data forwarding are performed at the network, hence the inclusion of such techniques within this section of the thesis.

These traffic management techniques aim to alleviate many of the concerns raised by unresponsive traffic (specifically P2P), enabling ISPs to gain a level of control over the traffic on their network. For example, BT (the ISP with the largest subscriber size in the UK (ISPReview 2010)) admit to controlling the usage of P2P traffic at their discretion, depending on network conditions (information retrieved from "BT Total Broadband Fair Usage Policy" (BT 2012)). BT are not alone in taking this view of P2P traffic, six of the UK's top ten ISPs (as listed in (ISPReview 2010)) are reported to implement a form of traffic shaping against P2P protocols, according to information from on the support pages of the most popular P2P client (Schulze and Mochalski 2009) Vuze (Vuze 2010). It is also common practice for ISPs to operate a *fair usage policy* which specifies a limit to the bandwidth a user can consume within a given time period (through either uploading or downloading), upon reaching this limit the user's connection is restricted to a fraction of its original capacity, affecting all of a user's subsequent activities, not just P2P.

2.3.7. Summary of QoS at the Network Layer

This section has reviewed the state of the art for Quality of Service provisioning at the network layer. Many mechanisms and approaches have been proposed over the past decade, all striving to improve upon the best-effort delivery of IP. This section focused heavily on the Differentiated Services architecture, which despite more than ten since it was first proposed still remains the widely accepted approach for QoS provisioning and subject of current research. Diffserv originally offered a scalable solution where Intserv had failed, however, as highlighted in this section many issues still exist.

Firstly, there exists the Diffserv model of prioritising traffic, which may be acceptable in managed environments such as Enterprise networks, but the rigid prioritisation of a single selected traffic types is a philosophy that contradicts that of this research project (See section 1.1 for the project aims). Aside from these philosophical disagreements, a large amount of research has considered how appropriate the Assured Forwarding PHB is for TCP traffic, given the complexity of TCP congestion avoidance and the relative simplicity of marking algorithms. This has led to attempts to integrate TCP models into the AF marking algorithm however, these proposals were often limited in considering either only the vulnerability of short flows or longer, steady-state flows, rarely addressing the needs of a realistic mix of TCP flows as likely to be found on the Internet. Moreover, the enhancements to the AF PHB frequently consider the question of fairness between flows and aggregates, but rarely consider provisioning resources from the perspective of application-layer QoS. In addition to improving the marking algorithms of Diffserv's AF PHB, a review was also given for a small number of dynamic Diffserv proposals, which aim to move away from the traditional static configuration. The limitations of these dynamic proposals provided insight when designing the specification for the novel architecture later in the thesis (section 4.3).

A common observation of all the proposed enhancements to Diffserv was an agreement that the coarse, purely-aggregate based handling of Diffserv cannot provide an optimal QoS for every traffic type, suggesting a compromise needs to exist between the granularity traffic is managed with and the simplicity of QoS solutions. It is also recognised that technological advances, such as routing in hardware and Deep Packet Inspection allow for more complexity than was feasible during the early designs of Diffserv, and therefore possibly present new opportunities for novel QoS solutions, which are considered during the design of the proposed solution in section 4.3.

While Intserv and Diffserv were described as methods of providing QoS within a managed environment, whether it is a single Autonomous System or a number of cooperating Systems, they fail to extend to the global Internet model (i.e. have the capability to extend beyond a single Autonomous System). Additional QoS solutions operating at the network layer were also reviewed in this section, highlighting the practise of traffic shaping in ISP networks. These non-end-to-end alternatives typically prioritise traffic in accordance with the business model of the ISP, favouring premium traffic types rather than aiming to provide an optimal user-experience. Some ISPs market this solution as an Olympic colouring system to traffic, where user traffic is classified as Titanium, Gold, Silver, Bronze and Best-effort. An interesting point was noted regarding the 'top' package offered by such a model, which promotes all of a user's traffic to the Titanium or Gold service class. This places P2P in the same classes as Web Browsing, Email and Streaming Video services, which due to the aggressive nature of P2P services could in effect degrade the quality of the other services in use by the user since they would be competing within the same traffic class. Furthermore, it is not stated whether this traffic classification model places all traffic of a specific colour into a single class, as this would allow a premium customer's "Gold P2P" to infringe on other user's Gold traffic.

2.4. QoS at the Transport Layer – Congestion control and Fairness

The previous section considered approaches towards QoS provisioning that are located at the network layer. It was highlighted that there is a lack of synergy between how traffic is handled within the network layer and how the transport layer responds to this handling, specifically referring to UDP traffic not responding to network congestion and TCP operating at a different granularity (per RTT) to the (per packet) handling at the network layer. This section of the chapter reviews recently proposed developments at the transport layer, which reconsider the functionality of transport layer protocols to aid QoS provisioning and introduce congestion controlled UDP-like protocols.

2.4.1. UDP and Unreliable Congestion Controlled Protocols

Section 2.1 introduced the rising number of services that are migrating towards IP networks, the growth in volume of real-time (such as VoIP and VTC) and streaming traffic (IPTV) has in turn resulted in a growth application data using the User Datagram Protocol (UDP), the small overhead combined with the fact that real-time applications ordinarily do not require the reliability (there is little value in recovering lost voice packets) favours UDP over TCP for real-time services. Recent studies conducted on Internet backbones observed more than 3 times as many UDP flows as TCP (Min, Dusi et al. 2009), illustrating a dramatic increase since 2002 and 2006, when UDP/TCP ratios were 0.11:1 and 1.06:1 respectively. This notable change in Internet traffic characteristics has raised concern over the fairness and stability of the Internet. As a result there have been a number of proposals that aim to provide congestion control to unreliable transport protocols.

In response to this rise in UDP traffic and the associated fairness issues (section 2.2.2.1), a number of *TCP-friendly* protocols have been developed over recent years, where a protocol is considered *TCP-Friendly* if its long-term arrival rate does not exceed that of a conforming TCP flow under similar network conditions (Floyd S. and Fall K. 1999). TCP-Real (Zhang and Tsaoussidis 2001) for example, extends the acknowledgement based binary feedback of standard TCP and estimates contention within the bottleneck link of a path by computing the receiving rate of back-to-back packets to measure network conditions. The resulting estimate is fed back to the sender as an attachment to the acknowledgements to inform the congestion control process, a larger inter-packet arrival gap indicates congestion within the network and instructs the source to reduce the transmission rate. A similar functionality is provided by TCP-Friendly protocols TCP-Westwood (Mascolo, Casetti et al. 2001) and TCP-Westwood+ (Mascolo, Grieco et al. 2004), which continuously measure the rate of incoming ACKs to measure and estimate the rate of a connection. TCP-Friendly Rate Control (TFRC) (Handley, Floyd et al. 2003), presents yet another alternative that adjusts the sending rate of data according to the level of congestion, inferred from the loss rate. Originally designed for applications sending packets of fixed size, a further enhancement to TFRC, TFRC-SP (TFRC small-packet) (Floyd S. and Kohler E. 2006) makes TFRC-SP a suitable transport protocol for future multimedia applications such as VoIP.

Whilst implementing congestion control in TCP-like protocols assists in congestion control and makes applications TCP-friendly, a compromise must be made between how responsive the congestion control algorithm is (in terms of how rapidly the application decreases its transmission rate) and how smooth the transmission rate is. The harsh sawtooth-like effect of the additive increase/multiplicative decrease (AIMD) algorithm can result in severe oscillations in transmission rates and in worst case scenarios transmission gaps, none of which are desirable for the close to constant bit rate characteristics of multimedia applications. As described by (Tsaoussidis and Zhang 2005) according to AIMD, a sender increases the congestion window (*cwnd*) by α packets each Round Trip Time (RTT) – given no packet loss, until the network capacity is reached. Following capacity being reached, a multiplicative decrease ratio is applied to prevent congestion collapse of the network decreasing the congestion window by βW. Tsaoussidis and Zhang document that within TCP-Friendly protocols the multiplicative decrease ratio β is increased to soften the sawtooth oscillations of traditional TCP's response to congestion. However, as investigated by (Chiu and Jain 1989; Lahanas and Tsaoussidis 2003) there is a fundamental trade-off between responsiveness (low α) and smoothness (high β), with different application types demanding different configurations of α and β.

Another congestion controlled unreliable protocol that has been proposed recently is the Datagram Congestion Control Protocol (DCCP) (Kohler, Handley et al. 2006). DCCP provides application flexibility in terms of the congestion control algorithm it can adopt. Using Congestion Control IDs (CCIDs) one of two integrated congestion control algorithms can be chosen. CCID2 provides a TCP-like AIMD congestion control algorithm that adapts quickly to make use of available bandwidth. Conversely, CCID3 selects TFRC congestion control, providing a smoother throughput adjustment at the cost of responsiveness.

The development of DCCP for real-time congestion aware transport functions has naturally led researchers to investigate the performance of applications such as VoIP when using DCCP compared with the traditional UDP approach. (Balan, Eggert et al. 2007) describes how when implementing VoIP over DCCP quality is less than VoIP over UDP, in light of this the authors propose a number of improvements to the DCCP specification that may alleviate this disappointing performance. One must keep in mind that DCCP was not developed for optimal VoIP call quality, a protocol that offers no transfer rate adaptation during congestion (i.e. UDP) is likely to provide a better perceived quality than one that adjusts to operate fairly.

2.4.2. QoS Enhancements to TCP-based protocols

Limited research has been conducted with regard to modifying the behaviour of TCP to introduce an awareness of QoS mechanisms within the network. This lack of research is likely to be due to there not being a single adopted solution to QoS at the network layer, thus making any such enhancements at the transport layer very specific to one particular QoS implementation at the network layer.

One example of a proposed TCP-like protocol is given by (Jourjon, Lochin et al. 2007), who describe the QoS-aware Transport Protoocl (QSTP). A variation of the TCP Friendly Rate Control (TFRC) algorithm is presented, which uses a selective acknowledgement mechanism to provide reliability. In addition to adding reliability the Jourjon et al. integrates a target rate parameter into the algorithm used by the sender when determining the transmission rate of the sender, resulting in their Guaranteed TCP Friendly Rate Control (gTFRC) congestion control algorithm. gTFRC is then demonstrated to address the problem of a congestion controlled host reducing its throughput below its target rate following congestion in an AF network, as introduced in section 2.3.3.1. The problem was described as being caused by a lack of awareness by the transport protocol that the dropped packets were indeed out-of-profile packets, exploiting excess available bandwidth. The gTFRC protocol proposes that through making the sender aware of the target rate, the transmission rate can be controlled to always be greater than the target rate.

This modification goes some way towards bridging the gap between network layer events and the subsequent response at the transport layer, by placing a minimum transmission rate (linked to the desired target rate) within the congestion control algorithm. However, this proposal is limited in a number of ways. Firstly, the details of the target rate for each flow are determined between the sender and the receiver, and are not inferred from the configuration of the QoS architecture within the network. This assumes that the network is over-provisioned and typically has sufficient capacity to accommodate of the users' traffic, given that the target rates calculated by the sender are not determined by the load of the network. Secondly, the target rate is used to control the throughput of each (gTFRC) flow, which assumes each application using the new transport protocol would require knowledge of its own desired target rate.

2.4.3. Response to Changing Traffic Dynamics

A limited number of proposals have been made to improve on the unfairness of applications opening and maintaining multiple TCP connections. Two examples are the congestion manager (Balakrishnan and Seshan 2001) and Weighted TCP (Ou. G. 2008), which essentially both present similar methods to aggregate TCP connections at the end host into per-application connections. For example, applications such as FTP or VoIP clients that only establish a single connection would operate as usual. However, applications such as P2P that establish multiple connections would have their connections placed into a virtual wrapper, which restricts the combined throughput to be closer to that of single connection-based applications.

One drawback severely limits the likely success of these approaches, that is, these proposals are modifications to the operating system of an end-host, and there simply is no incentive for a user of an application that establishes multiple connections (e.g. P2P clients) to update their operating system to employ a restriction on the throughput of their applications. From the perspective of the service provider, such solutions would provide a fairer distribution of bandwidth among flows within their network, potentially increasing the speed for non-P2P users.

2.4.4. Summary of Transport layer QoS mechanisms

This section of the chapter has introduced a number of enhancements to transport layer protocols. The majority of the novel protocols introduced in this section have provided alternatives to UDP, which integrate congestion control mechanisms for services not requiring the reliability or complexity that TCP offers. The Datagram Congestion Control Protocol (DCCP) is notably the most promising solution to alleviating concerns of rising levels of unresponsive UDP traffic, providing flexibility to application developers with regard to the congestion control algorithm that best suits their application. However early experimentation with the protocol demonstrates a lower performance index for VoIP services compared with UDP, and therefore further development is clearly required before DCCP will become a viable choice for the transport of real-time data.

In addition to enhanced transport layer functionality for unreliable protocols, this section described an attempt to integrate the target rate parameter from a Diffserv AF policy into the transmission rate equation of a TFRC sender. While this approach enables a sender to guarantee a transmission rate that is at least equal to a network defined SLA, the proposal operates on a per-flow basis and assumes the same per-flow configuration within the network. This renders the approach rather static in its design, given that the application behaviour at the end host must reflect the configuration within the network.

This review of recent developments at the transport layer has highlighted the realisation that congestion control is essential to network stability. Significant efforts have been made to present alternatives to using UDP for transport functionality, motivated by the advent of real-time services such as VoIP and VoD. However, all of the proposed solutions require modification to the end host, which given the number of end hosts currently estimated to be connected to the Internet considerably impedes the future deployment of such proposals. Moreover, to date the performance of these UDP alternatives has been demonstrated to be unable to equal or better that of UDP, further limiting their uptake by application developers.

2.5. QoS Provisioning at the Application Layer

Due to the lack of QoS mechanisms having been implemented on the Internet, applications have been developed to accept this best-effort nature and directed many to operate adaptively according to network conditions. This section of the review takes VoIP as an example to presents a number of application layer mechanisms that can be implemented to provide improved QoS to the end-user irrespective of an absence of QoS provisioning at lower layers.

2.5.1. Real-time Transport Protocol

Among the current standards for delivering audio and video over the Internet is through the use of the Real-time Transport Protocol (RTP) (Schulzrinne, Casner et al. 2003). RTP together with the Session Initiation Protocol (SIP) are reported to account for over 60% of all VoIP traffic on the Internet in all areas of the world aside from Eastern Europe and the Middle East, where Skype dominates, possibly due to low bandwidth connections (Schulze and Mochalski 2009). Although RTP is strictly defined as a transport layer protocol, nearly all implementations have been at the application layer, and therefore this review falls under the section detailing application layer mechanisms for QoS provisioning.

RTP operates on top of UDP and provides an application layer framework for delivering audio and video across an IP network. This framework offers a number of features that enable the protocol to deliver time-sensitive services over the best-effort Internet, while maintaining an acceptable level of QoS. These features include:

- Multicast and Unicast delivery of real-time application data
- Timing recovery between separate audio and video streams (synchronisation)
- Loss detection and correction
- Reception quality management
- Audio/video session management, e.g. management of participants

RTP is usually implemented alongside the Real-Time Control Protocol (RTCP), which is responsible for managing periodic out-of-band control messages, used by the RTP framework to manage the media transmission and for QoS purposes. Of the five types of RTCP message, three provide QoS information, they are:

- Sender Report (SR): Sent periodically from active senders containing absolute timestamps used to synchronise separate audio and video streams at the receiver.
- Receiver Report (RR): Periodic reports from non sending participants reporting on the received QoS of the RTP data, including statistics on the *cumulative number of packets lost,*the *loss fraction* and *inter-arrival jitter.*
- Source Description (SDES): Descriptive information about the data source (name, email address – not relevant within the context of QoS provisioning)

The original RTP/RTCP specification included capability to exchange QoS information between participants; it did not however include any recommendations regarding congestion control. A revised document (Schulzrinne and Casner 2003) advises that RTP receivers should monitor the level of packet loss, ensuring that packet loss is at an acceptable level. The level of packet loss is considered acceptable if a TCP flow across the same path and experiencing the same network conditions would achieve an average throughput not less than the RTP flow achieves. It is suggested that RTP receivers implement congestion control to ensure that transmission rates can be adapted to satisfy this throughput fairness during congestion. However, this is only a suggestion that RTP receivers *should* adhere to, and not a requirement that they *must* meet. Moreover, there is no relationship between how RTP handles the QoS and congestion control information provided by RTCP, meaning RTP may adapt a flow to ensure it behaves in a TCP-friendly manner but at the detriment of the QoS.

2.5.2. Packet Loss Concealment (PLC)

Packet Loss Concealment (PLC) is one technique for reducing the impact of packet loss on VoIP services. The latest codec to be added to the library of codecs used by popular VoIP client Skype, SILK (Vos, Jensen et al. 2010) describes its use of Forward Error Correction (FEC) to protect against packet loss. The SILK codec claims to detect onsets and transients within the speech, to perceptually determine important speech information. These important segments of speech information are encoded more than once and appended to subsequent packets, ensuring that if packet loss occurs there remains a high probability the important speech segments will still be delivered. FEC is only one method to conceal packet loss, a comprehensive survey of PLC techniques is given in (Perkins, Hodson et al. 1998) and are categorised into sender or receiver oriented. Sender oriented techniques include, Forward Error Correction (FEC), interleaving (re-ordering data units to reduce the impact of packet loss) and retransmission. Receiver oriented techniques include, packet repetition, silence or noise substitution, interpolating surrounding packets to cover loss, or regenerating lost audio using model-based recovery. The specific operations of these techniques are beyond the scope of this research, but can be found in (Perkins, Hodson et al. 1998) and cited materials.

2.5.3. Codec adaptation

A second mechanism that can be used to adapt voice data according to the network conditions is through the use of adaptive codecs. Applications can also use feedback information about the state of the network from control protocols; for example an RTP/RTCP VoIP application may use the feedback from the Real-Time Control Protocol (RTCP) to adjust the codec in use and better suit the network conditions. Experimental results performed in more recent studies provided evidence that adaptive VoIP applications offer a higher call quality and a reduction in packet loss as a result of codec switching, (Ng, Hoh et al. 2005; Usman and Sheikh 2005). It should be noted that codec adaptation is not a standard function of RTP/RTCP, and these works make informed decision of the optimal codec given network conditions described from these control messages. Again taking Skype as an example, a number of codecs are available to the application. Based upon the current network conditions Skype will choose the codec it considers the most appropriate. The SILK codec for instance supports four different modes, each of which provide a different audio sampling rate, frame size and bitrate, thus allowing Skype to evaluate the network conditions and choose appropriately.

2.6. Summarising the State-of-the-art for QoS provisioning

This chapter has provided an evaluation and critical assessment for the state of the art for QoS provisioning over IP networks. It began by defining what QoS is likely to be for a range of common IP services, both from the user and protocol perspective, discussing the impact network imperfections can have on service and protocol performance. Insight was also given into the changing nature of traffic dynamics, in particular the growth of P2P-based applications. This rise in bandwidth intensive applications, such as P2P, raises the issue of fairness between applications, when previously TCP-based applications could have been thought to be relatively fair to one another. These definitions clarified that there is a growing need for QoS provisioning in IP-based networks, but also a consideration for fairness at multiple levels, per-flow, per-application and per-user, given the ever increasing heterogeneity of services.

The chapter then focussed on a review of QoS provisioning efforts at the network layer, beginning with a study into how alternatives to taildrop queuing can be used to alleviate congestion. The negative effects of taildropping a large number of packets were identified, highlighting the benefits of preemptive mechanisms such as RED and ECN. Following on from queuing mechanisms was a review of Integrated Services, which although capable of providing end-to-end service guarantees, had two major shortcomings (scalability and a requirement for Intserv support end-to-end), which limited the deployment of Intserv to only small scale environments. The main focus of the research at the network layer expanded on Diffserv, the typical current method employed for QoS provisioning. Prior studies have shown Diffserv works well operating at an aggregate level for managed Enterprise environments, which have predictable traffic characteristics. Subsequently, this allows for QoS modelling and capacity planning to reflect the known services on the network. However, providing QoS over the ungoverned Internet is far more challenging, requiring ISP networks to provision for a far more diverse array of services and user expectations, and it is argued that no single service should be given explicit priority over others. Furthermore, although successful at an aggregate level, section 2.3.3.1 expanded on efforts which have aimed to improve Diffserv's ability to accurately manage traffic at a flow level. Also included in the network layer review were the results of an investigation into current ISP QoS practises. Although information surrounding ISP QoS policies and network configurations were understandably limited, it was ascertained that the common approach of managing user traffic is some form of traffic shaping at the edge, with a coarse Diffserv configuration across the core. The traffic policies used by ISPs are reported to give VoIP and Video services scheduling precedence over non favoured *elastic* services such as P2P, FTP and messaging (Kitz.co.uk 2009), illustrating a very different reality to the ideal scenario.

The chapter then focussed on the relationship between QoS and congestion control at the transport layer. The main focus of developments located at the transport layer has been to address the impact that rising volumes of unresponsive (UDP) cross traffic has on congestion controlled services. In response to this trend, a number of congestion controlled UDP alternatives have been proposed for real-time traffic. However, the main conclusions were that the proposals to date result in a reduction of service performance when compared with UDP, and therefore it is unlikely that application developers (or users) will opt to implement these transport protocols at the cost of application performance. Moreover, QoS provisioning efforts that require global protocol deployment to the estimated 2.1 Billion end hosts presents the equally taxing task of deployment.

A brief discussion of application layer mechanisms that optimise the performance of an application given the network conditions was also given. This section was provided primarily as a reference and to introduce the notion that without implementing QoS provisioning within the network, mechanisms at higher layers can only go so far towards absorbing the impact of loss and delay on Internet services.

In summary, it is believed fair to say that while many efforts have been made towards improving QoS for specific traffic types over IP networks, these solutions address individual services or scenarios. To date, none have been found that consider the task of QoS provisioning from the perspective of managing the entire multimedia user experience in a dynamic manner that reflects the current services that are in use.

Drawing on the above sections, a number of key attributes for a novel QoS provisioning approach can be established.

- Static AF configurations are not suitable for guaranteeing unknown activities
- Assured Forwarding for Diffserv can provide a guaranteed level of service to TCP, which is suitable for bulk transfers and continuous multimedia streaming in managed networks.
- TCP is unaware of whether a packet loss is *in* or *out* of profile.
- Blanket policies against P2P and large file downloads are not fair solutions
- All methods of controlling TCP operate at a per packet granularity, when TCP responds at a per-RTT granularity.
- P2P traffic is reportedly controlled through the use of Traffic Shapers, restricting the bandwidth available for such services.
- Priority services should not be exempt from congestion control.

3. An Empirical Study into Internet Services

3.1. Introduction

The previous chapter introduced the increasing need for QoS provisioning within IP networks, given the growing diversity of Internet services. To gain a better understanding of this diversity, this chapter provides an empirical study into a selection of current Internet services, which is later used when designing the novel architecture. A significant amount of research from the past decade has provided a number of mathematical models that describe Internet traffic, particularly at the aggregate level. Many have highlighted self-similarity and long-range dependence within traffic aggregates, as summarised in (Karagiannis, Molle et al. 2004), however, less research has considered the individual characteristics of multimedia Internet services, which is the perspective taken by this study. A combination of empirical data and related work is used to analyse and characterise each of the selected Internet services. This characterisation will later be used to aid and inform the novel QoS provisioning approach.

As described by the Ipoque study (Schulze and Mochalski 2009), Web browsing, file downloads (via HTTP/FTP), VoIP, Streaming video, gaming and P2P file transfers comprise over 90% of all Internet traffic. Of particular interest to this study were Web browsing, streaming video, P2P file transfers and VoIP. The analysis of traffic produced by online gaming was considered to be too complex to address at this time, given the large number of games across many different platforms. Therefore, it was not included as part of this study but is recommended to be the focus of future research.

The data was collected from four sources, Plymouth University and three residential locations each with a different ISP. Firstly a PC running Windows Vista was used to collect data on Web browsing and Streaming Video, but due to restrictions on the activities permitted on the University Network this PC could not be used to monitor P2P-based activities. Therefore, three further systems were used from residential locations to collect data for P2P-based services. The reason for collecting P2P data from three different locations was to ensure that any traffic shaping being performed on the P2P traffic by the ISPs could be identified (for example if one collection of data showed significant poorer performance than the others). Furthermore, two of the connections were ADSL and the third a cable connection. This study analysed 500 of the most popular webpages, the top 100 YouTube videos, a selection of on-demand programmes, 400MB of Skype traffic, traffic from 7 days of Spotify activity, more than 20GB of data via the BitTorrent protocol and a number of VoIP calls using four of the most popular free VoIP clients. It is fair to say that repeating this study at a larger scale would indeed provide a more comprehensive insight into the behaviour of Internet services. However, since the primary focus of this project is on QoS and not the characterisation of Internet services, the study is considered sufficient to obtain an up-to date overview of the behaviour of Internet services.

3.2. Web Browsing

Web browsing is the second largest contributor to Internet traffic, with HTTP being the dominant protocol in terms of usage on the Internet. The Ipoque study states that the average size of a website has continued to grow, no doubt supported by increasing Internet connection speeds. However, Ipoque merely state in their 2009 report that news portal websites measure approximately between 1-2MB in size, although go into no further detail of *the average webpage*. In fact there is very little academic research from recent years that has considered how the average webpage looks like, in terms of size (bytes), number of web objects (example objects being an HTML file, a styling file, an image, etc) or the behaviour of the protocols.

The first factor that could influence the behaviour of a web browsing session is whether version 1.0 or 1.1 of the Hyper Text Transfer Protocol (HTTP) is implemented at the web server. From a traffic analysis perspective the key differences between these two versions is the *persistent connection* feature in the latter. This feature allows a client to make multiple requests to a web server without the need to establish a TCP connection for each request. This dramatically changes the potential characteristics of the traffic generated from web browsing, given that the connection between the client and the server will serve all of the objects on a webpage (assuming all objects are located on the same server), hence the size and duration of a flow at the IP level now is dependent on the number of objects requested from the server and the size of each of them. Analysing each of the 500 most popular webpages (Alexa.com 2010), over 26,000 HTTP GET requests were made to web servers, of which 95% replied using HTTP 1.1. Identifying this instantly revealed a potential discrepancy in the well used assumption that HTTP traffic is predominantly characterised as short flows of less than 13kB, if all of the GET requests/responses for a webpage are carried through a single connection. The following analysis was aimed at verifying this hypothesis by establishing the average number of objects and the combined average size of a webpage.

The most recently known statistics that describe the average webpage are from 2008 (WebsiteOptimization.com), which present a comparison between findings from a 2007 paper by Domenech et al. and figures obtained from Gomez.com – a division of Compuware, who focus on web performance and experience management. WebsiteOptimization.com reports the average size of a webpage to have increased between 2003 and 2008, from 93.7kB to 312kB respectively. They also comment on the number of objects per page to have increased from 25.7 to 49.9 over the same time period. Using a list of the then top 100 visited URLs (Alexa.com 2010) and HTTP Analyzer v5 (IEInspector Software LLC 2010), a batch program was produced to visit each of the top 100 webpages in turn and obtain the average size and number of web objects of a webpage in 2010.

Based on these top 100 visited webpages within the UK it was observed that over 90% were below 500kB in size, with the remaining 10% ranging between 500kB and 1300kB. The mean of the 100 websites is 250kB, although this figure can be seen to have been skewed by a few large webpages when considering the mean for the top 95% of sites, which is 201kB. Figure 3.1 depicts this distribution of webpage sizes.

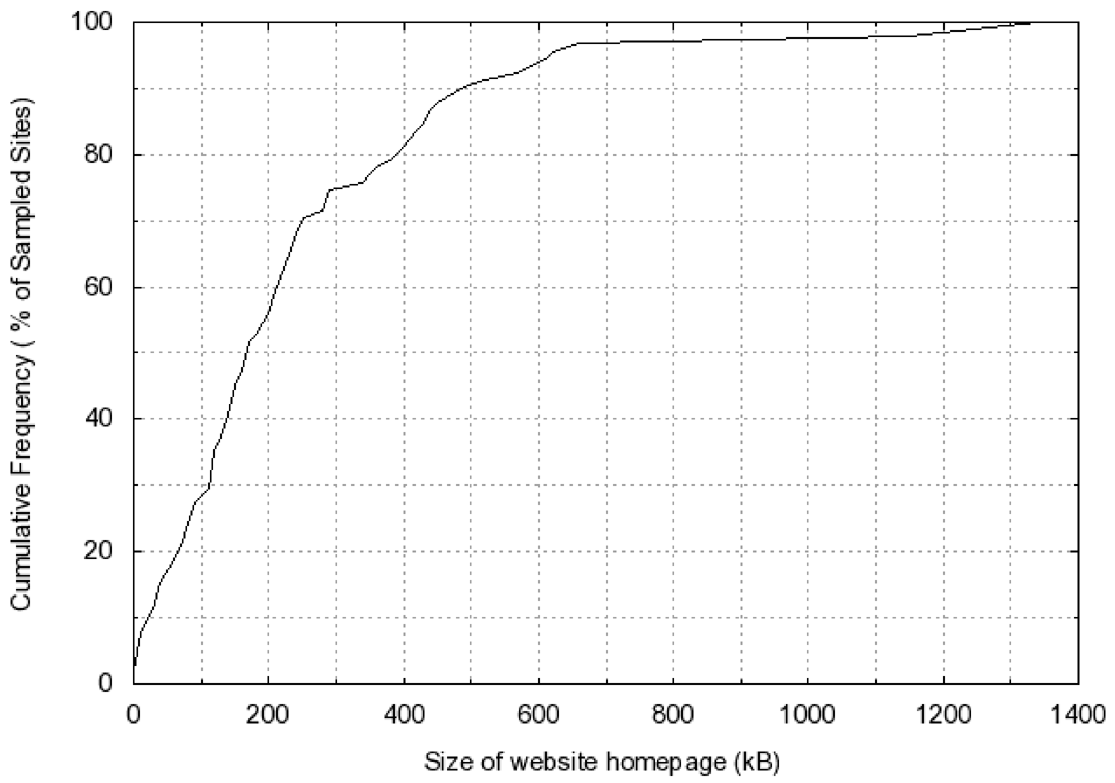


Figure 3.1: Cumulative Distribution Function for the size of the Top 100 UK websites

For the same top 100 webpages Figure 3.2 shows the distribution for the number of objects within each webpage, of which the mean number of objects per page is 60.

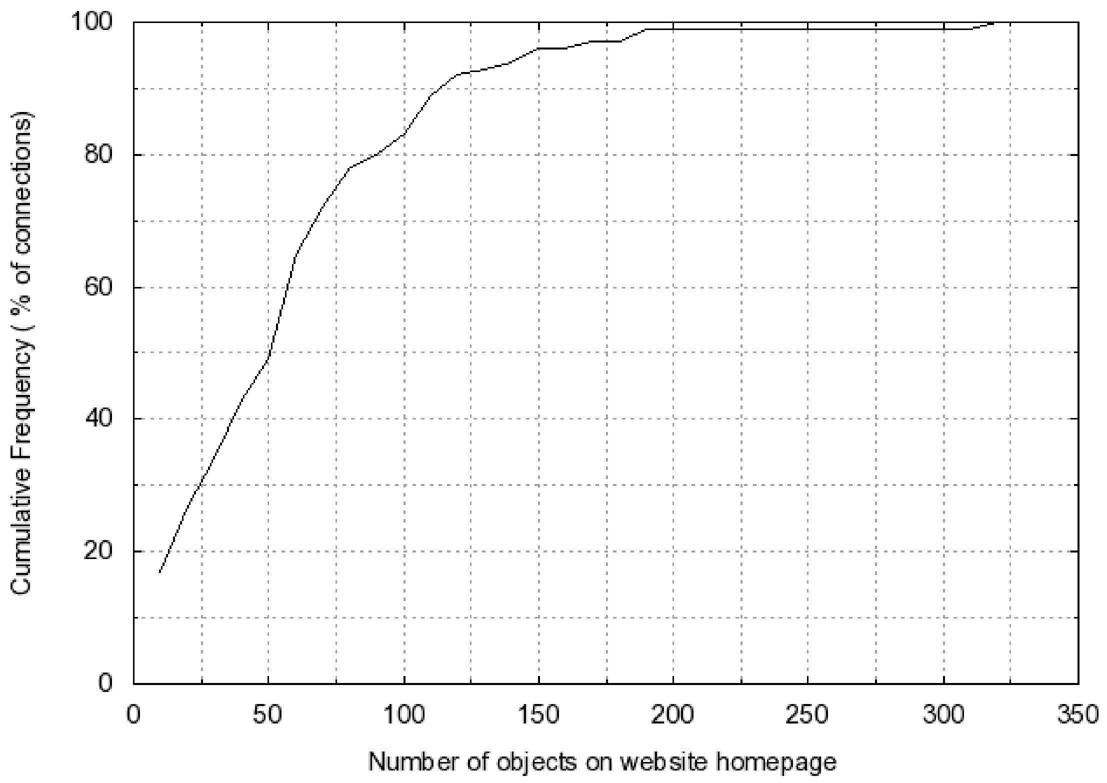


Figure 3.2: Cumulative Distribution Function for the number of objects within a web page

As introduced previously, if all of the data for a webpage is transferred over a single TCP connection then this study presents a significant difference between the widely used 13kB average webpage and a typical webpage visited today. Therefore, further analysis at the transport layer was also conducted, to reveal in particular the number of TCP connections that are established during the retrieval of the average webpage, in other words, does the prolific use of HTTP v1.1 mean that only a single connection is established, for which all data is sent along? TCPdump was used to capture the incoming traffic for each of the sampled webpages, and tcptrace (Ostermann 2003) to aggregate the data per TCP connection, Figure 3.3 illustrates that despite the high implementation of HTTP v1.1, data is not sent from a single source, using a single TCP connection. In fact, only one site from the top 100 (paypal.com) opened a single TCP connection to transmit its data, with the mean number of connections per webpage being 26 (Figure 3.3) with the number of GET requests per TCP connection ranging from 1-10, and a mean of 2.7.

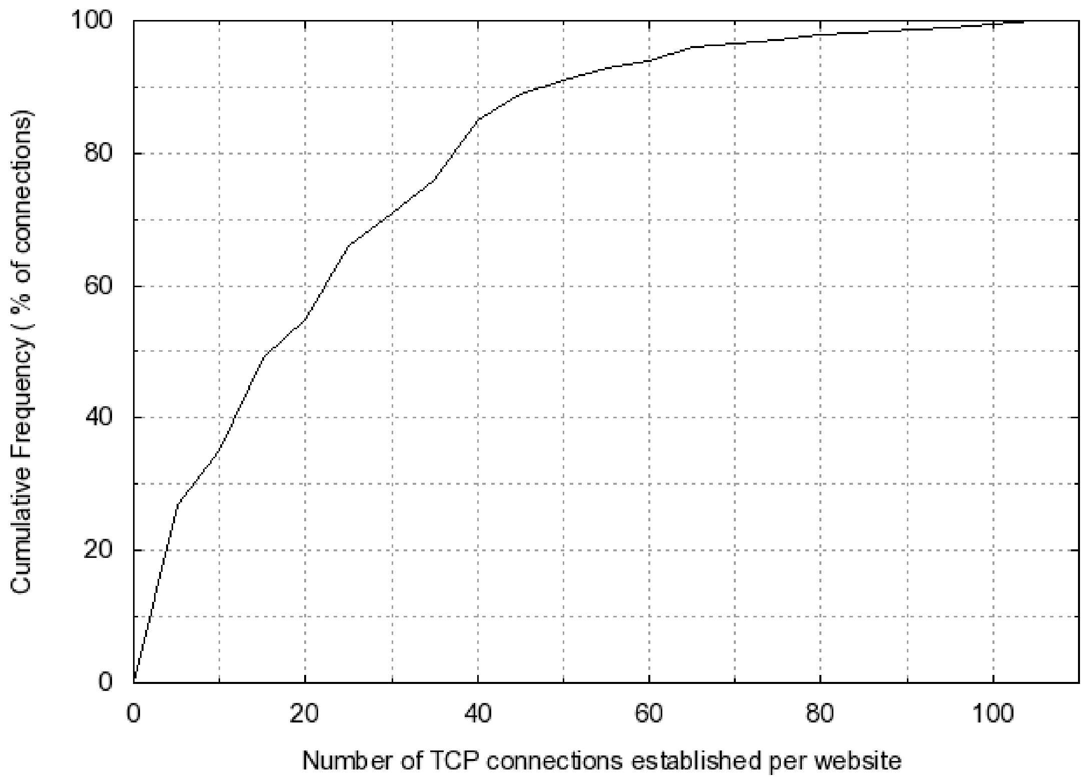


Figure 3.3: Cumulative Distribution Function for the number of TCP connections established per webpage

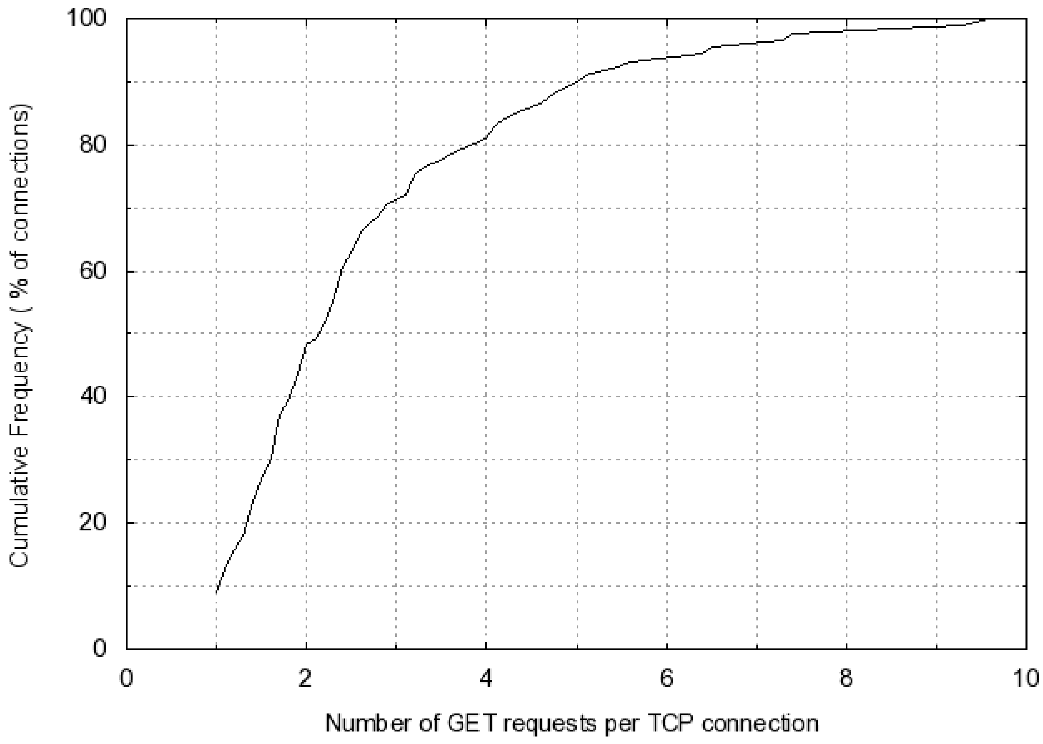


Figure 3.4: Cumulative Distribution Function for the number of GET requests per TCP connection

The study went on to consider the volume of data transferred by each HTTP-initiated TCP connection, illustrated by Figure 3.5. There is a difference of an order of magnitude between the range of volumes of data sent from the client to the server, and the amount of data uploaded. In the direction client to server, 95% of the connections are ≤ 6kB in size, with a mean volume of 1kB. In the reverse direction 95% of connections are ≤ 85kB, with a mean volume of 11kB, indicating that the average HTTP flow is still very close to the 13kB figure, despite the evolution of web content over recent years. It should be noted that this area of the study only considered opening the homepage for each of the websites involved.

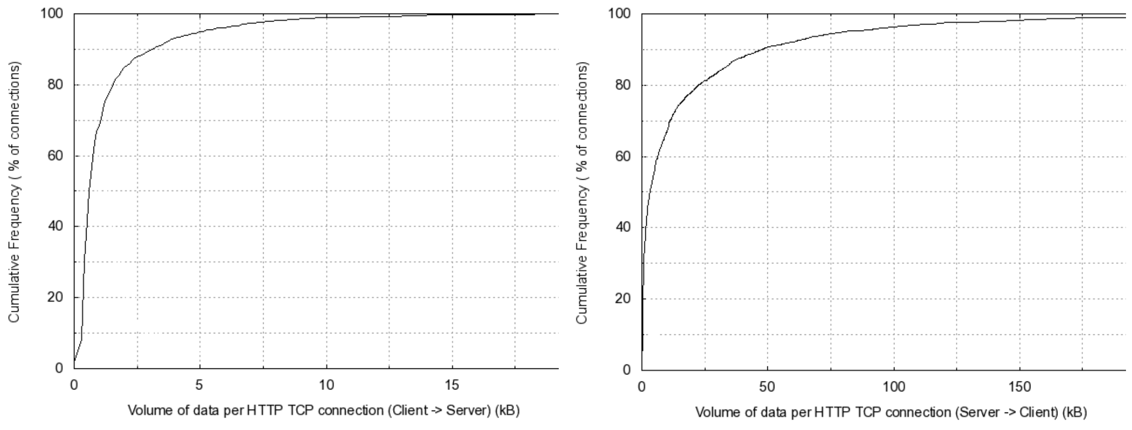


Figure 3.5: Cumulative Distribution Function for the volume of data per TCP connection

a). Client to Server **b). Server to Client**

The final characteristic of HTTP traffic that was considered was the number of packets per TCP connection. Figure 3.6 shows the distribution for the number of packets per connection for the observed websites. In both directions the number of packets transferred is very similar, with 95% of flows in the direction of Client to Server consisting of 50 or fewer packets, with a mean of 10.26. For flows in the opposite direction the number is slightly higher, at 70 or fewer packets with a mean of 11.79 for 95% of the flows.

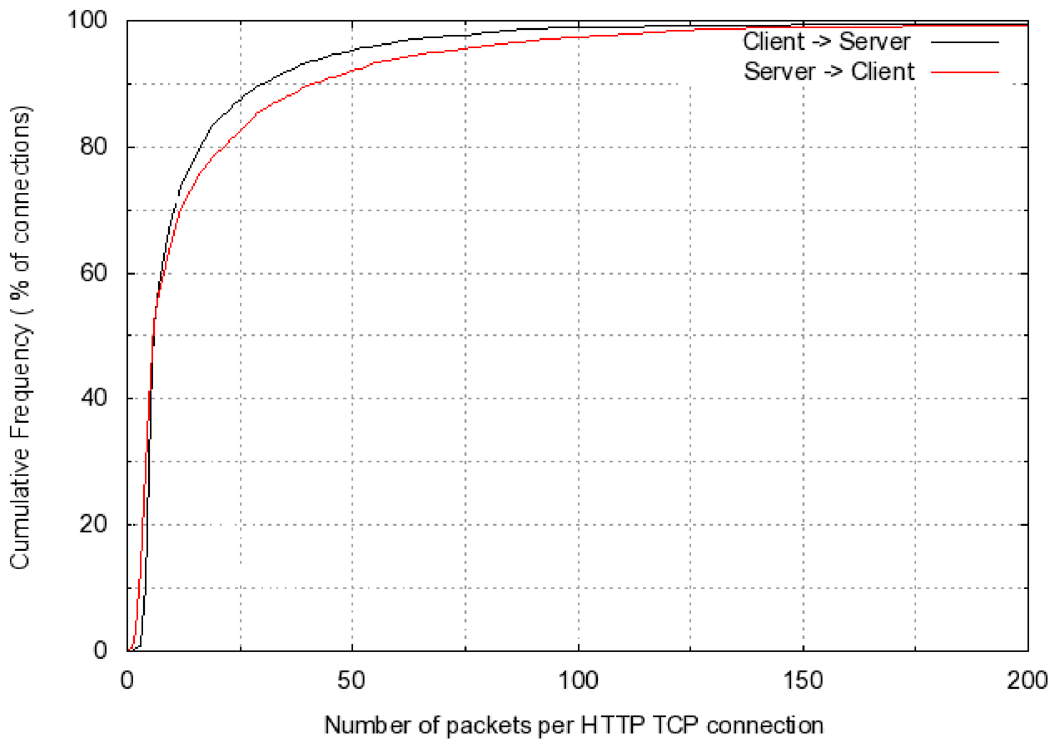


Figure 3.6: Cumulative Distribution Function for the Number of packets per TCP connection

This study into the size of the average webpage revealed that the size of the average webpage in 2010 has not changed dramatically since 2008. However, the results are limited to the homepage for each of the websites visited, if the nature of each of these websites is considered then the characteristics of web browsing change once again.

For example, 5 of the top 100 were search engines and 12 of the top 100 websites of 2010 were social networking sites. Both of these web site categories suggest that the user intends to perform a task by visiting the site, either through searching or logging in to a personal page. The web page analyser script was not programmed to provide any log in credentials or search criteria and therefore only recorded the size of the first page (typically a fast-loading page with a few graphics and a username and password form). However, it is highly unlikely that a user would visit Google.com or Facebook.com, and then not proceed to either search or log in and continue to browse through their online social network.

Taking Facebook as an example, the index page for Facebook.com is only 43kB in size and consists of 20 objects, but upon logging in an account the homepage is 185kB and consists of 57 objects. Based on viewing 50 random profile pages (from within a list of known friends), the average size of a profile page was 148kB, comprising of 31.4 objects. Visiting these 50 profile pages on Facebook a total of 10.1MB were transferred, via 1570 objects, and 256 TCP connections. Therefore, possible future research could consider how long a typical user spends on each website and how many pages from a site are subsequently visited.

3.3. Streaming media

The Ipoque study excludes streaming media websites from the category of web browsing, instead it is reported that streaming video (including Flash, Quicktime, Real Media, RTSP) represented on average7.65% of all Internet traffic in 2009, with Adobe's Flash platform contributing to between 60-83% of all streaming media in all regions, except the Middle East. As highlighted earlier in this thesis this figure has since grown substantially , with reports from 2011 indicating streaming video now represents as much as 40% of all Internet traffic (Sandvine 2011). While such traffic has been differentiated from regular web browsing, online video services are a rapidly growing sector and therefore cannot be ignored when characterising today's Internet traffic. Furthermore, video files are significantly larger in size than other types of web objects, and yet the user has the expectation the audio and/or video should play smoothly without interruption. Therefore, understanding the properties of streaming audio and video is essential when provisioning for QoS.

Without doubt, the most popular and well known online video repository is YouTube (YouTube 2010), which allows users to upload their own videos as well as view the videos uploaded by others. It was reported that in 2009 YouTube served over one billion video requests per day (Telegraph.co.uk 2009) and every minute 24 hours of new video is uploaded to YouTube (YouTube 2010), these figures have since grown significantly and now YouTube report themselves that 60 hours of video are uploaded to their servers every minute, and over 4 billion videos are viewed each day (YouTube 2012)

According to a study by (Gill, Arlitt et al. 2007), it is estimated that the average video size on YouTube is approximately 10MB, based on over 300,000 unique videos requested on a University Campus. The average duration of a video from YouTube was observed to be just over 4 minutes, and the bit-rate was estimated to be 394Kbps. A second study into YouTube suggests that despite the ever increasing number of videos on YouTube, the top 10% contribute toward towards over 80% of all views (Cha, Kwak et al. 2007). To obtain a perspective of current YouTube videos the traffic monitor requested and recorded statistics for 100 videos on YouTube, which were listed as being the most popular videos of all time by the site itself. Figure 3.7 shows the distribution of video file size for these files, where the median video size is 7MB and the mean size is 10.49MB. These figures support the findings of Gill's study, indicating little change in the characteristics of the most popular videos on YouTube between the two observations.

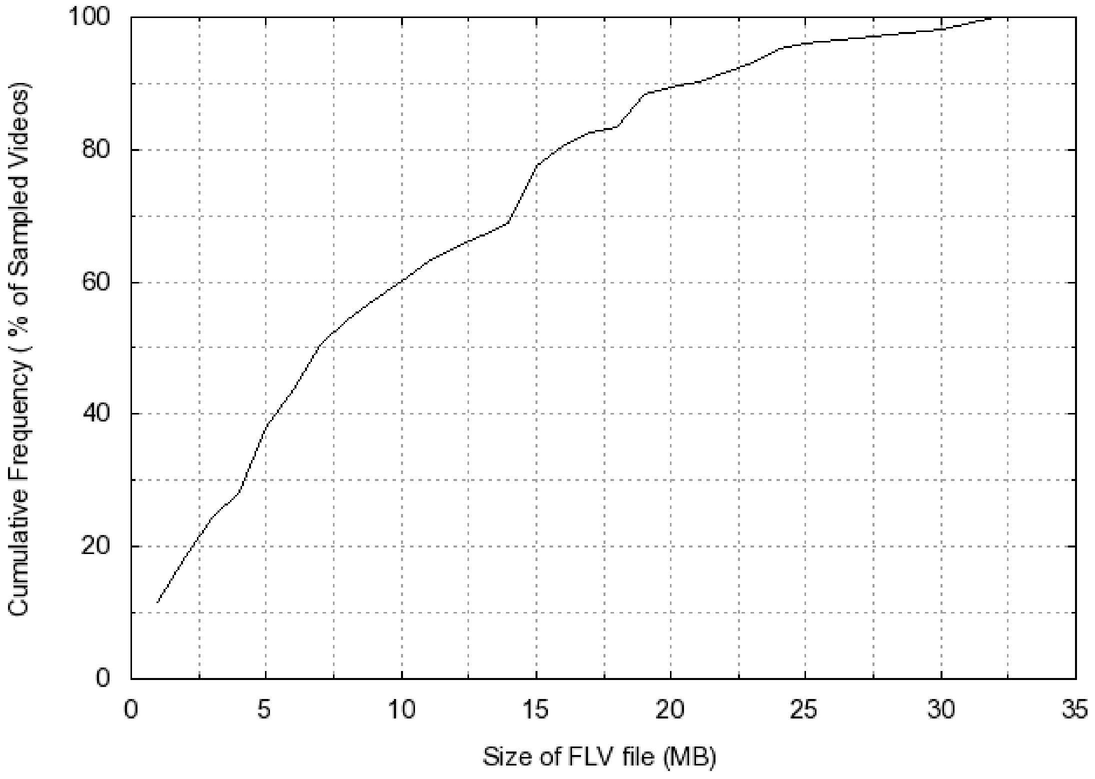


Figure 3.7: Distribution of video file size for 100 of YouTube's most popular videos

Gill et al. presented the average bitrate of a video on YouTube was 394kbps, it is hypothesised in this thesis that this figure is likely to be increasing. This hypothesis is based on recent additions to the supported bitrate YouTube allows users to upload at, coupled with the fact that higher resolution video recorders are now commonly integrated into mobile phone and media players. The video resolutions currently supported by YouTube along with their associated bitrates are provided in Table 3.1 below.

	Video Format				
	240p	**360p**	**480p**	**720p**	**1080p**
Approximate bitrate ranges (kbps)	200-300	300-600	600-1200	1100-1800	>1800

Table 3.1: YouTube video resolution and associated bitrates

Although the figure of 10MB for the average video size appears to have not changed over the last 3 years it should be noted that it represents videos added solely by individuals or promotional companies. Over the past 3 years television broadcasters have turned to the Internet for delivering on-demand content. The five major broadcasters in the UK (BBC, ITV, Channel 4, Five and Sky) all provide online video services. More recently Channel 4 and Five have partnered with YouTube to provide on-demand programs via the YouTube website, while the BBC, ITV and Sky currently use bespoke delivery systems, (although all are variants of Adobe's Flash Platform).

This advent of full length broadcasting over the Internet changes the nature of streaming video, whereas previously the average length of a YouTube video was just over 4 minutes, with broadcasters using the Internet to deliver programmes this figure is likely to rise as these services become more popular, as is apparent from the following details on the BBC iPlayer.

The BBC iPlayer has been publically accessible for the past 5 years and according to BBC statistics has seen more than a 100% increase in programme requests for the month of January since 2009 to 2010, going from 30.8 million TV requests to 67.4 million respectively (BBC-iStats 2010). This number increased to 116 million TV requests in January 2011 (BBC-iStats

2011), indicating a significant growth year-on-year. From a media perspective, the BBC state that their standard videos when viewed through an Internet browser are streamed at 800kbps, using the H.264 codec. The iPlayer distribution servers are reported to be capable of detecting if a user's connection can accommodate an 800kbps stream, if the user's connection cannot support 800kbps or in the presence of congestion, the service reduces the video quality to 500kbps – although it is stated that the video quality will not be increased should more bandwidth become available (this is listed as a future feature). A user also has the option to view the video in *High Quality*, upgrading the bit-rate to 1500kbps. Furthermore, both YouTube and BBC's iPlayer have recently began providing videos in *High Definition*, at resolutions of 1280x720, and at bit-rates of 3.2Mbps with an accompanying audio stream at 192kbps. Using these figures Table 3.2 below provides the volume of data transferred for a 30, 60 and 90minute programme, illustrating the requirements of delivering such services over the Internet. (bitrates for BBC's iPlayer obtained from (BBC-iStats 2009))

		Duration of Video (minutes)		
		30	60	90
Video bit-rate (kbps)	500	110MB	220MB	330MB
	800 (default)	175MB	350MB	525MB
	1,500	329MB	660MB	989MB
	3,500	700MB	1,400MB	2,100MB

Table 3.2: Approximate volume of data transferred for streaming videos

Aside from the vast difference in size compared with an average video on YouTube, a second key difference that should be noted is that videos on YouTube that are not provided by broadcasters (which represented, at the time of writing, the majority of the clips hosted by the website), are delivered in a pseudo streaming fashion. This means the video file is downloaded to the user's system as a discrete file, and through the use of Adobe's progressive download technology can begin playback before the file transfer is complete. Traditional streaming differs from this by establishing a streaming session between the user's system and the media server, which then sends the video stream in real-time. Traditional streaming methods facilitate the user to jump to a specific section of the video without having to wait for that portion of the video to have downloaded (as would be the case with pseudo streaming). However, traditionally streamed videos are not cached by the user's system and therefore if the user wishes to watch the video again the data stream will be transmitted again.

As with the HTTP analysis, the study also considered the characteristics of streaming media services from the perspective of the transport layer, specifically considering the number of connections established and the packet size.

When viewing a programme using the BBC iPlayer website, 7 connections were established, of which 6 were used to retrieve the web page (text, imagery, formatting and javascript files) and a single connection to a media server that provided the media stream using Adobe's Real-Time Messaging Protocol (RTMP) (Adobe Systems Incorporated 2010), over a TCP connection. Depending upon whether the client's system is behind a firewall or not influenced the port that the media stream was established on. In the absence of a firewall the connection used the standard RTMP port 1935, however, in the presence of a firewall blocking this port the connection was made using the standard HTTP port 80.

For all levels of quality the media stream connection sent full sized packets (1460 bytes), excluding the Live broadcast service, which operated at a reduced bitrate of 368kbps, and with a mean packet size of 597 bytes. These figures were observed to be constant for a number of recorded and live programmes and are therefore assumed to be indicative of the properties for all recorded and live transmissions.

Similarly for a video provided by a broadcaster, but hosted on YouTube (e.g. 4OD or Five), a number of connections are made "setting up" the page, complemented by a TCP connection to a media server, which again uses RTMP, and port 1935 if available, resorting to port 80 if necessary.

Since these measurements were made a study has been published that presents observations of the network characteristics for YouTube video streams (Rao, Legout et al. 2011). This study considers the characteristics of data transfer rather than the typical size or bitrate of a video, however, some interesting observations are made. The most notable is that YouTube servers appear to use mechanisms that delivers data in one of three fashions, buffering followed by steady state with no ON-OFF phases, buffering followed by a steady state of short ON-OFF phases and buffering followed by a steady state of long ON-OFF phases. These transfer strategies clearly deviate from the traditional behaviour of a file transfer, and were observed to vary depending upon platform (Web browser or dedicated mobile application), encoding bitrate and media container (Adobe Flash or HTML5). It was also observed that YouTube servers may artificially retain TCP window sizes during OFF periods, rather than reducing window sizes following an idle period and begin probing for available bandwidth at the next ON period. The maximum transmission rate from YouTube servers was also reported to be limited, possibly as a response to reports that a large proportion of users fail to view a video in its entirety (Gill, Arlitt et al. 2007).

The observations from the study presented in this thesis combined with those from related publications provide an important insight into streaming video characteristics. The identification of file size distribution, encoding rates, flow characteristics and adjustments to traditional TCP behaviour all provide crucial input for a novel QoS solution.

3.4. Peer-to-Peer Systems

In section 2.2.2.4 Peer-to-Peer (P2P) systems were introduced as presenting a threat towards network stability and fairness, largely through establishing multiple TCP connections to maximise throughput over coexisting applications. This section of the study into Internet services aimed to observe P2P applications in operation in order to characterise and better understand their behaviour. P2P architectures are implemented by many systems including, Sky’s Anytime Player, Skype (VoIP client, with value added services such as Video calling, file transfer, instant messaging and VoIP-to-PSTN calling), Spotify (a free on-demand music streaming service) and probably the most notorious use of P2P networks is for file sharing, for example Direct Connect, eDonkey, Gnutella and the most popular P2P protocol BitTorrent.

This study began by considering the P2P properties of proprietary systems such as Sky's Anytime player, Skype and Spotify. This process was complicated by the use of encryption and anti-reverse engineering techniques by the developers. However, some information can be inferred about their behaviour from observing and analysing the applications in use, which was the approach used in this study.

3.4.1. Peer-to-Peer properties of Skype

3.4.1.1. Overview of Skype's Peer-to-Peer Architecture

The development of Skype over the past 7 years has without a doubt promoted VoIP technologies into the consumer market. With Skype reporting over 500million registered accounts, and frequently exceeding 20million concurrent online users[2] the success of the application is evident. According the Ipoque study on Internet services, Skype represented an average of 35% of all VoIP protocols for all geographical areas considered, excluding Eastern Europe and the Middle East where over 80% of all VoIP traffic is attributed to Skype (Schulze and Mochalski 2009).

A number of characteristics of Skype have motivated the research community to hypothesis how the protocol, application and the P2P network operate, from which a picture can be established, although as Skype continues to evolve the operation remains partially unknown.

Skype was developed by the same organisation that developed the early P2P file-sharing application, Kazaa, and adopts a similar P2P architecture. Featured in literature (Baset and Schulzrinne 2004; Guha, Daswani et al. 2006; Bonfiglio, Mellia et al. 2009), and also identifiable through packet capturing, is the use of two-tiered overlay network, consisting of standard hosts and so-called super-nodes. The majority of systems with the Skype software installed will operate as standard hosts however, if a system has a public IP address (i.e. not

[2] Figures obtained from the Skype website and through information provided by the Skype client software.

behind a router with NAT) and has sufficient CPU power and a 'fast' Internet connectivity then it may be promoted to become a super-node. Super-nodes form the control plane of the Skype P2P network, to which standard hosts send availability information, instant messages, file transfers and VoIP session requests (Guha, Daswani et al. 2006). For VoIP calls, Skype attempts to establish a direct connection between the participants, using a number of techniques to circumvent NAT devices and firewalls, if these methods fail then the call is relayed via a super-node. Therefore to fully identify the P2P traffic characteristics of Skype, the operation of both standard nodes and super-nodes need to be observed.

3.4.1.2. Standard Skype node activity

Using a detailed previous study of the Skype protocol (Guha, Daswani et al. 2006), and verifying this behaviour using a number of packet captures from standard hosts operating the latest Skype software, it is observed that a standard Skype client sends/receives very little data (<1MB per day) when simply idle. Using packet captures from 5 separate locations, each lasting 24 hours and repeated for 10 consecutive days, the mean number of connections for a 24 hour period was 220. Over 95% of these connections were unidirectional flows containing only a few packets, sent from the local host and therefore likely to be control messages sent to super-nodes. In addition to the packet captures, the *shared.xml* file was copied from each system following the Skype application being restarted every 24 hour period - this file contains a list of 200 Skype super-nodes that the client updates periodically. When cross-examining the IP address of each super-node (from *shared.xml*) with the hosts from the list of established connections, approximately 25% of these connections were with known super-nodes, suggesting that the client updates the list of super-nodes more frequently than once every 24 hours, however this was not confirmed.

3.4.1.3. Skype Super-node activity

Having observed the connection properties of a standard Skype host, it was necessary to piece together some observations of Skype super-nodes. This aspect of the study is more inferred than characterising the traffic for a standard host for a number of reasons. Firstly, it can only assumed that a system is acting as a super-node after observing an increase in traffic volumes over time (given Skype is the only application operating). Secondly, the algorithm that retrieves the addresses of super-nodes when a Skype client starts is unknown, and therefore it is unknown whether certain super-nodes are preferred over others, raising the question, "Is there an average super-node?" Finally, the criteria for becoming a super-node are rather vague and therefore it cannot be guaranteed that a system will be promoted to a super-node, or how long it will remain a super-node having been promoted.

Despite these challenges some information about the behaviour of a super-node can be assembled from previous studies and observations from packet captures. In two early studies into Skype, figures of super-node bandwidth usage have been offered. In 2006 (Suh, Figueiredo et al. 2006) a 20 day capture revealed over 1GB of relayed Skype data, including control information and relayed VoIP calls. A parallel study states over 13GB of data was observed over a period of 135 days (Guha, Daswani et al. 2006), this second study goes further to segregate the traffic relayed by the super-node as either control/IM traffic or relayed VoIP. This segregation suggests that the vast majority of traffic sent via a super-node is within the control plane, rather than relayed VoIP data. In addition to these figures published in academic literature, Skype users frequently enquire via online forums and support groups as the bandwidth usage of Skype, with figures of 1-2GB per month not being uncommon.

To gain an insight into the behaviour of a super-node a system running Skype was left running on a 10Mbps Internet connection for 10 days. Although behind a router with NAT ports 80, 443 and the random port selected by Skype during installation were opened for the system within the router, thus emulating the known requirements for a Skype super-node. For control purposes a second system with Skype was left running for the same period, but without any port forwarding configured at the router.

After approximately 1 hour the volume of traffic observed at the control system was less than 1MB (it should be noted this initial figure include traffic generated during the login process), whereas the intended super-node had observed over 10MB of traffic. Following the experiment the total volume of traffic captured for the control node was ~ 11MB, which coincides with the figures observed during the earlier standard-node analysis. The second system, which featured the port openings at the router, observed a far greater volume of traffic, totalling just short of 400MB, indicating that it was highly probable this node had been operating as a super-node.

Over the 10 day period, the super-node witnessed 33,171 flows, of which 27,060 were TCP and the remaining 6111 UDP, a significant difference compared with the average 220 connections the standard node was observed to establish during 24 hours. Tcptrace (Ostermann 2003) was used to further analyse the packet captures, and revealed that 60% of the TCP connections were between 10 and 20 packets (in either direction), while the UDP flows were more evenly distributed, with 85% being less than 50 packets in length, despite there being fewer UDP connections in total. Tcptrace also revealed that 97% of the TCP flows were between 10 bytes and 10kB in length, which suggests the majority of TCP data at the super-node appears to be control information from other Skype users, rather than any form of relayed traffic. A similar distribution of flow size was observed for UDP flows, where 99% of flows were below 10kB.

While the increase in traffic volume suggested Skype had been more active on the "super-node" it was not sufficient to categorically say whether or not the system had been used to relayed data between Skype users' systems. Due to the large number of short flows with similar characteristics, in terms of number of packets and bytes sent, identifying relayed traffic among short flows was proved rather difficult. However, when filtering the traces to show only those flows with more than 400 packets the ability to identify potentially relayed traffic was greatly improved. Figure 3.8 provides an excerpt from the tcptrace output, where a selection of connections with more than 400 packets in either direction is shown. As tcptrace parses the packet capture it gives each new connection (determined by source and destination address, transport protocol and source and destination ports) a number, which are ordered sequentially. Therefore, when looking to initially identify a pair of relayed connections, two connections that appear adjacent in the tcptrace output, with a similar number of packets, provides a strong first impression that they could be relayed traffic. For example, Figure 3.8 shows seven pairs of connections that show signs of possibly being correlated.

Connection Number	Host A	Host B	Number of Packets A → B	Number of Packets B → A
51	90.205.6.208:1644	192.168.1.9:52382	513	690
52	90.202.7.103:51418	192.168.1.9:52382	497	673
161	70.153.84.116:65233	192.168.1.9:52382	1273	1511
162	92.3.148.167:55320	192.168.1.9:52382	2070	2002
520	88.224.189.224:13416	192.168.1.9:52382	630	642
521	85.108.15.19:3715	192.168.1.9:52382	575	637
640	212.183.140.53:7112	192.168.1.9:52382	10442	23268
641	94.253.228.208:52067	192.168.1.9:52382	23741	9643
~~1344~~	~~78.54.33.40:2421~~	~~192.168.1.9:52382~~	~~446~~	~~722~~
1406	78.54.33.40:3371	192.168.1.9:52382	655	749
1408	217.92.116.164:37554	192.168.1.9:52382	525	617
1537	78.54.33.40:4831	192.168.1.9:52382	446	512
1538	193.175.118.59:2651	192.168.1.9:52382	449	536
1620	212.183.140.53:52833	192.168.1.9:52382	1131	1276
1621	94.253.228.208:1080	192.168.1.9:52382	1313	1440

Figure 3.8: Tcptrace output for Skype super-node connections with greater than 400 packets sent

In order to prove/disprove any relationship between each pair of connections the analysis applied a filter to the original packet captures to display only each pair of connections. Then the trace for each connection pair was further analysed using metrics described by (Suh, Figueiredo et al. 2006) to identify relayed traffic. These metrics were:

- $S_{i,j} = |S_i - S_j|$ - the difference between the start times of flow i and j
- $E_{i,j} = |E_i - E_j|$ - the difference between the end times of flow i and j
- $B_{i,j} = \frac{B_i}{B_j}$ - the ratio of bytes carried by flow i and j

Using these metrics 115 pairs of TCP connections were identified as having a very high probability of being correlated with one another. While 230 connections represent a very small percentage of the overall number of TCP connections, this figure does not suggest that these were the only TCP connections to be relayed via the super-node. Moreover, connections of fewer packets are more difficult to categorically prove correlation between.

Due to TCP dynamics the TCP connection between Host A and the super-node was never perfectly symmetrical to the TCP connection between the super-node and Host B. However, when repeating the above method to identify potential pairs of UDP connections[3], the simplicity of UDP made identifying traffic relayed by the super-node far easier.

Firstly, UDP connections were filtered to show only those with more than 400 packets in either direction. This identified a number of potential pairs of connections (using the same visual identification methods associated with Figure 3.8). Following this initial identification each pair of connections was filtered from the original packet capture using TCPdump. An example of the TCPdump output is shown in Figure 3.9, which shows UDP packets arriving at the super-node (Host 192.168.1.9) and within a fraction of a second being relayed to the remote destination.

Timestamp of packet observed by monitor	Protocol	Source Address	Destination Address	Payload Length (Bytes)
10:57:39.938422	UDP/IP	86.11.1.125.46258	192.168.1.9.52382	97
10:57:39.938461	UDP/IP	192.168.1.9.52382	62.254.7.27.4119	97
10:57:39.943523	UDP/IP	86.11.1.125.46258	192.168.1.9.52382	80
10:57:39.943560	UDP/IP	192.168.1.9.52382	62.254.7.27.4119	80
10:57:39.953977	UDP/IP	62.254.7.27.4119	192.168.1.9.52382	75
10:57:39.954019	UDP/IP	192.168.1.9.52382	86.11.1.125.46258	75
10:57:39.958809	UDP/IP	86.11.1.125.46258	192.168.1.9.52382	78
10:57:39.958845	UDP/IP	192.168.1.9.52382	62.254.7.27.4119	78
10:57:39.988002	UDP/IP	86.11.1.125.46258	192.168.1.9.52382	91
10:57:39.988057	UDP/IP	192.168.1.9.52382	62.254.7.27.4119	91
10:57:39.990746	UDP/IP	62.254.7.27.4119	192.168.1.9.52382	71
10:57:39.990787	UDP/IP	192.168.1.9.52382	86.11.1.125.46258	71
10:57:40.003492	UDP/IP	86.11.1.125.46258	192.168.1.9.52382	94
10:57:40.003537	UDP/IP	192.168.1.9.52382	62.254.7.27.4119	94

Figure 3.9: Excerpt of a TCPdump output showing the relaying of UDP packets between two hosts via Super-node

The nature of the relayed traffic, for both TCP and UDP connections is largely left to assumptions. However, a variety of relayed traffic was collected and analysed, which included, TCP flows which could be files transfers and paired UDP connections with almost symmetrical

[3] Although UDP is by definition, a connectionless protocol, in this case a UDP connection is defined as communication between a source / destination pair (identified by IP and port numbers).

characteristics, which could be two-way VoIP/Video calls. Also observed were paired connections that were heavily asymmetric, which are hypothesised to be one-way video (e.g. webcam video) during an Instant Messaging session via Skype.

This section of the study has provided an insight into the P2P properties of the Skype application, while much is left unknown about the Skype protocol this study focussed on the network and transport layer properties rather than attempting to reverse engineer the actual operations of the application. Further characterisation of the VoIP features within Skype is provided later in this chapter, in section 3.5.

3.4.2. Spotify

Spotify is an on-demand streaming music application, which allows a user to search for and listen to music through the Spotify application. The terms of use for Spotify specify that users allow the 'Spotify network' to use their Internet connection and system to assist in the operation of the service, suggesting the possible use of a P2P-like architecture.

After analysing packet captures collected while using the Spotify application, it was determined that data is sent in 500kB bursts, over a single TCP/IP connection (resolved to show a *zzz*.Spotify.com address), Figure 3.10 illustrates this bursty behaviour. Data is sent using TCP over port 80, although is not traditional HTTP traffic despite using the same port, the packet payloads are encrypted. Periodically the application would establish TCP connections with (presumed) other Spotify peers, (this presumption is made given that resolving a selection of the IP addresses revealed hostnames describing ISP customers e.g. *f048184058.adsl.alicedsl.de*). However, these connections did not stay established for more than a few seconds and typically carried less than 5 packets in either direction. Spotify streams media at 160kbps for free user accounts and at a higher 320kbps for premium accounts[4], resulting in 4.8MB file for a 4minute song being transferred at standard quality and just under 10MB per song at premium quality.

[4] Figures obtained from Spotify's Wikipedia page.

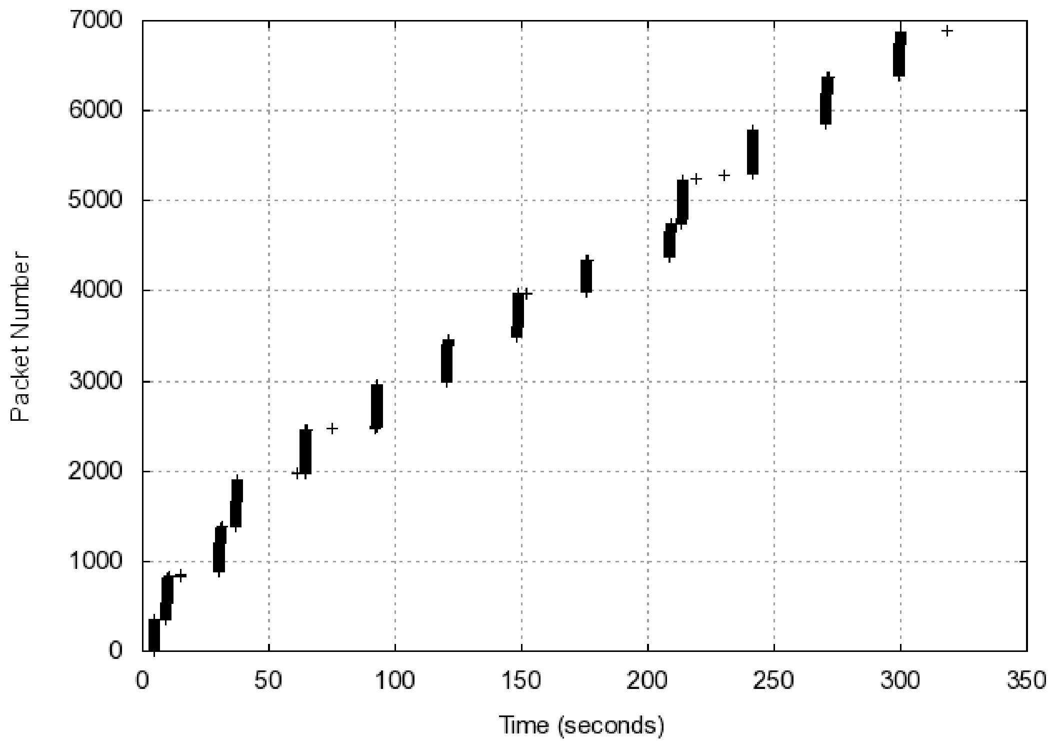


Figure 3.10: Observed incoming data bursts for Spotify application

During this study no evidence was collected of the Spotify client uploading to other users, however, it was noted that the media played through Spotify appeared to be cached locally in pieces ranging from 250kB to 1.5MB. The cached files also appeared to be encrypted and therefore the exact nature of these files is unknown, however, it is hypothesised that this caching may form the basis of Spotify's P2P operations, possibly indicating a similar architecture to Skype, where nodes with sufficient bandwidth relay their cached media to other users, although this remains to be proven/disproven in future research. A more recent separate study (Ellis, Strowes et al. 2011) has revealed that peer-to-peer activity between Spotify peers has been observed. The number of established peerings from a single Spotify client ranged between 20-40, and data transfer was recorded for a subset of those peerings (in addition to the client-server transfers identified from the aforementioned study). This indicated that the Spotify architecture can employ some level of peer-to-peer functionality to facilitate the delivery of media. However, the circumstances which invoke peer-to-peer behaviour were not identified, and a change in Spotify user policy has also taken affect between the two investigations. It is questioned whether or not Spotify's architecture makes use of Skype-like Super-nodes to distribute media, but without a larger scale investigation this cannot be confirmed.

3.4.3. BitTorrent

While proprietary applications using a P2P system are difficult to fully characterise, the behaviour of P2P file sharing protocols has been studied intensely over the past decade. Since 2006 the most popular P2P file sharing protocol has been BitTorrent, which contributes to over 40% of all P2P traffic (Schulze and Mochalski 2006; Schulze and Mochalski 2007; Schulze and Mochalski 2009).

Downloading a file via BitTorrent involves the user downloading a *torrent* from a torrent listing site or torrent search engine (such as BTJunkie.org, thepiratebay.se, mininova.org…). This torrent requires the user to have a BitTorrent client installed on their system, which can open the torrent and read the metadata held within. Amongst this metadata the torrent will provide the address of a *tracker* (a server on the Internet that maintains a database of clients downloading a specific file). The tracker provides the BitTorrent client with a list of peers that have (part of) the desired file. The BitTorrent client then negotiates with these peers via the BitTorrent protocol the exchange of 256kB pieces (further broken into 16kB chunks) of the desired file.

The exchange of pieces between peers comprises of a number of on-going processes, which aim to provide the client with the fastest possible download rate. Firstly, a connection to each of the peers obtained from the tracker is established, and information about the pieces each peer currently has is exchanged. Although established, these connections with peers do not guarantee the client will begin receiving data from connected peers. For each connection the client maintains four states variables, *am_choked*[5], *am_interested*[6], *peer_choked, peer_interested*, and keep-alive messages are sent (by default) every 120seconds to maintain the TCP connections.

The process of *choking* is performed to limit the number of concurrent upload connections for a peer, to maximise TCP performance. Based on throughput measurements of the download rates of connected peers (who are uploading to the client), the client *unchokes* the four peers who have provided the fastest download rates and are *interested* in the pieces the client currently has – enabling reciprocation. This *choking* and *unchoking* is performed every 10 seconds by default to ensure the client is downloading from the peers who provide the fastest connection, and in return uploads to each peer. Peers with better upload rates compared with the four unchoked peers, but aren't interested in downloading from the client are unchoked, to increase the download rate for the client. If one of these peers becomes interested in the client then they replace the slowest unchoked peer.

Periodically, (every 30 seconds by default), the client optimistically unchokes a single peer regardless of whether they are interested or not. This process is performed to provide newly connected peers a better chance of obtaining a complete piece, which can subsequently be used to upload.

The operation of the BitTorrent protocol demonstrates reciprocity in terms of data exchange, those peers providing the best upload rates are rewarded with an unchoked connection. If the client fails to receive any data from an unchoked peer for 60 seconds then it will assume it has been choked by the peer, and in response the client stops uploading.

---

[5] *Choking* refers to a peer temporarily refusing to upload to the client (or vice versa) while maintaining the connection open.

[6] A client will maintain whether it is *interested* in the pieces the remote peer has. This informs the remote peer downloading will begin if *unchoked*. Similarly, the peer informs the client if it is *interested* in pieces the client has.

The performance of the BitTorrent protocol is heavily dependent on a number of factors, which include:

- The connection rates of the peers provided *by the tracker*
- The popularity of the torrent listing site (less popular will result in fewer peers)
- The accuracy of the metadata provided by the tracker (the less often the tracker updates its records the higher the probability it will contained expired peers)
- The configuration of the BitTorrent client
- Any traffic shaping measures implemented by the service provider

While a user may be interested in finding the most popular torrent search engine (to increase the size of the peer-pool) it is doubtful that a typical user would consider the accuracy of the data provided by the tracker, therefore from here on its effect on the traffic characteristics is considered negligible. The top three torrent search sites were reported to be isohunt.com, thepiratebay.org and BTJunkie.org with 12.5million, 11million and 4.5million unique monthly visitors respectively (eBizmba.com 2010). These sites provided the sample torrent files for this study.

With regards to the configuration of the BitTorrent client used, Table 3.3 provides the default configurations for the top 5 BitTorrent clients (according to the survey of BitTorrent users by About.com (About.com 2010)).

BitTorrent Client	Download rate limit	Upload rate limit	Max concurrent Torrents	Max Connections (per-torrent)	Max Connections (total)	Upload Connection limit	Seed Limit
µTorrent	Unlimited	Unlimited	8	50	200	4	Seed while Ratio <=150%
Vuze	Unlimited	Auto (2% of measured download rate)	4	Unlimited	Unlimited	Unlimited	Min 600seconds
Deluge Torrent	Unlimited	Unlimited	8	Unlimited	200	4	Seed forever
ABC	Unlimited	Unlimited	2	Unlimited	Unlimited	5	Seed while ratio <=100%
Transmission	Unlimited	Unlimited		60	240	Unlimited	Seed forever

Table 3.3: Default configurations for the Top 5 BitTorrent clients

The aim of this study into the BitTorrent protocol was to observe a number of characteristics that describe the protocol's behaviour:

- The number of connections established with peers throughout the download
- The duration of each of these connections
- How the distribution of established connections vary during the file transfer period
- Ratio of uploaded data to downloaded data
- Ratio of upstream connections to downstream

3.4.4. Methodology of BitTorrent Analysis

The monitoring of P2P (BitTorrent) traffic was limited by a number of factors. Firstly, due to the bandwidth intensive nature of P2P applications, the use of P2P protocols within the Plymouth University network is prohibited. To alleviate this constraint all of the experiments described in this section were performed on residential broadband connections. The second limiting factor was that ISPs often implement traffic shaping measures against P2P traffic during *peak-hours* (for example the evenings). To minimise any impact that ISP traffic shaping may inflict on the results a number of test downloads were performed throughout the day recording the average throughput during a 10-minute download period. It was established through a series of preliminary measurements that the BitTorrent protocol performed at its best between 12:00 and 15:00, and therefore all subsequent tests were performed during this "unshaped window".

It was decided that a 10-minute window of observation per downloaded file would be sufficient to observe connection characteristics. The only preserved files were TCPdump traces, which had had the packet payloads stripped to leave only the headers – required for analysis. Over 100 unique files were partially downloaded using BitTorrent, which provided sufficient data to be able to ascertain a traffic profile. These 100 files were comprised largely of Linux ISO files (a typical Linux distribution provides a number of unique files, e.g. DVD image, CD images, Live images, USB images) and open source software packages.

The first characteristic to be investigated was how many TCP connections were established throughout the duration of the download. Using the TCPdump traces collected during the experiment, tcptrace (Ostermann 2003) was used to analyse the captures and determine the number of fully established connections that were open at an interval of 5 seconds, for the duration of the download. Within Figure 3.11 between 60-80 established connections is the most densely populated region of the graph. It is noted that Figure 3.11 indicates that the number of established connections exceeded the prescribed limit imposed by the BitTorrent client (See Table 3.3). This is attributed to the limit imposing a restriction on the number of concurrent active downstream connections, thus allowing additional connections to be established providing they are currently choked, this behaviour describes Figure 3.11 where more than the maximum number of connections are reported as being established.

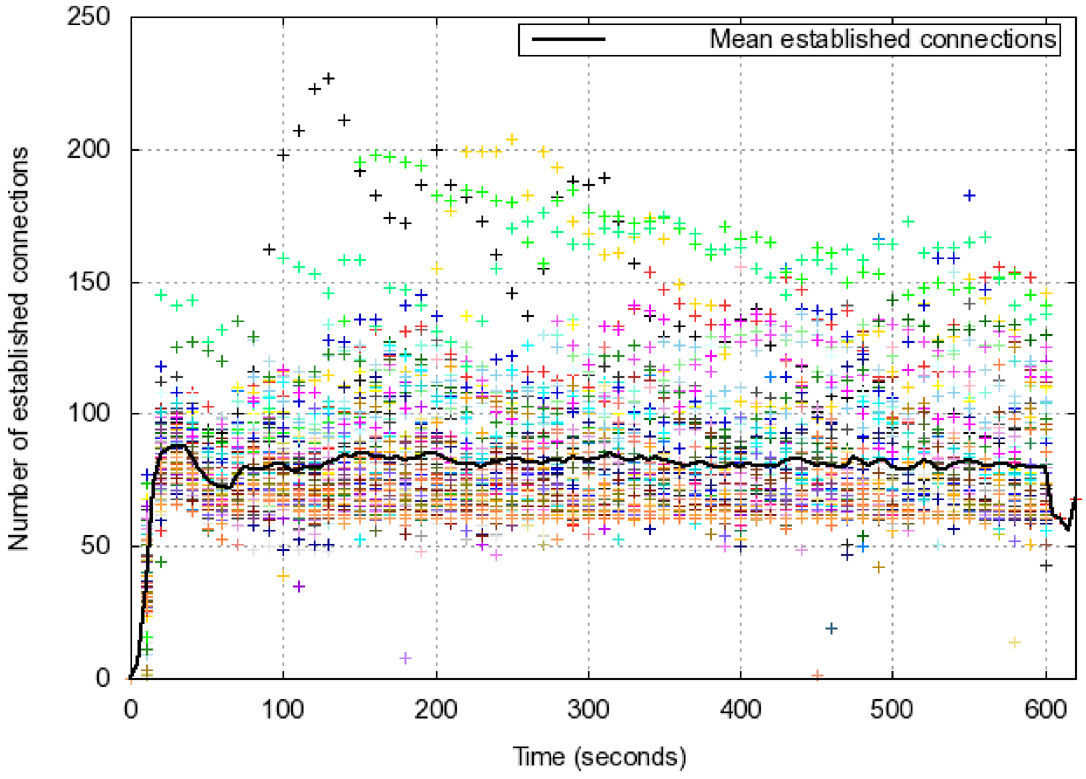


Figure 3.11: Established TCP connections over time for 100 BitTorrent transfers (one colour per connection)

The variation of the number of connections over time is observed to be very low after the first 15 seconds, when the number of connections exponentially increases. Following this start-up period the standard deviation for the number of connections is just 2.5 throughout the remainder of the observed period. It should be noted that the number of open connections does not describe how many of these are actively engaged in the exchange of BitTorrent pieces. It is known that the BitTorrent protocol will open many connections but unless a peer unchokes the connection, it may well see no traffic. However, the sending of keep-alive messages restricted the ability to filter the traces to only those connections exchanging pieces of the file. Despite this constraint, the study does go on to investigate the size of each flow, which reveals some information as to the number of active flows.

From the BitTorrent protocol specification, it is known that despite the large number of connections that are established, only those peers that *unchoke* the client will provide data. Therefore, the majority of data is downloaded from a small number of the connections, leaving the remaining connections exchanging keep-alive messages and control information. Based on the BitTorrent traces analysed, 10% of the flows were responsible for 99.9% of the total data transferred. 90% of the flows recorded were less than 10kB in size (as shown in Figure 3.12), which indicates a high proportion of the flows sent only control data. The upper 10% of flows were of lengths between 10kB and 446MB, of which 67% of these flows were less than 1MB and 95% less than 10MB, which demonstrates that a very small number of connections are responsible for the majority of the data transferred. Furthermore, Figure 3.12 also illustrates that for the upper 10% of connections the sizes for flows upstream are of an order of magnitude less than the downstream connections. It is noted that the volume of data transferred is a reflection of the capacity of the Internet connection used.

Next was to determine the ratio of data uploaded to data downloaded, therefore also obtaining the ratio of upstream connections to downstream. The Tshark (Wireshark 2009) program was used to calculate the volume of data in each direction for each flow observed. From these statistics it was observed that for 90% of the connections the ratio of uploaded to downloaded data was approximately 1:3. This statistic alone provides a misleading impression of the traffic, given that these flows predominantly carried only control traffic between peers, and therefore carried very little data. In terms of data observed in each direction, over 28GB of data was analysed, of which 92% was downstream traffic, indicating a much lower upload to download ratio from the perspective of data volumes. It should be noted that these figures are representative of the ratio of traffic transferred during a file download using BitTorrent. Whether a user opts to continue uploading (seeding) pieces of a file after he/she has acquired the complete file is their prerogative, although this will naturally impact the ratios.

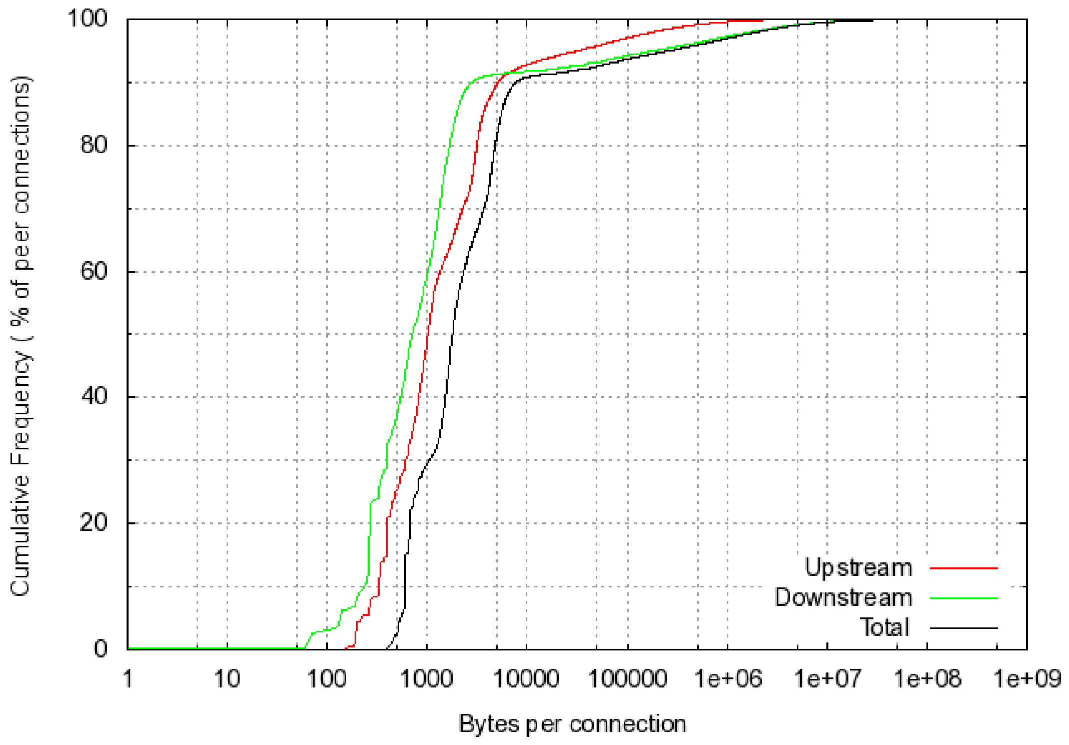


Figure 3.12: Cumulative Distribution Function for the volume of data uploaded to downloaded

Further investigations into the characteristics of BitTorrent flows led the study towards determining the duration of each established connection. This area of the study was complicated by the use of keep-alive messages, as detailed earlier in the description of the BitTorrent protocol, this made filtering out connections sending more than just keep-alives very difficult.

However, despite this, the distribution is given in Figure 3.13, which indicates that the majority (80%) of connections lasted for less than 60 seconds. The upper limit of 600 seconds is bound by the observation period, and so does not suggest an upper limit for connections.

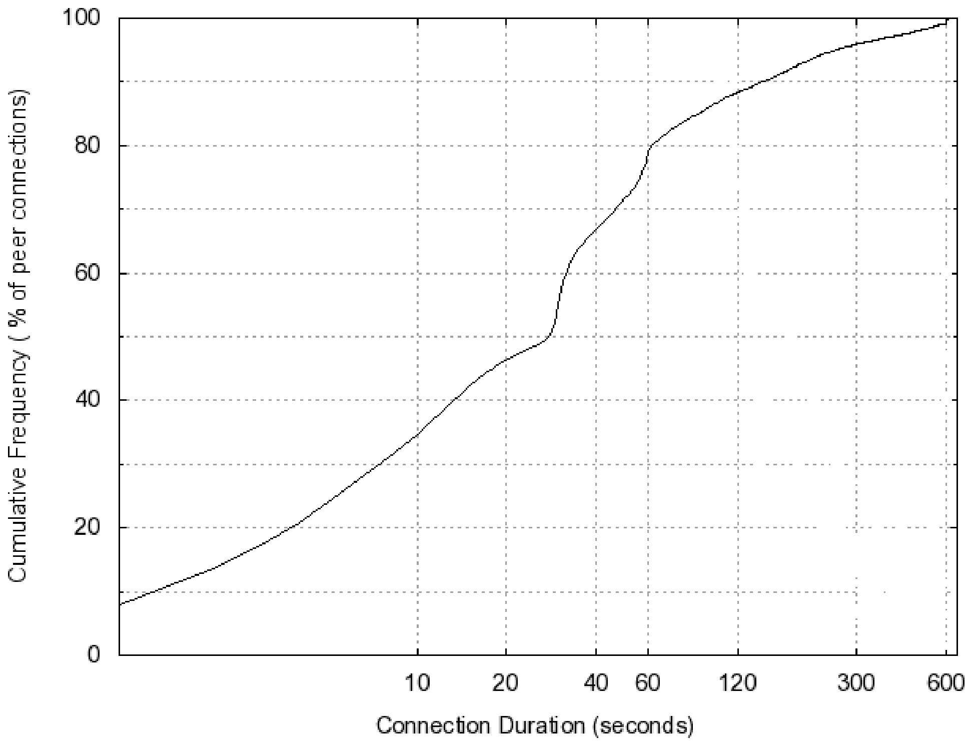


Figure 3.13: Cumulative Distribution Function for connection duration

With regard to the underlying transport layer protocols used by BitTorrent, all of the connections analysed using the packet captures were TCP. This is contrary to the findings of (John, Tafvelin et al. 2008) were small (less than 3 packets) UDP flows accounted for 4% of the P2P traffic analysed in a similar study. Possible explanations for this presence of UDP traffic include that other P2P protocols were involved that use UDP, or that the traffic captures included the client loading the *.torrent* file which involves contacting a server via HTTP and therefore involves a DNS lookup (over UDP), however, this study only observed the data transferred during the download period and as a result no UDP traffic was identified.

3.5. Voice over IP

As introduced earlier in this chapter the use of Voice over IP technologies has witnessed considerable growth over recent years as a cost-efficient alternative to traditional voice-call technologies. Skype has already been introduced as the leading application for VoIP within the consumer marker, however alternative software based VoIP clients exist. This section of the Internet traffic study details the characteristics of a number of leading VoIP applications in use today, specifically, Skype, Windows Live Call (Messenger), Yahoo Messenger and Google Talk, which were responsible for over 50% of all VoIP traffic on the Internet(Schulze and Mochalski 2009).

Many of the details of these VoIP clients are unknown, given that they are all closed-source proprietary applications. However, unlike Skype, Google Talk, Yahoo Messenger and Windows Live call are all reported to use a derivative of SIP/RTP for their VoIP functionality, and do not use a P2P architecture like Skype. Both Microsoft and Yahoo make no official statements on the audio codecs that are used by their products, although third party information suggest that Yahoo uses the Truespeech and the Internet Low Bit Rate Codec (iLBC), and Microsoft use the Siren 7 codec, which is a slightly modified implementation of G.722.1. Google and Skype report that their clients support a number of different codecs, any of which may be selected given current network conditions and the support of the remote client. The algorithms used by each client to determine which codec is selected are unknown for all clients. Previous versions of Skype (prior to version 4.0) could be engineered to use a specific codec, however, this ability appears to have been removed from version 4.0 onwards. This information for each of the four VoIP clients is summarised and presented in Table 3.4.

VoIP Client	Codec	Sampling Frequency (kHz)	Frame Size (ms)	Bitrate (kbps)
Google Talk	iLBC	8	20 / 30	13.33 / 15.2
	PCMA (G.711)	8	20	64
	G.723.1	8	30	5.3 / 6.3
	Speex	8 / 16 / 32	30 / 34	2 - 44
Skype	SILK (default for PC-to-PC)	8 / 12 / 16 / 24	25	6 - 40
	iLBC	8	20 / 30	13.33 / 15.2
	G.729 (default for PC-to-PSTN)	8	10	8
	iPCM-wb	16	10 / 20 / 30 / 40	80
Windows Live	Siren 7 (G.722.1)	16	20	16, 24, 32
Yahoo Messenger	iLBC	8	20 / 30	13.33 / 15.2

Table 3.4: Popular VoIP clients and their associated voice codec parameters

The aim of this element of the Internet traffic study was to profile each service from the network and transport layer perspective, identifying application layer features (such as silence suppression) where and when possible.

The method used to evaluate each VoIP client involved placing a call over the Internet between two residential locations. The capacity of each Internet connection was measured using two leading speed test websites, speedtest.net and broadbandspeedchecker.co.uk - each connection had at least a 2Mbps downstream connection with at least 0.5Mbps upstream. The tests for each VoIP client were performed between the hours of 2pm and 4pm, to avoid peak-time traffic spikes. Prior to each VoIP-client being tested the speed tests were repeated at each location to ensure the connection remained relatively constant for each test. Located at one end of the connection was a network monitor running TCPdump to capture both data leaving the local host and the incoming data from the remote destination.

To ensure the packet captures for each VoIP client were as similar as possible two pre-recorded sound clips were used to simulate a conversation between the two locations. The sound clips used were approximately 4minutes long, featuring on-off periods of voice with a mean on period of 10 seconds, separated by increasing periods of silence, ranging from 1 second to 15 seconds. This approach was used to identify the behaviour of each client given varying periods of silence.

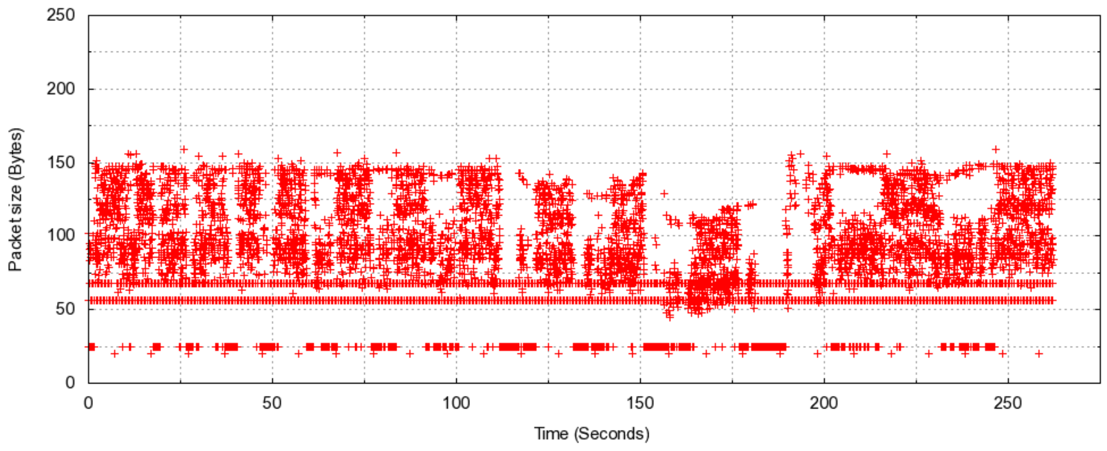
Having collected the packet captures for each of the tests the captures were filtered to contain only traffic originating from the local source. Figure 3.14 shows a scatter plot of payload sizes for each of the VoIP clients. For each of the clients the on-off periods of speech and silence clearly have a visible effect on the payload size. Starting with GoogleTalk (Figure 3.14-a) the majority of payloads are distributed between 70-150bytes. Two continuous streams of packets of 56 and 68 bytes are also visible, which upon further analysis were revealed as STUN request/response packets, used to negotiate NAT routers. Additional packets of 25 and 20 byte payloads are also visible every 300ms, both of which are assumed to be control data being sent to the remote destination. These control packets were filtered out of the capture to consider the packet and bit rate of the voice data. During speech periods GoogleTalk had a mean packet rate of 33 packets/second, and a bitrate of 25kbps, although the bitrate is highly variable. The packet rate dropped to between 2-10 packets/second during silent periods, and the bitrate consequently was reduced to approximately 10kbps. Referring back to Table 3.4 the observed behaviour of GoogleTalk correlates to the properties of the Speex codec, which suggests this was the codec used.

The second plot represents the payload sizes for traffic sent by the Skype client. In contrast to GoogleTalk traffic there appeared to be significantly less control traffic being sent, although it is not known whether control data is appended to speech packets. There is however a 3 byte packet sent every 10 seconds, which is assumed to be some form of control data. The payload plot shows two discrete populations, which correlate to the on-off periods of the sound clip. During speech payloads are distributed between 90-140, and during silent periods the payload is reduced to between 60-90, indicating that a form of silence suppression is in use. Applying a filter to remove any assumed control data the packet rate and bitrate for the Skype voice traffic was inferred. The packet rate was almost constant at 50 packet/second (± 1 packet). The mean bitrate during periods of speech was 44kbps, which was reduced to 30kbps during silent periods, indicating that silence suppression was not used by Skype during the test call.

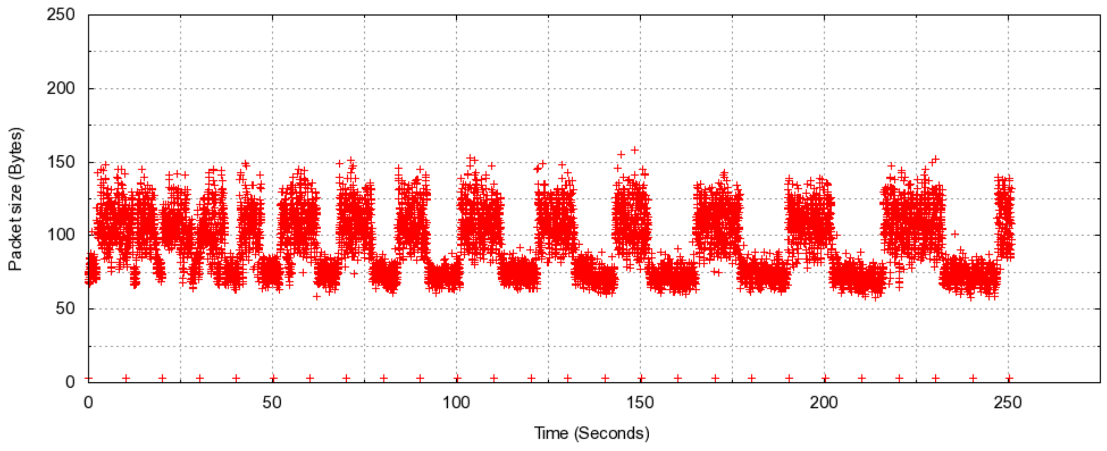
The plot for the third VoIP client, Windows Live Call, shows discrete bursts of packets with very little data transmitted in between these bursts. The packet bursts again correlate to the periods of speech, and indicate that Windows Live Call is implementing a form of silence suppression. In addition to this three potential control streams are also visible on the plot, with packets of 24, 28 and 224 bytes being sent periodically throughout the trace. Regarding the voice data, the packet payloads are higher compared with the alternative clients, tightly distributed between 125 – 190 bytes during speech periods. For silent periods greater than 5 seconds (illustrated at ~45 seconds in Figure 3.14-c) a distinct gap in transmission occurs, this behaviour is repeated for all subsequent silent periods, suggesting a threshold of 5 seconds of silence is required before transmission is suppressed. Applying a similar filter as with the previous two captures, the control traffic was removed to investigate the packet rate and bitrate of the voice data. Similar to Skype, Windows Live Call had an almost constant packet rate of 50 packets/second (±1 packet), which provided a mean bitrate of 60kbps during speech periods. Interestingly this bitrate did not correlate with the characteristics of the Siren 7 codec, however, the Siren 14 codec (also a derivative of G.722.1) can provide a 64kbps bitrate, possibly indicating that this codec is used by Windows Live Call, however this hypothesis is unproven.

The final VoIP client that was evaluated was Yahoo Messenger, for which the payload scatter plot is given in Figure 3.14-d. A number of initial observations can be made about the plot, firstly two streams of (assumed) control data can be identified with payloads of 12 and 64 bytes. The second observation is a clear difference between the data sent during speech periods and silent periods. During speech periods the payload size is distributed between 80-150 bytes, while during silent periods the payload size is constant at 21 bytes. During a speech period the packet rate (excluding control data) was almost constant at 33 packets/second (± 1 packet), providing a bitrate of approximately 40kbps. Subsequently, during silent periods the packet rate reduced to 10 packets/second, which resulted in a bitrate of 2kbps. Similar to Windows Live

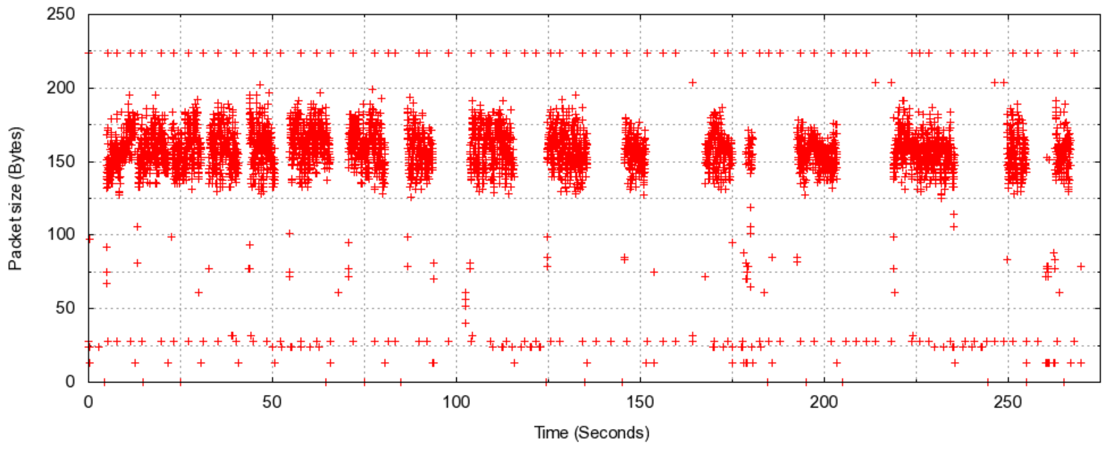
Call the characteristics of the traffic for Yahoo Messenger did not correlate with the codec in Table 3.4, which indicates that an alternative codec is being used.



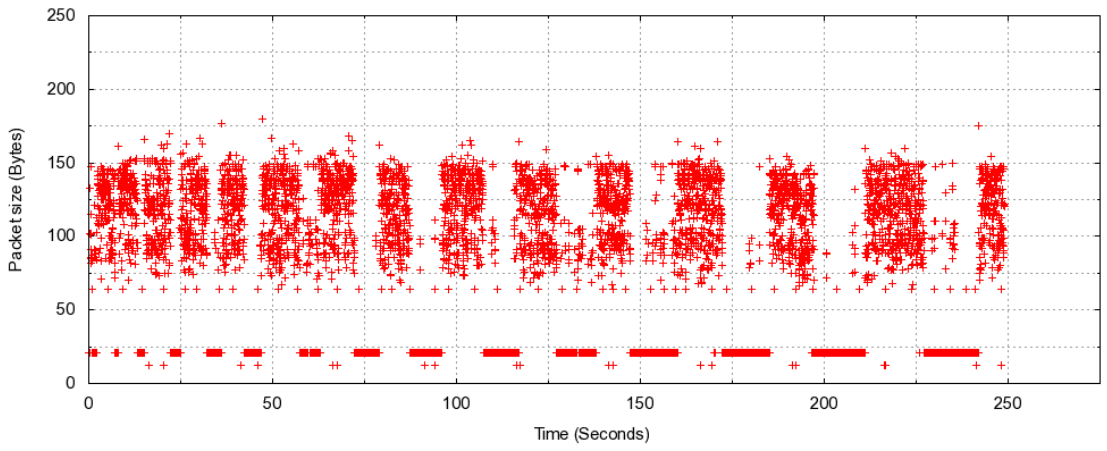
a) GoogleTalk VoIP Client



b) Skype PC-to-PC VoIP Client



c) Windows Live VoIP Client



d) Yahoo Messenger VoIP Client

Figure 3.14: Scatter plot for payload sizes of packets from a VoIP call

These observations were made using a test topology with sufficient bandwidth for the traffic involved. However, a number of previous studies have considered the performance of Skype and MSN messenger (an early version of Windows Live Messenger) in more bandwidth-constrained networks. For example, (Chiang, Xiao et al. 2006) presents an evaluation of both Skype and MSN messenger under constrained network conditions. Chiang et al. observe that over a low bandwidth connection (< 32kbps) Skype reduces both its sending rate and packet size, maintaining a smooth transmission of packets, with low jitter. On the other hand MSN is observed not to reduce either its sending rate or the packet size, and as a result has a lower performance (MOS score) than Skype. Chiang et al. also investigated both clients in lossy environments, concluding that the MSN Messenger client featured no error resilience mechanism, whereas Skype increased its packet size (from approximately 100 bytes to 200-300 bytes) in the presence of loss. This increase in packet size is likely to be attributed to a form of Forward Error Correction (FEC) being used by Skype, although Chiang observes beyond 10% packet loss the performance of the VoIP call decreased while the throughput continued to increase, suggesting the FEC mechanisms in use were not optimised.

A similar, more recent study into the behaviour of Skype under varying network conditions is presented by (Bonfiglio, Mellia et al. 2009), which again observes Skype decreasing the packet size and increasing the inter-packet gap in bandwidth constrained networks. Bonfiglio also presents results illustrating Skype increasing the packet size during lossy periods, again suggesting the used of FEC to compensate for packet loss.

At this time, no studies have presented a complete comparison of the four major VoIP clients evaluated in this section of the thesis. Windows Live Call (MSN Messenger) is generally agreed to provide poorer performance than Skype, however, a further comparison including GoogleTalk and Yahoo Messenger has not been identified, and therefore as future work this study is likely to be extended to provide such a comparison. Similarly, the video-call features of each client have not been evaluated but could form part of future research.

3.6. Summary

This chapter of the thesis provides a detailed insight into a number of Internet services, which together contribute towards a large proportion of the Internet-based activities performed today. Using a combination of empirical data collected for a variety of services and drawing upon observations from related research, a comprehensive description of the "multimedia Internet" has been provided, not only from the application layer perspective, but also considering the network and transport layer characteristics of these services.

A number of key observations were made by this study, for instance the surprisingly distributed nature of website content, which results in many short connections being established to retrieve a typical webpage. Further analysis of (non-live) streaming media over the Internet revealed two primary methods of content delivery, the first approach being pseudo-streaming where data is buffered (as fast as possible) on the user's system ready for playback in real-time, and the second approach, which operates more closely to the word streaming, removing the storing of media locally. Recognising the nature of Internet traffic and the underlying delivery mechanisms are considered a key factor for the novel QoS architecture.

The primary purpose of this study was to obtain an application and protocol level understanding of a selection of modern Internet services. It was recognised that despite the large number of users, and content providers there appears to exist a number of prevailing technologies that are responsible for a large proportion of Internet traffic. The application and protocol characteristics determined in this chapter provided the opportunity to integrate this knowledge into a novel QoS architecture. This architecture is presented in the following chapter.

4. A Novel Approach to QoS Provisioning

4.1. Introduction

The previous chapters examined existing approaches of providing QoS within IP networks and also presented an in depth analysis of current Internet traffic. The outcomes from these chapters were used to design a novel architecture for user-centric QoS provisioning. The architecture is located at the network layer of the Internetworking model. This decision to develop at this layer was taken based on the previously described difficulties related to deploying modifications to the protocol implementations within end-hosts, and the inability to enforce the uptake of such modifications across the Internet. It is also believed that within the network layer, the architecture can fully evaluate the current network conditions, and directly control traffic flows. Although located at the network layer and concerned with managing IP packets, the novel architecture operates a cross-protocol approach, acknowledging the transport and application layer mechanisms implemented by different Internet services, which provided a more intelligent method of management.

4.2. Motivations for the novel approach

Chapter 2 examined the merits and shortfalls of the current approaches that are available for QoS provisioning. The challenge of providing QoS within IP networks continues to be the subject of much research, with efforts focussing across all the layers of the Internetworking model. One of the most prevalent methods for QoS provisioning was reviewed as the Diffserv architecture, which has widely been adopted within large managed network environments (e.g. Enterprise networks). QoS provisioning in such an environment allows for certain luxuries, such as being able to dictate traffic policies, control (or even outlaw) specific protocols and *tune* the QoS configuration according to known traffic profiles – a process that can be repeated periodically or following the addition of a new network service.

For unmanaged networks, such as those of ISPs, the services in use are less predictable, with new services continually emerging, altering user behaviour and expectations, complicating the task of *tuning* a QoS configuration to match the current network traffic. Moreover, the capacity of Enterprise networks is typically the result of forward-planning, with knowledge of the services that will be used. This contrasts highly with the traditional ISP network, designed to exploit a high contention ratio, based on the assumption not every user would be using their connection simultaneously, and traditional Internet services, such as web browsing, are bursty in nature. The emergence of new services over the Internet, such as VoIP, VoD, media purchasing and gaming has transformed the usage pattern of home Internet connections, pressuring ISPs to consider some form of traffic management on their networks. However, this has generally been performed using traffic shaping techniques at the ingress of the ISP network to provide precedence only for premium services, rather than adopting a system that considers the entire spectrum of user services.

While this approach has allowed real-time services such as VoIP to be marketed over ISP networks, it fails to address the QoS requirements for those users not subscribed to premium services or those services which may be falsely classified as non real-time. For example, securing an online purchase in the last remaining seconds is a real-time activity, but is unlikely to be regarded as such, and hence not receive the desired precedence across the network. This concept of providing a more user-centric QoS model was reviewed in section 2.3.3.2, where Diffserv-based frameworks were presented that allow the user to configure the QoS parameters for their individual applications. Despite their described limitations, these user-centric frameworks do advocate a move away from static, policy driven QoS solutions, and a move towards integrating user expectations into the provisioning process.

Moreover, Internet users are no longer using single services at any one time; instead they are engaged in many concurrent services expecting each to receive an acceptable level of service. This draws attention to the inadequacies of the methods used by ISPs to manage their user traffic, favouring individual traffic types. Based on this review of existing QoS mechanisms a distinct need exists for a QoS provisioning method that can adapt to the needs of each user; optimising their entire Internet experience, not solely focussing on a single preferred traffic, therefore only serving one user-group an optimised service.

4.3. The requirements for a novel approach to QoS provisioning

4.3.1. Ideal Solution

The ideal solution for QoS over the Internet would satisfy the expectations of each and every user, irrespective of the services they choose to use. As mentioned in the chapter 1, if the capacity of the network could always exceed the bandwidth demand of the users then there would be no requirement for QoS provisioning. However, since bandwidth is a finite resource and multi-streamed applications (such as P2P) are designed to obtain any available bandwidth, this is neither a practical substitute nor a long term solution. Therefore, an ideal QoS solution, which addresses the limitations of static pre-configured solutions, such as Diffserv, would be to have advanced knowledge describing which of the concurrent activities a user is engaged in they consider being the most important, and providing priority to that service. This approach would ideally have the following functionality:

- The network would interact with the user to obtain their perspective of their service precedence.
- This interaction would have to be non-intrusive, to avoid disrupting the user experience.
- The QoS policies within the network should adapt as the user's usage varies, not be statically configured and require manual evaluation.
- The user's service precedence would have to be upheld from end-to-end.
- The desired QoS for one user must not impair that of others.
- Consider the entire user experience and not solely focus on providing QoS to a single service.

While this level of synergy between user expectations and delivered QoS would provide the ideal user-centric QoS solution, the practicality of such as system needs to be considered.

Firstly, obtaining a user level opinion regarding the level of precedence for a service is particularly challenging. The only guaranteed method to obtain the order of precedence for the services in use by a user would be to consult them. However, examples of this method were reviewed in section 2.3.3, and were concluded as being intrusive to the user experience – contradicting the second *ideal functionality*. A more pragmatic alternative would be to recognise the services in use by the user and evaluate how best to maximise the user experience based on a number of assumptions about the concurrent usage of Internet services. The assumptions required to maximise the quality of the user experience are discussed later in this chapter, in the design of the traffic management component of the architecture (see section 4.6).

The fourth ideal functionality was to uphold the user's order of service-precedence from end-to-end. While Intserv and Diffserv may provide end-to-end QoS, they are typically restricted to within a single Autonomous System or among cooperating Autonomous Systems. The complex peering agreements of the Internet impede the ability to guarantee end-to-end QoS, however, analysis of flows transiting an Internet node can be used to infer the current network conditions of the end-to-end path.. For example, knowing the behaviour patterns of a TCP flow under normal, delayed and lossy conditions (as described in section 2.2.2.2) would allow the architecture to monitor the behaviour of TCP flows and infer the health of the network from end-to-end. As a result the architecture could adjust the handling of a flow in order to counter-act the adverse conditions.

The fifth functionality listed for the ideal solution is a matter of enforcing fairness among users. It has previously been highlighted that certain applications can behave unfairly, and therefore require management within the network. However, it has also been highlighted that traditional static QoS configurations, such as Diffserv, can result in an unfair distribution of resource among users. Therefore, a method to ensure fairness among user traffic is critical for the novel architecture.

The final ideal function stated that the solution should consider the entire user experience and not solely focus on optimising the performance of a single service, this is one of the primary objectives for the research and so the proposed architecture should meet this requirement.

4.3.2. Pragmatic approach to providing the ideal solution

Using the motivations for a novel approach to QoS provisioning from section 4.2, and considering the constraints of the ideal solution from the previous sub-section, a specification for the novel architecture can be produced. The architecture requirements are defined under five headings; functionality, reliability, usability, performance and supportability. These requirements are the result of considering the shortfalls of previous QoS alternatives and the constraints of an ideal QoS solution based in reality, also in conjunction with discussions and meetings with members of the UK ILAB of France Telecom, and are presented in Table 4.1.

Functional Requirements	
F.1	Adjust network QoS policies to reflect the services in use by the user.
F.2	Buffer and schedule outgoing packets according to the inferred user expectations.
F.3	Acknowledge the impact queuing and scheduling techniques may have on application QoS.
F.4	Employ rate control techniques on application traffic if necessary - controlling the ratio of traffic under congested conditions.
F.5	Instruct congestion controlled traffic sources of the need to adjust transmission rates if the network is becoming or has become congested.
F.6	Manage application access to the network ensuring the balance between QoS with fairness is maintained.
F.7	Identify/Infer network conditions through network/traffic monitoring techniques.
Usability Requirements	
U.1	The allocation of resources between end point application traffics reflects changes at the end points, however, remain transparent to the user(s).
Performance Requirements	
P.1	The adjustment of traffic conditioning should be harmonised with changes in application usage by the user.
Supportability Requirements	
S.1	The architecture requires no change to the underlying IP protocols, or application layer functionality.
S.2	Processed packets must be routable by current Internet routers, and conform to Internet Protocol standards (including but not limited to RFC 1122, (IETF 1989)

Table 4.1: Requirements for a Novel QoS Architecture

4.4. Proposed User-centric QoS Provisioning Architecture

The requirements in the previous section provide the specification for the novel architecture, defining a new network layer architecture, capable of providing an adaptive user-centric QoS solution. These requirements were used further to develop an abstract architecture, which can be integrated into existing IP infrastructures, optimising network traffic from the perspective of the user-experience and based upon the current state of the network.

To be able to manage user-traffic with respect to the entire user-experience the architecture must be located at a point within the network where services aggregate together. In order to condition traffic optimally, the architecture should condition the aggregating services prior to them reaching the bottleneck link, which as previously discussed is typically located at the 'contended-by-design' access network. The bottleneck of a network is a much disputed topic, with many factors involved when trying to specify its location. For the purposes of this research it is considered appropriate to state the core uplink from the aggregation node at the edge of the access network as the network bottleneck. This is the location where multiple end users aggregate at a single point, and where the sum of their individual connections exceeds the uplink capacity. The conditioning of traffic at the bottleneck may be complimented with support from subsequent network nodes, i.e. within an ISPs network however, this functionality is not a necessity since the conditioning is primarily achieved at the ingress of the bottleneck.

Figure 4.1 illustrates the intended location of the architecture at the edge of the ISP network, and also gives an expanded view of the logical architecture, where the solid dark arrows represent the flow of packets through the system, and the broken white arrows represent control-plane information within the architecture.

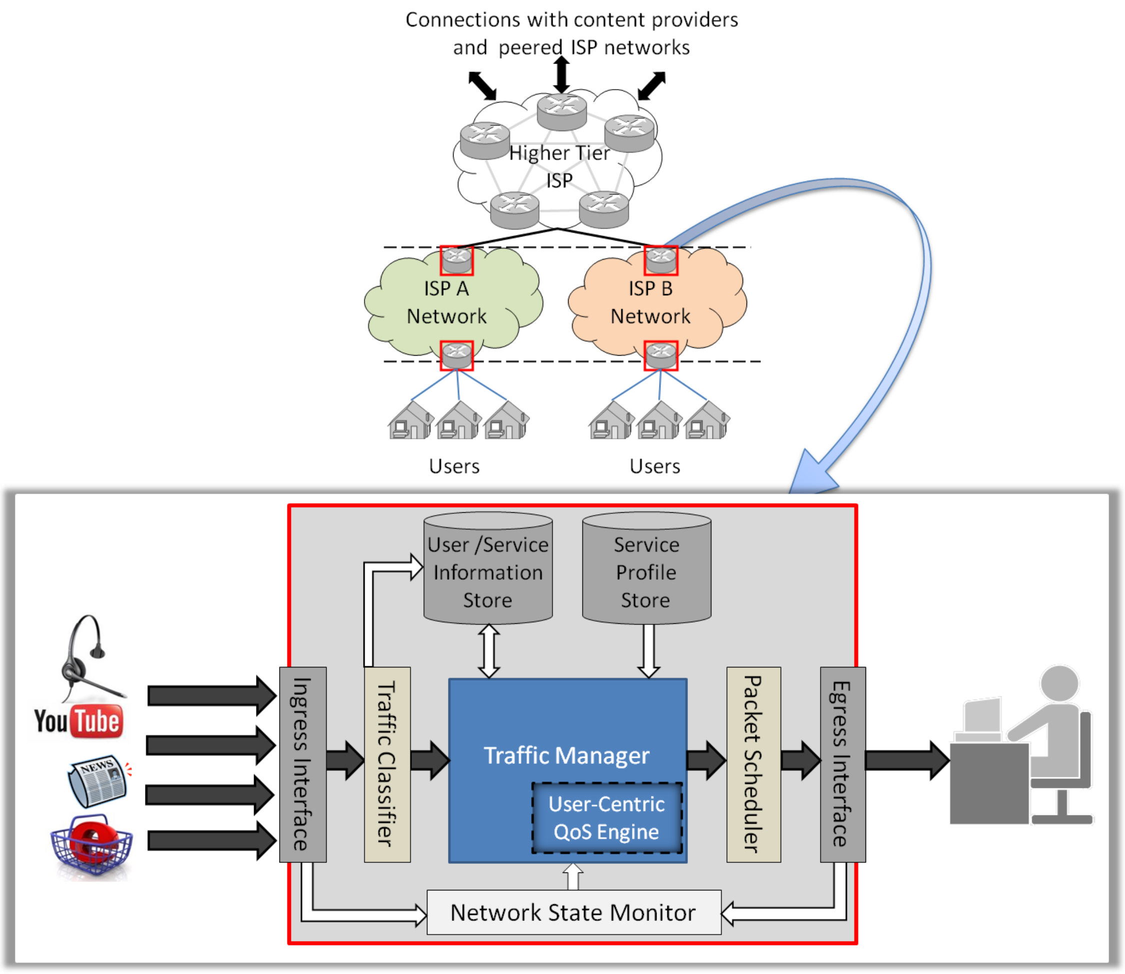


Figure 4.1: Block diagram of the proposed QoS architecture

The following sub-sections provide a description for each of the architectural components, expanding on its functionalities and interactions with the other elements in Figure 4.1. A reference to the associated design requirement (as defined in Table 4.1) is provided where applicable.

4.4.1. Ingress and Egress Interfaces

The ingress and egress interfaces describe the entry and exit points of the architecture for user data. The terms ingress and egress are noted as logical labels, since a single physical interface will act as both ingress and egress depending upon whether the attached host is the source or destination of the data. For packets arriving at the ingress interface they are passed onto the *Traffic Classifier* component. The ingress interface is also responsible for measuring the packet arrival rate of incoming packets and forwarding this control plane information to the *Network State Monitor.* Similarly, the egress interface monitors its current utilisation and forwards this information to the *Network State Monitor*.

4.4.2. Traffic Classifier

The traffic classifier component is responsible for identifying the incoming packets in terms of protocol and application type. This identification can be achieved using a number of techniques, for example identifying traffic coming from known source IP addresses, well known layer 4 port numbers (i.e. 80 = HTTP, HTTPS = 443, 21 = FTP..), or using more complex methods of Deep Packet Inspection (DPI) to identify layer 7 attributes or traffic signatures. For the scope of this research it is sufficient to acknowledge such identification methods exist. The traffic classifier is also responsible for updating the *User / Service Information Store* with a record that describes new incoming services.

4.4.3. User / Service Information Store

This component is responsible for maintaining two data structures, one that describes each service in use (FlowTable), and also an aggregated summary of the traffic for each user (UserTable)[7]. These data structures store information describing service usage and performance, which enable the *Traffic Manager* to evaluate this performance against the current network health. The exact design of these structures will be dependant on the services the architecture is configured to manage. However, an example is provided in Appendix D for VoIP, HTTP, FTP

[7] Throughout the subsequent sections field names from the FlowTable are denoted by the subscript i . Fields from the UserTable are denoted by the subscript j.

and P2P, which were the traffic types used during the validation of the architecture in section 5. As part of the maintenance of these data structures, the *User / Service Information Store* is responsible for periodically scanning the records and removing any data relating to expired services[8].

The final responsibility of this component is to answer the queries of the *Traffic Manager*, providing performance statistics when required, for example average throughput, average delay and packet drop history.

4.4.4. Services Profile Store

The purpose of the services profile store is to maintain a record of the properties for known Internet services. These properties can include expected bitrate, packet sizes, adaptability (i.e. codec adaptation or F.E.C.). The services profile store is responsible for informing the *Traffic Manager* of these properties to enable an informed management decision to be made with regard to quality of service. The service profile store can be updated either with service properties provided by content providers, or through the monitoring of new Internet services. The details surrounding the online, real-time learning of new Internet services is out of the scope of this research, but could provide a promising avenue for future study.

4.4.5. Network State Monitor

The network state monitor plays a critical role in the operation of the proposed architecture. It is responsible for receiving control plane information (e.g. link utilisation and packet arrival rate) from both the *ingress* and *egress interface,* in order to evaluate and report to the *Traffic Manager* the current state of the network. Further information regarding the granularity of the information that is collected and evaluated by this component is provided in section 0.

[8] A service is considered expired if no traffic has been observed beyond a defined expiration period.

4.4.6. Traffic Manager

The traffic manager is the central component in the proposed architecture, responsible for aggregating together the information provided by the *Network State Monitor, User / Service Information Store and Service Profile Store.* This information is used by the *User-centric QoS engine* (part of the traffic manager) to make an informed and dynamic decision about how best to handle incoming traffic. A detailed description of the decision process is provided in section 4.6. Following the decision (and if necessary subsequently action) of the traffic manager, the packet is forwarded to the *Packet Scheduler* component, ready to be queued for the *egress Interface*.

4.4.7. Packet Scheduler

The packet scheduler is responsible for managing the outbound buffer(s), where the conditioned packets await forwarding towards their destination. A number of considerations were made in the design of the queuing and scheduling for the proposed architecture, these are provided in sections 4.9.1 and 4.9.2 respectively.

Having expanded on the overall architecture of the proposed system the next section presents a number of trade-offs that were made during its development. These trade-offs describe the need to balance granularity of the control plane data collected and processed, with the overall system complexity.

4.5. Design Trade-offs

The previously reviewed study on Internet services (Schulze and Mochalski 2009) suggests that a very high proportion of Internet usage is accounted for by a relatively small number of discrete services. Standardisation of application protocols results in a relatively small number of underlying technologies and platforms being responsible for the delivery of the majority of Internet services. The level complexity of a user-centric architecture compared with that of a coarsely granular class based approach is understandably higher, however, observations like those by Schulze can be advantageous to the proposed design. For example, rather than each flow having individual QoS requirements (as per the design of Intserv), the new architecture can cross-reference a store of common service profiles. Once a matching profile is identified, the QoS requirements for the flow can be interpreted.

Taking Video on Demand (VoD) as an example application, there are a finite number of commonly used bitrates (as presented in Section 3.3). The novel QoS management architecture can store a profile for each of these bitrates (also including information such as adaptable codec), which can then be used to inform the traffic management component how a particular flow should be handled.

The *User / Service Information Store* and the *Network State Monitor* are both responsible for the storing and collection of information on network traffic and conditions, which enables the *Traffic Manager* to optimise user-traffic given the current conditions. This data collection is conducted in a passive manner, meaning the data is collected through monitoring incoming traffic flows, and measuring interface utilisation. A more active method of information collection could have been used for example; injecting test flows into the network between dedicated devices may have offered an increase in the accuracy and detail of the information collected. However, would have also meant that the traffic from which performance information was based upon was not real user-data.

The *User / Service Information Store* maintains a record of the services that are in use by each user. The exact information held for each user's services will vary depending upon the individual service, but example fields may include a flow ID, source and destination IP addresses, application type, average throughput and VoIP quality rating (more detailed consideration of the data required for each service is given later in this chapter). The decision to maintain information about traffic flows and users raised scalability concerns in terms of the volume and granularity of the information being recorded. However, it was concluded that any shift from a coarse class-based QoS architecture towards the desired user-centric approach would introduce an intrinsic requirement to monitor and store characteristics for flows and applications in a stateful manner. The limits in terms of how far the architecture could scale were decided to be reserved for future research, given that such limits are likely to be bound by hardware (ASIC) capabilities. Incidentally, it is believed that although this is an area for future consideration, the stateful storing of information about Internet traffic flows would not fundamentally inhibit the success of the architecture. This belief is based upon the fact that there are a number of technologies and network appliances that rely upon the collection (and in some cases interaction) with live Internet flows, for example, Intrusion Detection / Prevention Systems, Cisco's Netflow accounting feature (Cisco Systems Inc. 2010) and server load-balancing devices.

One alternative active approach could have been to modify either the end-user system (i.e. the operating system) or by integrating cooperation with the gateway device of the user's network. For example, the operating system could be modified to send periodic summaries of the services currently in use by the user to the QoS architecture within the network. While this method may have provided a description of the applications in use without monitoring and inspection at the edge node, a number of limitations out-weighed this benefit. Firstly, this approach would require modification to the operating system, which as discussed throughout this thesis presents a significant deployment issue. In addition, as with any method of active measurement, this approach would inject additional data onto the network, which during periods of high utilisation would add to the problem of congestion. Furthermore, using information provided by external nodes opens the QoS system to the possibility of receiving rogue control information, which could be used against the system, disrupting the management of user-traffic.

The *Network State Monitor* could also increase the variety and accuracy of the data it collects by actively engaging with other network nodes. For example, rather than modifying the end-user's system, simply installing a piece of software on their system that could be probed periodically to provide information such as bottleneck bandwidth, delay, and loss statistics of the network path. However, once again this active approach of measuring the network condition injects additional data onto the network, and only describes the path taken by that specific flow, which is unlikely to be indicative of the conditions experienced by all other flows.

With regard to measuring and inferring the network health many trade-offs must be made in terms of the accuracy and proportion of the network that is measured. The more knowledge about the network that is known, the more complex the architecture becomes, and the less scalable the solution. Therefore, given that it was assumed that the access network is the most likely location for congestion, and that the routers at the edge of the ISP network can be informed of link capacities, the method of determining link/network utilisation was to compare the arrival rate of the incoming packets to the capacity of the egress interfaces.

Although this provides a very limited perspective of the network health investigating the benefit that using additional network parameters could have on the efficiency of the *Traffic Manager* is reserved for future work.

This section has highlighted the compromises that were needed to realise the specification given in section 4.3.2, and develop it into a feasible architecture, capable of delivering user-centric QoS. The following section expands on the design and operation of the traffic management component. The section describes the performance metrics used in decision process of the traffic manager and defines the how each traffic type is controlling and conditioned.

4.6. Design of the Traffic Manager

The novel architecture is required to optimise the QoS for each user service, rather than simply providing precedence to specific traffic types, with the aim to consider the entire user-experience. Given that bandwidth is a finite resource and it is not possible to create additional bandwidth, only reduce the amount a service uses, the role of the *Traffic Manager* is to employ the most appropriate control mechanism to each service, ensuring any degradation of quality is distributed across all services, in a manner that minimises the overall impact on the user-experience.

In the first instance the architecture ensures the bottleneck capacity is shared equally among users, ensuring per-user fairness rather than favouring premium service users. This allocation is described by equation 10, where C is the bottleneck capacity and $r_{(i)AS}$ is the bottleneck entitlement for user *i*, given *n* users. α denotes a weighting function that reflects the subscription plan of user *i*, however, from here on it shall be assumed all users have an equal subscription, and α omitted.

$$\sum_{i=1}^{n} \alpha . r_{(i)_{AS}} = C \tag{10}$$

Equation 10 gave $r_{(i)AS}$ as the fair share entitlement of the bottleneck bandwidth for user *i*, which assumes each user will utilise all of their share of the bandwidth. In the (likely) case that every user does not saturate their connection, the bandwidth available to user *i* can be described as an equal share of the bottleneck capacity, C_{total} plus an equal share of any unused bandwidth from other users, $C_{residual}$.

$$r_{(i)_{AS\_max}} = \frac{C_{total}}{n} + \frac{C_{residual}}{n} \tag{11}$$

This approach of resource allocation is similar to the per-aggregate Assured Forwarding in Diffserv networks, in the sense that it guarantees a proportion of the bandwidth to a flow aggregate (in this case the traffic for a single user). However, the novelty of the architecture comes from if the network becomes congested and $r_{(i)} > r_{(i)AS}$. Under these conditions the *Traffic Manager* reduces $r_{(i)}$ so that $r_{(i)} \leq r_{(i)AS}$, by combining the information about the services a user is currently engaged in (stored in the *User / Service Information Store)* with the requirements and characteristics of each service (stored in the *Service Profile Store)*, to provide an application-aware, user-centric traffic management system.

There are two approaches to managing network traffic, loss management and delay management; for example, if one wishes to improve the quality of a service then packets for that service can either be given queuing priority over concurrent services (delay management), or packets from other services can be dropped (loss management).

The following sections outline the traffic management policies for a variable mix of web browsing, single source file downloads (via FTP/HTTP), VoIP, streaming video, and P2P file transfers. While this list does not encompass every Internet service, these services are responsible for over 90% of all Internet traffic (Schulze and Mochalski 2009) and are therefore considered sufficient to demonstrate the principles of the architecture. Moreover, the *Service Profile Store* enables the architecture to be updated to recognise the latest protocols and applications. Following this outline of the management policies, the next chapter describes how the novel architecture could be integrated into existing network infrastructures, and the mechanisms required to implement and enforce these policies.

4.6.1. Traffic Management of UDP-based Applications

As introduced in section 2.4.1 the transmission rate of an application using UDP is not affected by loss or delay, however, it has also been highlighted that typical services that use UDP, for example VoIP, are highly sensitive to loss and delay. Therefore, it is the responsibility of the *Traffic Manager* to maintain such services at a point of equilibrium between network health and acceptable QoS. Of the services studied in Chapter 3 (VoIP, web browsing, streaming media and P2P), only web browsing and VoIP were observed to send packets over UDP. For web browsing activities, the UDP traffic refers to DNS resolutions that are performed in order to acquire the IP address for the desired web server. While this process does not involve the transmission of actual webpage-data, the process is critical to the retrieval of a webpage, and on account of this packets associated with this process are protected as part of the browsing experience.

4.6.1.1. Management of VoIP traffic

The traditional approach used for QoS provisioning for VoIP traffic is to allocate the required amount of bandwidth to accommodate the voice flow(s), and implement priority queuing to ensure delay is kept to a minimum. While this is one possible method, the *Traffic Manager* must be able to optimise the use of the available bandwidth, which may require compromising the proportion of bandwidth VoIP traffic is allocated. In section 2.2, the E-model was introduced as providing one method of evaluating the quality of a VoIP call, its *R-Factor* - derived using quantitative measurements (including packet loss and one way delay). Only when a VoIP call has an R-Factor ≤ 70 does user dissatisfaction being to occur, therefore it is proposed that during congestion the *Traffic Manager* aims not to reserve a fixed amount of bandwidth for a VoIP flow, but rather consider this quality rating, and aim to maintain an R-Factor ≥ 75.

The reduction of the E-Model provided by (Cole and Rosenbluth 2001) is given in equation **12**, where Ie (**13**) and Id (**14**) are the impairment factors for packet loss and delay respectively. In equation **13** λ_1, λ_2 and λ_3 are codec specific parameters derived by Cole and Rosenbluth, and e represents the current loss rate of the flow. In equation **14**, the One Way Delay[9] is given by d and $H(x)$ is a step function where $H(x) = 0$ if $x < 0$ or $H(x) = 1$ if $x \geq 0$.

[9] The One Way Delay is assumed to be equal to half the estimated RTT.

$$\mathrm{R} = 94.2 - \mathrm{Ie} - \mathrm{Id} \tag{12}$$

$$Ie = \lambda_1 + \lambda_2 \ln(1 + \lambda_3 \mathrm{e}) \tag{13}$$

$$Id = 0.024d + 0.11(d - 177.3)H(d - 177.3) \tag{14}$$

Figure 4.2 shows the decreasing R-factor of a G.711 VoIP flow as a function of the experienced packet loss and delay. Two distinct gradients can be seen in Figure 4.2, the first where delay < 177ms and the other where delay > 177ms, which is attributed to the step function used in equation **14**.

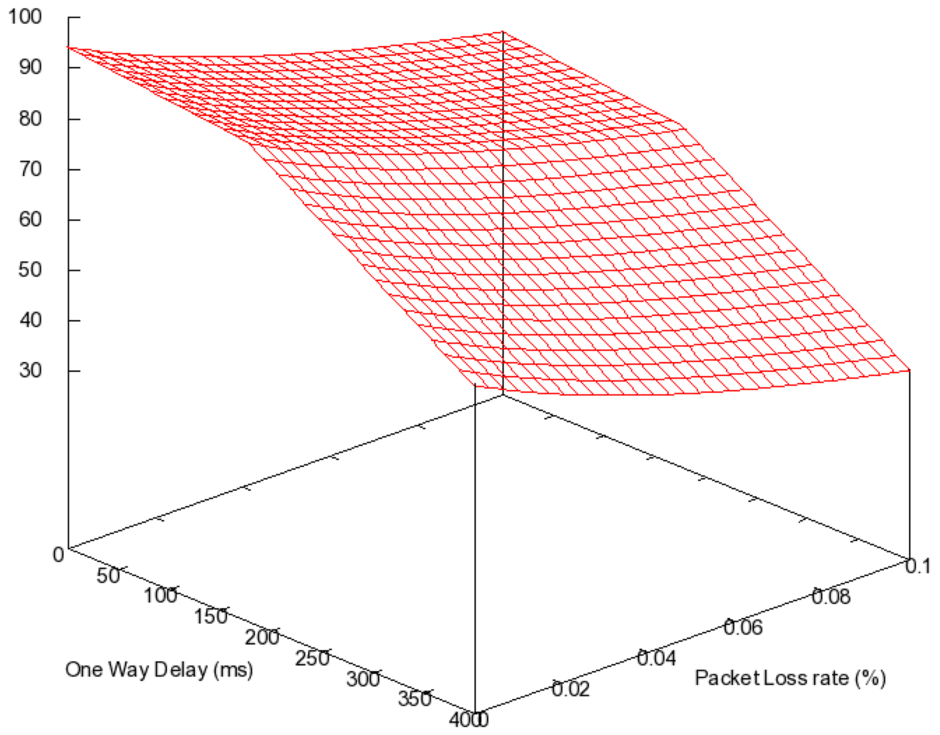


Figure 4.2: 3D plot showing decreasing VoIP quality (R-factor z-axis) as a function of delay and packet loss

Integrating the E-Model into the *Traffic Manager* is possible using two methods. The first and recommend method is by using RTCP Extended Reports (RTCP XR) (Friedman, Caceres et al. 2003), an extension of the standard RTCP functionality, (introduced in section 2.5.1). RTCP XR include a number of metrics that enable the evaluation of a VoIP call, including *loss rate, round trip delay, signal level, noise level, MOS values* and *E-Model's R-Factor*. As a VoIP flow supporting RTCP XR traverses the *Traffic Manager* the flow's performance statistics are recorded by the *Traffic Manager* and determine how packets from that flow should be treated as a function of the current QoS for the call.

If RTCP XR is not implemented, then inferring the QoS for a VoIP requires additional computation within the architecture to determine the *packet loss rate*, *one way delay* and *voice codec*– this computation is performed by the *VoIP QoS Engine*, a sub-component of the *Traffic Manager*.

The *packet loss* and *voice codec* can be easily determined using information carried within standard RTCP Sender Reports (SR) and the RTP header. Each SR_{i+1} provides the *fraction of packets lost* since SR_i was generated, and the voice codec is described by the *Payload type* field of the RTP header. However, standard RTCP reports do not provide sufficient information to determine the RTT between VoIP call participants. Despite each RTCP report having a *Timestamp* field this cannot be used to calculate the RTT since the starting point for these timestamps is chosen randomly by each RTP participant at the start of the session, which makes calculating the RTT since the timestamping impossible. Furthermore, the receipt of an RTCP SR or RR does not trigger an instant response, and so no method exists to observe the delay between a request and response to infer a flow's RTT (as can be used as a method of inferring the RTT for a TCP flow, later described in section 4.6.2.2). To this end, inferring the RTT between VoIP participants cannot be achieved passively, and therefore there is a requirement to periodically probe the participants of the VoIP call, using ICMP echo-request packets, measuring the response time from the novel architecture within the network to both VoIP parties.

Using this method to obtain an estimate for the *one way delay,* and the *packet loss* and *codec* determined from the RTCP and RTP packets, the *Traffic Manager* can calculate the current R-factor for the VoIP call. Since the *fraction of lost packets* can be misleading (1 drop from a total of 5 packets is less significant than 200 drops from a total of 1000), the significance is determined using the *cumulative packets sent* value, also in the SR, allowing the *Traffic Manager* to gauge the severity of the loss.

A moving R-factor is calculated for the previous 10 seconds, to provide insight into the recent call quality. It was considered that *10* seconds was enough of a period to evaluate given that events that happened beyond *n* seconds ago in a real-time service bear little impact on the current QoS, however this was balanced with a need to assess a period long enough to ensure a level of control. The moving calculation is performed on a time-basis rather than a per-packet one, given that the average will decay at the same rate for every VoIP flow, irrespective of its sending rate.

This integration of the E-Model allows the *Traffic Manager* to use the calculated current R-factor, current packet loss and current one way delay values in the decision process for how each VoIP flow should be treated. This permits for an intelligent management of VoIP packets with an understanding of the impact that further queuing or dropping will have on the perceived QoS, avoiding the traditional static reservation of resources that is typically implemented.

Figure 4.3 provides the pseudo-code for the management of VoIP traffic by the *Traffic Manager.* Initially the current utilisation of the bottleneck is used to determine whether or not the network is congested. During periods of no congestion the R-factor is evaluated to ensure the VoIP flow is above an acceptable level (recommended to be an R-factor ≥ 75). In the event that the R-factor is lower than the acceptable limit, the *Traffic Manager* queries the packet loss for the VoIP flow to determine whether the flow has experienced excessive loss, in which case it can be assumed there is congestion elsewhere on the end-to-end path between the VoIP parties, and the *Traffic Manager* aims to alleviate this by adapting its management of other services in use by the user. If the packet loss is acceptable but the one way delay is greater than a delay threshold the *Traffic Manager* will increase the priority given to the VoIP packets in an effort to reduce the delay and increase the R-factor above 75. In an uncongested state, where the

VoIP flow has an R-factor greater than or equal to 75 the VoIP packets are simply forwarded under normal operating conditions.

If the bottleneck utilisation indicates that there is congestion the *Traffic Manager* first evaluates the current R-factor for the VoIP flow. For cases when the R-factor is less than the lower VoIP threshold (e.g. R-factor = 75), other concurrent services for the user in question are conditioned more intensively to make available additional resources, in an effort to increase the R-factor. For cases when the bottleneck is congested but the VoIP flow is maintaining an R-factor greater than a specified upper threshold (e.g. R-factor = 80) then the *Traffic Manager* calculates the packet loss budget using an inverse E-Model, which is derived from the standard equation $\mathrm{R} = 94.2 - \mathrm{Ie} - \mathrm{Id}$ factoring in either the current *packet loss* or *delay* with a *target R-factor* to determine the upper limit for either *packet loss* or *delay* respectively.

```
Inputs: CurrentUtil, UtilThreshold, LowerVoIPThreshold, UpperVoIPThreshold, DelayThreshold, LossThreshold,
R-factor_i, PacketLoss_i, OWD_i,

if CurrentUtil < UtilThreshold
        then
            if (R-factor_i ≤ LowerVoIPThreshold) AND (PacketLoss_i > LossThreshold) AND (OWD_i < DelayThreshold)
                then reduce priorities of other services from this user & forward VoIP packets as normal
            else if (R-factor_i ≤ LowerVoIPThreshold) AND (PacketLoss_i < LossThreshold) AND (OWD_i > DelayThreshold)
                then increase VoIP packets priority
            else if (R-factor_i >= LowerVoIPThreshold)
                then forward VoIP packets as normal
else
            if (R-factor_i <= LowerVoIPThreshold )
                then increase VoIP packets priority and reduce priorities of other services from this user
            else if (R-factor_i > UpperVoIPThreshold)
                then calculate LossBudget_i
                        Drop packets based on LossBudget_i
```

Figure 4.3: Pseudo-code for the novel management of VoIP traffic by the Traffic manager

$$R\text{-}factor = F(packet\ loss, one\ way\ delay, codec\ parameters)\mathrm{R} = 94.2 - \mathrm{Ie} - \mathrm{Id} \qquad \mathbf{15}$$

$$loss\_budget = F(Target\ R\text{-}factor,\ one\ way\ delay,\ codec\ parameters)\mathrm{Ie} = \lambda_1 + \lambda_2 \ln(1 + \lambda_3 \mathrm{e}) \qquad \mathbf{16}$$

$$delay\_budget = F(Target\ R\text{-}factor,\ packet\ loss,\ codec\ parameters)Id = 0.024d + 0.11(d - 177.3)H(d - 177.3) \qquad \mathbf{17}$$

The calculations for the *loss_budget* or *delay_budget* can be performed using one of two methods. The *Traffic Manager* can either calculate periodically for each VoIP flow an exact figure using the inverse E-Models given in **16** and **17**, or a less computationally intensive method is to maintain a profile for a range of popular VoIP codecs, which describes the loss/delay budgets required to achieve the desired R-factor. Table 4.2 provides an example of such a profile for the G.711 codec to achieve an R-factor of 75.

Packet Loss Experienced (%)	One Way Delay bounds (ms)
0-1%	285 - 256
1-2%	256 – 230
2-3%	230 – 205
3-4%	205 - 183

Table 4.2: Delay bounds for given varying degrees of packet loss, while maintaining an R-factor of 75

The study into Internet services given in Chapter 3 revealed that many of the most popular VoIP clients employ encryption techniques often on proprietary protocols, thus restricting ability to evaluate call quality. However, also presented in section 3.5 were methods to infer packet loss and additional delay by analysing variation in the packet arrival rate and packet payload size. (Huang, Huang et al. 2010) studied a number of Skype codecs under varying loss conditions, observing adjustments made by the Skype application to the bitrate and payload size, Figure 4.4 shows their results (the red line represents packet loss ranging from 0-10% in 1% increments increasing every 180seconds). Huang et al. observed that each of the three codecs reacts to an increase in packet loss by increasing the packet size and also the bitrate, indicating the use of Forward Error Correction techniques. For example, the iSAC codec sends packets with a payload between 50-150 bytes for up to 4% packet loss, beyond which the payload is increased to between 150-300 bytes. The bitrate lies within the range of 20-40kbps for up to 3% packet loss, increasing to 40-50kbps at 4% loss, and further increasing to 50-70kbps for loss above 5%. While these observations alone do not provide indication of the QoS, by way of an R-factor, they can be used by the *Traffic Manager* as an indication of congestion, and a method of inferring the loss characteristics for the VoIP flow.

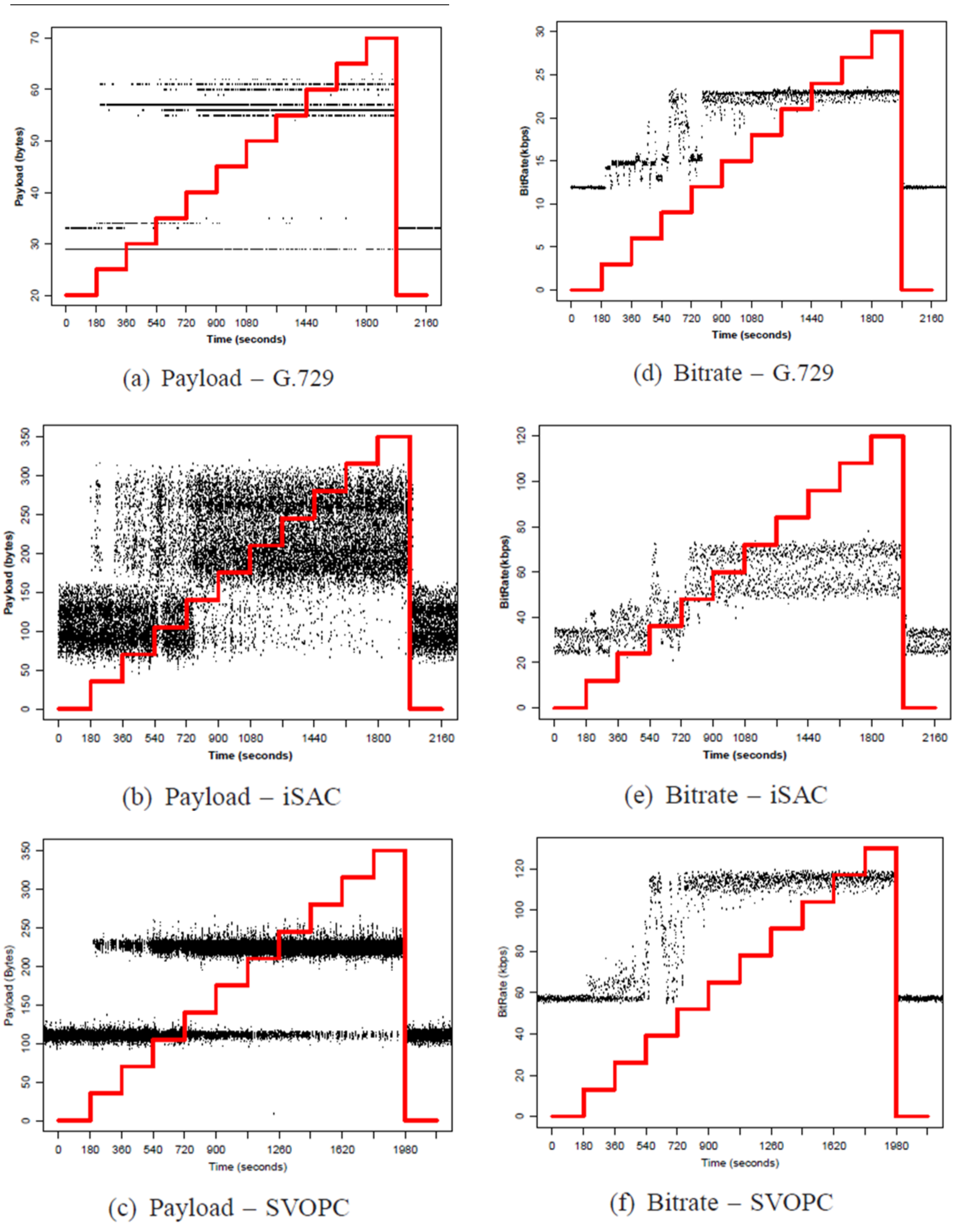


Figure 4.4: The impact of increasing loss rate on the payload size and bit rate of Skype packets using three different codecs. (Image source: (Huang, Huang et al. 2010))

While being able to infer the approximate packet loss or delay for an encrypted VoIP flow is a step closer to dynamic QoS management, the ability to integrate a more comprehensive assessment of the call quality (e.g. The ITU-T E-Model) would be more desirable. One solution would be for the codec developers to release call quality guidelines that describe the performance of their protocol under varying loss conditions. For example, Skype provides two graphs that illustrate the MOS score for a VoIP call using the SILK codec at varying bitrates and under different packet loss conditions (Figure 4.5). This information could be combined with the method of inferring loss for a Skype flow to estimate the QoS of a call, which could subsequently be used by the *Traffic Manager*.

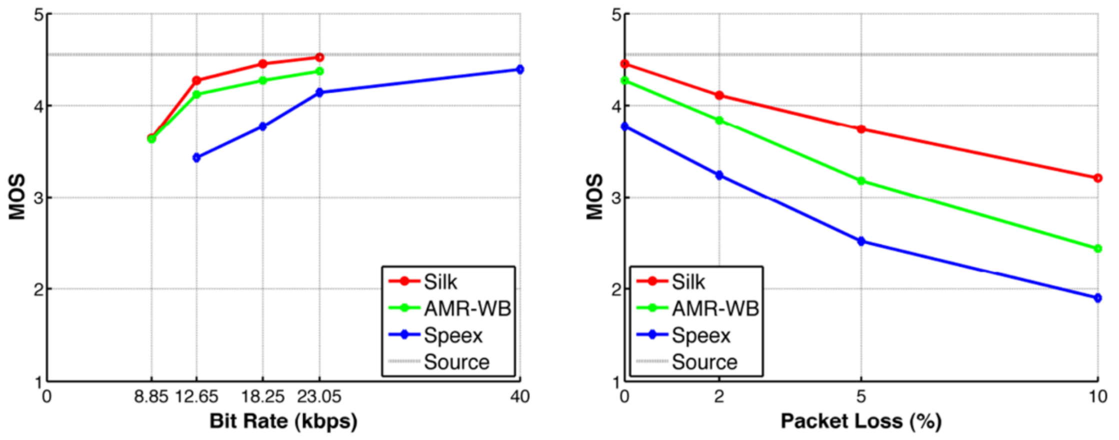


Figure 4.5: Mean Opinion Scores for VoIP codecs for varying bitrates and packet loss (Image Source: (Skype 2010))

4.6.2. Traffic Management of TCP-based Applications

Previous methods of QoS provisioning for TCP flows were reviewed in sesction 2.3.3.1, which primarily focused on improving the fairness of resource allocation between heterogeneous flows, by managing the throughput of flows or aggregates. While each of the following sections describe in detail the management for a number of TCP-based services, the fundamental metric being controlled is throughput. Once again a compromise must be made between the degree of accuracy the throughput of a flow is controlled with, and the resulting system complexity. At one extreme, a full TCP model can be integrated into the *Traffic Manager*, enabling accurate calculations of the loss and delay budgets to condition a TCP flow to achieve a precise target throughput, a technique similar to that reviewed in section 2.3.3.1 by (El-Gendy and Shin 2003).

However, as highlighted in the earlier review, this approach requires the estimation of TCP variables, including RTT, RTO, *ssthresh* and the Receiver window, severely adding to the system complexity, with *ssthresh* and the Receiver window not being able to be calculated without end-point cooperation. Furthermore, this approach only models steady-state TCP-Reno flows (assuming the well known ‘Padhye’ model is used (Padhye, Firoiu et al. 2000)), this approach was considered to be too complex, while still incomplete as it did not encompass all variants of TCP, nor does it begin to take into consideration the diverse application layer requirements of modern Internet services. To this end, the *Traffic Manager* chooses arguably less accurate methods of controlling the throughput of a flow, but instead, combines features from TCP-aware packet handling (Mellia, Stoica et al. 2003) with application-layer requirements.

The operations of the *Traffic Manager* for Streaming video, HTTP, FTP and Peer-to-Peer are described in the following sections.

4.6.2.1.Management of Streaming Video traffic

Streaming video, although not strictly a real-time service does have an explicit requirement for a minimum level of service beyond which the user-experience is noticeably impaired. Two primary delivery techniques for streaming video were identified in Chapter 3, pseudo-streaming and traditional streaming, the key difference being the former allows video data to be transmitted at a rate greater than the playback speed, utilising temporary caching and buffering of the entire video on the receiver host, whereas traditional streaming may buffer only a few seconds worth of video at a time. Fundamentally, the throughput of the video stream is the critical factor that determines the QoS perceived by the user - less than the bitrate of the video and playback will be intermittent while sufficient data is buffered to allow playback to commence. While this delivery style provides more resilience against adverse network conditions, (Lei, Songqing et al. 2005) suggest that 87% of all streaming media is abandoned by the user within the first 10seconds, wasting up to 20% of the server’s bandwidth; which could be allocated to other services during periods of congestion. It was discussed in Chapter 3 that YouTube servers have been observed to limit the maximum throughput of certain video streams (Rao, Legout et al. 2011), however this behaviour cannot be assumed across all platforms or content providers. Furthermore, there exists no known mechanism that ensures a video stream receives sufficient Internet bandwidth to operate smoothly and without interruption, and for these reasons the following proposal is provided. This section describes how the *Traffic Manager* manages both pseudo-streaming and traditional video streams to ensure an acceptable level of service is delivered to the user.

Introduced in Chapter 3, YouTube is undoubtedly the most popular online video repository, primarily using pseudo-streaming delivery for its videos (traditionally streamed videos via YouTube are considered later in this section). When uploading a video to YouTube the source video is encoded into a number of formats, which provide different bit rates for the viewer. The video quality requested by the user can be obtained within the network by analysing the *Request URI* field, within the HTTP GET request packet, and identifying the *fmt* argument (an example capture of this field is provided below, taken from a Wireshark capture from July 2010).

```
HTTP GET /get_video?video_id=yQ5U8suTUw0
                  &t=vjVQa1PpcFOryuogR5j6U8OEtSozhbUS1cfQYNDq1RI=
                  &el=detailpage
                  &ps=
                  &fmt=37
                  &asv=2
                  &noflv=1
```

Using the value of *fmt* from the HTTP GET Request, combined with a Service Profile for YouTube videos (stored in the *Service Profile Store*), the *Traffic Manager* knows the typical bandwidth requirements for a video of the desired quality. Knowing the approximate bitrate of the video, the *Traffic Manager* ensures that during a period of congestion, the TCP connection for the video stream maintains a throughput as close to the required bitrate, providing smooth playback while limiting any buffering to provide other services access to resources. Table 4.3

provides an example of a Service Profile for YouTube based on the characteristics collected in section 3.3.

	Video Format				
	240p	**360p**	**480p**	**720p**	**1080p**
fmt value	5	34	35	22	37
Approximate bitrate ranges (kbps)	200-300	300-600	600-1200	1100-1800	>1800

Table 4.3: YouTube video formats and associated bitrate

To further improve the treatment of streaming video data, the *Traffic Manager* can also determine the size, in bytes, of the video, using the *content length* field in the packet sent by the server in response to the GET request. This value can be stored in the *User / Service Information Store* in conjunction with a cumulative sum of the packet sizes to date, which enables the *Traffic Manager* to determine the percentage of the video that has been transferred to-date. This information can then be used if a period of low utilisation occurs, determining if a short acceleration of the transmission rate would complete the transfer, freeing network resources for subsequent services.

When addressing the management of traditional streaming video, client-side caching is minimal, therefore cannot be relied on. However, from the perspective of QoS provisioning, traditional methods of video streaming allow for a simpler style of management. Based on the findings from section 3.3, traditional streaming is favoured by broadcasters over pseudo-streaming for the delivery of television programmes. The method of management for these video streams involves identifying the quality (bitrate) of the requested video, using the same method of packet inspection described earlier in this section for standard YouTube videos, and ensuring the video stream achieves a throughput approximately equal to the video bitrate. It is believed based on observations of 30 YouTube videos that the *asv* argument refers to the streaming delivery method to be used, where *asv=2* and *asv=3* relate to pseudo-streaming and traditional streaming respectively.

```
HTTP GET /get_video?noflv=1
             &t=vjVQa1PpcFPupTQhBdOOT_UKHm6U1sqXhesyZHooyew=
             &fmt=34
             &asv=3
             &video_id=4bUKGqNjPFk
             &el=detailpage
```

In contrast to the streaming services currently provided by YouTube, ITV and Sky, the BBC iPlayer features additional functionality that enables it to detect the connection speed of a user, and if necessary lower the quality of the video to ensure a smooth playback is maintained. During periods of prolonged congestion the *Traffic Manager* exploits this server-side quality adaptation, along with the known bitrates for the different qualities support (provided by the *Service Profile Store*) to restrict the throughput of the media stream to a lower alternative. This provides the user with continuous viewing, while also aiding towards easing network congestion by reducing the bandwidth requirements of their service.

```
    Inputs: CurrentUtil, UtilThreshold, DataDelivered_i, PacketSize, LowerBitrate,
1   RequiredBitrate ← Retrieved using HTTP GET analysis
2   ContentLength ← Retrieved using HTTP GET analysis
3   $DataDelivered_i = DataDelivered_i + PacketSize_i$
4   if $CurrentUtil > UtilThreshold$
5       Switch (Streaming method)
6           case 1: pseudo-streaming
7               Maintain $rate_i \approx RequiredBitrate$
8           case 2:  traditional-streaming but not adaptive bitrate
9               Limit/maintain $rate_i \approx RequiredBitrate$
10          case 3: traditional-streaming and adaptive bitrate
11              Reduce $rate_i \approx LowerBitrate$
12  else if $CurrentUtil < UtilThreshold$
13      then
14          if ($rate_i < RequiredBitrate$)
15              then increase video priority and reduce priorities of other services for this user
16          else if pseudo-streaming
17              Calculate $CompletionTime = F(ContentLength, RequiredBitrate, DataDelivered)$
18              if $CompletionTime < AcceleratedCompletionThreshold$
19                  then expedite packet to accelerate download
20              Else Forward packets as normal
21          Else Forward packets as normal
```

Figure 4.6: Pseudo-code for the novel management of Streaming Video traffic by the Traffic manager

4.6.2.2. Management of HTTP/FTP traffic (Web browsing and File Transfers)

The QoS perceived by a user engaged in web browsing was described as being the culmination of many parameters, (section 2.2.1). However, since traffic management techniques are unable to address aesthetic or usability issues, the focus is limited to ensuring the timeliness of the data delivery process. A method of TCP-aware packet handling is presented by (Mellia, Stoica et al. 2003), which provides protection to TCP flows during vulnerable periods (*slow-start* and *Fast Recovery*). Mellia proposes that the first 5 packets of a flow should be protected, to assist the flow in exiting the sensitive *slow start* phase, and a number of packets protected following the detection of a loss event, to assist a flow during *Fast Recovery*. The novel architecture extends this protective method of packet handling, to better reflect the characteristics of HTTP traffic that were observed in section 3.2. Rather than protecting the first 5 packets, it is proposed that the first 12 packets should be protected, as a reflection of the average number of packets per TCP connection for web browsing.

In addition to this adjustment, Mellia's proposal is further extended to be able to distinguish between a file transfer via HTTP, and a legitimate HTTP v1.1 connection, where the *persistent connection* mechanism may result in a number of sequential, but discrete transactions over a single TCP connection. In both these cases the number of packets is likely to exceed the 12-packet threshold associated with short HTTP flows ($packet\_count_i > HTTPflowsize$). To this end, the *Traffic Manager* records the time that the last packet for each flow was observed, to identify idle periods between HTTP v1.1 packet bursts, which do not appear in the continual stream of packets when downloading a file. At this stage the value of *idle_threshold* has not been considered, (given that the implemented prototype does not consider HTTP flows of this nature), and so remains for future investigation.

In the case of the file downloads via HTTP/FTP, the user-perceived QoS solely reflects the time taken for the file transfer to complete, which is dependent on TCP maximising a steady throughput, with minimal packet losses or excessive delay. For multiple TCP connections to maximise their throughput there must be a degree of fairness enforced, which alleviates the problem of heterogeneous TCP flows achieving different throughputs (reviewed in section 2.2.2.2). Therefore, in addition to protecting a number of packets at the start of the flow (to allow exit from *slow start*) the *Traffic Manager* also aims to provide fairness between heterogeneous TCP flows, by using the equation given by 18 and subsequently dividing this value by the number of TCP flows currently established.

$$\begin{array}{c}\textit{Bandwidth}\\ \textit{available for}\\ \textit{TCP services}\end{array} = r_{(i)} - (VoIP\_throughput + Streaming\_throughput + HTTP\_throughput)^{10} \quad \textbf{18}$$

On the arrival of a packet the average arrival rate for that packet's TCP flow is calculated and compared with the fair share, and if found to exceed this fair share the *Traffic Manager* queues the arriving packet with a higher drop probability, with the aim that if the queue is highly congested the packet will be dropped and TCP will reduce its throughput. This behaviour is similar to the Assured Forwarding PHB, however, it addresses the previous shortfall of AF marking not operating on the same timescale as TCP. Traditionally AF operates on a per-packet basis, despite TCP reacting to network conditions on a per-RTT basis. To overcome this the *Traffic Manager* periodically estimates the RTT for long-lived TCP flows and waits 2xRTT between induced packet drops, which gives TCP opportunity to respond to the loss and reduce its throughput without unnecessarily dropping packets which have already been transmitted.

The RTT for the flow is estimated by recording the sequence number, TCP-header timestamp (t_1) and current time (t_2) of the arriving TCP packet. The sequence number for each subsequent ACK is compared with the recorded sequence number, and if greater the TCP-header timestamp (t_3) and current time (t_4) of the ACK is recorded. Subtracting the arrival times from the

[10] The calculation for bandwidth available to long-lived TCP flows is based upon the assumptions for concurrent Internet services, provided in section 4.7.

timestamps within the TCP-headers provides an estimate of the time taken to travel from either the server or client to the *Traffic Manager*. The sum of these times can be multiplied by 2 to estimate the RTT for that packet, which can then be used to calculate the average RTT (SRTT) using the same equation that is used by the TCP sender **19**. This process is illustrated in Figure 4.7.

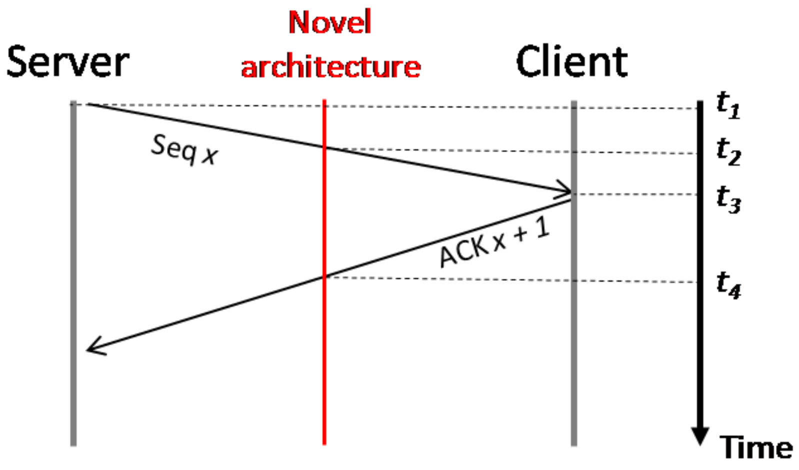


Figure 4.7: Process of passively estimating the RTT for a TCP flow

$$RTT = 2.\left((t_2 - t_1) + (t_4 - t_3)\right) \tag{19}$$

$$SRTT = (1-\propto).SRTT + (\propto.RTT) \tag{20}$$

Given that TCP flows with short-RTTs can achieve a higher throughput than a flow with a larger RTT (due to the rate *cwnd* is increased), the per-RTT management of TCP flows aims to take one step towards alleviating this, in the sense that the rate at which a flow is actively managed is a function of its RTT as well as packet arrival rate.

The Traffic Manager provides further control over TCP flows by considering its *congestion history*. This term refers to recording the behaviour of a flow for a specified period of time. For example, each time a flow is policed in response to it exceeding its fair share of the available bandwidth, the *Traffic Manager* records this policing event, providing an insight into how responsive/well-behaved a flow has been over time. This information is then used to determine the severity of any subsequent policing that is required, i.e. a flow that has repeatedly exceeded its fair share of bandwidth will be policed more aggressively. Similarly, as a flow responds to being policed, its *congestion history* will be updated to reflect this, thus calming subsequent policing.

4.6.2.3. Management of Peer-to-Peer traffic

The QoS of Peer-to-Peer downloading is very similar to HTTP/FTP file downloads, in the sense that the user's primary desire is to complete the file transfer as quickly as possible. However, rather than downloading from a single source, pieces of the file are acquired from many peers, of which the four fastest peers are also uploaded to in return (reciprocation). The nature of P2P allows the application to obtain a disproportionate share of the network bandwidth, and therefore, the *Traffic Manager* aggregates the throughput of all P2P flows (for a user), and aims to achieve a fair share of bandwidth between applications, rather than between flows. For example, if a file is being downloaded via HTTP and via P2P then it is the goal of the *Traffic Manager* that each application should receive an equal share of the bandwidth, not each TCP flow.

The key finding from the study presented in section 3.4 on the BitTorrent protocol was that despite a BitTorrent application establishing on average 60-80 TCP connections with other peers, only a very small percentage of these connections were responsible for the delivery of the majority of data. In fact, less than 10% of the connections transferred more than 5MB of data. From a management perspective it is imperative that the *Traffic Manager* control each flows proportionally to its achieved throughput. Furthermore, there is no benefit in terms of QoS from dropping packets containing control information sent between peers, such as *keepalives*, since these packets do not impact the throughput of the flows responsible for obtaining large amount of the bandwidth.

The reciprocal nature of P2P can be exploited to maximise the performance of the BitTorrent network, while also conserving bandwidth for additional services. If a user limits the rate they upload to other peers, then they risk being *choked* by the peer and impairing their own download rate. However, the client maintains more than just reciprocal connections with peers, and opens connections with other peers willing to provide BitTorrent pieces without them having to upload and data in return. Since limiting the upload rates of reciprocal connections may lead to impairing two peers QoS (the client and also the remote peer), it is recommended the *Traffic Manager* limits the "gratuitous" connections, where the client is merely downloading to increase their own download rate.

4.6.2.4. Pseudo-code for the novel management of TCP-based traffic

The pseudo-code that follows describes the novel handling of TCP-based applications (excluding streaming video, which was provided separately in section 4.6.2.1), field names from the FlowTable are denoted by the subscript i . Fields from the UserTable are denoted by the subscript j. As introduced in the previous section the term *CongestionHistory* is used to record the responsiveness of a flow over time – those flows that repeatedly fail to behave fairly will be managed more aggressively than those doing so for the first time.

```
Inputs: $Linkspeed, UserCount, FTPThroughput_j, VoIPThroughput_j, StreamingThroughput_j, HTTPThroughput_j, FTPCount_j, now, FlowTimestamp_i, PacketCount_i, HTTPFlowsize, rate_i, RTTEstimate_i, CongestionHistory_i, LastTimePoliced_i, P2PThroughput_j, P2PCongestionHistory_j$
$UserFairShare = linkspeed / UserCount$
$FlowFairShare = (VoIPThroughput_j + StreamingThroughput_j + HTTPThroughput_j)$
$P2PFairShare = (FTPThroughput_j / FTPCount_j)$
Switch (Traffic type)
    case 1: HTTP
        If ($PacketCount_i \leq HTTPflowsize$)
            Then
            $Packet\_count_i + 1$
            $FlowTimestamp_i = now$
            Forward packet as normal
        Else if ($PacketCount_i > HTTPFlowSize$) AND (($now - FlowTimestamp_i$) > $IdleThreshold$))
            Then
            $PacketCount_i + 1$
            $FlowTimestamp_i = now$
            Forward packet as normal
        Else if ($rate_i > FlowFairShare$) AND (($now - LastTimePoliced_i$) > $2*RTTEstimate_i$))
            Then
            $PacketCount_i + 1$
            Police packet as a function of $CongestionHistory_i$
            $CongestionHistory_i + 1$
            $LastTimePoliced_i = now$
        Else
            Forward Packet as normal
    case 2: FTP
        if ($rate_i > FlowFairShare$) AND (($now - LastTimePoliced_i$) > $2*RTTEstimate_i$))
            Then
            $PacketCount_i + 1$
            Police flow$_i$ as a function of $CongestionHistory_i$
            $CongestionHistory_i + 1$
            $LastTimePoliced_i = now$
        Else
            Forward Packet as normal
    case 3: P2P
        if ($P2PThroughput_j > P2PFairShare$) AND (($now - LastTimePoliced_i$) > $2*RTTEstimate_i$))
            Then
            $PacketCount_i + 1$
            Police flow$_i$ as a function of $CongestionHistory_i$ and $P2PCongestionHistory_j$
            $CongestionHistory_i + 1$
            $P2PCongestionHistory_j + 1$
            $LastTimePoliced_i = now$
        Else
            Forward Packet as normal
```

Figure 4.8: Pseudo-code for the novel management of TCP traffic by the Traffic manager

4.7. Additional assumptions for Concurrent Internet Services

Ordinarily, the *Traffic Manager* aims to minimise the over-all negative effect of traffic management by distributing any delay or packet loss across all of the user's services, while ensuring application-specific thresholds are achieved. However, there may be periods of extreme network congestion when the *Traffic Manager* cannot meet the minimum requirements for a user's services[11]. During these periods it is necessary to make some assumptions about which services the user would favour over others since actively involving the user in this process would conflict with the required transparency of the architecture.

Web browsing, VoIP, and streaming video are described as interactive services, each having a *minimum level of service*, below which renders the service unusable for the user. In contrast the user requirement for a file download is to retrieve the file as quickly as possible. In the event that a user is downloading one or more files, while also engaged in VoIP, Video and Web browsing sessions, it is proposed they will accept an increase in the time taken for their download to complete, providing the interactive services they are engaged in remain within the bounds of acceptability.

This traffic management is considered to improve upon traditional static QoS provisioning methods, such as Diffserv or simply class based queuing by initially aiming to proportionately degrade services to their respective minimum acceptable levels, prior to discriminating against non-interactive services.

4.8. Integrating User-centric QoS provisioning into Diffserv

The previous sections have described the novel traffic management policies that will provide a more application-aware, user-centric approach to QoS provisioning. The remaining sections within this chapter present the integration and enforcement of these policies into the Diffserv framework, which together provides the Congestion Aware Packet Scheduler (CAPS). The

[11] Repeated occurrences of this situation may be the result of a heavily over-subscribed ISP network or the result of a user reaching the limit of their subscription, in the case of the former it would be in the interest of the ISP to consider upgrading the capacity of their network, or in the case of the latter the user should consider upgrading their subscription – after all, the proposed architecture can only optimise bandwidth usage, it cannot create additional bandwidth.

decision to integrate the novel traffic management policies into the Diffserv framework was based primarily upon the prevalence of Diffserv support by router manufacturers (irrespective of whether it is currently in use or not). This provided an existing platform that could simply be updated with the CAPS algorithm with relative ease. Furthermore, a full Diffserv implementation (edge and core routers), is not explicitly needed in order to utilise the CAPS traffic management policy; since the management is performed solely at the network edge, it does not rely on Diffserv core router functionality, although having a network core that can interpret and act upon Diffserv markings would naturally be advantageous.

It is stressed that the proposed architecture was designed to allow integration into any existing IP infrastructure, either as the presented enhancement to the Diffserv edge router, or implemented as a standalone device situated at the network edge. That is, the traffic management is performed at the edge of the network with knowledge of the end-to-end characteristics for each service, rather than relying on end-to-end control over each service. This extends to being able to work transparently *with* any additional traffic management mechanisms that may be employed between the source and destination, since from the perspective of the novel architecture, any benefits these provide will simply be observed in the end-to-end characteristics of the traffic.

4.9. Enhancing the Diffserv Edge Router

The typical Diffserv edge router framework was presented as featuring four operational components, a *classifier*, *meter*, *marker* and a *shaper/dropper* (section 2.3.3). The components of the edge router are easily analogised to those of the novel architecture, described in the previous chapter. The *classifier* is comparable with the *Traffic Classifier*, the *shaper/dropper* performs the same role as the *Packet Scheduler*, and the roles of the *meter* and the *marker* are in the *Traffic Manager*. Specific details of the queuing mechanisms employed by the *shaper/dropper* were not covered in the previous sections, this was done in order to separate the traffic management philosophy from the enforcement techniques used.

In order for the traditional edge router components to conduct traffic management in accordance with the CAPS policy a number of enhancements are required. The *classifier* requires modification to include Deep Packet Inspection (DPI), which enables the retrieval of application-layer properties from incoming packets required by the CAPS policy. The use of both Internet service profiles and historical flow/user data requires for the two data stores, the *Service Profile Store* and the *User/Service Information Store,* to be also integrated into the edge router framework. The integration within the edge router is illustrated in Figure 4.9, which shows the addition of the two data stores, and also indicates the interaction between the components presented in the novel architecture and the integrated solution, the flow of packets is illustrated by the dark arrows, while control data within the architecture is represented by the white arrows.

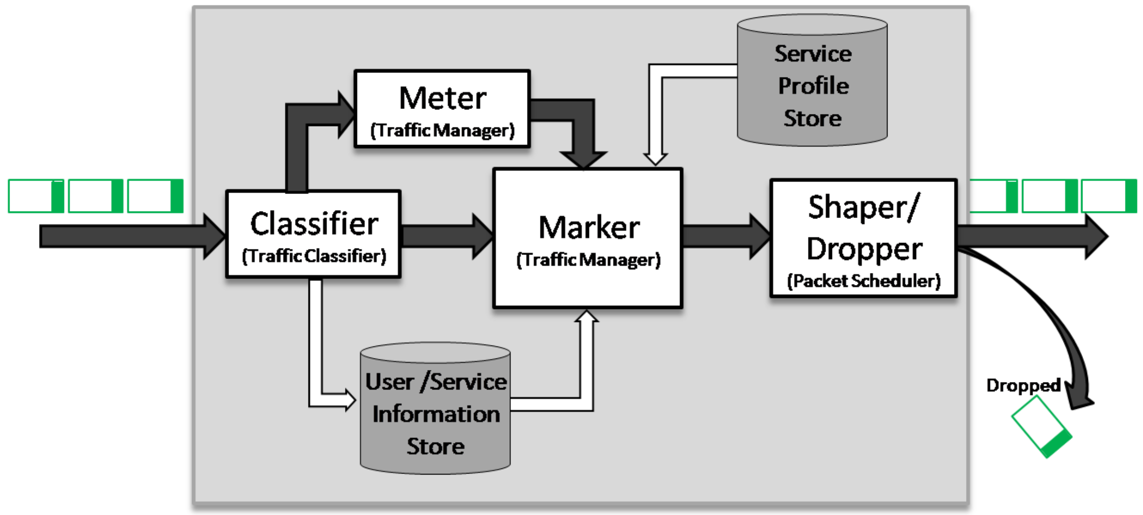


Figure 4.9: Integration of the CAPS traffic management system within a Diffserv Edge Router

4.9.1. Traffic Policy Enforcement

In essence, the CAPS traffic management policy aims to protect the most sensitive packets within a flow, making informed decisions regarding how packets should be queued or dropped during periods of congestion. The traditional queue configuration in a Diffserv network features the EF, AF and Best Effort traffic classes, with the network operators deciding the priority of an Internet service and assigning it to an appropriate class. A decision was made not to use the EF

PHB in the novel architecture since its operation is considered to conflict with one of the fundamental aims of this research – that is to deliver QoS to Internet users in a fair and non-preferential manner. This statement is justified by the fact that when using the EF PHB two configurations are possible, both of which are considered to violate the aim of user fairness. The first configuration when using the EF PHB is to implement strict priority over all other traffic classes, with no upper limit on the proportion of bandwidth allocated to the EF PHB. In a worst case scenario this configuration will lead to resource starvation for all other traffic types. In the second configuration an upper limit can be imposed on the proportion of bandwidth the EF PHB can occupy. Although other traffic types are now protected against resource starvation there is an intrinsic risk of denial of service (or absence of QoS guarantee) for any EF traffic arriving at the router once this upper limit has been reached. For these reasons the use of an EF PHB in the CAPS traffic management system has been excluded.

Similarities can however be drawn between the AF PHB and the philosophies of the proposed traffic management approach, in the sense that the AF PHB allows traffic to benefit from a service guarantee, providing the traffic complies with the parameters defined within a Service-Level Agreement (SLA). The management system is designed to maintain traffic above a minimum QoS threshold, which is deemed to be this aforementioned SLA. The earlier description of the AF PHB (section 2.3.3) describes how traffic is expected to adhere to a Committed Information Rate (CIR), compliance will result in incoming packets being marked with a particular DSCP value relating to a transmission queue and drop precedence (RED with IN and OUT (Clark and Fang 1998)), conversely, violating the CIR will result in the incoming packets being marked with a less favourable DSCP value (relating to a transmission queue with a higher drop probability). Shortfalls in the standard operation of the AF PHB were highlighted as being a lack of acknowledgement for the impact on transport and application layer protocols at a micro-level during the marking process. This meant in its standard form, AF PHB was only suitable for low granular aggregate flow handling.

However, the additional information collected and calculated by the *User / Service Information Store, Service Profile Store* and *Traffic Manager* (including the *User-Centric QoS Engine*) allows the AF PHB to be extended within the novel architecture to provide a higher granularity of protocol information, which enables a more user-centric QoS solution. The traditional AF PHB considered only whether or not a flow (or aggregate of flows) was adhering to a configured CIR. It is presented herein that an extension of this behaviour allows the CAPS traffic management system to dynamically adjust the QoS policy to ensure the evaluated parameters are relevant to the current mix of user traffic. For example, a VoIP flow achieving a specific R-value, FTP/P2P flow(s) achieving a fair share of bandwidth and Video flows achieving a throughput ≥ the video bitrate – the minimum acceptable level of QoS that was defined in section 4.6.

The synergy between the AF PHB, RED queues and the proposed management techniques of CAPS provided a promising queue configuration to be explored, while also raising a number of questions. As mentioned earlier in section 2.3.1, the tuning of RED parameters is a challenge in itself, one which is lessened by self-tuning variants such as Adaptive RED (Floyd S., Gunmamadi R. et al. 2001), but beyond this point, a question of suitability remained over whether RED (or similar AQM techniques) are indeed appropriate for multimedia traffic – after all, VoIP traffic is typically placed into a simple (albeit expedited) taildrop queue. A number of previous studies have investigated the impact of various AQM techniques on multimedia and VoIP traffic flows. Evidence can be found in (Hollot C. V., Misra V. et al. 2002; Wydrowski B. and M. 2002) that suggests AQM techniques can in fact provide a reduction in queue length (delay) and queue oscillations (jitter) for multimedia traffic flows when comparing against taildrop configurations. Extending this research to consider the perceived user QoS of VoIP flows (Reguera, Álvarez Paliza et al. 2008) evaluated the performance of various AQM techniques using the MOS scale, again concluding that AQM techniques can offer a significant improvement on user perceived QoS for VoIP, with Adaptive RED and Adaptive Virtual

Queues (AVQ) (Kunniyur S. and Srikant R. 2003) offer the best VoIP performance (from a user's perspective).

In addition to considering the suitability of AQM for VoIP traffic, further studies have also evaluated the performance offered for HTTP traffic, with two widely cited studies being (Le, Aikat et al. 2003; Weigle, Jeffay et al. 2006). Both of these works conclude that AQM techniques can offer an improvement to HTTP response times over taildrop queuing, but in both cases the use of Adaptive RED with ECN enabled sources providing the best results, standard RED queues offered little improvement over taildrop configurations.

It is highlighted that although these studies conclude certain AQM techniques can offer significant improvement for multimedia services, standard RED queues were not the best performing configuration, rather self-tuning and adaptive AQM techniques out-performed standard RED. Despite this it was decided that the novel architecture would employ RED queues for traffic management enforcement. This decision was based upon 3 main points; 1. Although other AQM mechanisms out-performed RED, at no point did RED perform worse than taildrop; 2. No one single AQM mechanism served all traffic types optimally - employing the optimal AQM mechanism for each traffic type would rely upon the appropriate interoperability between mechanisms, which to date has not been explored; 3. While RED has been widely implemented in production routers, a number of the alternative AQM mechanisms remain as mathematical models, only implemented in network simulation packages. This could limit real-world development of CAPS.

Following the decision to use standard RED queues for CAPS, the next step in the design was to determine how many RED queues would be needed in order to achieve sufficient control over the traffic, and the values of min_{th}, max_{th}, and max_p for each queue. In section 4.6 the outlined operation of CAPS describes a multi-stage approach to penalising flows, which is enabled through recording the *CongestionHistory* for each flow – a variable that records the number of times a flow has had its packets downgraded in service. In the first instance all packets are assigned to the default queue. In the event that congestion occurs packets from misbehaving flows are placed into a stricter queue with a higher drop probability. In the event a traffic source does not respond to this action by reducing its throughput then packets from offending flows will be placed into an even stricter queue. This operation dictates there are at least three RED queues, with increasingly stricter parameters. For the purposes of providing proof of concept to the CAPS design three RED queues were used. It was considered that little gain could be achieved through adding further degradation queues.

When deciding the values of the RED parameters guidance was taken from two publications by Sally Floyd (Floyd S. 1997; Floyd S., Gunmamadi R. et al. 2001) and also from sampling a large number of research papers that implement RED queues. In accordance with Floyd's recommendation the chosen value of max_{th} was 3x that of min_{th}, and max_p was chosen to be 0.02, 0.05 and 0.1 for each of the three RED queues (in ascending strictness). Table 4.4 below provides the specific values for each RED parameter (the unit of min_{th} and max_{th} is packets).

RED Queue	min_{th}	max_{th}	max_p
Default	40	120	0.02
Downgrade level 1	*15*	*45*	*0.05*
Downgrade level 2	10	30	0.1

Table 4.4: RED Queue Parameters for CAPS architecture

4.9.2. Queue Scheduling Algorithm

Following the development of the queuing mechanisms used in the CAPS architecture some thought had to be given regarding the scheduling of the packets from each of the three RED queues. The primary purpose for having more than one queue is to enforce the traffic management policy, rather than segregate different types of traffic. However, any packets queued in either of the penalty queues still has a greater chance of being forwarded than dropped, and in which case it would be detrimental to the overall QoS of the service to excessively delay their transmission. For this reason a simple round robin scheduling system was considered the most appropriate method to service each of the queues.

4.10. Summary of the Novel Approach to Traffic Management

This chapter has reflected upon the findings from chapters 1 and 3 to revise and affirm the motivations for a novel approach to QoS provisioning. The chapter began by expanding on these motivations to construct a specification for an ideal solution for user-centric QoS, which although idealistic required realisation in order to produce a pragmatic design. This specification was then used to develop an architectural blueprint, defining each component of the proposed solution in detail. The role and function of each component were described in turn, considering both data plane and control plane responsibilities. Following the introduction of the proposed architecture the chapter continued, describing design trade-offs that were needed to balance a desirable degree of granularity and control, with the overall complexity of the system.

The second part of the chapter considered the design of the traffic management component, and specifically focussed on the management of five popular Internet services, VoIP, streaming-video, FTP, HTTP and P2P. Using performance metrics that can be monitored and measured for incoming flows, a key new threshold for each service was realised. The *minimum acceptable level of service* is derived from network, transport and application layer measurements, and represents the minimum performance a service can be delivered to the user, without introducing dissatisfaction or unfairness. These thresholds were; R-Factor for VoIP, throughput aligned to bitrate for Video-on-Demand, bandwidth shared fairly among single and multi-flow TCP applications (i.e. FTP and P2P) and protection for the first 12 packets of an HTTP flow (to prevent loss during the early stages of TCP). It is the role of the proposed traffic manager to ensure these thresholds are adhered to in a dynamic fashion, offering QoS for the entire multimedia user experience. The chapter continued to discuss additional assumptions regarding traffic management should these thresholds not be achievable in extreme network conditions.

Following from the design of the traffic management policies, the chapter considered how these policies should be enforced, and how such functionality could be integrated within a network architecture. The decision was made to take advantage of the architecture of a Diffserv edge router, given existing level of support. A number of considerations were made when deciding the optimal enforcement technique, balancing complexity with supportability. The decision was made to implement a number of RED queues to provide policy enforcement, configured in a manner that would provide a multi-tiered management approach. The chapter concluded with a brief justification of the round robin scheduling algorithm chosen to service the aforementioned RED queues.

5. Validation of CAPS through Simulation

Following the integration of the CAPS algorithm into the Diffserv edge router, functional validation was required to demonstrate the key aspects of the novel approach to traffic management, these were:

- An ability to provision for QoS without prior per-user configuration, for any Internet user, irrespective of the services that are in use.
- Evaluate Internet services using application-layer QoS metrics (e.g. E-model's R-factor, video bitrate)
- Manage Internet services based on application-layer information. (e.g. Ensure the R-factor of a VoIP flow remains above the lower threshold, and during congested periods is managed to be between the upper and lower thresholds).
- Limit the throughput of P2P traffic to be equal to any concurrent FTP/HTTP services from the same user.
- The benefit that performing TCP traffic management on a per-RTT basis has compared with the standard per-packet approach.
- The ability to police flows that repeatedly exceed their fair share more aggressively than those that comply with the traffic policy.

The decision was made to perform the validation of the CAPS algorithm using the *ns*2 Network Simulator (ns2 Network Simulator 2010), a discrete event-driven simulator, developed at the University of California, Berkeley. The simulation engine is implemented in C++, and uses OTcl as the command and configuration interface. Thus, modifications made to the simulator are made to the C++ source code (see Appendix B), which then requires recompiling, and simulation topologies and configurations are written in Tcl (see Appendix C), which are then interpreted by the simulator. This decision to use *ns*2 for the modelling of the CAPS algorithm was based on a number of factors:

- *ns*2 has been in development for over 20 years, and continues to be developed with substantial contributions from the research community.
- *ns*2 is the leading network simulator used within academic research.
- The popularity of *ns*2 has resulted in an active online development community, which not only offers technical support for developers, but the widespread use of *ns*2 allows for peer validation of implemented prototypes.
- Although *ns*3 was available at the time the validation was undertaken, it remained less mature in terms of documentation and support compared with *ns*2.
- The commercial funding from France Telecom made this research project ineligible for an academic license of OPNET (OPNET 2010), which would have otherwise been an alternative option for a development environment.

An alternative to validation through simulation would have been to implement the CAPS algorithm as part of a real-life Diffserv router. One method of achieving this would have been to use the Traffic Control framework (tc), part of the Linux operating system. While this implementation would have enabled validation with live Internet traffic, which could have provided a subjective method of evaluation, this method was not pursued for the main reasons that the equipment budget would not have permitted validation beyond more than a few terminals, preventing essential testing at larger scales, and subjective testing is also expensive in terms of the time needed.

5.1. Differentiated Services in *ns*2

The validation work made extensive use of the Differentiated Services framework for *ns*2, developed by Nortel Networks (Pieda, Ethridge et al. 2000), and is included with the standard *ns*2 installation. The Nortel Diffserv implementation focuses largely on the AF PHB, servicing incoming packets across multiple RED queues according to their conformance with the agreed policy, providing an ideal foundation to implement the CAPS algorithm on. However, a full Diffserv configuration can also be modelled using the Nortel framework to include the EF and

BE PHBs by configuring additional queues, which are then serviced according to a priority-based scheduling algorithm (this configuration is described in further detail in section 5.2.2). This framework provided both a platform to implement the CAPS algorithm and also the ability to benchmark its performance against alternative traffic management schemes (including traditional Diffserv).

5.1.1. Limitations of the Standard *ns*2 Diffserv Implementation

Although the *ns*2 framework offered basic Diffserv functionality, it was limited in a number of aspects, which required addressing prior to the validation of the CAPS algorithm.

- By default the edge router requires one traffic policy for each source-destination pair (up to a maximum of 40 pairs), no method existed to apply a policy to multiple source-destination pairs. While the CAPS algorithm does not required per-source-destination configuration, the traditional Diffserv configuration does, and therefore, *ns*2 was modified to allow large-scale topologies to be created for evaluating against Diffserv.
- Packet marking based upon the service type was not possible, instead marking was performed by source-destination pair only; for example, using the standard *ns*2 implementation all traffic between nodes A and B will be marked the same irrespective of the traffic type. To resolve this issue an enhancement allowing marking based also on traffic type was made (see Appendix B for the modified *ns*2 source code).
- The only traffic sources that could be simulated were UDP (CBR, Pareto ON-OFF and exponential sending rates), FTP (via TCP) and HTTP. The FTP traffic source is further limited since it can only be configured to start and stop at specific times, rather than specifying a volume of data to transfer and the source stop transmitting once this volume has been delivered. A number of enhancements were made to address these traffic limitations, details of which are provided in section 5.2.1.

- Further modifications were needed to extract performance metrics for later analysis and evaluation. These included VoIP R-factor, One-way delay, Round Trip Time, aggregate P2P throughput and fair bandwidth share.

5.2. Simulation Methodology

In order to fully demonstrate the functionality of CAPS the simulation trials were divided into two phases. The first simulation was designed to exhibit the key features of the novel framework for a single user, highlighting the novel management of different traffic types. The second phase of simulations was designed to evaluate the overall performance of the CAPS algorithm when considering it in the context of a medium-large scale hierarchical Internet topology.

Each of the simulations from Phase-1 were ran a minimum of 10 times, with the mean performance value given in this results section. For Phase-2, the results provided are the arithmetic mean for a number of simulated nodes (i.e. users/destinations). The exact number of simulated nodes varies depending upon traffic type, but was always within the range of 30-90 hosts. This representation of results was considered to better represent the overall performance of CAPS rather than focusing on a single user.

5.2.1. Simulated Traffic Sources

As introduced previously, the standard installation of *ns*2 offers a relatively limited selection of possible traffic sources, namely; FTP over a variety of TCP implementations (e.g. Tahoe, Reno, New Reno), Constant Bit Rate (CBR) / Pareto / Exponential over UDP, and HTTP-like traffic using the Packmime-HTTP Web Traffic framework (Cao, Cleveland et al. 2004). While VoIP services, FTP file transfers and HTTP traffic were easily achieved using *ns*2 extensive enhancement to the simulation software would have been required in order to validate the CAPS algorithm modelling all of the Internet services reviewed to this point. Based on the evaluation of current streaming video techniques (section 3.3), it was decided not to model streaming video in *ns*2, and subsequently only include traffic models for VoIP, FTP, HTTP and P2P in the simulations. This decision to exclude a traffic model for streaming video was made given that from the transport layer perspective, streaming video is in essence an HTTP transfer over a TCP connection. The CAPS algorithm is described to aim to ensure in the case of traditional streaming methods the TCP connection maintains a throughput in keeping with the approximate bitrate, which is inferred using DPI, and in the case for pseudo streaming video flows the bitrate is limited to no less than the inferred approximate bitrate. Given that the novelty of this management approach is the bitrate inference method and not the throughput management of a TCP flow, it was considered an acceptable compromise to make.

5.2.1.1. Simulated VoIP Parameters

The simulated VoIP traffic was based on the G.711 codec (ITU-T. 1988), using a CBR traffic source over UDP. The CBR source was configured to transmit packets of 160 bytes at a rate of 64Kbps, in line with the standard parameters for the codec. Although this configuration features a higher bitrate than many of the VoIP codecs evaluated in section 3.5, a large number of these have been developed from the G.711 PCM algorithm, using compression and silence suppression techniques, hence the reduction in bitrate. Furthermore, the G.711 codec has been, and still is, used prolifically throughout research, which includes but is not limited to (Balan, Eggert et al. 2007; Reguera, Álvarez Paliza et al. 2008), and so for these reasons was considered a suitable VoIP codec to model. For simplicity reasons when evaluating the QoS of the VoIP traffic, the simulated flows were uni-directional, with speech travelling in one direction only, however, this did not hinder the validation of using the R-factor of the call for QoS provisioning. No additional speech mechanisms, such as silence suppression, Forward Error Correction or adaptive bitrates were implemented.

5.2.1.2. Simulated FTP Parameters

Long-lived FTP transfers over TCP were configured to represent the rising trend of one-click hosting services for large file transfers, such as RapidShare and MegaUpload. Despite these services operating over HTTP in reality, TCP file transfers within *ns*2 are modelled using the

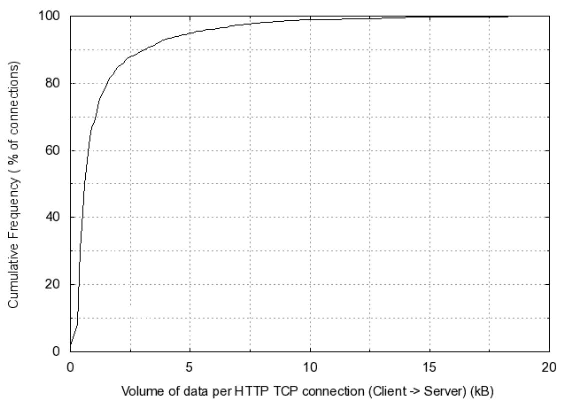
FTP traffic source, which for the purposes of this validation did not present a problem. Each host was configured with the TCP Reno implementation, sending full sized packets with payloads of 1460 bytes.

5.2.1.3. Simulated HTTP Parameters

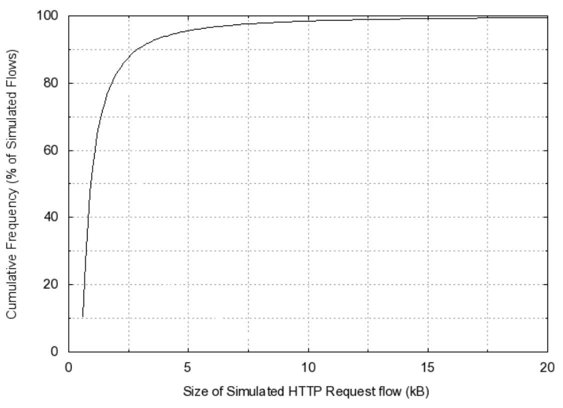
Whereas all other traffic sources within *ns*2 generate traffic for a single application, the Packmime HTTP traffic source represents a cloud of HTTP clients or servers, requesting and responding to multiple HTTP connections at any given time.

This limits the ability to simulate HTTP traffic on an individual user basis, as part of a larger network topology. However, during validation two Packmime clouds (one HTTP clients and the other servers) were configured to provide HTTP cross traffic within the simulated network, which although not destined toward individual users / destinations, could be evaluated to determine the performance of the CAPS policy in handling HTTP traffic.

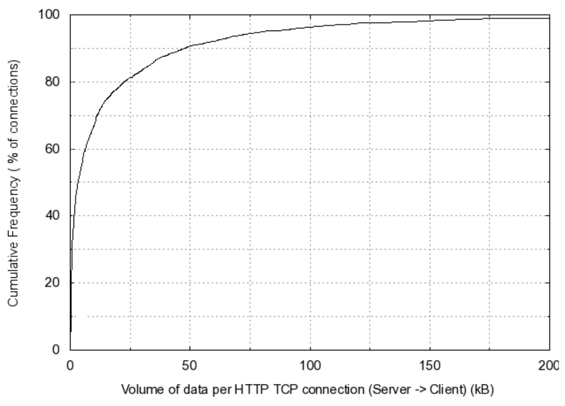
Each pair of HTTP clouds were configured to represent, as accurately as possible, typical HTTP cross traffic, given the scale of the simulation. The configuration parameters were chosen in accordance with the data collected in section 3.2 that describes the "average webpage". The HTTP request and response flow sizes were configured as Pareto distributed random variables, with mean values of 1.5kB and 22kB respectively. Comparing the CDF plots for the synthetic web traffic with the observed real-world traffic (Figure 5.1), the Pareto distributions are highly similar, with comparable 95% percentiles of approximately ≤ 6kB for the request size, and ≤ 85kB for the response flow sizes.



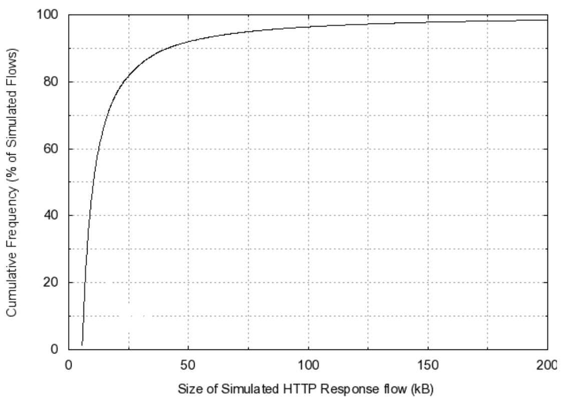
Real-world Observations



Simulated Traffic



Real-world Observations



Simulated Traffic

Figure 5.1: Comparison of Real-World HTTP Request/Response Flow Sizes with Simulated Equivalents

The rate at which new HTTP connections were established was dependant on the number of users the HTTP clouds were intended to represent, which varied between simulation trials. More precise details are provided in the description of the simulations, later in this section.

5.2.1.4. Simulated P2P Parameters

The standard installation of *ns*2 does not provide the capability to simulate P2P-like traffic. A number of modular solutions have been developed within the research community, namely "Gnutellasim" (He 2003) and "BitTorrentSim" (Eger, Hoßfeld et al. 2007). The former of these options was dismissed due to it being heavily designed around the Gnutella P2P architecture, which has been in steady decline as a popular P2P architecture over the past four years (Schulze and Mochalski 2006; Schulze and Mochalski 2007; Schulze and Mochalski 2009), and so was not considered overly relevant to represent current Internet services. The latter offering is designed around the BitTorrent P2P protocol, which as discussed in Chapter 3 is currently the most popular P2P technology according to the Ipoque Internet study(Schulze and Mochalski 2009). However, while this offering provided a high-level replication of the complex BitTorrent mechanisms, including the *choking/unchoking* algorithms, the framework was designed to evaluate the performance of the protocol itself under varying configurations, rather than simply providing a BitTorrent traffic source.

The inclusion of P2P-like traffic was considered paramount to the validation of the CAPS algorithm, and therefore a bespoke BitTorrent-like configuration was developed. The decision to focus on the BitTorrent protocol was made based on its prevalence on the Internet, and a better understanding of the operations compared with alternatives. The BitTorrent-like traffic model was simplified to emulate only the following attributes of the real-life protocol:

- Simulate both Seeds (uploaders only) and Peers (upload and downloaders)
- Each Seed/Peer should select four other peers at random.
- TCP connections should be established with each of these selected peers and data transferred for a given period of time (defined by a random variable).

While this functionality is simplified compared with the full BitTorrent protocol, it does provide a traffic model that emulates the exchange of data between a swarm of peers. The selection of four peers is reflective of the limit imposed by the *choking* algorithm, (which allows a maximum of four simultaneous uploads to current peers by default), while the randomised nature of the peer selection process results in some peers receiving more incoming flows than others.

The BitTorrent-like implementation did not consider the role of the *tracker*, nor was it considered necessary to establish connections with larger numbers of peers purely to facilitate a model for control traffic. These decisions are justified by the negligible proportion of bandwidth consumed by control traffic and during communication with the *tracker*, as detailed in section

3.4.3, and the fact that the CAPS algorithm focuses on managing the BitTorrent flows responsible for the majority of the data transfer.

5.2.2. Validation Cases

For both simulation phases the "ISP network segment" simulated network topologies (later presented in sections 5.3.1 and 5.5.1) were configured to represent four different traffic management schemes, Best-effort (no traffic management policy), Traditional Diffserv, static Weighted-RED (WRED) and using the novel CAPS algorithm. A static Weighted-RED configuration was included to differentiate between the dynamic user-centric provisioning methods of the CAPS algorithm and the static preconfigured methods of WRED, which was questioned on a number of occasions during the development of the novel architecture. It should be noted that the performance of Traditional Diffserv and WRED is entirely dependent on the mapping of services to traffic classes (queues). Thus, the presented results are reflective of the chosen configurations, which were based upon the known traffic models of the simulated network, combined with common consensus for traffic preference models, where VoIP is primarily considered the most critical traffic type, and P2P is frequently cited as the least favoured. This challenge of configuring Diffserv and WRED further emphasises the benefit of the CAPS architecture and algorithm, in the sense that no prior knowledge or configuration is required.

A description and justification for the configuration of each of the alternative validation cases follows, while CAPS was implemented according to the recommendations in the previous chapter.

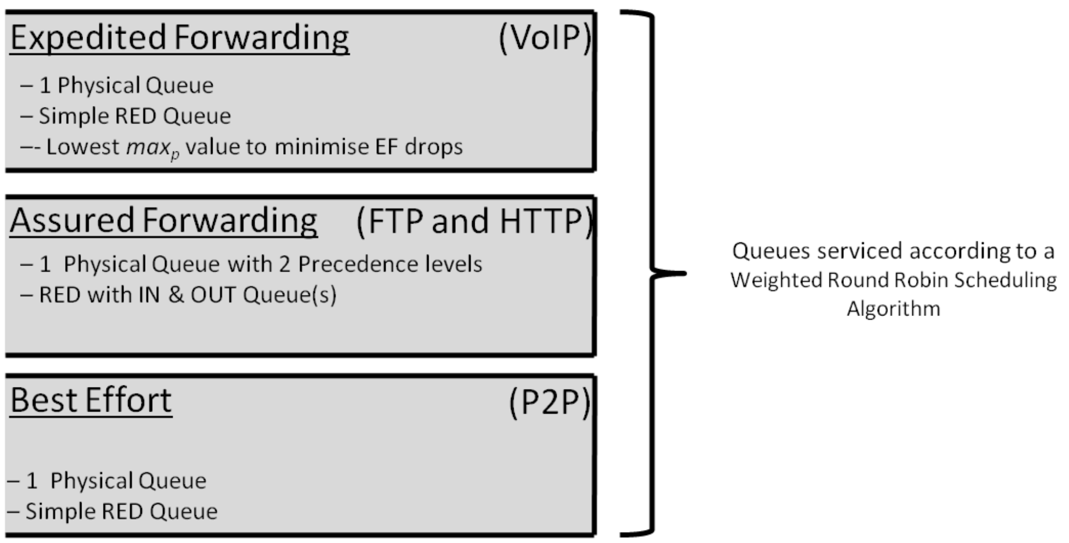
Best Effort Configuration

The Best-effort configuration was achieved with a single RED queue configuration, which all traffic was associated with. This configuration was used to represent the Best-effort Internet, where by design (i.e. assuming no additional traffic management systems are being used) every

IP packet is forwarded with equal importance. As per the design of the RED parameters for the CAPS architecture (section 4.9.1) the Best-effort RED queue was configured in accordance with (Floyd S. 1997), with $min_{th} = 40$, $max_{th} = 120$ and $max_P = 0.02$, where the threshold limits are expressed in packets (default queue unit within *ns*2).

Traditional Diffserv Configuration

As described in section 5.1 the Nortel Diffserv framework can provide a full Diffserv configuration, with Expedited, Assured and Best-effort Forwarding (EF, AF and BE PHBs). This configuration was achieved during the validation by configuring three physical RED queues, one for each PHB. These physical queues are then serviced in accordance with the Weighted Round Robin algorithm, which provides preferential treatment to the EF queue, and least favours the BE queue (a strict priority scheduler was not available in *ns*2 and as such a WRR approach was used instead, heavily weighted in favour of the EF queue). The AF PHB is implemented using one physical queue, with a multiple virtual queues, which provide the IN/OUT-of-profile handling. This queue configuration along with service mappings is illustrated in Figure 5.2. While the configuration of a Diffserv network is at the discretion of the network operator this configuration was in line with the traffic prioritisation models in use by ISPs (summarised earlier in section 2.3.7) and therefore considered a fair representation of Diffserv.


Expedited Forwarding
(VoIP)
– 1 Physical Queue
– Simple RED Queue
–- Lowest max_p value to minimise EF drops
Assured Forwarding
(FTP and HTTP)
– 1 Physical Queue with 2 Precedence levels
– RED with IN & OUT Queue(s)
Best Effort
(P2P)
– 1 Physical Queue
– Simple RED Queue
Queues serviced according to a Weighted Round Robin Scheduling Algorithm
Figure 5.2: Queue configuration for Traditional Diffserv and Service mapping

Similar to the configuration of the Best-effort RED queues, the RED parameters for the Diffserv configuration were chosen based upon the recommended settings, and are summarised in Table 5.1.

Queue	min_{th}	max_{th}	max_p
EF	40	120	0.02
AF Virtual Queue 1 (IN of Profile traffic)	40	120	0.02
AF Virtual Queue 2 (OUT of Profile traffic)	10	30	0.1
BE	20	60	0.02

Table 5.1: RED Queue parameters for Traditional Diffserv Configuration

<u>Weighted-RED Configuration</u>

The WRED configuration was achieved using three RED queues, for which the RED parameters were increasingly more aggressive. VoIP traffic was associated with the highest priority queue, FTP and HTTP to the second queue, with marginally stricter RED parameters than the first, and P2P to the third queue. The RED parameters are summarised in Table 5.2.

Queue	min_{th}	max_{th}	max_p
Queue 1 (VoIP traffic)	40	120	0.02
Queue 2 (FTP and HTTP traffic)	40	120	0.05
Queue 3 (P2P traffic)	40	120	0.1

Table 5.2: RED Queue parameters for WRED configuration

5.3. Simulation Phase-1 – Demonstrating the features of CAPS

5.3.1. Simulated Network Topology

The network topology used for the simulations in Phase-1 (Figure 5.3) was designed to represent a user of each type of Internet service (FTP, VoIP, HTTP and P2P), and demonstrate the operations of the CAPS algorithm when provisioning for each of these services over a congested network link (between E1 & E2 and highlighted red in Figure 5.3).

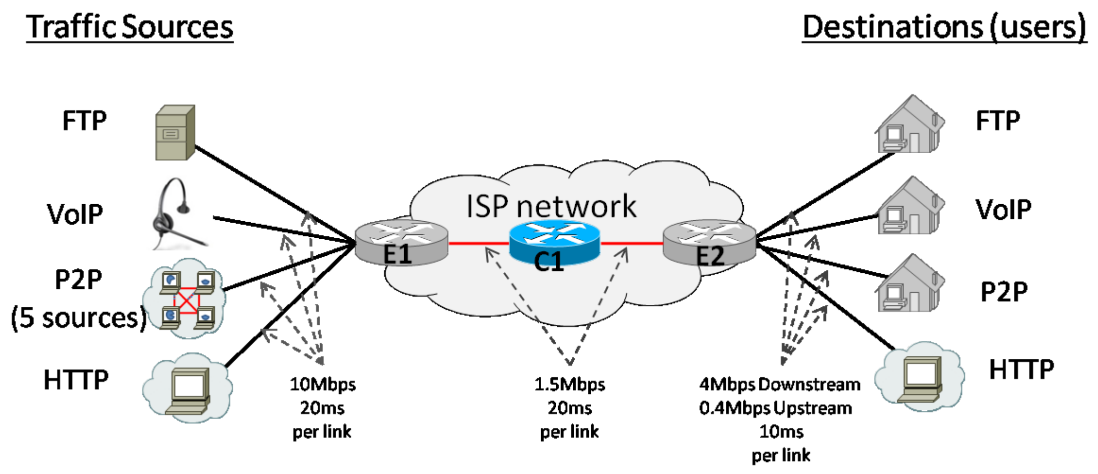


Figure 5.3: Simulated Network Topology for Trial A

The topology was based upon the well known "dumbbell network", with traffic sources and destinations either side of a congested bottleneck link, which in this case is the ISP network, analogous to the typically contended access network that occurs within real-world networks. Each of the destination nodes were connected to the ISP network via two simplex links. The downstream link for each destination was configured at 4Mbps with 10ms delay, and the upstream at 400kbps with 10ms delay. These values were chosen as an accurate representation of a typical UK Broadband connection, according to (Ofcom 2008). The capacity of the ISP network was deliberately scaled down from the real-world equivalent in order to cause congestion, thus providing means to evaluate the performance of the CAPS algorithm. Furthermore, the capacity of the ISP network reflects the scale of the topology (i.e. four destinations compared with the likely thousands a real-world ISP network would support). Table 5.3 summarises the link capacities and delay values for the Phase-1 network topology.

Link	Bandwidth (Mbps)	Delay (ms)
FTP Source ↔ ISP Edge Router (E1)	10	20
VoIP Source ↔ ISP Edge Router (E1)	10	20
P2P Source ↔ ISP Edge Router (E1)	10	20
HTTP Server Cloud ↔ ISP Edge Router (E1)	10	20
E1 ↔ C1	1.5	20
C1 ↔ E2	1.5	20
E2 → Destination node	4	10
Destination node → E2	0.4	10

Table 5.3: Link Parameters for Phase-1 network topology

The traffic sources used were as described previously in section 5.2.1, with the exception of the P2P traffic. Due to the reduced scale of this topology, the BitTorrent-like traffic model could not be used. Therefore, the traffic destined for the P2P user was modelled using a collection of five FTP servers, all configured to send data to a common destination (the P2P user). This design modelled sufficiently the use of multiple TCP connections, destined for a single user-application. All traffic sources were started at 0.1 seconds and terminate at 450 seconds (the duration of the simulation).

5.4. Analysis of Phase-1 Results

This section presents the results and observations obtained from the Phase-1 simulation trials, which were conducted to demonstrate the key aspects of the CAPS architecture and traffic management algorithm. These features are presented through performance evaluation of the simulated traffic sources - a comparison between the results obtained using the CAPS algorithm and each of the alternative configurations is given. A number of analysis scripts were used to compute the presented performance metrics from the *ns*2 trace files; these are provided in Appendix D.

5.4.1. Performance Analysis for VoIP traffic

The first aspect that is presented is the novel management of VoIP traffic by CAPS, which uses a moving average of the R-factor(s) for the VoIP flow(s) to determine the handling of VoIP packets, in contrast to explicitly allocating bandwidth. Figure 5.4 provides a plot of the R-factor over time for each of the simulation cases. As expected, the traditional Diffserv configuration achieves the highest average R-factor of 87.9, given that under this configuration sufficient bandwidth (64kbps) is explicitly reserved for the flow, and the EF queue is serviced at a ratio of 4:2:1 compared with the AF and BE queues respectively. There were no packet drops for the Diffserv configuration, so the small degradation in quality was introduced via delay only (this was unavoidable due to a lack of strict priority scheduling within *ns*2). Finally, a relative standard deviation of 5.7% and a modal scale of 91.4 indicate the low variation of the R-factor over time under traditional Diffserv.

With regard to the VoIP performance using CAPS Figure 5.4 illustrates the flow being actively managed based upon its current average R-factor, aiming to maintain an average moving R-factor of 75, to avoid user dissatisfaction. In fact, CAPS achieves a mean R-factor of 74.8, demonstrating this aim of providing an acceptable quality of VoIP to the user was achieved without any static configuration or reservation. The relative standard deviation under CAPS is marginally higher compared with Diffserv, at 9.4%, however this can be attributed to the active management technique, which in the event of congestion introduces controlled packet drops / additional queuing to VoIP flows, to prevent resource starvation to other users.

Both the Best-effort and WRED configurations struggle to deliver the VoIP traffic with any degree of acceptable quality (later observed to be due to the aggressive nature of the concurrent P2P flows on the network, section 5.2.1.4). The poor performance from the WRED configuration highlights the difference between merely placing traffic in weighted RED queues (WRED) and actively managing with regard to application/user –layer metrics.

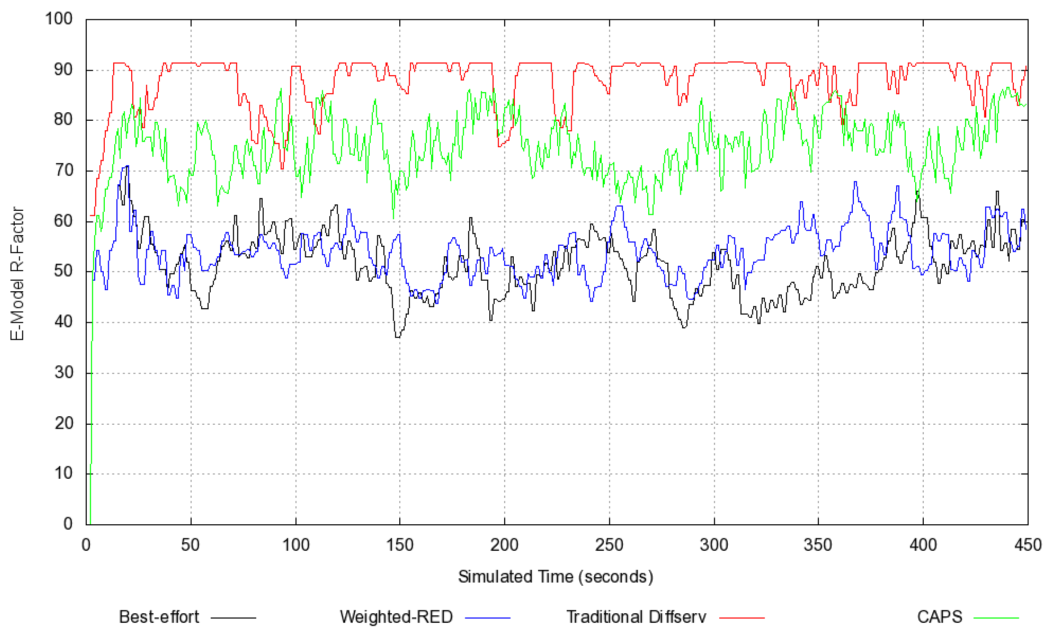


Figure 5.4: Average R-factor over time for a G.711 VoIP flow

5.4.2. Performance Analysis for FTP traffic

The second traffic type to be evaluated is FTP traffic, which modelled the long lived TCP connections established during FTP/HTTP file downloads. Figure 5.5 illustrates the average throughput over time of the FTP transfer for each of the validation cases, where the performance under CAPS is noticeably better than the alternative configurations. CAPS achieved a mean throughput over the duration of the simulation of 0.55Mbps, with a standard deviation of 25%, compared with 0.25Mbps for Diffserv and WRED, and 0.21Mbps under Best-effort conditions. The relative standard deviations for the alternative configurations were also poorer at 24%, 36% and 44% for Best-effort, Diffserv and WRED respectively.

The cause of the poor performance from the alternative configurations is once again due to the aggressive nature of the concurrent P2P traffic on the network, which obtains a disproportionate share of bottleneck bandwidth by establishing multiple TCP connections, (this behaviour is discussed in detail in the next section 5.4.3). However, one of the aims of the CAPS algorithm is, “to ensure each user receives a fair share of the bandwidth, plus an equal share of any residual”, Figure 5.5 confirms this aim has been achieved. In the simulated topology there are four users, three of whom are using TCP-based services (FTP, HTTP and P2P), and one VoIP client. The previous section presented how the VoIP user's traffic was managed with regard to the R-factor, which once satisfied leaves the remainder of the 1.5Mbps bottleneck capacity available to the other three users (approximately 1.45Mbps). Therefore, CAPS can allocate just under 500kbps to each user's traffic (plus any further residual that arises from any of these sources not utilising their full allocation). The 0.55Mbps obtained by the FTP user is evidence that this method of fair bandwidth allocation was achieved. It is noted that FTP traffic performs significantly closer to the fair bandwidth share of 0.5Mbps using CAPS compared with the three alternative configurations. This performance is largely attributed to the per-application fairness that CAPS aims to achieve, which prevents applications using multiple TCP flows (i.e. P2P) from obtaining a disproportionate share of resources.

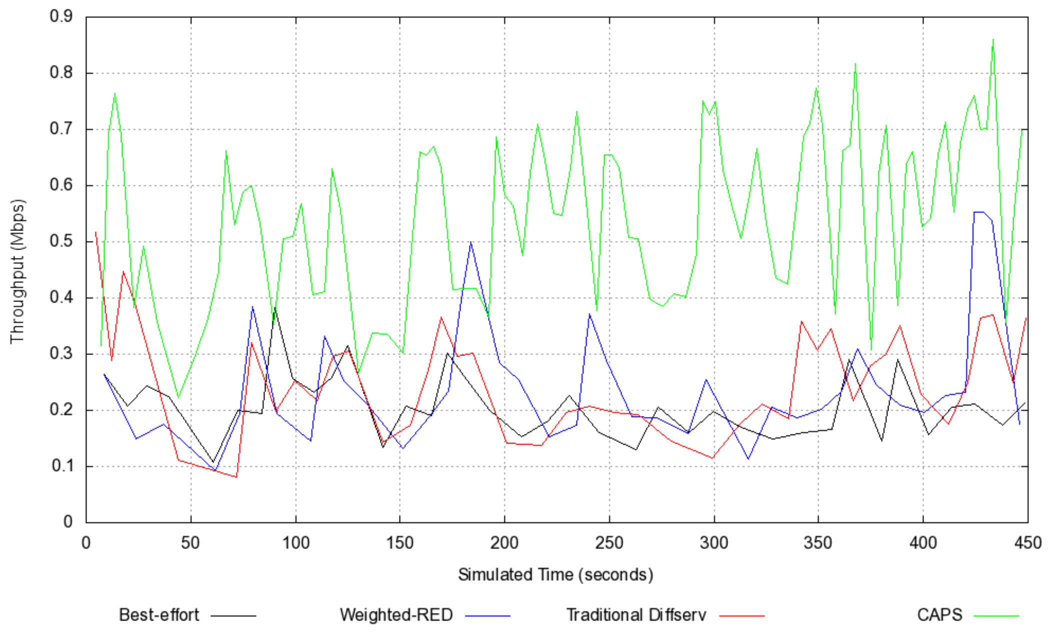


Figure 5.5: Average throughput over time for an FTP application

To further evaluate the performance of the FTP traffic the number of packets received by the FTP client and the number of FTP packets dropped under each configuration is provided in Table 5.4. Two observations can be made from this data; firstly, by controlling the aggressive nature of P2P traffic the FTP source was able to send more than double the amount of data compared with the alternative configurations. Secondly, less than half the number of proportionate packet drops were observed when compared with the alternative simulations, demonstrating a more efficient and fairer operation for FTP traffic.

Configuration	**Number of FTP packets sent from the source**	**Number of packet drops observed**
Best-effort	7693	198
WRED	8534	179
Traditional Diffserv	8686	115
CAPS	20216	109

Table 5.4: Number of Packet Drops observed for the FTP traffic

5.4.3. Performance Analysis for P2P traffic

The previous discussions on the management of VoIP and FTP traffic has suggested that the uncontrolled P2P traffic on the network was responsible for degrading the QoS of other user services in the Diffserv, WRED and Best-effort configurations. The aggressive nature of P2P traffic is a topic that has been heavily reviewed throughout this thesis, and the configuration of the alternative scenarios were chosen to reflect the discriminatory viewpoint that is apparent of network operators. However, despite P2P being configured as the least favoured traffic for both Diffserv and WRED, the poor performance of the FTP service indicated that the P2P traffic is still degrading the QoS for other, concurrent users.

Figure 5.6 provides the average aggregated throughput for P2P traffic, for each of the validation cases. The aggregated throughput using CAPS is noticeably lower than the alternative configurations, averaging 0.45Mbps over the simulated period, compared with 0.89Mbps, 0.86Mbps and 0.79Mbps for the Best-effort, WRED and Diffserv scenarios. It should be noted that under the Diffserv configuration the aggregated throughput is marginally lower than under Best-effort or WRED, given that FTP and HTTP are favoured over P2P by Diffserv, so the P2P management is slightly more aggressive.

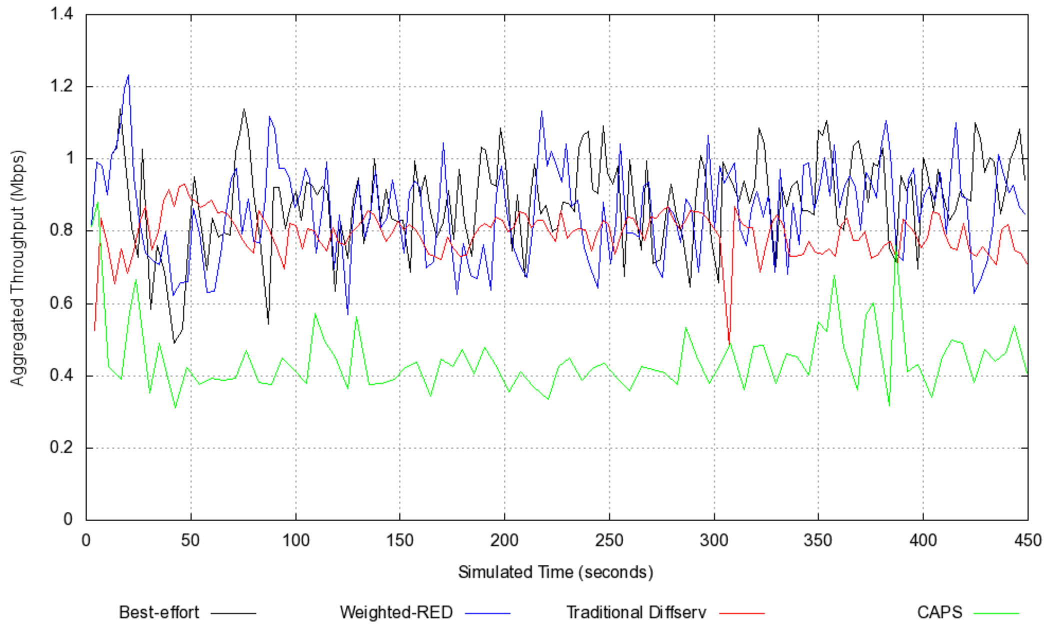
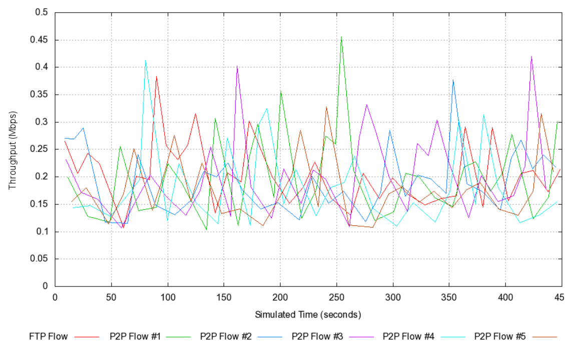
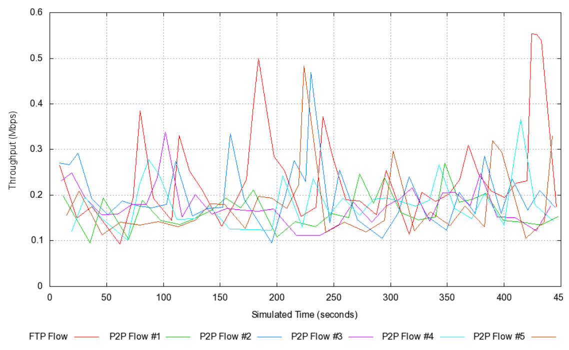


Figure 5.6: Average Aggregated Throughput over time for a P2P application

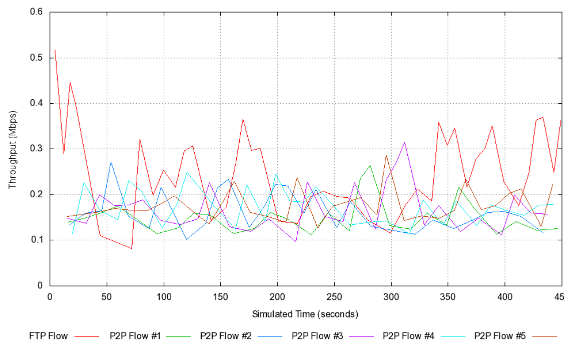
To further demonstrate how CAPS manages P2P traffic in a fair manner Figure 5.7 provides the throughput over time for the FTP flow and also the individual throughputs over time for each of the component P2P flows. Recalling that CAPS will police the component P2P flows for a user to ensure their combined throughput does not exceed that of a coexisting FTP/HTTP flow (section 4.6.2.3), Figure 5.7 shows each of the five P2P flows being maintained well below the throughput of the single FTP flow, in the knowledge that at the application-level FTP and P2P perform almost equally.



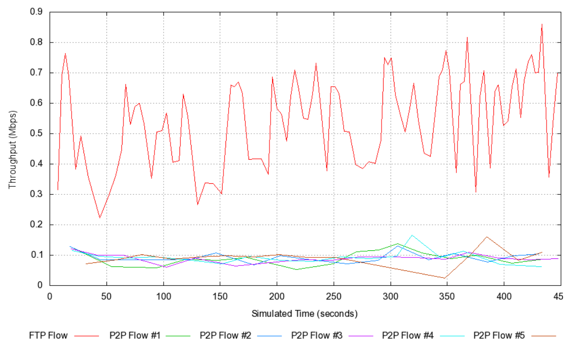
a)



b)



c)



d)

Figure 5.7: Average Throughput over time for concurrent FTP and P2P TCP connections
a) Best-effort; b) Weighted-RED; c) Traditional Diffserv; d) CAPS

In section 2.2 the aggressive nature of P2P application was introduced and illustrated to show the disproportionate share of bandwidth such applications can obtain when coexisting with traditional services. Figure 5.8 shows how CAPS changes this distribution of bandwidth between the two applications.

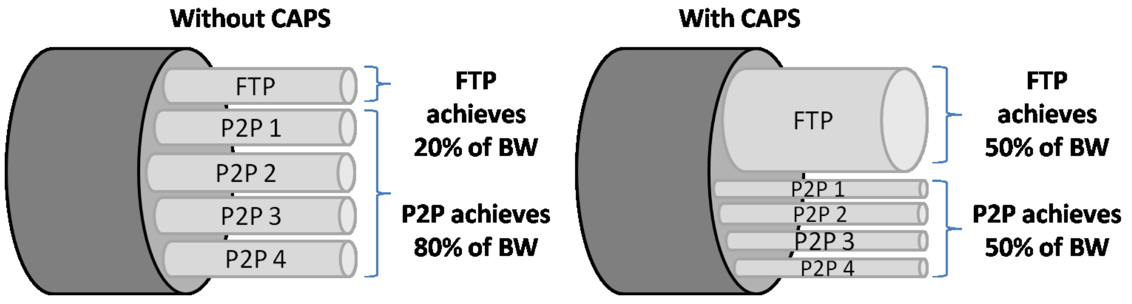


Figure 5.8: Distribution of bandwidth between FTP and P2P applications, with and without CAPS

5.4.4. Performance Analysis for HTTP traffic

The final Internet service that was simulated in phase-1 was HTTP. The CAPS algorithm aims to optimise the handling of HTTP traffic by protecting the first 12 packets of an HTTP flow to minimise risk to the application-level performance during the sensitive *slow start* phase of the TCP connection. The Packmime HTTP web traffic framework outputs an additional trace file along with the standard *ns*2 trace, which provides the response time for each simulated HTTP request-response event. Given that this is a key metric relating to application performance for HTTP traffic, the cumulative distribution function for request-response times were plotted.

Figure 5.9 below shows the CDF plot for the simulated HTTP traffic. There is a noticeable difference in the performance of CAPS when compared with the alternative configurations. It is observed that under the management of CAPS, 90% of the HTTP flows completed in approximately 5.5seconds, whereas for the alternative configurations the plot indicates that it's closer to 12.5seconds for 90% of the flows to complete, demonstrating a sharp increase in performance under the novel management.

Further investigation was conducted to understand how CAPS achieved this sharp difference in performance.

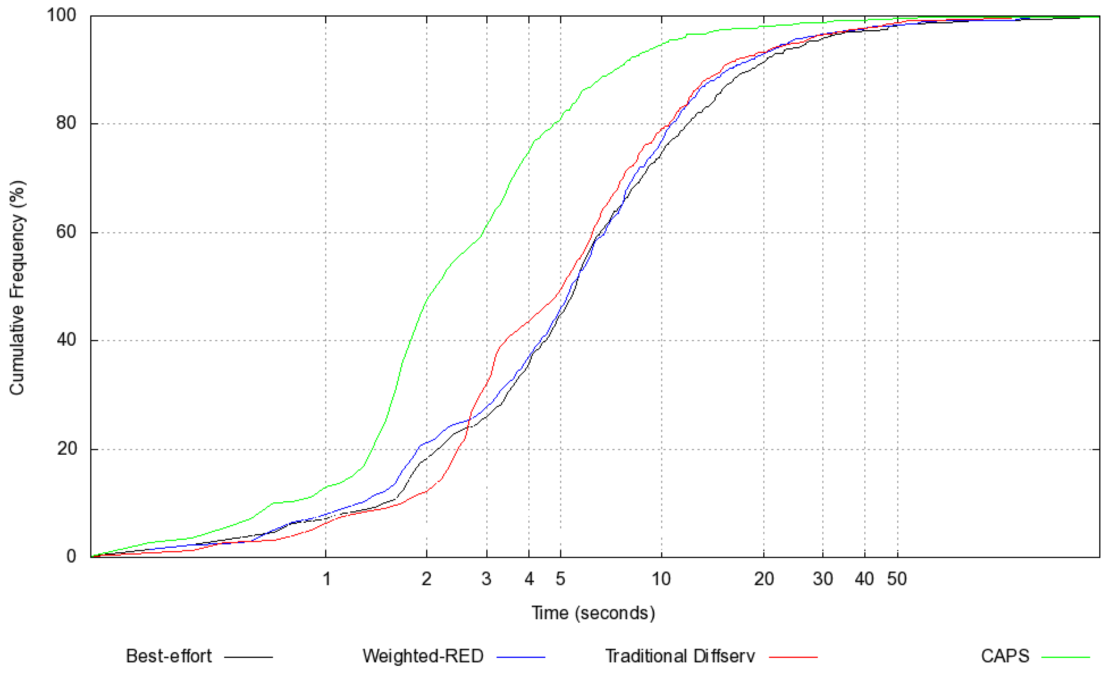


Figure 5.9: Cumulative Distribution of HTTP Request-to-completed-Response time (seconds)

Further analysing the trace files from the simulations revealed the number of HTTP packet drops that occurred throughout the simulation. Table 5.5 provides the number of packet drops for each of the tested configurations. The results show that for Best-effort and WRED configurations there were 380 and 422 packet drops respectively, whereas for the Diffserv and CAPS configurations this number was approximately reduced by 50%. It is highlighted that despite the reduction in packet drops for Diffserv and CAPS, the number of discrete HTTP flows is almost identical across all configurations; dispelling the possibility that lower number of packets could be due to fewer active HTTP flows.

Rather, for Best-effort the poor performance is not unexpected, there exists no preference for any of the traffic types, resulting in packets drops across all traffic types due to the network congestion. For WRED the poor performance is attributed to a number of limiting factors of this configuration. Firstly, although three traffic classes are defined with varying drop probabilities this doesn't offer any explicit protection for sensitive traffic types, it merely raises the probability that packets from less favoured services will be dropped first. In a congested network packets are still dropped from all of the traffic classes, just with varying probabilities. Looking more closely at the number of packets dropped for each traffic type under WRED, it can be seen that both VoIP and HTTP flows experienced higher drop rates than FTP and P2P, which is not expected behaviour. Further investigation revealed that the *ns*2 WRED algorithm calculates the queue size using the number of packets in the queue, without regard for the size of these packets, inadvertently favouring flows with larger packets (a traffic source sending a large number of small packets will occupy more of the queue than a source sending fewer larger packets). For both VoIP and HTTP traffic the packet sizes were a lot smaller than for FTP and P2P, resulting in an unfair distribution of drops for these traffic types.

Configuration	Number of HTTP Flows	Number of Dropped HTTP Packets	Mean Request-Response Delay (seconds)
Best-effort	659	380	9.65
WRED	658	422	8.73
Traditional Diffserv	662	179	8.1
CAPS	669	212	4.49

Table 5.5: Packet drops and average delay for HTTP traffic

In summary for the handling of HTTP traffic, CAPS successfully provides protection to short TCP flows, protecting from packet drops due to policing during the *slow-start* phase of TCP. This combined with the ability to prevent aggressive applications to obtain an unfair share of the bandwidth resulted in a significant improvement in HTTP response times over alternative configurations.

5.4.5. Analysis of Per-RTT Handling of TCP Flows

The last feature of the CAPS algorithm to be discussed is the effectiveness of the per-RTT handling for TCP flows. Earlier in this thesis in section 2.2.2.2 the damaging effects of packet loss to a TCP flow were explored, primarily focussing on TCP Reno, which was considered a good base-implementation. It was observed that following a loss event (duplicate acknowledgements due to packet drop or expiration of the RTO timer) when using TCP Reno the congestion window would be reduced by either 50% or reset to 1, depending upon the type of loss event. During the development of the CAPS algorithm (section 4.6.2.2) it was proposed that traditional methods of traffic policing were too aggressive, given that they operate on a per-packet basis, while TCP requires at minimum one round trip time to react to packet loss (indicating congestion). To this end it was proposed that the CAPS algorithm would estimate the RTT of the TCP connection and use this to determine the frequency that traffic management would be enforced at. The aim of this method is to prevent dropping additional packets from a TCP flow before the sender side has had the opportunity to detect the first loss and react.

However, the decision to use RED queues for traffic management in the CAPS architecture reduced the effectiveness of per-RTT handling of TCP. This was because packets from misbehaving flows were not explicitly dropped, but rather enqueued into a more aggressive RED queue. The random drop characteristics of RED make it highly unlikely that two adjacent packets are dropped from a single flow. Although the original intentions of per-RTT based handling were reduced somewhat by the use of RED queues, a comparison between a traditional per-packet based policing approach and the novel per-RTT approach is included in the following text to investigate any impact on TCP performance.

A set of simulations were conducted in which two heterogeneous TCP flows were configured, one with a round-trip time of 150ms and the second of 300ms (flows 1005 and 1006 respectively in Figure 5.10). In the absence of any traffic management mechanisms (best-effort behaviour) it was observed, as expected, that the flow with a shorter RTT obtained a great share of the bandwidth, Figure 5.10

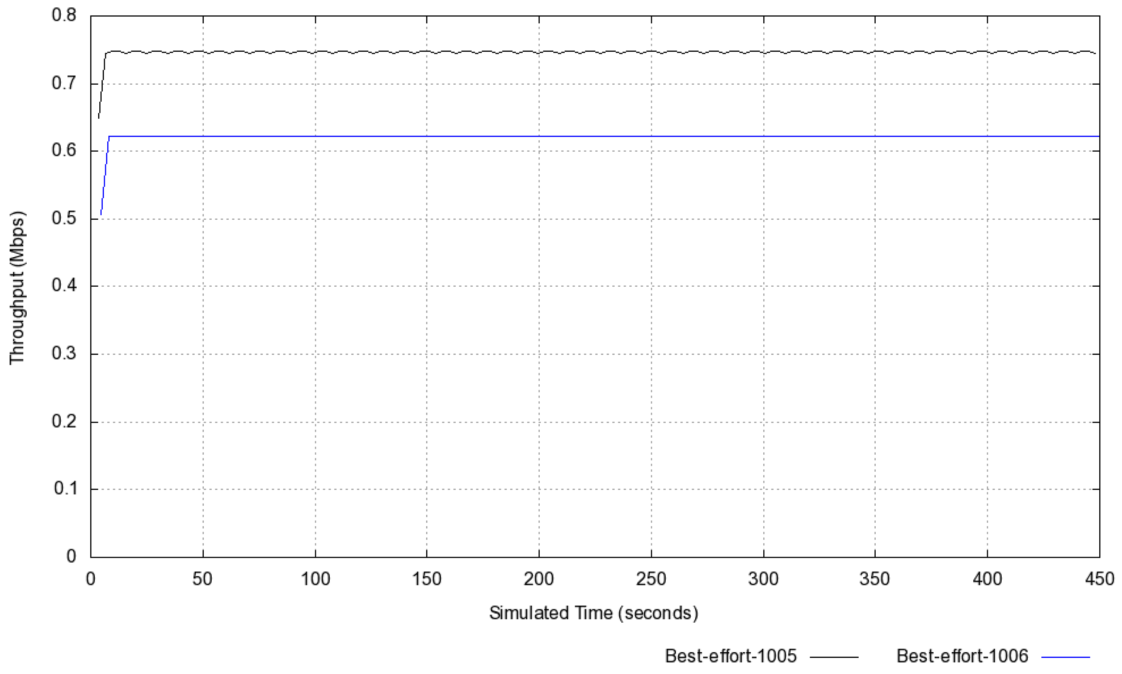
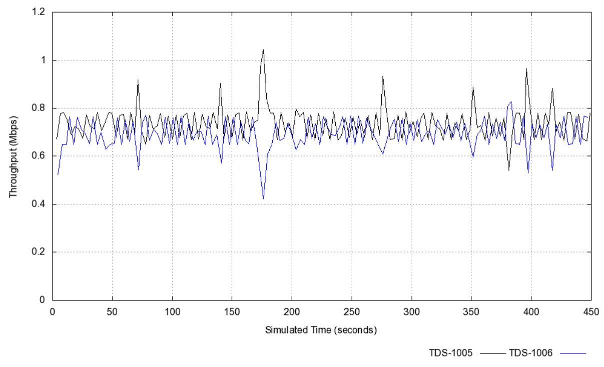
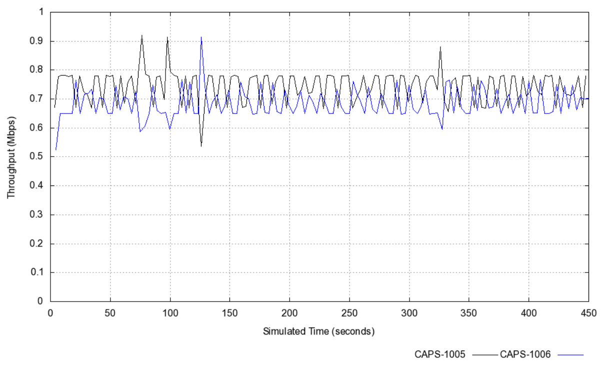


Figure 5.10: Heterogeneous TCP flows achieving dissimilar throughputs in Best-effort conditions

The same flows were then simulated using a Traditional Diffserv per-packet policer and the per-RTT policing of CAPS, with each scenario attempting to police the flows evenly. Figure 5.11 provides the throughput for each flow under each scenario. The per-packet handling of Traditional Diffserv resulted in almost constant policing of the 'shorter' flow (1005) to allow the 'longer' flow (1006) to obtain an equal share of the bandwidth, with average throughputs being 0.73Mbps and 0.69Mbps for flow 1005 and 1006 respectively. The plot for per-RTT handling (Figure 5.11 (b)) shows significantly fewer oscillations between the two flows, yet the average throughputs for each flow were almost identical to the per-packet handling, at 0.74Mbps and 0.685Mbps for flows 1005 and 1006 respectively. In recognition that a pure average throughput can sometimes misrepresent the performance of a flow, the total volume of data delivered to each destination node was also calculated for each scenario. Under per-packet handling the destination for flow 1005 received 41.2Mbytes, and the destination for flow 1006 38.5Mbytes; whereas using per-RTT handling 41.6Mbytes were delivered to the destination of flow 1005 and 38.4Mbytes to the destination of flow 1006, demonstrating that a gentler policing mechanism can provide fairness between heterogeneous flows at no cost to flow performance.



(a) Traditional Diffserv per-packet policing



(b) CAPS per-RTT policing

Figure 5.11: Comparison of Throughput for heterogeneous TCP flows under different policing mechanisms

This behaviour is quantified in Table 5.6 which provides the number of times a flow was policed to a more aggressive RED queue and the number of subsequent packet drops for each scenario.

Flow	Total Packets Sent	Number of Downgrade Events	Number of Packet Drops
Short-RTT Per-packet handling (1005)	28,874	2,963	67
Long-RTT Per-packet handling (1006)	26,998	1,357	10
Short-RTT Per-RTT handling (1005)	29,149	1,719	48
Long-RTT Per-RTT handling (1006)	26,842	449	3

Table 5.6: Comparison of per-Packet and per-RTT Policing Downgrade Events

5.4.6. Summary of Phase-1 Results

This section of the thesis has provided the results from the first evaluation phase of the CAPS architecture. The architecture was developed using the *ns*2 simulation package, and a small-scale network topology was created to demonstrate how CAPS handles four different traffic types: VoIP, FTP, P2P and HTTP. The first test case that was considered was for VoIP traffic. The proposed architecture was shown to be able to evaluate the current performance of a VoIP flow through the calculation of its R-factor. The traffic manager then used this metric as a target, adjusting resource allocation among co-existing flows, to ensure an acceptable R-factor was achieved. The results from this section provide evidence that this management was successful, with CAPS being able to provide an average R-factor of 74.8, without the need for static resource reservation. When compared against best-effort and weighted-RED, CAPS performed significantly better, and only marginally behind traditional Diffserv, despite placing no precedence on VoIP traffic.

The second test-case considered the management of FTP traffic, for which CAPS aimed to ensure a throughput at least equal to the fairshare of bandwidth entitled to the service. The results confirmed that CAPS was able to allocate at least this fairshare, also permitting FTP to exploit any residual bandwidth when available. When comparing the performance of CAPS against the alternative configurations it was clear that the explicit priority placed on VoIP traffic by Diffserv was at the detriment of co-existing flows, with FTP being able to achieve only a fraction of its entitlement. Similarly, an absence of control or management over P2P traffic led to a reduced performance for FTP under all alternative configurations.

The third test-case considered P2P applications, which CAPS aimed to ensure could not achieve a disproportionate share of available bandwidth simply through establishing multiple TCP connections. The results from the simulation successfully demonstrated the management of P2P traffic, restricting the aggregate application throughput did not exceed that of an FTP application, however, without explicitly discriminating against users wishing to engage in such services. The alternative services were demonstrated to allow each TCP flow to obtain a per-flow share of the bandwidth, resulting in a significant unbalance between FTP and P2P throughput.

The final traffic type to be evaluated was HTTP, for which the CAPS architecture aimed to protect the first 12 packets of each flow. This protection reduced the likelihood of packet drops for short HTTP flows, which rarely exit the vulnerable *slow-start* phase of TCP. The measures taken were shown to result in less than half the number of packet drops observed under best-effort and weighted-RED configurations, and a similar number to the Diffserv configuration. However, the average request-response time observed under CAPS was 50% faster than that of Diffserv, which is hypothesised to be due to explicit priority for VoIP traffic provided by Diffserv.

The final test-case to be reviewed in this section was evaluating the performance of the per-RTT handling provided by the CAPS architecture, compared with the per-packet handling of traditional mechanisms. The proposed mechanism was shown to be able to provide a similar level of control over the throughput of a flow, with fewer packet drops. However, the decision to integrate RED queues into the proposed architecture vastly reduced the likelihood of bursty packet drops, which reduced the effectiveness of the proposed method.

5.5. Simulation Phase 2 – Evaluation of CAPS for a large scale network

The previous section demonstrated the operation of the CAPS algorithm for a number of key traffic types. For each of the traffic types CAPS was able to provide an acceptable level of service to the user without the need for explicit resource reservations or prior knowledge of the network traffic profile.

Having proven that CAPS can provide an optimised delivery for a mixed user base the next stage of evaluation was to increase the scale of the simulations.

This section of the thesis presents the second phase of the simulations, which involved simulating a multi-ISP network with thousands of simultaneous flows between hundreds of customer nodes.

5.5.1. Simulated Network Topology for Phase-2

This large scale network topology is illustrated in Figure 5.12, it consisted of three ISP networks that were interlinked via an Internet Exchange Point (IXP) (for simplicity the IXP was a router with sufficient capacity and negligible delay to ensure that no loss or meaningful delay was incurred transiting this node). Attached to the backbone were three FTP servers, responsible for sourcing FTP application data to client nodes upon request. In the previous section the Packmime HTTP framework was used to generate web-like traffic for evaluation. However, it was not possible to use this framework to simulate large scale HTTP services for the purpose of evaluating CAPS. This was due to the Packmime framework using just two *ns*2 nodes as the source and destination of all HTTP traffic (each node is simulating a 'cloud' of HTTP users). While the source node could be configured to generate a higher volume of traffic, thus representing more HTTP users than the previous phase, as far as the CAPS algorithm and the alternative *ns*2 QoS configurations were concerned, all HTTP traffic would be for a single destination, invalidating any per-user QoS calculations. Unfortunately, because the standard FTP/TCP traffic model in *ns*2 can not be configured to send a specific volume of traffic (rather a start and a stop time are used to turn the source on or off) this could not be used as an alternative to the Packmime framework given that the performance indicator for HTTP is request-response time. In an attempt to circumvent this issue, 90 HTTP sources were also attached to the IXP router, and 30 HTTP sinks were attached to the customer-side of each ISP network. Each source/sink pair were configured as per the previous simulation, each representing a single HTTP user.

VoIP and BitTorrent traffic was sent between ISP client nodes, to best simulate the nature of such traffic.

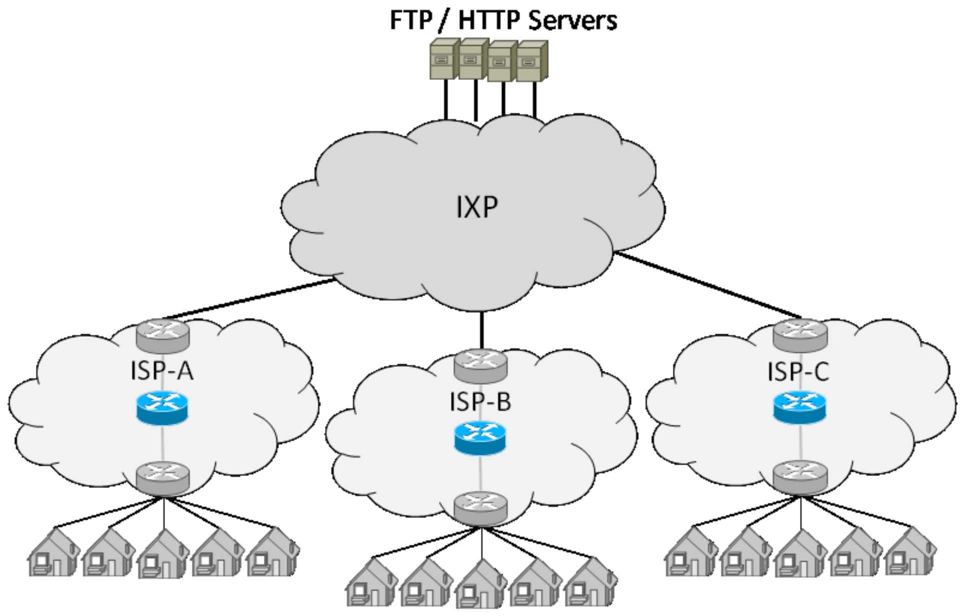


Figure 5.12: Simulated Network Topology

Each ISP network had 100 clients attached, via two simplex links, to model an asymmetric Internet connection of 4Mbps downstream and 400Kbps upstream (again by way of reflecting the average UK residential Internet connection as described in (Ofcom 2008)). Core link capacities within the network were increased accordingly; however, bottleneck links remained contended enough to ensure congestion. The individual bandwidth and delay parameters for each link within the topology are provided in Table 5.7.

Link	Bandwidth	Delay
ISP Edge Router - > ISP Core Router	10Mb	20ms
ISP Core Router -> ISP Edge Router	10Mb	20ms
ISP Edge -> Internet Core	10Mb	20ms
File Server -> Internet Core	50Mb	0.1ms
HTTP Server Cloud -> Internet Core	50Mb	0.1ms
HTTP Client Cloud -> ISP Network	10Mb	5ms

Table 5.7: Link Parameters for Phase-2 Network Topology

5.5.2. Traffic Sources

To bring the phase-2 simulation closer to a realistic environment the configuration of the traffic sources were enhanced from the previous simulation. Instead of constant traffic flows from the start of the simulation trial to the end, the traffic sources were randomly distributed throughout the simulated time. The randomisation of start and stop times for the traffic sources was configured in a manner that was consistent across all of the alternative configurations, to ensure the same traffic load was observed in each case, meaning each QoS configuration was presented with the same volume and mix of traffic. Furthermore, to ensure that traffic sources were not configured for too short a period, or were distributed too sparsely so that the bottleneck never became congested, certain limits were placed on the random time generator. FTP traffic started within 0 – 150 seconds and had durations lasting between 30-240 seconds, BitTorrent flows started between 0 – 150 seconds and lasted between 30-200 seconds, VoIP flows started between 0 – 150 seconds and lasted between 60 – 180 seconds. Finally, the HTTP clouds began their flow generation at 5 seconds and continued throughout the duration of the simulation. Aside from start and stop times, all of the traffic sources were configured with the same parameters as were described in section 5.2.1.

The pairing of customer nodes for the exchange of VoIP and BitTorrent traffic was configured to ensure that traffic transited between ISP networks where possible, this was to ensure limited amounts of local-routing occurred whereby any QoS configuration on the ISP edge would not be applicable.

5.5.3. Traffic Profiles

To further contextualise the simulation closer to reality, each ISP was configured to have customers with different traffic profiles, with the aim to observe how CAPS serves users engaged in concurrent services. In total seven different traffic profiles were simulated; VoIP, FTP, P2P, VoIP & FTP, VoIP & P2P, VoIP FTP & P2P and finally HTTP.

5.6. Analysis of Phase-2 Results

This section of the thesis presents the results from the phase-2 simulations, using a large scale simulated environment. The presentation and analysis has been divided into 7 sub-sections, one for each of the traffic profiles, followed by a summary of the overall performance for each configuration.

5.6.1. Performance Analysis for 'VoIP only' users

The first traffic profile includes users that engaged in solely VoIP services. As per the previous analysis the evaluation for VoIP QoS is conducted by plotting the R-value over time for each of the QoS configurations. Figure 5.13 below illustrates the mean R-factor over time for users only engaged in VoIP. At 50 seconds the network bottleneck began to congest, resulting in a noticeable drop in performance for all configurations. Under a Best-effort configuration a sharp and continued drop in achieved R-factor is observed as the network bottleneck becomes congested and the delivered QoS for the VoIP traffic falls below acceptable bounds. Weighted-RED also struggled to provide an acceptable level of service to the VoIP-only customer traffic, only marginally improving on the behaviour of Best-effort. Unsurprisingly, the traditional Diffserv configuration achieved the highest average R-factor throughout the simulation by providing adequate bandwidth guarantees and favouring the EF queue during queue servicing. The slight reduction in R-factor for the VoIP traffic under the Diffserv configuration was caused by a small number of packet drops and largely an increase in delay, both of which occurred during periods of high congestion. The CAPS algorithm performed closely behind Diffserv, maintaining a mean R-factor of 78.4 between 50 and 250 seconds, which was highest period of most congestion, this is a 10% and 17% improvement on the performance of WRED and Best-effort respectively. The achieved R-factor is comfortably above the target value of 75 and further validates that the dynamic application-level method of management for VoIP traffic still performs as desired in larger scale environments.

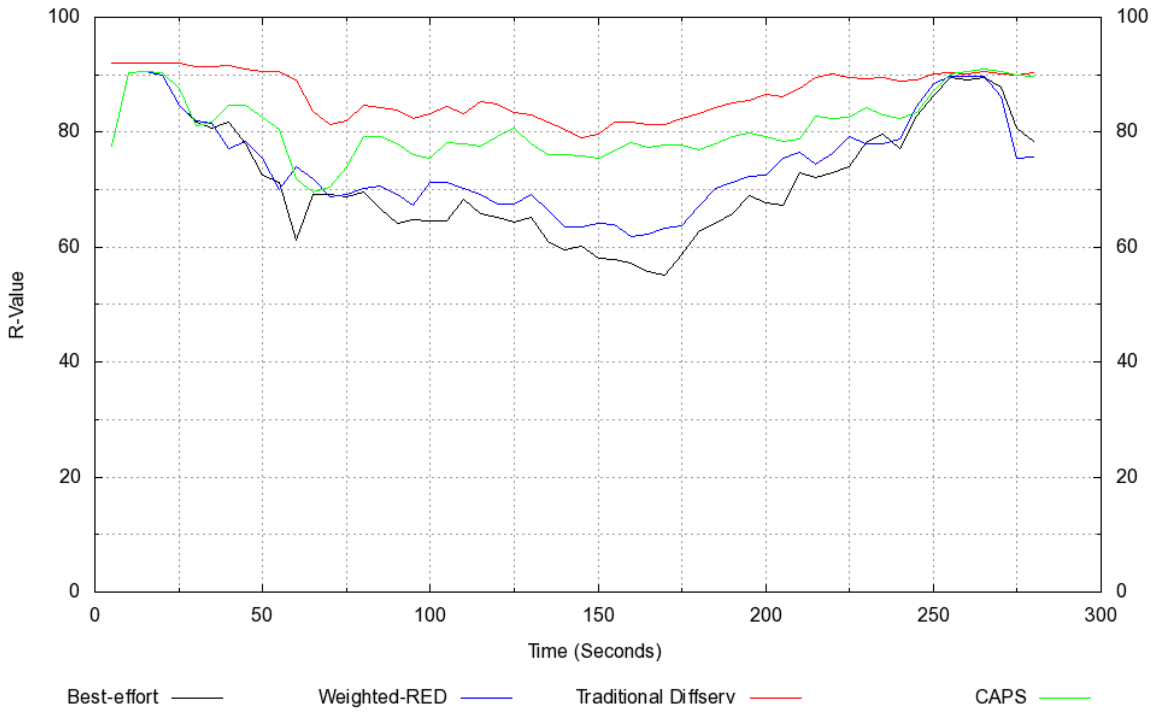


Figure 5.13: Average R-factor over time for 'VoIP only' users

5.6.2. Performance Analysis for 'FTP only' users

The second traffic profile used simulated users of only FTP services, the average throughput over time for each configuration is plotted in Figure 5.14 along with the statistical bandwidth share that each user was entitled to. The first observation is the poor performance under Traditional Diffserv, which achieved a mean throughput of just 0.15Mbps throughout the simulation, just 40% of the statistical fairshare for that flow. Although steady at this rate, the explicit priority provided to guarantee VoIP performance contributes towards bandwidth starvation for co-existing traffic. Furthermore, the chosen configuration of Diffserv attempted to offer an assured service for FTP and HTTP traffic (target CIR) however, it is clear that this configuration struggles to enable flows to rapidly exploit any residual bandwidth that may become available.

The average FTP throughput during congestion (between 50-250seconds) under the CAPS configuration is 0.78Mbps. Although this value is above the statistical fairshare it can be seen in

Figure 5.14 that the throughput is policed on a number of occasions (at 80-105 seconds and 155 seconds) closer to the fairshare. This behaviour demonstrates how CAPS allows for residual bandwidth to be benefited from when possible, but that the algorithm retains control over these flows to reclaim bandwidth when needed in order to satisfy the fairshare requirements of other users / flows.

Best-effort and WRED configurations performed very similarly with respect to FTP throughput. For both of these configurations the FTP throughput far exceeded the fairshare, claiming more than 300% of the user's fairshare during the highest period of congestion. Although utilising available bandwidth is a positive attribute from the perspective of the FTP application (and user), this result has been achieved at the detriment of co-existing user traffic (as seen in the previous section and is also seen in the following sections).

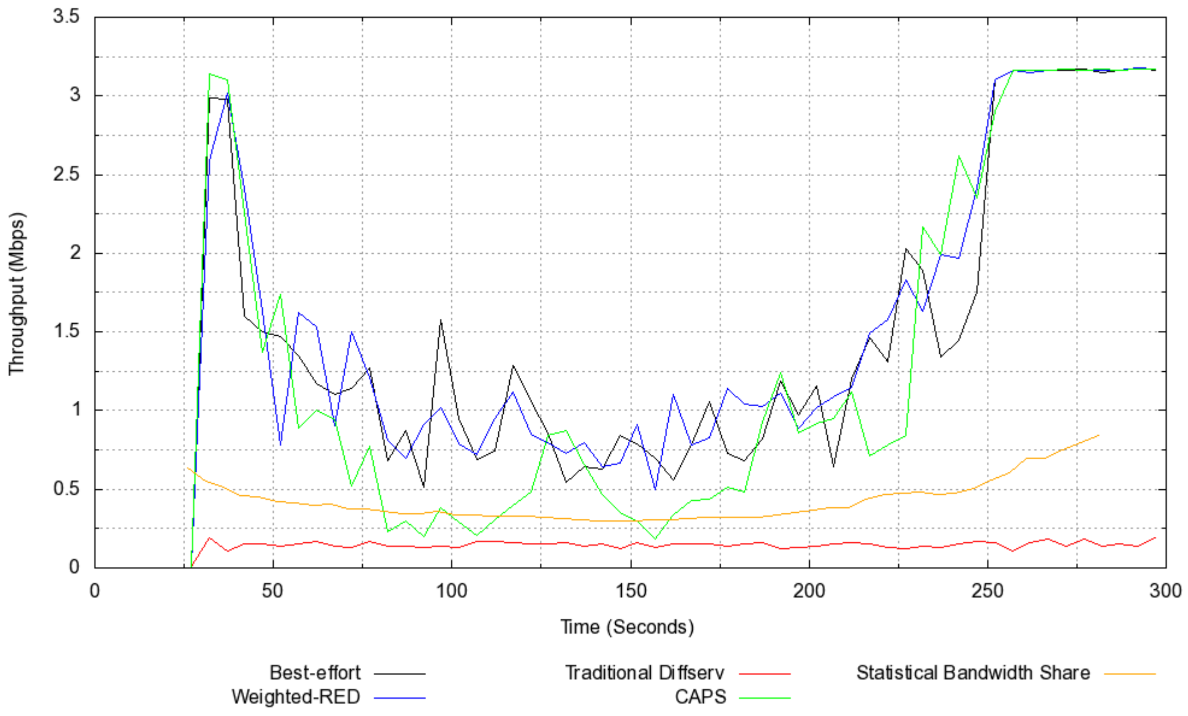


Figure 5.14: Average FTP throughput for 'FTP only' users

5.6.3. Performance Analysis for 'P2P only' users

The third traffic profile represented users of only P2P traffic, which compared with the previous two profiles differs in the sense that the traffic is composed of multiple TCP flows from multiple sources (as per the P2P model). Figure 5.15 provides the average throughput achieved by P2P-only users.

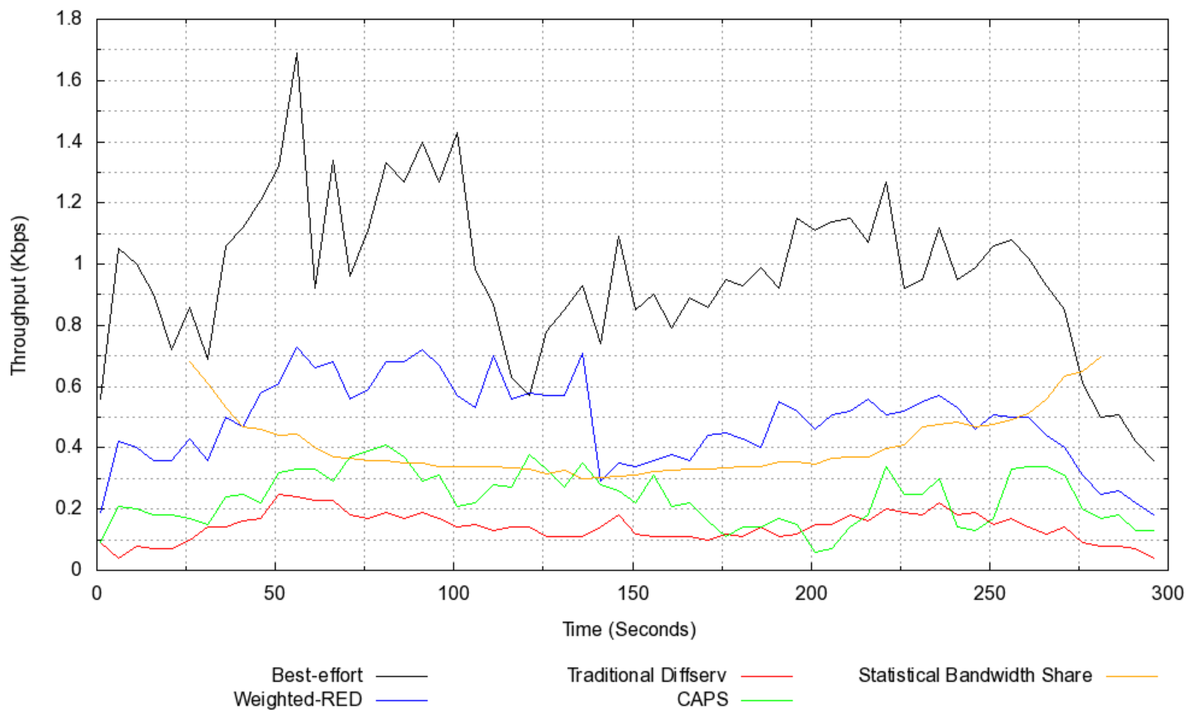


Figure 5.15: Average aggregated P2P throughput for 'P2P only' users

The aggressive, bandwidth hungry nature of P2P traffic is demonstrated in the Best-effort configuration, where the mean P2P throughput was close to 1Mbps for the full duration of the simulation – far exceeding the fairshare the users were entitled to. Weighted RED was configured to favour P2P traffic the least, the benefit of this is illustrated in Figure 5.15 where the average throughput is shown to be approximately 50% of that achieved in the Best-effort environment. In the case of Traditional Diffserv the achieved throughput remains below the fairshare value for the entire simulation, averaging around 0.15Mbps, just 50% of the fairshare. The Traditional Diffserv configuration was designed to limit the throughput of P2P traffic (in addition to guaranteeing VoIP performance) and the negative effect of this approach for users of 'less preferred' traffic is clear from this test. The CAPS configuration achieved an average throughput within 10% of the fairshare for the duration of the simulation, which demonstrates the ability to manage aggressive application traffic without the need for static configurations. There is a clear drop in throughput across all configurations around 120 seconds, this was attributed to a number of the P2P sources completing their transfers, which in turn resulted in a drop in aggregated throughput. This was not due to external policing or traffic management.

5.6.4. Performance Analysis for 'VoIP & FTP' users

Having considered three traffic profiles for users of a single service the results that follow describe how the CAPS algorithm performs when a user is engaged in multiple activities at once. The first traffic profile that is considered is for users engaged in VoIP and FTP services; Figure 5.16 provides a plot for the average R-factor over time for these users.

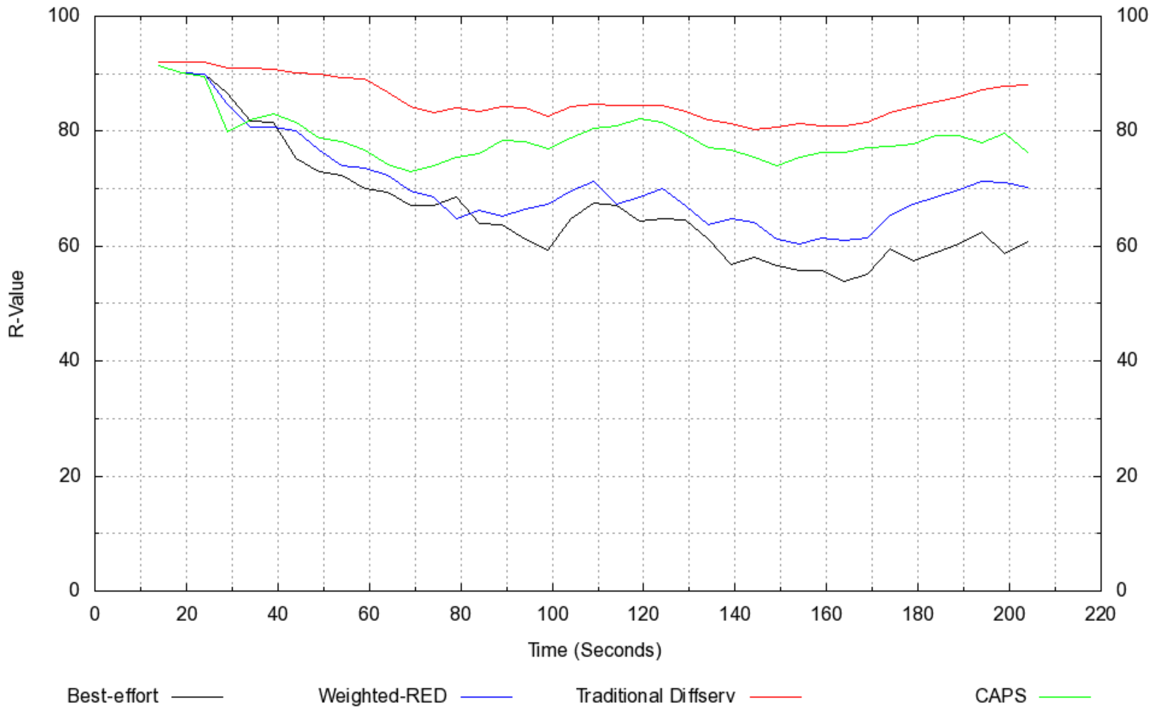


Figure 5.16: Average R-factor over time for 'VoIP & FTP' users

The performance of VoIP traffic under Best-effort and WRED configurations continually degraded in quality (achieved R-factor) as the level of congestion increased, following a similar trend to the previous similations. Traditional Diffserv once again sustained the highest R-factor throughout the simulation, with a mean value of 85, degraded only marginally due to a few packet drops and increased delay (due to having to use WRR scheduling rather than a strict priority implementation). In a manner similar to that shown for users of just VoIP services, CAPS managed to successfully maintain the R-factor close to the target of 75, producing a mean R-factor of 78.

Figure 5.17 below plots the average FTP throughput achieved by the users of VoIP and FTP along with the statistical bandwidth share.

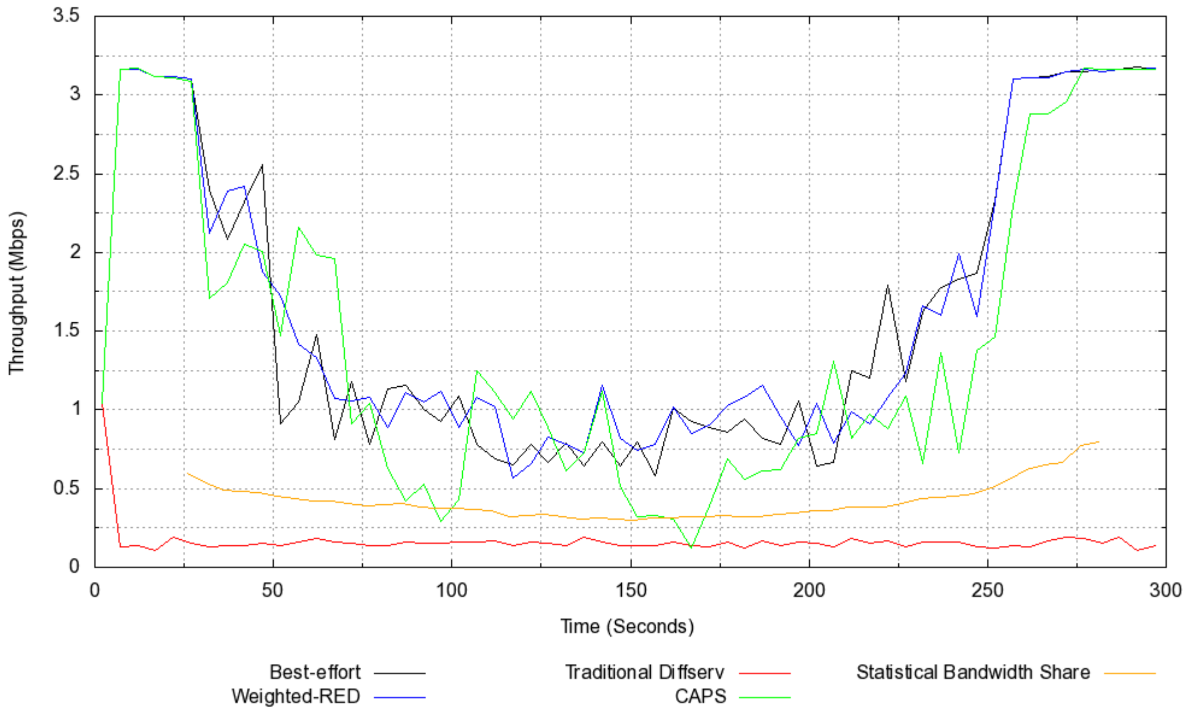


Figure 5.17: Average FTP throughput for 'VoIP & FTP' users

The first point to note from the above graph is that again Traditional Diffserv has failed to provide adequate resource to the FTP traffic, limiting its throughput by ensuring guarantees for VoIP performance. Throughout the simulation the Traditional Diffserv configuration prevents the FTP flows from obtaining its fairshare of bandwidth, once again resulting in an unfair service delivery for unfavoured traffic types.

In contrast to this, Figure 5.17 shows how Best-effort, WRED and CAPS allow the FTP traffic to achieve a far greater throughput, averaging approximately 1Mbps for Best-effort and WRED and 0.85Mbps for CAPS during the period 50 – 250 seconds. Although under CAPS the FTP traffic typically exceeded the fairshare, there are occasions (75–100 seconds and 150-175 second) when CAPS can be seen to reduce the throughput of FTP in order to reallocate resources.

5.6.5. Performance Analysis for 'VoIP & P2P' users

The fifth traffic profile that was simulated represented users of VoIP and P2P applications, with Figure 5.18 and Figure 5.19 below illustrating the performance of each traffic type.

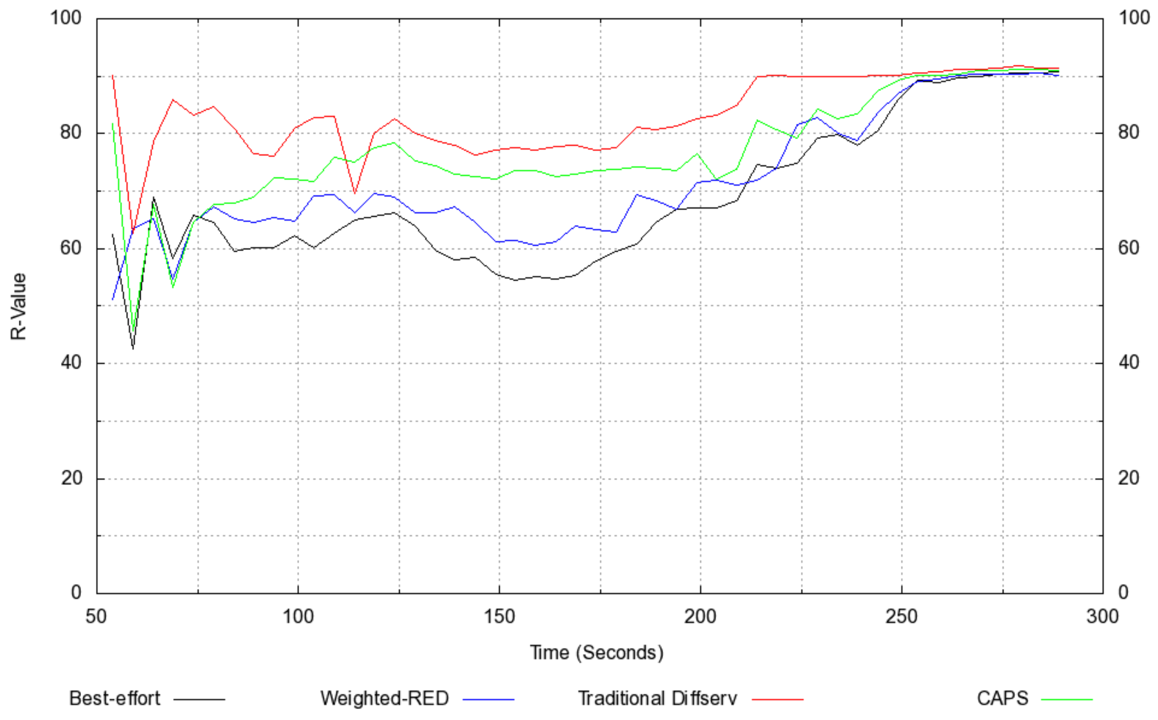


Figure 5.18: Average R-factor over time for 'VoIP & P2P' users

Comparing the graph above with the previous VoIP performance plots, Figure 5.18 illustrates the negative effect P2P traffic has when co-existing with VoIP flows is evident. For all configurations the performance of VoIP has been negatively affected. Unsurprisingly Best-effort and WRED perform the worst, managing an average R-factor of 64.8 and 68.5 respectively for the period between 50 and 200 seconds. Although Traditional Diffserv did experience a number of short periods of degradation it again achieved the highest R-factor value, averaging 82.5 during the highest period of congestion, 50-200 seconds. For the CAPS configuration there was a short period between 50-75seconds where the R-factor was below 70. This is accounted as due to a delay in CAPS being able to sufficiently police co-existing traffic, for which TCP had already established itself and was exploiting available bandwidth. Despite this initial behaviour the average R-factor achieved by CAPS between 50-200 seconds was 74.7, again remarkably close to the target R-factor of 75. Beyond 200 seconds the overall level of network congestion was reducing, which allowed all configurations to benefit from additional network resource.

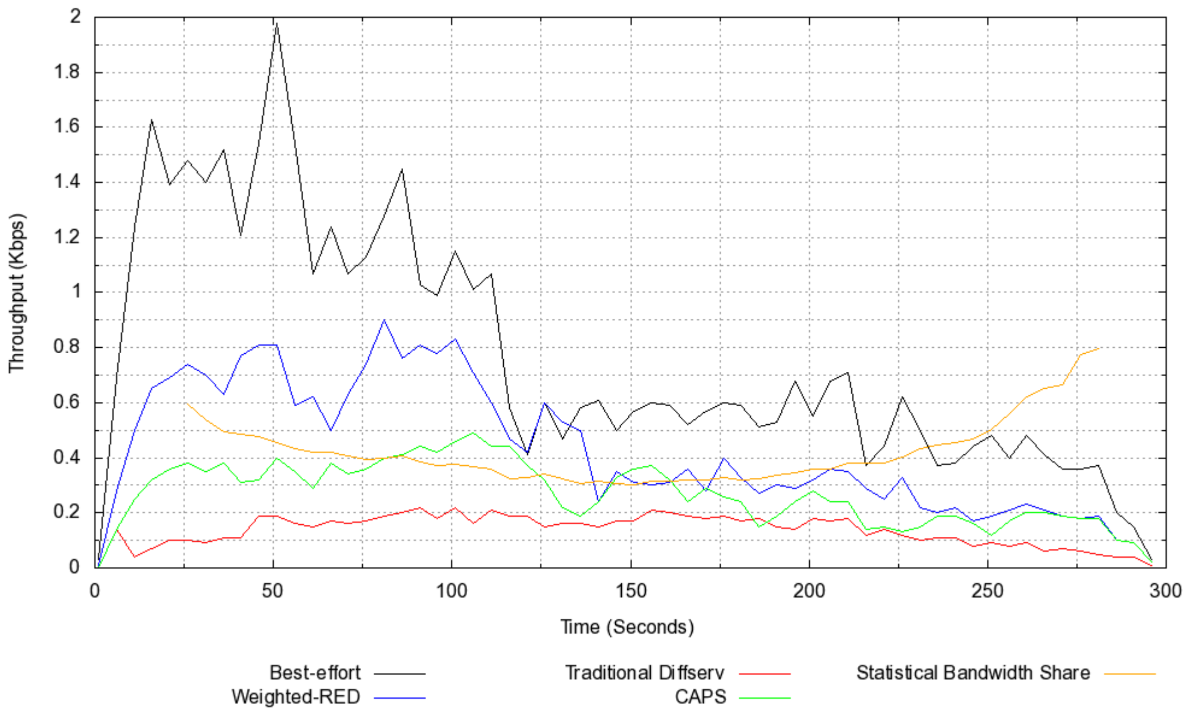


Figure 5.19: Average aggregated P2P throughput for 'VoIP & P2P' users

Figure 5.19 provides the average aggregate throughput over time for VoIP and P2P users for each of the configurations. As has been observed in the previous results, Best-effort does nothing to limit or restrict the throughput of multi-sourced applications, such as P2P. As a result it can be seen that the P2P traffic under Best-effort conditions was able to obtain far beyond its fairshare of bandwidth, for the first 125 seconds averaging 1.2Mbps, nearly 4 times its fairshare. Beyond 100 seconds the randomly distributed sources begin to tail off, which is evident by the drop in throughput after this point. Weighted RED demonstrated that its mechanism for dropping P2P traffic over co-existing traffic assisted in limiting its throughput, although it only managed to restrict P2P to 0.66Mbps for the first 125 second, which was still more than twice the statistical fairshare.

In contrast to Best-effort and WRED, Traditional Diffserv continued to starve non-favoured services of bandwidth, with P2P achieving a steady throughput of approximately 0.2Mbps, slightly below the fairshare allocation.

In keeping with the previous results CAPS managed to maintain the average throughput of P2P close to the fairshare, preventing P2P from disproportionately obtaining any residual bandwidth. As with the other configurations beyond 100 seconds the number of traffic sources reduced, which accounts for the tailing seen in the graph.

5.6.6. Performance Analysis for 'VoIP, FTP & P2P users

The last composite traffic profile was designed to see how CAPS would perform if given the task of managing a user's traffic if they engage in all three services at once.

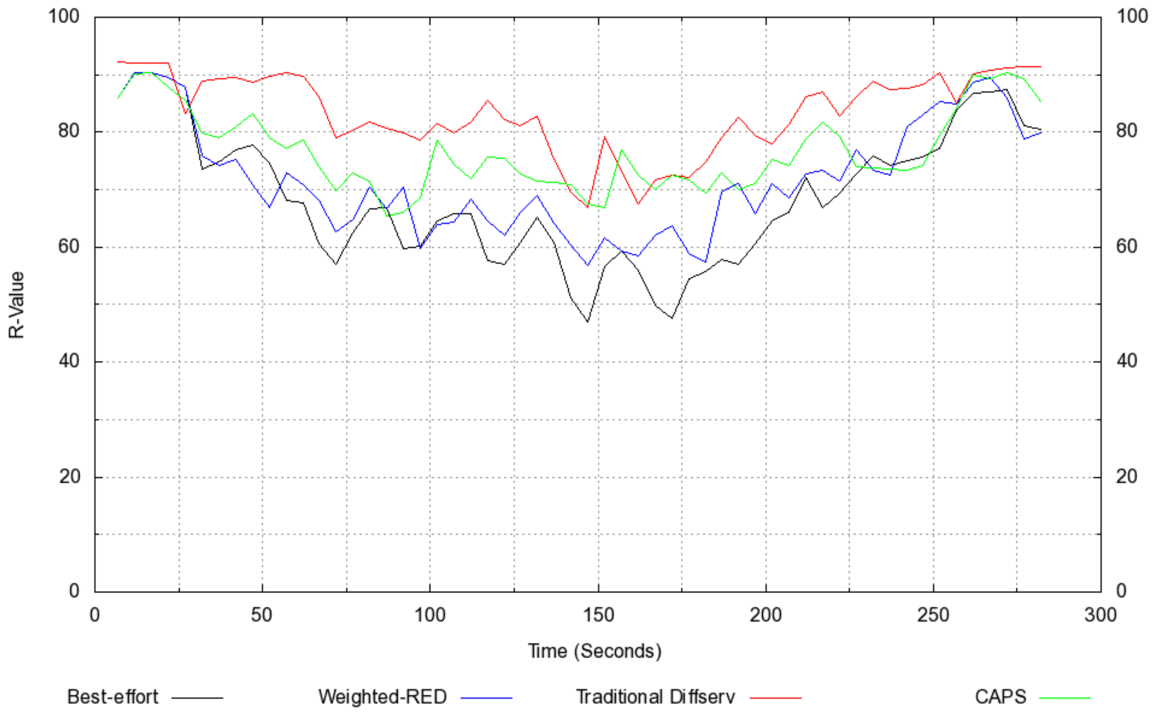


Figure 5.20: Average R-factor over time for 'VoIP, FTP & P2P' users

The first traffic type to be evaluated is VoIP, which Figure 5.20 provides a similar result to the previous evaluations. Best-effort and WRED offered the worst performance, averaging an approximate R-factor of 60 during periods of high congestion, and with Best-effort dropping to below 50 on two occasions. In such situations this level of QoS delivered to a user would cause serious dissatisfaction, and in all likeliness result in the user terminating the service, therefore, it is not an acceptable delivery. In contrast to how Traditional Diffserv has performed in previous tests, when faced with provisioning for multiple services at once, there were fluctuations in the delivered R-factor. In a real-world implementation of Diffserv this fluctuation is unlikely to have occurred, since a strict priority queuing model would have been used. However, this being said, serving EF traffic with strict priority could *only* have a negative impact of co-existing traffic, so while VoIP may have performed better the same can not be said for all traffic.

The performance of VoIP flows for this traffic profile when using CAPS was marginally lower than previous results, with an average R-factor of 73.2 for the period 50-225 seconds. However, this proves that CAPS can maintain an R-factor very close to the target, even under network conditions that proved difficult for Traditional Diffserv to handle.

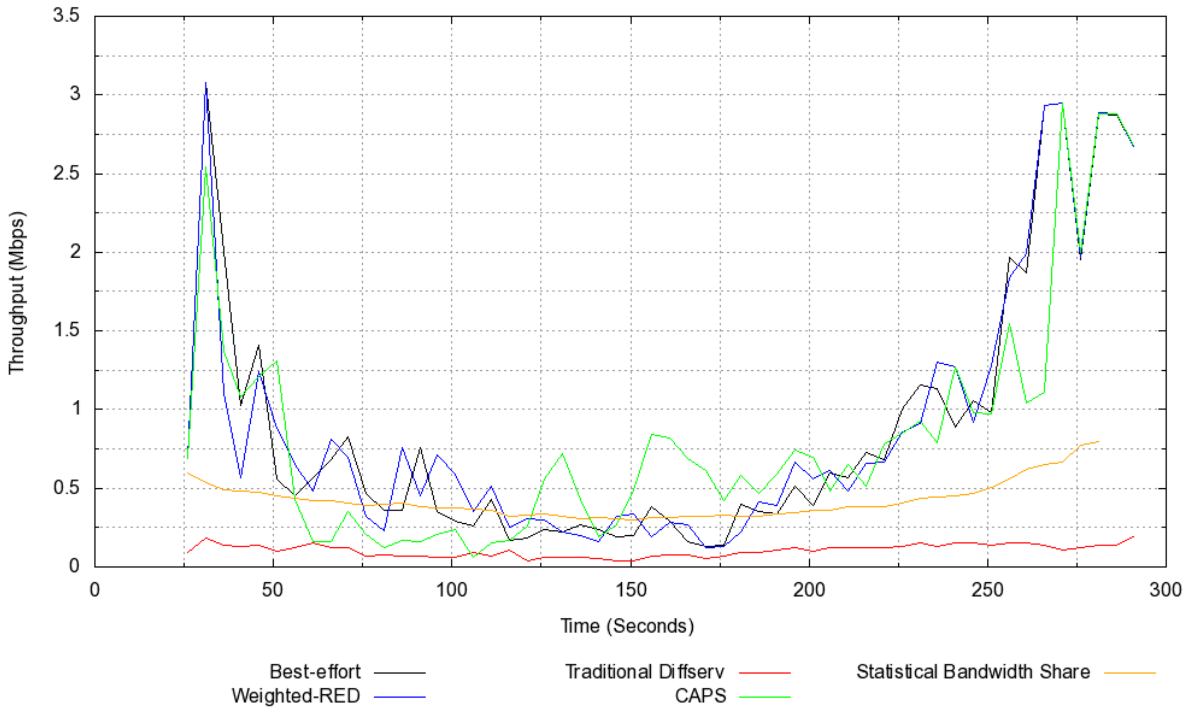


Figure 5.21: Average FTP throughput for 'VoIP, FTP & P2P' users

Figure 5.21 shows the throughput for the co-existing FTP services, and for Traditional Diffserv the results echo those from previous traffic profiles – explicit guarantees for VoIP lead to resource starvation and poor performance for non-favoured traffic. Best-effort and WRED performed very closely under these circumstances, both achieving approximately 0.5Mbps between 50-200 seconds, which was the most heavily congested period. The performance of the FTP traffic under CAPS appears at first to loosely follow the trend of Best-effort and WRED, however, differences can be identified when Figure 5.21 is considered alongside Figure 5.22 (average aggregated P2P throughput for VoIP, FTP & P2P users). It is observed that CAPS manages the FTP and P2P services in a manner that ensures they each achieve similar throughputs of 0.49Mbps and 0.42Mbps respectively for the period of high congestion (50-200seconds).

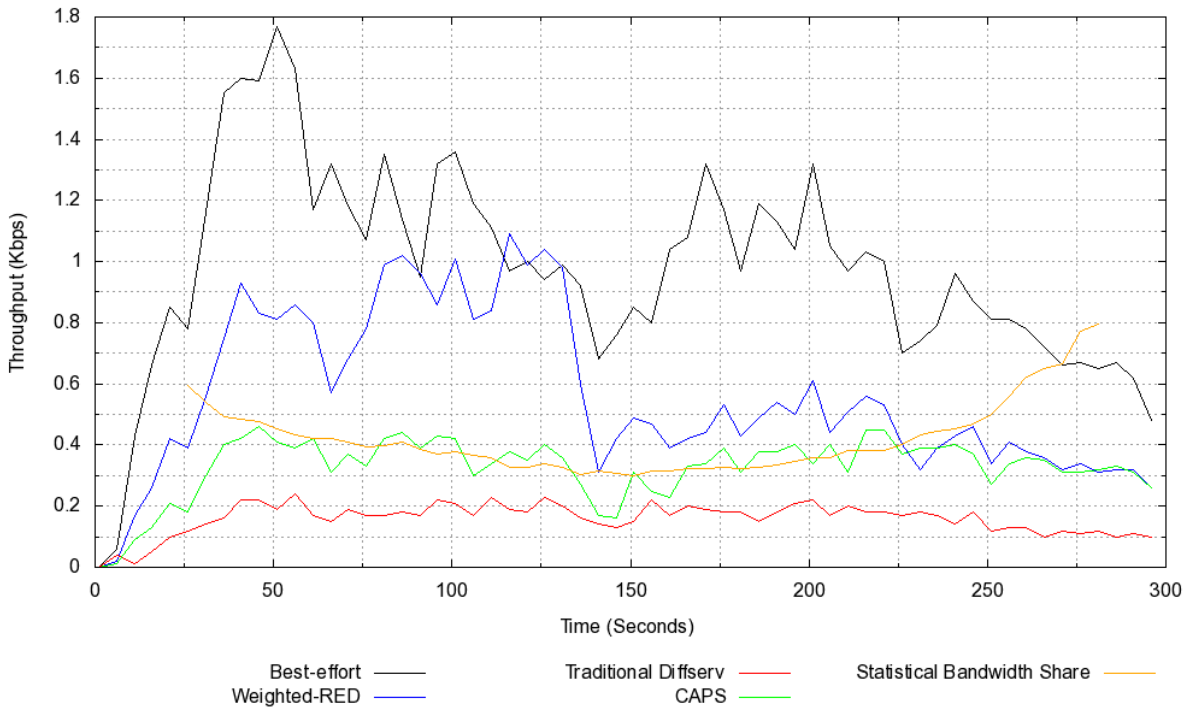


Figure 5.22: Average aggregated P2P throughput for 'VoIP, FTP & P2P' users

The performance results of Best-effort, WRED and Traditional Diffserv were very similar to those presented in section 5.6.5, with Best-effort and WRED failing to prevent P2P traffic from obtaining far beyond its fairshare. The plot for Traditional Diffserv throughput illustrates once again the configuration to guarantee QoS for VoIP combined with actively discriminating against P2P traffic results in an unfair delivery to users of such services.

5.6.7. Performance Analysis for HTTP Traffic

The last traffic type to be evaluated in the large scale topology was HTTP. In section 5.4.4 CAPS was demonstrated to provide a significant improvement to the request-response time for web-like traffic, simulated by the Packmime framework for *ns*2. As mentioned in section 5.5.1 there were a number of difficulties in scaling the HTTP configuration. The method chosen for evaluation was to use 30 HTTP source/sink pairs for each ISP network, and configure each pair as a single HTTP user. Although this allowed for each HTTP pair to be considered a discrete end-user from the perspective of the QoS algorithms, the results were not as had been hoped. Figure 5.23 provides the distribution for the average request-response times for each validation case, which is significantly different to the results from phase 1 (see section 5.4.4). In addition to CAPS performing almost identically to Best-effort, Traditional Diffserv and WRED behave in an almost identical manner.

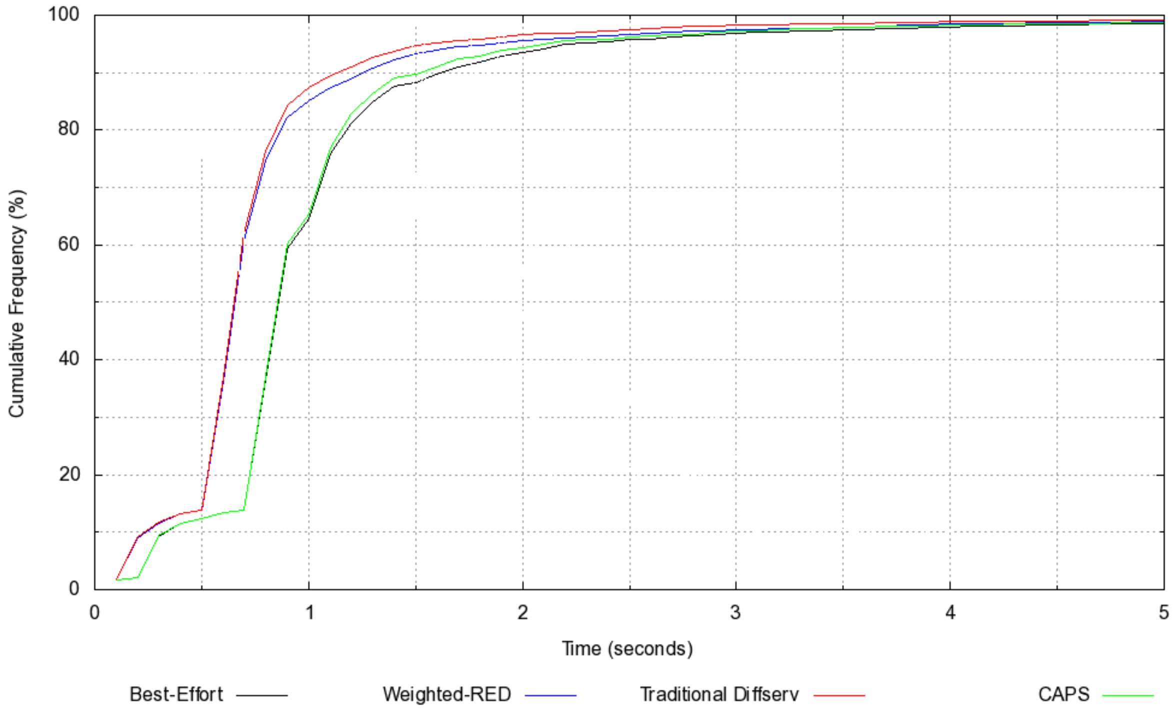


Figure 5.23: Cumulative Distribution of HTTP Request-to-completed-Response time (msec)

Following a detailed analysis of the simulation trace files the cause of the anomaly was identified. For all four validation cases the HTTP flows were modelled identically, that is to say the cause was not due to varying traffic models, rather the issue was caused by three main factors; 1) the volume of short TCP flows involved in the HTTP simulation (resulting in a large number of 40byte ACKs being sent from the HTTP sink back to the source); 2) CAPS and Best-effort both using a single physical RED queue for all traffic types, whereas Traditional Diffserv and WRED placed HTTP into a separate RED queue and 3) a limitation in the *ns*2 Diffserv implementation of RED queues that restricted the calculation of the average queue size to be calculated in packets rather than bytes. These three factors resulted in a disproportionate number of ACKs (generated by the HTTP sinks) being dropped by the RED queues, which in turn caused the increase in request-response time. This anomaly could have been rectified by altering the *ns*2 source code, however, the additional effort required was considered too great given that the Packmime framework was not ideally suited to per-user HTTP modelling.

5.6.8. Summary of Large Scale Simulations

The results from the large-scale simulations provided further validation of the CAPS algorithm, demonstrating that even when scaled it can successfully manage traffic flows of varying type with the aim of optimising end-user QoS.

The novel method of managing VoIP traffic was shown to be able to deliver VoIP to the destination at an acceptable level of QoS, without any static bandwidth configurations, something believed to be unique to this architecture. This behaviour was achieved in all scenarios, even when co-existing with bandwidth intensive services such as P2P.

The handling of TCP-based applications such as FTP and P2P at large-scale were supportive of the results provided in section 5.3, with bandwidth being distributed fairly among applications rather than the flows. This resulted in a noticeable performance increase for users of single-flow applications such as FTP, since their applications were not starved of bandwidth by aggressive applications such as P2P. Similarly, the dynamic traffic management of CAPS removed the need to statically allocate bandwidth proportions among applications, which meant that each user's traffic received a fair share of the bandwidth, with no single traffic being discriminated against – as was the case in Traditional Diffserv environments. The negative implications of static bandwidth allocation among traffic types were demonstrated by the simulations for Traditional Diffserv, when despite excelling in being able to offer the highest level of QoS for VoIP this came at the cost of all other traffic types.

6. Discussion and Conclusions

This chapter concludes the thesis by summarising the achievements of the research. The chapter proceeds with a discussion of areas for future research that have been identified during this project and that extend the work presented herein.

6.1. Achievements and Contributions

This research programme began with an investigation into the various efforts made to provide Quality of Service in IP networks, exploring the effect network degradation can have on Internet protocols and traffic. It also identified the need to consider QoS as more than simply the prioritisation of a single service, focussing on the combined user experience as a whole. The study evaluated the characteristics of a number of popular Internet services that could subsequently be used as inputs for a novel QoS architecture. The proposed architecture addresses the need to shift QoS provisioning from a static pre-configured design, towards a dynamic, user-centric model. Simulation was used in order to evaluate the capabilities of the novel architecture against several alternative QoS mechanisms.

The overall aim of this study was to investigate and propose a novel architecture capable of evaluating and adapting to changing network conditions and traffic levels, from a user-centric perspective. Through a series of experimental studies and simulations this study has been successful. There were four significant outcomes from this research, details of which follow:

1. Chapter 3 presented the analysis and characterisation of a selection of modern Internet services from a number of different observation points (Internet connection types). The analysis considered characteristics for HTTP, VoIP, Video-on-demand, and P2P applications, identifying packet, flow (TCP/UDP) and application layer characteristics. These included average packet-rate and size, the number of TCP connections for a given online activity, connection duration / activity and ratio of downstream to upstream traffic. The analysis included a method of observing how an application that

chooses to obfuscate its payload can be observed to react to packet loss and delay, providing a novel mechanism to infer performance degradation for encrypted flows, which is subsequently used as an input for the dynamic QoS engine (See section 3.5). One surprising, yet important observation was made for HTTP traffic, where it was revealed that despite mechanisms such as persistent connection and HTTPv1.1, the majority of webpage-content came from a number of distributed sources (See section 3.2) – resulting in many short TCP flows. This was contrary to the assumption that HTTPv1.1 was prolifically deployed and used to deliver the majority of data over a single connection. The abundance of short TCP flows motivated a specific need for protection from unnecessary packet drops, given the increased impact adverse network conditions have during the early stages of a connection. A study into the delivery of YouTube content identified key methods of passively inferring the characteristics of the video stream, such as bitrate, resolution and content length (size), which can be used to inform the proposed QoS architecture. This allowed for dynamic resource allocation aligned to the specific properties of a video stream, without the need for prior or static configuration (See section 3.3).

2. Chapter 4 presented the development of a novel user-centric QoS architecture, which featured awareness of the requirements and behaviour of Internet services. The Congestion Aware Packet Scheduler (CAPS) offers a user-centric approach to the challenge of QoS for an ISP network, whereby an optimum QoS is provided for all the concurrent services in use by an Internet user rather than prioritising a single service at the network level. CAPS combines real-time performance evaluation with traffic management in order to dynamically adjust QoS policies to reflect changing network conditions and traffic variations. The architecture included several novel elements, such as proposing the use of a *minimum acceptable level of QoS.* This threshold defined a target that the CAPS architecture would try to achieve for each traffic type, enabling a

dynamic management of resources with direct effect on application performance, without the need for a static class-based pre-configuration.

The concept was first developed and exemplified for VoIP traffic, measuring the R-factor of a call to determine user satisfaction (See section 4.6.1.1). In the event the R-factor was below an acceptable value, CAPS would adjust the ratio of co-existing services in order to increase the R-factor (thus improve the QoS). Similarly, if the network was congested and the R-factor was far exceeding an upper limit, CAPS would reclaim resources used by the VoIP flow in order to benefit co-existing services, while still ensuring an acceptable level of QoS was achieved by VoIP.

Once the concept had been proven for VoIP traffic, thresholds for additional services were defined, these were; throughput aligned to bitrate for Video-on-Demand and bandwidth shared fairly among single and multi-flow TCP applications (i.e. FTP and P2P).

Further to using these thresholds as traffic management targets, CAPS recognises the impact of network events (loss and queuing) on upper-layer protocols. For example, based on the findings from Chapter 3 CAPS protects the first 12 packets of an HTTP flow in order to prevent loss during the early stages of TCP, which would severely affect TCP performance.

Another novelty of the CAPS architecture is its ability to combine performance evaluation of user traffic with traffic management - whereas traditional QoS architectures only react to local congestion, if the CAPS architecture detects a degradation in delivered quality it will adjust the ratios of co-existing traffic (for a user) with the aim to improve the delivered QoS.

3. Chapter 5 began by presenting the results from an investigation into the impact of the proposed architecture on traffic and application performance. A prototype of the CAPS architecture was developed within *ns*2. Four different users were configured within the

*ns*2 prototype, namely, VoIP, FTP, P2P and HTTP, all competing for a share of the highly contended bottleneck bandwidth. It was shown that without any prior configuration of the network, and solely through real-time monitoring and dynamic management of user traffic, CAPS was successfully able to provision between the four users in a non-discriminatory manner.

The traffic management for VoIP traffic was designed to maintain a moving average R-factor of 75, which was identified as an acceptable threshold before user dissatisfaction occurs. Through the use of real-time monitoring and traffic management CAPS successfully provided an average R-factor of 74.8 (See section 5.4.1), this was achieved without the need for priority queuing or dedicated bandwidth allocation. CAPS was also successful in the management of resources among concurrent FTP and P2P applications. Once again, without the need for a preferential policy or prior configuration, CAPS was able to dynamically monitor the performance (throughput) for each of these applications, actively managing the flows to ensure each application received a fair share of the bottleneck bandwidth, irrespective of the number of TCP flows the application established (See section 5.4.2 and 5.4.3).

The final traffic type to be evaluated under CAPS was HTTP, for which chapter 3 had identified a number of key characteristics. As previously mentioned, despite mechanisms such as HTTPv1.1 and persistent connection, it was observed that HTTP delivery still features many short TCP connections. Due to the sensitive nature of short TCP connections the CAPS architecture provides protection for the first 12 packets of an HTTP flow, in order to minimise the risk of packet loss and TCP retransmission timeouts. It was observed that the protection of short flows, combined with the management of P2P traffic, and a non-preferential management policy, allowed CAPS to complete 90% of HTTP transactions 50% faster than the next best alternative QoS configuration, reducing request-response times from approximately 10 seconds to under 5 seconds (See section 5.4.4).

The CAPS architecture also proposed a novel method of traffic management for TCP based applications. It was hypothesised that the use of per-RTT rather than per-packet management would reduce the number of adjacent packet-drops, while providing the TCP sender sufficient time to react before the connection experienced a subsequent loss. It was demonstrated that using per-RTT management two heterogeneous TCP connections could be controlled to behave fairly towards each other with 30% fewer packet drops with no reduction in flow performance (See section 5.4.5).

4. Chapter 5 also provided the results from tests benchmarking the performance of CAPS against alternative QoS solutions, identified from within the state of the art. A large scale topology was developed using *ns*2, which allowed the evaluation of CAPS in a scaled environment, featuring hundreds of users, configured with a mix of seven different traffic-profiles, all competing for a share of a highly contended bottleneck. The large scale simulations proved that CAPS can successfully provision network resources between users, applications and flows without prior configuration or a preferential management policy. The novel method for managing VoIP QoS was consistently successful in achieving the objective of maintaining an average R-factor of 75, even for users engaged in VoIP, FTP and P2P simultaneously (See sections 5.6.1, 5.6.4, 5.6.5 and 5.6.6). The performance of VoIP under CAPS was seen to be typically 10-20% better compared with Best-effort and WRED configurations, and only marginally behind Traditional Diffserv, despite its explicit prioritisation of VoIP traffic.

   The issue of fairness was stated as one of the primary objectives for the novel architecture, ensuring that a user receives a fair share of network resources regardless of the services they choose to use. Even in a scaled environment CAPS was successful in its management of aggressive P2P traffic, policing the aggregate throughput of a P2P application to within a user's fairshare of the bottleneck. It was shown that CAPS could manage P2P traffic to remain within a user's fairshare, while under Best-effort and

WRED configurations P2P obtained up to 300% of its fairshare, reducing the bandwidth available for concurrent users and services. The preferential model of Diffserv severely restricted the bandwidth available for P2P services, meaning users were able to only achieve 50% of their fairshare of the bandwidth, highlighting the fairness deficiency of this configuration (See sections 5.6.3, 5.6.5 and 5.6.6).

The successful management of aggressive P2P traffic enables CAPS to ensure fairness can also be provided to traditional TCP based services, such as FTP. Sections 5.6.2, 5.6.4 and 5.6.6 illustrate that FTP throughput under CAPS is typically closer to a user's fairshare of bandwidth than was possible with alternative configurations. In contrast to Traditional Diffserv, the absence of static pre-configuration and preferential policies allows flows under CAPS to take advantage of residual bandwidth. This was evident from the performance of FTP flows, which can be seen to burst above the fairshare when possible, however, unlike Best-effort and WRED this exploitation of residual bandwidth is controlled, with throughputs seen to be reduced back towards the fairshare when necessary.

In summary, it is believed that this research programme has reached all of the objectives defined in Chapter 1, and achieved the research aim.

6.2. Areas for Future Work

Throughout the research a number of tangents were identified, each of which provides an area for future research:

1. Expansion of the study of Internet services included in Chapter 3 would provide a number of benefits; Firstly, analysing Internet content over a lengthened period of time would allow the evolution of services to be assessed, identifying whether or not a service profile has a predictable validity. The study included in this research only considered a subset of Internet services, therefore it would benefit from further research to characterise additional traffic types, such as online gaming and video-calls, in order to determine QoS requirements for these services. The question of how to manage traffic from a user whose system is acting as a relay for other Internet users still remains. On one hand it is unfair to discriminate against the relayed traffic, but it doesn't contribute towards the user-experience for the owner of the relaying system. In addition to a study on Internet services it is considered a study into the distribution of current TCP implementations would be greatly beneficial to the research community, particularly those considering novel QoS mechanisms and traffic management solutions.
2. An extension of the CAPS prototype in an alternative development environment will provide further performance results for the architecture. The restrictions of the traffic sources available in *ns*2 were described in section 5.1.1, resulting in only a subset of the traffic analysed in Chapter 3 from being reproduced (with an acceptable degree of realism). Future work in this area could make use of newer simulation environments, such as *ns3* or OPNET, and may involve the inclusion of live (or replayed) traffic streams (rather than simulated traffic models), from which subjective evaluation of the delivered QoS could be investigated. Further development could also explore more scalable methods of simulating HTTP traffic for the *ns*2 / *ns*3 environments.

3. Additional methods of inferring network health to inform resource allocation. In the proposed architecture the method of inferring network health focuses on the bottleneck link utilisation, with the assumption being that the architecture is located on an edge node, adjacent to the bottleneck. Future research could include a method for detecting non-adjacent bottlenecks, which appear in the path of transiting flows. This information would then be used to manage Internet flows based upon their path. Additional parameters to link utilisation would also be inferred and collected, possibly through an out-of-band distributed communication mechanism. This would then be able to evaluate the reliability of a path over time to predict network conditions. The prediction of network health could make use of neural networks in the identification of recurring trends.
4. Alternatives to dropping packets. It is hypothesised that there is a point in the data path where dropping a packet from a responsive flow does not aid to relieve congestion, and is certainly detrimental to QoS. For example, nearing the destination of the packet the majority of the forwarding path has already been transited. In the event of a congested link, a router may chose to drop the packet, invoking TCP to reduce its transmission rate, at the cost of a retransmission. Alternatively, it is proposed that deliberately delaying a packet will increase the measured RTT at the sender, also invoking a reduction in the transmission rate. However, this method does not require retransmission of a packet, so does not further contribute network congestion, but does still deliver the packet. The thresholds of buffering a packet for an increased period of time would require investigation, along with any impact on application performance.

6.3. Concluding Remarks

The integration of the Internet into every aspect of modern day life brings with it many challenges, such as content providers having to ensure their media lives up to user expectations. The shift from simply purchasing access from a service provider to now purchasing a connection with multimedia capabilities has already begun, with over-the-top services being released to market constantly. Furthermore, mobile devices have evolved from a simple communication tool into a multimedia hub, capable of not only acquiring services from the Internet, but also being the source of new online content. Tasked with the role of delivering the ever increasing volume of data to the world lies the Internet, which without the addition of an intelligent mechanism to manage resources, in real-time, will continue to operate as best-effort.

This research has focussed on the subject of User-Centric Quality of Service, a shift from the traditional service-led approach of QoS. This research has two main beneficiaries, firstly the ISP, who can use the Internet traffic analysis, combined with the proposed architecture to evaluate the value their network delivers to the user. Secondly, the end users stand to benefit from any subsequent optimisation of the ISP's network. It is hoped that an ISP would acknowledge that a change of the QoS paradigm to a user-centric model (CAPS) would result in satisfying a larger proportion of customers, rather than only a handful with a static, rigid QoS design.

References

About.com. (2010). "The Best Torrent P2P software - About.com " Retrieved 03-03-2010, 2010, from http://netforbeginners.about.com/od/peersharing/tp/best_torrent_software.htm.

Adobe Systems Incorporated. (2010). "Real-Time Messaging Protocol (RTMP) Specification." Retrieved 02/03/2010, 2010, from http://www.adobe.com/devnet/rtmp/.

Alcaraz, S., K. Gilly, C. Juiz and R. Puigjaner (2007). Handling HTTP flows over a DiffServ framework. Proceedings of the 4th international IFIP/ACM Latin American conference on Networking, ACM: 95-101.

Alexa.com. (2010). "The Web Information Company." Retrieved 01/03/2010, 2010, from http://www.alexa.com.

Allman, M., V. Paxson and E. Blanton (2009). TCP Congestion Control. IETF RFC 5681.

Allman, M., V. Paxson and W. Stevens (1999). TCP congestion control. IETF RFC 2581.

Apostolopoulos, G., D. Williams, S. Kamat, R. Guerin, A. Orda and T. Przygienda (1999). "QoS Routing Mechanisms and OSPF Extensions". IETF RFC 2676.

ARPA (1981). "Internet Protocol". IETF RFC 791.

Babiarz J., Chan K. and Baker F. (2006). Configuration guidelines for Diffserv service classes IETF RFC 4594.

Baker, F., C. Iturralde, F. Le Faucheur and B. Davie (2001). Aggregation of RSVP for IPv4 and IPv6 Reservations. IETF RFC 3175.

Balakrishnan, H. and S. Seshan (2001). "The Congestion Manager". IETF RFC 3124.

Balan, H. V., L. Eggert, S. Niccolini and M. Brunner (2007). An Experimental Evaluation of Voice Quality Over the Datagram Congestion Control Protocol. INFOCOM 2007. 26th IEEE International Conference on Computer Communications., Anchorage, USA.

Baset, S. A. and H. Schulzrinne (2004). An Analysis of the Skype Peer-to-Peer Internel Telephony Protocol. Technical Report CUCS-039-04. Computer Science Department, Columbia University, New York, NY.

BBC-iStats. (2009). "BBC iPlayer goes HD and adds higher quality streams." Retrieved 21-04-2010, 2010.

BBC-iStats. (2010). "BBC iPlayer Publicity Pack for March 2010." Retrieved 21-04-2010, 2010, from http://www.bbc.co.uk/blogs/bbcinternet/img/BBC_iPlayer_Publicity_pack_March_2010.pdf.

BBC-iStats. (2011). "BBC iPlayer Monthly Performance Pack, July 2011." Retrieved 31-03-2012, 2012, from http://www.bbc.co.uk/blogs/bbcinternet/2011/08/18/BBC_iPlayer_performance_monthly_1107_FINAL.pdf.

Bernet, Y., P. Ford, R. Yavatkar, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski and E. Felstaine (2000). "A Framework for Integrated Services Operation over Diffserv Networks". IETF RFC 2998.

Blake, S., D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss (1998). An architecture for differentiated services. IETF RFC 2475.

Bonfiglio, D., M. Mellia, M. Meo and D. Rossi (2009). "Detailed Analysis of Skype Traffic." IEEE Transactions on Multimedia 11(1): 117.

Braden R., Clark D. and Shenker S. (1994). "Integrated Services in the Internet Architecture: an Overview", Request For Comments 1633.

BT. (2012). "Broadband Usage Policy, BT.com." Retrieved 25-03-2012, 2012, from http://bt.custhelp.com/app/answers/detail/a_id/10495/~/broadband-usage-policy.

Cao, J., W. S. Cleveland, Y. Gao, K. Jeffay, F. D. Smith and M. C. Weigle (2004). "PackMime: Synthetic web traffic generation in ns-2, Available from: http://dirt.cs.unc.edu/packmime."

Cha, M., H. Kwak, P. Rodriguez, Y. Y. Ahn and S. Moon (2007). I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. Proceedings of the 7th ACM SIGCOMM conference on Internet Measurement, ACM: 14.

Chiang, W. H., W. C. Xiao and C. F. Chou (2006). A Performance Study of VoIP Applications: MSN vs. Skype. Proceedings of Multicomm 2006. Instanbul, Turkey.

Chiu, D. M. and R. Jain (1989). "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks." Computer Networks 17(1): 1-14.

Cisco Systems Inc. (2010). "Cisco IOS Netflow." Retrieved 01-08-2010, 2010, from http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html.

Clark, D. D. and W. Fang (1998). "Explicit allocation of best-effort packet delivery service." IEEE/ACM Transactions on Networking (TON) 6(4): 373.

Cole, R. G. and J. H. Rosenbluth (2001). "Voice over IP performance monitoring." ACM SIGCOMM Computer Communication Review 31(2): 9-24.

Crawley, E., R. Nair, B. Rajagopalan and H. Sandick (1998). A framework for QoS-based routing in the Internet. IETF RFC 2386.

Cruvinel, L. and T. Vazao (2011). Profile-Based Adaptive DiffServ Policing with Learning Techniques. Internation Conference on Computer Communications and Networks (ICCCN). Maui, Hawaii, IEEE: 1-7.

Davie, B., A. Charny, J. C. R. Bennett, K. Benson, J. Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu and D. Stiliadis (2002). An expedited forwarding PHB (per-hop behavior). IETF RFC 3246.

eBizmba.com. (2010). "Top 15 torrent websites - April 2010." Retrieved 01-04-2010, 2010, from http://www.ebizmba.com/articles/torrent-websites.

Eger, K., T. Hoßfeld, A. Binzenhöfer and G. Kunzmann. (2007). "BitTorrent Implementation for *ns*2 Network Simulator." from http://www.tu-harburg.de/et6/research/bittorrentsim/index.html.

El-Gendy, M. A. and K. G. Shin (2003). "Assured forwarding fairness using equation-based packet marking and packet separation." Computer Networks 41(4): 435-450.

Ellis, M., S. Strowes and C. Perkins (2011). An Experimental Study of Client-side Spotify Peering Behaviour. IEEE Local Computer Networks 2011. Bonn, Germany.

Elshaikh, M. A., M. Othman, S. Shamala and J. M. Desa (2008). "A new fair marker algorithm for DiffServ networks." Computer Communications 31(14): 3064-3070.

Firoiu, V. and M. Borden (2000). A study of Active Queue Management for Congestion Control. INFOCOM 2000. 19th IEEE International Conference on Computer Communications. , Tel Aviv, Israel.

Floyd, S., T. Henderson and A. Gurtov (2004). The NewReno Modification to TCP's Fast Recovery Algorithm. IETF RFC 3782.

Floyd S. (1997). "Random Early Detection queue parameters." Retrieved 05/12/2009, 2010, from http://www.icir.org/floyd/REDparameters.txt.

Floyd S. (2000). "Recommendation on using the gentle variant of RED: Technical note." Retrieved 14/04/2010, 2010, from http://www.icir.org/floyd/red/gentle.html.

Floyd S. and Fall K. (1999). "Promoting the use of end-to-end congestion control in the Internet." IEEE/ACM Transactions on Networking (TON) 7(4): 458-472.

Floyd S., Gunmamadi R. and S. S. (2001). "Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management." Retrieved 10/06/2010, 2010, from http://www.icir.org/floyd/papers/adaptiveRed.pdf.

Floyd S. and Jacobson V. (1993). "Random early detection gateways for congestion avoidance." IEEE/ACM Transactions on Networking (TON) 1(4): 397-413.

Floyd S. and Kohler E. (2006). TCP Friendly Rate Control (TFRC): The Small-Packet (SP) Variant. Internet Draft.

Friedman, T., R. Caceres and A. Clark (2003). RTP control protocol extended reports (RTCP XR). IETF RFC 3611.

Gill, P., M. Arlitt, Z. Li and A. Mahanti (2007). Youtube traffic characterization: a view from the edge. Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, ACM.

Guha, S., N. Daswani and R. Jain (2006). "An experimental study of the Skype peer-to-peer VoIP system."

Ha, S., I. Rhee and L. Xu (2008). "CUBIC: A New TCP-Friendly High-Speed TCP Variant." ACM SIGOPS Operating System Review 42(5): 64-74.

Handley, M., S. Floyd, J. Padhye and J. Widmer (2003). "TCP Friendly Rate Control (TFRC): Protocol Specification". IETF RFC 3448.

Hardin G. (1968). "The Tragedy of the Commons." Science 162(3859): 1243-1248.

He, Q. (2003). "Packet-level peer-to-peer simulation framework and gnutellsim." Retrieved January 2010, from http://www.cc.gatech.edu/computing/compass/gnutella/.

Heinanen, J., F. Baker, W. Weiss and J. Wroclawski (June 1999). Assured forwarding PHB group. IETF RFC 2597.

Heinanen J., F. Le Faucheur, L. Wu, B. Davie, S. Davari, P. Vaananen, R. Krishnan and P. Cheval (2002). "Multi-Protocol Label Switching (MPLS) Support of Differentiated Services", Request for Comments 3270.

Hollot C. V., Misra V., Towsley D. and G. W. (2002). "Analysis and Design of Controllers for AQM Routers Supporting TCP Flows." IEEE Transactions on Automatic Control 47(6): 945-959.

Huang, T. Y., P. Huang, K. T. Chen, A. Sinica and P. J. Wang (2010). "Could Skype Be More Satisfying? A QoE-Centric Study of the FEC Mechanism in an Internet-Scale VoIP System." IEEE Network: 3.

IEInspector Software LLC. (2010). "IEInspector HTTP Analyser v5." Retrieved 01/03/2010, 2010, from http://www.ieinspector.com/httpanalyzer/download.html.

IETF (1989). Requirements for Internet Hosts - Communication Layers. (Editor Braden R.). IETF RFC 1122.

ISPReview. (2010). "ISPReview - Top 10 UK ISPs." Retrieved January 2010, from http://www.ispreview.co.uk/review/top10.php.

ITU-T. (1988). Pulse Code Modulation (PCM) of voice frequencies.

ITU-T. (2000). The E-Model, a computational model for use in transmission planning.

ITU-T. (2001). End-user multimedia QoS categories, *ITU-T Recommendation G.1010*.

ITU-T. (2001). ""Perceptual evaluation of speech quality (PESQ): An Objective method for end-to-end speech quality assessment of narrow-band networks and speech codecs", *ITU-T Recommendation P.862*, Study group 12."

Jacobson, V. (1995). "Congestion avoidance and control." ACM SIGCOMM Computer Communication Review 25(1): 157-187.

John, W., S. Tafvelin and T. Olovsson (2008). "Trends and differences in connection-behavior within classes of internet backbone traffic." Passive and Active Network Measurement: 192-201.

Jourjon, G., E. Lochin and P. Sénac (2007). "Design, implementation and evaluation of a QoS-aware transport protocol." Computer Communications 31(9): 1713-1722.

Karagiannis, T., M. Molle and M. Faloutsos (2004). "Long-range dependence ten years of Internet traffic modeling." IEEE Internet Computing 8(5): 57-64.

Kitz.co.uk. (2009). "Kitz.co.uk - UK ADSL/Broadband Consumer Information Site." Retrieved 15/02/2010, 2010, from http://www.kitz.co.uk.

Kohler, E., M. Handley and S. Floyd (2006). Designing DCCP: congestion control without reliability. SIGCOMM '06 Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications, Pisa, Italy, ACM Press New York, NY, USA.

Kung, H. Y. and F. W. Kuo (2006). Dynamic user-oriented QoS specification and mapping for multimedia differentiation services. 5th IEEE International Conference on High Speed Networks and Multimedia Communications.

Kunniyur S. and Srikant R. (2003). "Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management." Transactions from Networks 12(2): 286-299.

Kuzmanovic, A. (2005). The power of explicit congestion notification. Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications, Philadelphia, USA.

Lahanas, A. and V. Tsaoussidis (2003). "Exploiting the efficiency and fairness potential of AIMD-based congestion avoidance and control." Computer Networks 43(2): 227-245.

Le, L., J. Aikat, K. Jeffay and F. D. Smith (2003). The effects of active queue management on web performance. Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, Karlsruhe, Germany ACM.

Lei, G., C. Songqing, X. Zhen and Z. Xiaodong (2005). Analysis of multimedia workloads with implications for internet streaming. Proceedings of the 14th international conference on World Wide Web. Chiba, Japan, ACM.

Lin, D. and R. Morris (1997). Dynamics of Random Early Detection. Proceedings of the ACM SIGCOMM '97 conference on Applications, technologies, architectures, and protocols for computer communication, Cannes, France.

Lochin, E. and P. Anelli (2009). "TCP throughput guarantee in the DiffServ Assured Forwarding service: what about the results?" Annals of Telecommunications 64(3): 215-224.

Mascolo, S., C. Casetti, M. Gerla, M. Y. Sanadidi and R. Wang (2001). TCP westwood: Bandwidth estimation for enhanced transport over wireless links, ACM Press New York, NY, USA: 287-297.

Mascolo, S., L. A. Grieco, R. Ferorelli, P. Camarda and G. Piscitelli (2004). Performance evaluation of Westwood+ TCP congestion control. Proceedings of Internet Performance Symposium (IPS 2002), Taipei, Taiwan, Elsevier.

Mathis, M., J. Mahdavi, S. Floyd and A. Romanow (1996). TCP Selective Acknowledgement Options. IETF RFC 2018.

Matowidzki, M. (2003). ECN is fine- but will it be used? 21 st IASTED International Multi-Conference on Applied Informatics.

Medina, A., M. Allman and S. Floyd (2005). "Measuring the evolution of transport protocols in the internet." ACM SIGCOMM Computer Communication Review 35(2): 37-52.

Mellia, M., I. Stoica and H. Zhang (2003). "TCP-aware packet marking in networks with DiffServ support." Computer Networks 42(1): 81-100.

Min, Z., M. Dusi, W. John and C. Changjia (2009). Analysis of UDP Traffic Usage on Internet Backbone Links. 9th Annual International Symposium on Applications and the Internet, 2009. SAINT '09. .

Miniwatts-Marketing-Group. (2012). "World Internet usage statistics news and population stats." Retrieved 25-03-2012, 2012, from http://www.internetworldstats.com/stats.htm.

Molnar, K. and V. Vrba (2008). DiffServ-based user-manageable quality of service control system. Proceedings of the 7th International Conference on Networking, Cancun, Mexico, IEEE Computer Society.

Ng, S. L., S. Hoh and D. Singh (2005). Effectiveness of adaptive codec switching VoIP application over heterogeneous networks. 2nd International Conference on Mobile Technology, Applications and Systems, 2005.

ns2 Network Simulator (2010). Available from: http://isi.edu/nsnam/ns/.

Ofcom (2008). UK broadband speeds: initial findings of consumer experience of broadband performance.

OPNET. (2010). "OPNET Modeler." Retrieved 05/05/2010, 2010, from http://www.opnet.com/solutions/network_rd/modeler.html.

Ostermann, S. (2003). TCPtrace, available from http://www.tcptrace.org.

Ou. G. (2008). "Fixing the unfairness of TCP congestion Control." Retrieved 01/04/2008, 2008, from http://blogs.zdnet.com/Ou/?p=1078&page=4.

Padhye, J., V. Firoiu, D. F. Towsley and J. F. Kurose (2000). "Modeling TCP Reno performance: a simple model and its empirical validation." IEEE/ACM Transactions on Networking (ToN) 8(2): 133-145.

Paxson, V. and M. Allman (2000). Computing TCP's Retransmission Timer. IETF RFC 2988.

Perkins, C., O. Hodson and V. Hardman (1998). "A survey of packet loss recovery techniques for streaming audio." IEEE Networks 12(5): 40-48.

Pieda, P., J. Ethridge, M. Baines and F. Shallwani (2000). "A network simulator differentiated services implementation open IP, Nortel Networks."

Ramakrishnan, K., S. Floyd and B. D. (2001). "The Addition of Explicit Congestion Notification (ECN) to IP". IETF RFC 3168.

Rao, A., A. Legout, Y. Lim, D. Towsley, C. Barakat and W. Dabbous (2011). Network characteristics of video streaming traffic. Proceedings of the Seventh Conference on emerging Networking Experiments and Technologies. Tokyo, Japan, ACM.

Reguera, V. A., F. Álvarez Paliza, W. Godoy Jr and E. M. García Fernández (2008). "On the impact of active queue management on VoIP quality of service." Computer Communications 31(1): 73-87.

Sandvine. (2011). "Global Internet Phenomena Report 2011." Retrieved 08-08-2011, 2011, from http://www.sandvine.com/downloads/documents/10-26-

2011_phenomena/Sandvine%20Global%20Internet%20Phenomena%20Report%20-%20Fall%202011.pdf.

Schulze, H. and K. Mochalski. (2006). "Ipoque P2P Survey 2006." Retrieved January 2010, from http://www.ipoque.com/resources/internet-studies/p2p-survey-2006.

Schulze, H. and K. Mochalski. (2007). "Ipoque Internet study." Retrieved January 2010, from http://www.ipoque.com/resources/internet-studies/internet-study-2007.

Schulze, H. and K. Mochalski. (2009). "Ipoque Internet study." Retrieved January 2010, from http://www.ipoque.com/resources/internet-studies/internet-study-2008_2009.

Schulzrinne, H. and S. Casner (2003). ""RTP Profile for Audio and Video Conferences with Minimal Control"." IETF RFC 3551

Schulzrinne, H., S. Casner, R. Frederick and V. Jacobson (2003). "RTP: A Transport Protocol for Real-Time Applications". IETF RFC 3550.

Schumacher, J., M. Dobler, E. Dillon, G. Power, M. Fiedler, D. Erman, K. De Vogeleer, M. O. Ramos and J. R. Argente (2010). Providing an User Centric Always Best Connection. Second International Conference on Evolving Internet (INTERNET) 2010, IEEE: 80-85.

Seddigh, N., B. Nandy and P. Pieda (1999). Bandwidth assurance issues for TCP flows in a differentiated services network. Proceedings from GLOBECOM 1999, Rio de Janeireo, Brazil.

Skype. (2010). "SILK: Super Wideband Audio Codec from Skype." Retrieved 02/02/3010, 2010, from http://developer.skype.com/silk?action=AttachFile&do=get&target=SILKDataSheet.pdf.

Su, H. and M. Atiquzzaman (2003). "ItswTCM: a new aggregate marker to improve fairness in DiffServ." Computer Communications 26(9): 1018-1027.

Sudha, S., S. Maddipati and N. Ammasaigounden (2008). A new adaptive marker for bandwidth fairness between TCP and UDP traffic in DiffServ. IEEE Region 10 Conference TENCON 2008-2008, TENCON 2008: 1-5.

Suh, K., D. R. Figueiredo, J. Kurose and D. Towsley (2006). Characterizing and detecting relayed traffic: A case study using Skype. INFOCOM 2006. 25th IEEE International Conference on Computer Communications., Barcelona, Spain.

Tan, K., J. Song, Q. Zhang and M. Sridharan (2006). A Compound TCP approach for High-Speed and Long Distance Networks. IEEE Infocom, Barcelona, Spain.

Telegraph.co.uk. (2009). "YouTube hits one billion views a day." Telegraph Retrieved 24-04-2010, from http://www.telegraph.co.uk/technology/google/6281439/YouTube-hits-one-billion-views-a-day.html.

Tsaoussidis, V. and C. Zhang (2005). "The dynamics of responsiveness and smoothness in heterogeneous networks." IEEE Journal on Selected Areas of Communications 23(6): 1178-1189.

Usman, M. and N. M. Sheikh (2005). Performance Analysis of Adaptive Source Rate Control Algorithm (ASRC) for VoIP. TENCON 2005 2005 IEEE Region 10.

Vos, K., S. Jensen and K. Soerensen (2010). SILK Speech Codec draft-vos-silk-01, available from http://tools.ietf.org/html/draft-vos-silk-01.

Vuze. (2010). "A list of ISPs reported to implement a form of traffic management against P2P protocols." Retrieved January 2010, 2010, from http://wiki.vuze.com/.

Weigle, M. C., K. Jeffay and F. D. Smith (2006). "Quantifying the Effects of Recent Protocol Improvements to Standards-Track TCP." Computer Communications 29: 2853-2866.

Wireshark (2009). available from http://www.wireshark.org/.

Wydrowski B. and Z. M. (2002). GREEN: An Active Queue Management Algorithm for a Self Managed Internet. Proceedings from IEEE Internation Conference on Communications, ICC 2002. New York, USA. 4: 2368-2372.

Yeom, I. and A. L. N. Reddy (2001). Adaptive marking for aggregated flows. Proceedings from IEEE Globecom 2001, Texas, USA.

YouTube. (2010). "YouTube - Broadcast Yourself." from www.youtube.com.

YouTube. (2010). "YouTube Fact Sheet." from http://www.youtube.com/t/fact_sheet.

YouTube. (2012). "YouTube Statistics." Retrieved 31-03-2012, 2012, from http://www.youtube.com/t/press_statistics.

Zhang, C. and V. Tsaoussidis (2001). TCP-real: improving real-time capabilities of TCP over heterogeneous networks, ACM Press New York, NY, USA: 189-198.

Appendices

The following appendices contain work supporting this thesis. Appendices A – D are provided on the accompanying DVD, provided at the back of this thesis, while Appendix E is printed.

Appendix A – Traffic Analysis Data & Scripts

Appendix A includes a number of packet captures files for the Internet services evaluated in Chapter 3, along with various shell scripts that were used to analyse the captures.

Appendix B – Modified *ns*2 Source Code

Appendix B.1 – Modified ns2 diffserv package

Appendix B.2 – Modified n2 trace package

Appendix B.3 – Output from diff against original ns2 source code

Appendix B.4 – FlowTable and UserTable Data Structures integrated into *ns*2

Appendix C – Simulation TCL Scripts

Appendix C.1 – Phase 1 Simulation scripts

Appendix C.2 – Per RTT TCP Tests

Appendix C.3 – Phase 2 Simulation scripts

Appendix D – Analysis and Processing Scripts

Appendix D.1 – Phase 1 Processing Scripts

Appendix D.2 – Phase 2 Processing Scripts

Appendix E – Publications

Culverhouse M., Ghita B., Reynolds P., Wang X., (2010) Optimising quality of service through the controlled aggregation of traffic, In Proceedings of The Seventh International IEEE Conference for Internet Technology and Secured Transactions, United Kingdom, November 2010.

Culverhouse M., Ghita B., Reynolds P., (2011) User-Centric Quality of Service Provisioning, In Proceedings of The 37th IEEE Conference on Local Computer Networks (LCN), Bonn, Germany, November 2011.

Optimising quality of service through the controlled aggregation of traffic

M. E. Culverhouse, B. V. Ghita, P. Reynolds and X. Wang
Centre for Security, Communications, and Network Research, University of Plymouth, UK
info@cscan.org

Abstract

The multimedia enriched transport supported by the Internet today continues to pose a challenge to engineers who attempt to provide QoS for its users. The popular method of using Diffserv solutions is typically too static to meet the mixed user traffic model. This paper introduces a novel user-centric approach of dynamically evaluating and policing incoming Internet flows to control the ratio of traffic types for individual users. After describing its theoretical rationale, the proposed method is presented as an integrated architecture (Congestion Aware Packet Scheduler – CAPS), which allows seamless integration with existing Diffserv networks. An ns2 implementation of the CAPS architecture is presented and investigated for a number of different scenarios. The evaluation of CAPS indicates that the architecture outperforms best effort, traditional Diffserv and weighted-RED alternatives, providing a better, dynamic QoS balance for a wide range of traffic profiles without the need for explicit predetermined precedence on traffic types.

1. Introduction

The Internet has evolved into a ubiquitous communications network with an estimated 1.6 Billion users [1]. Throughout its evolution, many different methods were proposed to improve the best-effort nature of IP and to provide the user with a satisfactory level of Quality of Service (QoS). It is suggested that merely configuring a service provider's network to give predetermined precedence to specific traffic types, as can be seen with traditional Differentiated Services (Diffserv) implementations [2], can result in an unfair level of service for customers who may produce a different traffic mixture. Furthermore, it should not be assumed that real-time services such as VoIP will always be considered a priority over other traffic from a single user; for example, real-time online purchases may be incorrectly classified as a non real-time activity and not get the priority a user may desire. The change in nature of Internet traffic has also made provisioning for QoS progressively difficult. An increase in UDP-based traffic, which is unresponsive to congestion, has meant that there is no regard for coexisting flows, which can lead to impairing the performance of congestion controlled traffic when they compete for available bandwidth.

The Diffserv architecture is a popular choice for service providers to offer QoS on their networks. Diffserv comprises of edge and core routers, where the former are responsible for inspecting incoming flows and mark the IP header of the packet with a Diffserv Code Point (DSCP). The DSCP is determined according to a policy defined by the network operators, with Diffserv core routers using the DSCP to determine with what precedence the packet should be forwarded within the presence of congestion. Typically a Diffserv network will be configured to map different traffic types onto one of three different Per Hop Behaviours (PHBs): Expedited Forwarding (EF) [3], Assured Forwarding (AF) [4] and Best Effort (BE). It is recommended to map real-time services onto the EF PHB to meet the stringent QoS requirement (low packet loss, jitter and delay) of such services. Diffserv AF is traditionally a static, policy based traffic management system which can be configured by the network operators to provide a specific level of service to a customer. It supports up to four different traffic classes, each of which has up to three drop precedence levels. A flow that adheres to its agreed policy (for example remains within its committed information rate (CIR)) is said to be *in-profile* and receives a default level of service. If a flow fails to adhere to its policy (for example exceeding its CIR) then it is described as being *out-of-profile*, and its packets are marked with a lower precedence DSCP value, thus facing a higher drop probability in the event of congestion.

Although a good starting point, specific services should not automatically be given priority at the cost of concurrent flows, based on the philosophy that QoS should consider the entire user experience and not necessarily favour just a single traffic type. Additional reasons for not following the trend of using the EF PHB for VoIP traffic (or other real-time conversational services) are given by [5] where it is suggested that the EF configuration is limited by a number of aspects, such as having a load limit on the EF capacity (to prevent starvation of capacity to other traffic types), which could result in high priority traffic connections being refused even when residual capacity is available in lower priority classes. Moreover, this configuration is static and predetermined by the network operator rather than a dynamic system that acknowledges the current user activities and network climate.

The architecture introduced in this paper aims to overcome this limitation by proposing a dynamic approach of classifying and prioritising traffic, through the continuous observation of the user traffic levels. There have been many proposed improvements to the original operation of Diffserv over the past 10 years, which suggest a more dynamic approach to resource management; for example [6] presents a user-oriented QoS framework which provides the user with an application-layer interface to control resource allocation among applications in a friendly, easy-to-user manner. This information is used to dynamically configure the Diffserv network. Whilst such research promoted user-oriented QoS, the method employed requires user-interaction which can be intrusive from the user experience perspective. In [7], the authors proposed a modification to the static Diffserv configuration to use bandwidth measurements from the network to influence the edge routers during the metering and policing process, however the architecture does not provide policing mechanisms to enforce fairness or control over misbehaving hosts. The research in [8] describes a method for dynamically configuring the bandwidth between the traffic classes within a Diffserv network, ensuring over or under provisioning is minimised. It is noted that a large amount of research has focussed on improving specific aspects of Diffserv architectures, such as admission control and link utilisation; however, there has been a very limited amount of research conducted in offering user-centric Diffserv solutions, further emphasising the novelty of this research.

Inadequacies in these proposed enhancements motivated the conception of a new approach of providing QoS and the development of this concept. By moving away from a static, predetermined prioritisation scheme, the presented architecture considers the combined multimedia experience of a single ‘Internet user’ rather than focussing on a ‘single service’. The bandwidth management system aims to balance resources between flows, applications and ultimately each paying customer to ensure fairness between coexisting traffic is maintained.

To enable this novel architecture it is necessary to consider the combined multimedia experience of each user. In order to achieve this, CAPS must recognise how network-oriented events (such as delay and packet drops) impact the QoS of each user service.

For example, TCP-based applications benefit from low packet drop rates to allow the additive increase multiplicative decrease algorithm to reach and maintain the maximum throughput possible, given the available bandwidth [9]. Frequent packet drops prevent the growth of the TCP congestion window and therefore CAPS should be cautious when dropping packets from TCP applications to avoid preventing a flow from obtaining a maximum throughput. [10] models the performance of TCP Reno to give Equation (1), which describes the relationship between the sending rate of the TCP Reno connection *T*, given a loss rate *p*, TCP retransmit timeout value T_{RTO} and packet size of *s*.

$$T = \frac{s}{RTT\sqrt{\frac{2p}{3}} + T_{RTO}\left(3\sqrt{\frac{3p}{8}}\right)p(1+32p^2)} \tag{1}$$

Real-time applications, such as VoIP, are sensitive to a number of factors which can seriously degrade the quality received by a user. The performance metrics of interest for such applications are delay, packet loss ratio and jitter (inter-packet delay). A widely accepted method to assess the QoS of a voice call is the ITU-T G.107 E-Model [11] which rates the QoS between 0 (incomprehensible) and 100 (perfect) - depending upon the impairment by delay and packet loss ratio (PLR). [12] presents a reduction of the E-Model for use in packet switched networks and provides the equation used by the CAPS architecture. This formula is given in equation (2). Where Ie (3) and Id (4) are the impairment factors given the PLR (e) and delay (d) respectively, λ_1, λ_2 and λ_3 are codec specific parameters derived in [12], (which for the G.711 codec later described as having been used in this work are 0, 30 and 15 respectively). H(x) is a step function where H(x) = 0 if x< 0 or H(x) =1 if x ≥ 0.

$$R = 94.2 - Ie - Id \tag{2}$$

$$Ie = \lambda_1 + \lambda_2 \ln(1 + \lambda_3 e) \tag{3}$$

$$Id = 0.024d + 0.11(d - 177.3)H(d - 177.3) \tag{4}$$

The monitoring/policing functionality of the CAPS architecture needs to be located at the point of aggregation within the network, where traffic from multiple content providers can be conditioned prior to transiting across the core network. Placing the system at this point of aggregation means that modifications to Internet end-hosts are avoided and the need to assume end hosts can be trusted to behave in a respectful manner to other hosts is also removed. Incorporating CAPS within a Diffserv edge router allows traffic to be conditioned on the ingress interface before it transits the core network towards the destination, and also facilitates the integration and utilisation of existing Diffserv infrastructures. However, unlike Diffserv, CAPS does not require a pre-emptive configuration and features a far more dynamic mode of operation, as highlighted in section 2.

The rest of the paper is organised as follows: Section 2 provides an overview of the features and functionality of the CAPS architecture. Section 3 describes the development of the architecture within the *ns*2 simulation software. Section 4 presents and analyses the results. A summary of the paper's findings will be given in section 5 along with a future direction for the research.

2. Architecture overview

The CAPS architecture comprises of edge routers to classify and police traffic, and core routers for subsequent forwarding. It is loosely based upon the AF PHB group [4], providing an assured level of service if certain conditions are adhered to, and utilises a Random Early Detection (RED) queue management [13] system to schedule outgoing packets. The CAPS architecture enhances the original operation of an AF edge router to consider all traffic to be within a single traffic class and aims to maximise the QoS for each flow, through the recognition of how network events impact application QoS, as discussed in the introduction of this paper. The CAPS algorithm has been designed to work with this understanding to identify the point of equilibrium where the QoS of each service can be maximised. CAPS aims to ensure that each TCP flow operates at a transmission rate no greater than its statistical fair share (available link bandwidth divided by number of TCP flows) and that any active VoIP flows achieve an R value between 70 and 80. If a VoIP call is present and has an R value lower than 70 then coexisting TCP flows are policed more aggressively in order to make available additional resource and increase the R value of the VoIP flow. Maintaining the R-value of a VoIP call between 70 and 80 has been chosen based upon recommendation G.107 [11], which states that an R-value of 70 is the lower limit before dissatisfaction amongst users occurs (A traditional PSTN call has a target R-value of 70). Therefore, the value to a user of a VoIP call with an R-value below 70 is considered unacceptable from the user experience perspective.

Unlike previous QoS provisioning alternatives, the traffic policing approach employed by CAPS does not give explicit priority to VoIP flows. Instead, the edge routers balance the performance of each flow, to achieve an overall improvement to an Internet user's experience, and in the case of VoIP traffic ensures the service remains valuable to the user.

The policing of traffic is performed using of a series of RED queues with increasingly more aggressive RED parameters (lowering the max_{th} and increasing the packet drop probability). This use of RED ensures that during times of high utilisation packets will inevitably be dropped. The dropped packets will, in turn, instruct congestion controlled transport protocols (such as TCP) to throttle back their transmission rates, and control the transmission rates of unresponsive flows (such as VoIP) from exceeding what is needed to achieve a satisfactory level of QoS.

A flow table is held by the edge router that stores statistics on currently active flows. For each arriving flow the following statistics are maintained: the most recent arrival timestamp of a packet belonging to the flow, the congestion history of the flow (CHV), time of the last degradation, and average rate of the flow. Furthermore, statistics solely for VoIP flows, average one way delay, packet loss ratio and the calculated R value for that flow are also stored. The term congestion history (CHV) refers to whether a TCP flow has behaved fairly to coexisting flows. As a packet arrives at the edge router the performance metric(s) for the parent flow of that packet are checked. If the flow is misbehaving (i.e. a TCP transmitting beyond its fair share) and the network is congested (bottleneck utilisation exceeds a configurable threshold) the edge router will assign the packet a degraded DSCP, instructing the router (and subsequent core routers) to enqueue the packet in a more aggressive queue. The CHV is incremented and a timestamp for when this DSCP marking occurred is appended in the flow table. If subsequent packets from that flow arrive within a round trip time (RTT) of the last degradation then these packets will not be placed in the more aggressive queue and simply enqueued in the default queue. This back-off routine allows the sending node to act upon any packet loss which has occurred as a result of enqueuing the degraded packet in a higher drop-probability queue. In the event that all existing VoIP flows are maintaining an R value of 80 or above TCP flows with a CHV greater than 10 are subject to a decreasing function to the CHV.

Pseudo code for the CAPS optimisation algorithm is given in Figure 1, which describes the traffic policing performed by CAPS. Within Figure 1 *current_util* refers to the current utilisation of the bottleneck link, *util_threshold* is the threshold beyond which CAPS will begin policing the traffic. *R-value$_i$* is the E-model R-value for flow *i*. *ICP* is the Initial Code Point that is the default DSCP marking which is associated to the default queue.

```
For VoIP Flows
if current_util < util_threshold
        then DSCP = ICP
if (current_util > util_threshold)
    & (R-value_i > UpperVoIPThreshold)
        then DSCP = ICP + 1
For TCP flows
if (current_util > util_threshold)
    & (rate_i > BW/Flowcount)
    & ($\bar{R}$ < LowerVoIPThrehold)
    & (TimeSinceLastDegrade_i > RTT)
        then
case (CHV_i < CHVThreshold1 )
        CHV_i ++
        LastDegradedTime = now
        DSCP = ICP
case (CHV_i <= CHVThreshold2 )
        CHV_i ++
        LastDegradedTime = now
        DSCP = ICP + 1
case (CHV_i <= CHVThreshold3 )
        CHV_i ++
        LastDegradedTime = now
        DSCP = ICP + 2
case (CHV_i > CHVThreshold3)
        CHV_i ++
        LastDegradedTime = now
        DSCP = ICP = 3
else if ($\bar{R}$ > UpperVoIPThrehold)
        then CHV_i --
            DSCP = ICP
else
        DSCP = ICP
```

Figure 1: The CAPS Optimisation Algorithm

ICP+1, ICP+2 and ICP+3 refer to the subsequent DSCP markings for the increasingly aggressive RED queues. *CHV$_i$* is the congestion history value for flow *i*.

It is acknowledged that the use of the flow table could become computationally and memory expensive, increasing the workload of a router. However, the CAPS algorithm is within an edge router, which in comparison with a core router serves only a fraction of Internet clients. Furthermore, to alleviate future scalability issues the architecture could adopt Netflow [14] (or similar) functionality, a proven flow-based system used to maintain large Internet-flow tables.

3. Testing and validation

3.1. CAPS Implementation

The CAPS architecture was implemented using the *ns*2 network simulator [15], using the Nortel Networks Differentiated Services Implementation framework [16]. The network topology created in *ns*2 features the CAPS architecture embedded within the edge router, configured with four precedence (RED) queues. Within the simulations the edge router has the role of maintaining a record of packet loss ratios and one way delays for each VoIP flow used to calculate the R values. However, in a real-life scenario this calculation could utilise extended RTCP report between the VoIP call participants, which would give access to the R value of the call without having to calculate it at the router [17].

3.2. Simulation topology and traffic parameters

The network topology in Figure 2 includes a number of traffic sources aggregating at the edge router of a service provider's network (E1) via 1Mb duplex links with 10ms of delay. They share a bottleneck through the Diffserv network towards the client destination (D), it is appreciated that a bottleneck of this capacity is not analogous to a true Internet core. However, this configuration represents only a single destination (client) node and an estimation of a single user's share of an ISP's network.

The traffic sources in the simulation have been configured to represent two VoIP calls, 3 long-lived FTP transfers over TCP, HTTP cross traffic using the Packmime web traffic generator for *ns*2 [18], and a variety of non specific background traffic (*BGT* in Figure 2). The two VoIP sources and are simulated using constant bit rate sources each operating at 64kbps with a packet size of 160bytes, making them comparable with the popular G.711 speech codec [19]. The FTP file transfers are based upon the TCP Reno implementation, and the non specific background traffic feature two CBR sources and a Pareto on-off traffic source, configured with mean on-off periods of 1500ms and 1000ms respectively. This traffic source is analogous of a VoIP codec using Voice Activity Detection (VAD) [20].

The simulation duration is 315seconds with traffic sources starting between 0-200 seconds and ending between 215-315 seconds. The simulation was run using four traffic management schemes, Best effort, traditional Diffserv, WRED and the CAPS architecture. The Traditional Diffserv configuration featured three traffic classes, EF, AF and BE. The two VoIP flows were assigned to the EF group providing they adhered to their CIR, the FTP flows and non-specific background traffic were assigned to the AF group. The HTTP traffic was assigned to the BE group, the justification for this is that short-lived HTTP flows will not achieve such high throughputs as the long-lived FTP flows and therefore required less precedence under this model. The WRED configuration featured three queues, the two VoIP flows and the HTTP traffic were assigned to their own queues, and the remaining traffic was assigned to the third queue. The VoIP queue had a higher maximum threshold and a lower drop probability than the other queues. The Best Effort configuration featured a single RED queue, to which all traffic was assigned. The CAPS configuration featured a single traffic class with four precedence queues within that class. No single traffic type was given explicit priority prior to the simulation beginning, nor were any CIR values declared for any of the flows. Two configurations of traffic sources were ran the first with the two VoIP flows active; the second was without these VoIP flows.

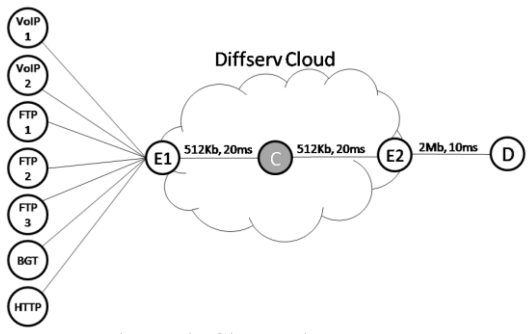


Figure 2: Simulation Topology

4. Analysis of results

The performance metrics of interest were throughput for TCP flows and packet loss, delay and jitter for the real-time VoIP flows (contributing factors to the R-value). Figure 3 shows the mean throughput (in kbps) for each of the three long-lived FTP flows. The results illustrate that the CAPS algorithm managed to allow each of the FTP flows to reach a similar throughput achieved by each of the alternative configurations (within 10%).

Figure 4 and 5 show that, while the CAPS algorithm resulted in a 5-10% performance reduction for the long-lived FTP flows, the network resource reclaimed by this policing is utilised by CAPS to achieve a significantly noticeable improvement on the QoS for both the VoIP flows, averaging an R-value 20% higher than achieved by any of the alternate configurations. Both Figure 4 and 5 illustrate both VoIP flows maintaining an R-value above 70 (the minimum threshold for acceptable VoIP) throughout the duration of the simulation. It should also be noted that as congestion in the network increases CAPS lowers the R-value of the VoIP flows to lie within the upper and lower VoIP thresholds. The introduction of additional traffic flows throughout the simulation is indicated by the vertical lines, representing the three FTP start times and the start and stop times for the HTTP cloud.

The 20% performance increase of the VoIP flows under CAPS is despite VoIP offering no explicit priority over coexisting traffic, in contrast to the traditional Diffserv configuration. The relative poor performance of traditional Diffserv is believed to be due to Diffserv attempting to serve the CIR for not only the EF VoIP flows but also those flows within the AF group under such oversubscribed conditions. Whereas under the CAPS architecture, flows were not guaranteed a proportion of bandwidth to any traffic, but instead CAPS aims to guarantee an acceptable level of QoS for each flow.

As expected, a WRED configuration does not result in the same performance as when RED queues are coupled with the CAPS traffic management algorithm, differentiating the operation of CAPS from a standard WRED environment.

To validate the performance of the CAPS algorithm when VoIP flows were not present, a second set of simulations were ran with the two VoIP flows disabled. Figure 6 shows that without VoIP present the long-lived FTP flows achieved throughputs equal to that of a best-effort network. This result is due to the CAPS algorithm trying to ensure that the available bandwidth is shared amongst the existing TCP flows fairly, something which the TCP congestion avoidance algorithm already achieves, and therefore since there are no VoIP flows to maintain R-values for, CAPS does not intervene.

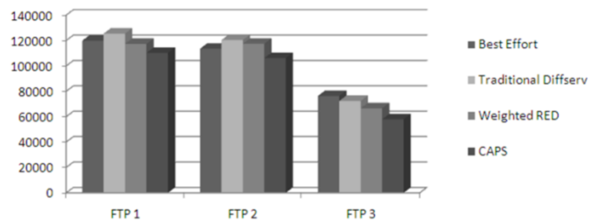


Figure 3: Throughput (Kbps) for the long-lived FTP flows

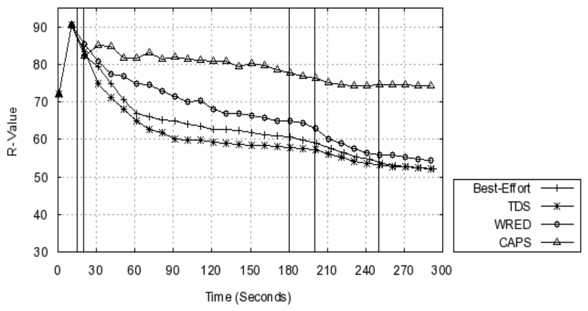


Figure 4: R-value against Time for the VoIP flow1

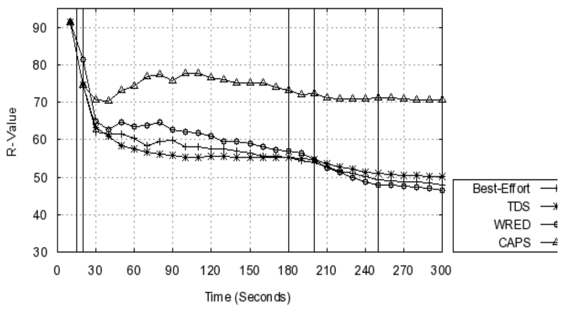


Figure 5: R-value against Time for the VoIP flow2

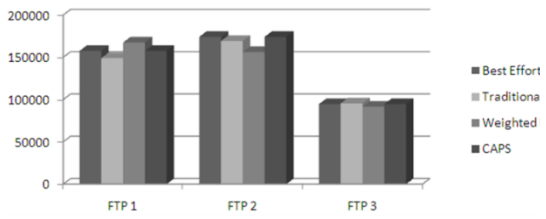


Figure 6: Throughput (Kbps) for FTP flows without VoIP present

5. Conclusions

This paper introduced a novel approach for QoS provisioning for multimedia-rich traffic. Unlike the traditional approach of statically configuring networks to provide tightly bound resource allocation for various traffic types, the presented architecture recognises the changing nature of Internet traffic, and the differing needs of individual users rather than traffic types. Furthermore, the architecture requires no modification to Internet end-hosts, operating at the point of aggregation within a service provider's network, drastically reducing potential deployment difficulties.

The simulation results have shown that the user-centric CAPS algorithm successfully observes the services in use and dynamically controls the ratios of traffic across the bottleneck to maximise the received QoS across the applications in use. In contrast to a traditional Diffserv configuration no bandwidth allocations or reservations are required by the CAPS algorithm, nor does the algorithm require prior configuration. Further differences include the fact that even in the presence of VoIP flows no priority is given to individual packets and all packets are treated equally regardless of the type.

Future work will include extending the CAPS algorithm to accommodate larger network topologies and a wider variety of services being simulated – namely Peer-to-Peer traffic. CAPS will also be developed to include per-user and per-application fairness in addition to the per-flow fairness presented herein, in order to ensure that fairness can be balanced with QoS at a number of levels.

6. References

[1] Miniwatts-Marketing-Group, "World Internet usage statistics news and population stats." vol. 2010, 2010, p. Internet User Statistics.

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services." vol. IETF RFC 2475, 1998.

[3] B. Davie, A. Charny, J. C. R. Bennett, K. Benson, J. Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis, "An expedited forwarding PHB (per-hop behavior)." vol. IETF RFC 3246, 2002.

[4] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured forwarding PHB group." vol. IETF RFC 2597, June 1999.

[5] Q. Yang and J. M. Pitts, "Guaranteeing Enterprise VoIP QoS with Novel Approach to DiffServ AF Configuration," in *IEEE International Conference on Communications*, 2007, pp. 640-645.

[6] K. Hsu-Yang and K. Fu-Wen, "The Dynamic User-oriented QoS Specification and Mapping for multimedia Differentiation Services," in *IEEE International Conf. on High Speed Networks and Multimedia Communications*, 2002, pp. 365-369.

[7] T. Ahmed, R. Boutaba, and A. Mehaoua, "A measurement-based approach for dynamic QoS adaptation in DiffServ networks," *Computer Communications,* vol. 28, pp. 2020-2033, 2005.

[8] H. Shimonishi, I. Maki, T. Murase, and M. Murata, "Dynamic fair bandwidth allocation for DiffServ classes," in *IEEE International Conf. on Communications*, 2002, pp. 2348-2352.

[9] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control." vol. IETF RFC 2581, 1999.

[10] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling TCP Reno performance: a simple model and its empirical validation," *IEEE/ACM Transactions on Networking (ToN),* vol. 8, pp. 133-145, 2000.

[11] ITU-T., "The E-Model, a computational model for use in transmission planning," 2000.

[12] R. G. Cole and J. H. Rosenbluth, "Voice over IP performance monitoring," *ACM SIGCOMM Computer Communication Review,* vol. 31, pp. 9-24, 2001.

[13] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance,"

IEEE/ACM Transactions on Networking (TON), vol. 1, pp. 397-413, 1993.

[14] Cisco Systems Inc., "Cisco IOS Netflow," 2007.

[15] ns2 Network Simulator, "Available from: http://isi.edu/nsnam/ns/," 2010.

[16] P. Pieda, J. Ethridge, M. Baines, and F. Shallwani, "A network simulator differentiated services implementation open IP, Nortel Networks," 2000.

[17] T. Friedman, R. Caceres, and A. Clark, "RTP control protocol extended reports (RTCP XR)." vol. IETF RFC 3611, 2003.

[18] J. Cao, W. S. Cleveland, Y. Gao, K. Jeffay, F. D. Smith, and M. C. Weigle, "PackMime: Synthetic web traffic generation in ns-2, Available from: http://dirt.cs.unc.edu/packmime," 2004.

[19] ITU-T., "Pulse Code Modulation (PCM) of voice frequencies," 1988.

[20] J. Ramírez, J. M. Górriz, and J. C. Segura, "Voice Activity Detection. Fundamentals and Speech Recognition System Robustness," *Proc. Robust Speech Recognition and Understanding,* pp. 460 - 482, 2007.

User-Centric Quality of Service Provisioning

M. E. Culverhouse, B. V. Ghita and P. Reynolds.

Centre for Security, Communications & Network Research
University of Plymouth
Plymouth, United Kingdom
info@cscan.org

***Abstract*— This paper presents a novel extension to the Assured Forwarding per-hop-behaviour for a Diffserv-based network, enabling a more user-centric approach when provisioning for Quality of Service over IP networks. Presented is the Congestion Aware Packet Scheduler (CAPS), a novel traffic management algorithm that uses application performance metrics to adjust resource allocation. The CAPS algorithm has been validated through extensive simulations of a large Internet topology coupled with realistic traffic models. The results show that irrespective of the combination of user traffic, the CAPS algorithm successfully manages the allocation of bandwidth across applications to provide the optimal aggregate-QoS, offering an improvement over alternative network configurations.**

Keywords- QoS, Dynamic, User-centric Diffserv, Congestion Aware Packet Scheduler, Adaptive QoS

INTRODUCTION

The Internet today provides a worldwide delivery network to an ever expanding array of multimedia services used by an estimated 1.6 Billion users [1]. However, this growth could not have been foreseen almost 30 years ago when the Internet Protocol (IP) was designed as a best-effort delivery service. Despite services such as Voice and Video services migrating to the Internet, IP offers no service guarantees and still only operates as best-effort. Therefore, a significant amount of research focussed on providing Quality of Service (QoS) for the various traffic types transiting the Internet. Two main philosophies were developed by the IETF, Intserv (Integrated Services) [2] and Diffserv (Differentiated Services) [3]. Diffserv, which marks and forwards IP packets based on their traffic type, prevailed and is currently proving to be the accepted approach of providing QoS across the Internet.

Within Diffserv networks, edge routers monitor incoming IP packets and mark them with a Diffserv Code Point (DSCP) determined by a policy defined by the network operators. The DSCP value then indicates to core routers of its forwarding requirements, typically relating to onto one of three different Per Hop Behaviours (PHBs): Expedited Forwarding (EF) [4], Assured Forwarding (AF) [5] and Best Effort (BE). It is normally recommended to use the EF PHB for real-time services, such as Voice over IP (VoIP), in order to meet the stringent QoS requirements (low packet loss, jitter and delay) typically associated with such services. The AF PHB provides an assured level of service to a flow providing it adheres to an agreed policy. If the flow exceeds a committed information rate (CIR) then it stands to receive a lower priority DSCP marking and is queued accordingly. The BE PHB operates as the name suggests and makes no guarantees to deliver the packets.

While traditional Diffserv configurations have until now provided a popular mechanism for operators to offer service guarantees across their networks, it is argued in this paper that simply configuring a Diffserv network to offer a predetermined precedence to specific traffic types can result in unfairness for customers not using favoured traffic types. For example, a stock-broker may consider transactional data to be their most important traffic type, but such traffic may be marked as a non-real-time service and hence not receive the desired priority.

There have been a number of proposed methods for providing a more dynamic, user-centric approach to QoS configurations. For example, [6] and [7] present similar concepts that introduce a user interface that allows a user to evaluate the Diffserv traffic classification for their traffic and adjust if necessary. However, whilst these concepts do attempt to move away from the traditional static Diffserv configurations, it does require user interaction which may be intrusive to the user experience or add an element of confusion for inexperienced users if asked to configure their QoS settings. A less intrusive modification to a Diffserv network is proposed by [8], which, presents a dynamic bandwidth allocation algorithm that achieves per-flow bandwidth guarantees based upon user-subscribed rates, coupled with an allocation algorithm for any residual bandwidth.

Diffserv implementations may also utilise a Bandwidth Broker for admission control purposes, only granting EF flows access to the network if sufficient resources are available to satisfy its requirements. Dynamic admission control within Diffserv networks is also a topical research area. Presented in [9] is an architecture where Diffserv edge routers consult a broker when determining how to handle out of profile packets. The decision made by the broker is based on periodic network statistics provided from the network routers. Alternative methods of enhancing the bandwidth broker have been proposed, however, it is more common for admission control mechanisms to focus on adjusting resource allocation in order to maximise bottleneck utilisation without regard for the resulting QoS of the services in use.

To further complicate the task of providing QoS the nature of Internet traffic has changed over recent years, meaning protocols that could have previously been accepted to operate harmoniously with concurrent flows now can no longer be assumed to do so. Unresponsive real-time applications are no longer the only threat to network congestion, the rise in popularity of Peer-to-Peer (P2P) architectures have placed an additional demand on networks. P2P technologies have been promoted by file sharing applications and now account for almost 70% of all Internet traffic [10]. In essence, all participants (peers) act as both providers and consumers of resources. A P2P-based file sharing application will open numerous connections with other peers who have an identical copy of the desired file, and download pieces of the file from these peers. As a user obtains more and more pieces of the file being downloaded, it will also upload these pieces to other peers, creating a swarm of peers exchanging pieces of a file. In addition to obtaining an unfair share of the bandwidth over other applications (through multiple TCP connections), the rise of P2P has changed Internet traffic dynamics from asymmetric to a more symmetrical model. The apparent result of this is that many ISPs now implement traffic shaping or throttle back P2P traffic. For example, BT has the largest broadband subscription size in the UK [11] and claim to throttle the usage of P2P traffic at their discretion, depending on network conditions (information retrieved from "BT Total Broadband Fair Usage Policy"). BT are not alone in taking this view of P2P traffic, six of the UK's top ten ISPs [11] are reported to implement a form of traffic shaping against P2P protocols, according to [12] (a support page from the P2P client 'Vuze', the most popular P2P client [10]). It is also common practice for ISPs to operate a *fair usage policy,* which specifies a limit to the bandwidth a user can consume (through either uploading or downloading), which upon reaching invokes a restriction on the user's connection to a fraction of its original capacity, affecting all of a user's subsequent activities, not just P2P.

These enhancements to Diffserv networks, whilst a good starting point, often focus on making Diffserv more dynamic from the perspective of reallocating unused bandwidth from traffic classes to where they it can be most profitable to the ISPs, and not necessarily by focussing on satisfying the demands of each user. Any form of brokering in a network introduces an inherent risk of denial of service (access), and is therefore not considered to be fair to all users. Limitations in these proposals and the generally negative attitude towards P2P traffic have motivated a new philosophy when providing QoS. The Congestion Aware Packet Scheduler (CAPS), which was first introduced in [13], can be incorporated into a Diffserv edge router, allowing for seamless integration with existing networks. The algorithm moves away from placing precedence on a single traffic type, and instead focuses on maximising the combined QoS of all user services. This paper presents and builds extensively on the preliminary proposals given in [13], and details how the CAPS architecture has been developed and validated using the *ns*2 simulation software [14] with large scale network topologies and realistic traffic modelling.

The rest of this paper is organised as follows: Section 2 gives an overview of the features and functionality of CAPS. Section 3 describes the development of the architecture within the *ns*2 simulation software, detailing the validation scenarios that were used. Section 4 presents the results from these simulations, followed by a summary of the findings of the research and concluding remarks in Section 5.

Architecture Overview

The CAPS algorithm is located within Diffserv edge routers, classifying, policing and marking packets for subsequent forwarding by core routers. CAPS is based upon the AF PHB [5] and aims to provide a guaranteed minimum level of service (a fair share of the available bandwidth) to Internet flows. The CAPS architecture is designed to operate without an EF PHB, and instead all traffic belongs to a single class but is metered, policed and scheduled across Random Early Detection (RED) managed queues for an optimal aggregate-QoS. The decision not to use EF for real-time services was taken because it is suggested by the authors that the EF PHB cannot provide the optimum aggregate QoS for all users. This statement is justified by the fact that when using the EF PHB an upper limit of bandwidth is allocated to the PHB in order prevent starvation to other classes. Therefore, when this volume of EF traffic has been reached there is a risk of denial of service to subsequent flows. Furthermore, as highlighted in the introduction, using the EF PHB may impose an unfair advantage over users not using real-time services. The packet queuing component of the CAPS architecture utilises a number of RED queues, each configured with increasingly more aggressive parameters (by lowering the $max_{threshold}$ and increasingly the drop probability). Each queue is associated with a DSCP value, hence when the CAPS algorithm marks a packet with an appropriate DSCP value this instructs subsequent routers of the RED queue that the packet should be placed in.

Quality of Service Metrics

In order to optimise the combined QoS for a user the CAPS algorithm must recognize the factors which contribute to the perceived quality of each service. For example, TCP-based applications such as FTP benefit from low packet loss rates to allow the Additive Increase Multiplicative decrease (AIMD) algorithm to reach and maintain the maximum possible throughput for the bandwidth available [15].

Real-time services such as VoIP are far more sensitive to network conditions, requiring low delay, low packet-loss and low jitter levels in order to obtain a good overall quality of service. The actual perceived QoS of a real-time service is largely a subjective measure, based on the expectations of the users. However, a widely accepted parameter based method to express the QoS of a voice call is the ITU-T E-Model [16] and has been adapted for packet switched networks [17]. The E-Model expresses the quality of a voice call as having an R-value between 0 and 100, with 0 being incomprehensible and 100 perfect). The formula to which is given in (1), where Ie (2) and Id (3) are the impairment factors for the voice call given the loss rate (e) and delay (d) respectively, λ_1, λ_2 and λ_3 are codec specific parameters derived in [17]. H(x) is a step function where $H(x) = 0$ if $x< 0$ or $H(x) =1$ if $x \geq 0$.

$$R = 94.2 - Ie - Id \quad (1)$$

$$Ie = \lambda_1 + \lambda_2 \ln(1 + \lambda_3 e) \quad (2)$$

$$Id = 0.024d + 0.11(d - 177.3)H(d - 177.3) \quad (3)$$

The CAPS Policy

For regular TCP-based applications the CAPS algorithm assumes the congestion avoidance and control mechanism within TCP will enforce a moderate level of equilibrium between co-existing TCP-based flows. Therefore, unless the total aggregate throughput for all a user's TCP flows exceeds their statistical share of the bandwidth (link capacity divided by the number of users) CAPS will not intervene. However, if the bottleneck link is congested and a TCP flow exceeds its statistical fair share of bandwidth, then CAPS may chose to place packets from the flow into a more aggressive RED queue, which may result in a packet drop and instruct TCP to throttle back its transmission rates.

Typical HTTP traffic (associated with web browsing and not large file downloads over HTTP) can be described as being relatively short flows (≤ 10 packets per web object). Due to the interactive nature of HTTP the user expects HTTP request-response transactions to be completed as quickly as possible. Furthermore, due to the short-lived nature of HTTP traffic the flows do not possess the longevity for the AIMD algorithm to establish a fast transfer rate. Therefore, for packets identified as being short-lived HTTP, CAPS will always mark with the default DSCP value, aiming for the optimal service for HTTP.

As suggested by our earlier research, it is argued by the authors whether a VoIP call with an R-value < 75 has any usefulness or value to the user, given that below 75 dissatisfaction amongst users typically occurs [16]. Therefore, whilst the CAPS algorithm makes no explicit resource reservation for VoIP traffic, if a user is using VoIP CAPS will maintain the R-value of the service above 75. In recognition of the fact VoIP flows typically use the unresponsive transport protocol UDP, CAPS also imposes an upper threshold on the R-value of a VoIP call in the event of network congestion to permit other active flows to achieve a maximised performance.

The threat that P2P applications, such as the BitTorrent protocol pose to a network was previously highlighted. Typically used to download large files from multiple sources, such applications establish multiple TCP connections in order to obtain a disproportionate share of the available bandwidth [18], leading to the frequent use of traffic shaping by ISPs. However, the CAPS algorithm approaches P2P traffic from a different perspective and suggests that users of P2P applications are doing so in order to download a file as quickly as possible. Therefore, placing a blanket throttling or shaping policy over this traffic is not providing the user with the best possible QoS for their traffic. CAPS aims to manage a user's traffic to ensure the aggregate throughput doesn't exceed a user's share of the bottleneck capacity, and as a result if a user is using solely a P2P application then his/her aggregate throughput may reach but not exceed their fair share (given the network is experiencing congestion). In addition to this handling of P2P traffic when a user uses a P2P based application concurrently with a regular TCP-based application during a period of congestion, CAPS controls the P2P traffic to limit its aggregate throughput to be not greater than that achieved by the co-existing TCP-based applications.

The policing mechanisms for the different traffic types described above are performed by CAPS using an evaluative method on arriving packets at an edge router. The CAPS architecture meters the incoming packets and maintains a flow table for currently active flows (having been observed within the last *x* seconds – finding the optimal *x* with regards to router load is for future investigation). The statistics held on each flow are: packet count, last time observed, congestion history value (CHV), last time policed and average arrival rate for the flow. These statistics are stored for all types of flow, with some addition statistics stored for VoIP flows in order to calculate the flow's R-value; CHV refers to the number of times a flow has been observed to have been behaving in an unfair manner.

The CAPS algorithm

The pseudo code for the operations performed by the CAPS algorithm on each arriving packet is given in Fig.1, where ICP is the default, or Initial Code Point (ICP) that a packet will receive under normal network conditions. ICP + 1, ICP +2 and ICP +3 refer to downgraded DSCP values associated with more aggressive RED queues. The term CHV_i represents the Congestion History Value for $flow_i$, and describes whether $flow_i$ has been policed in the past. Upon a flow being policed (a packet receives a downgraded DSCP value) CHV_i will be incremented by 1. The use of $_{dst}$ within the algorithm refers to the variable value for a particular destination. For example $Flowcount_{dst}$ represents the number of flows observed that were destined for a specific host.

```
For VoIP Flows
if current_util < util_threshold
        then DSCP = ICP
if (current_util > util_threshold)
    & ($R\text{-}value_i$ > UpperVoIPThreshold)
        then DSCP = ICP + 1
For TCP flows not identified as P2P
if ($packet\_count_i$ < HTTPflowsize)
        then DSCP = ICP
else if (current_util > util_threshold)
    & ($rate_i$ > BWShare/$Flowcount_{dst}$)
    & ($\bar{R}$ < LowerVoIPThrehold)
        then
    for (j = 1; j++; $j \leq 3$) )
        if $CHV_i < CHVThreshold_j$
            $CHV_i$ ++
            DSCP = ICP + j
    done
else
        DSCP = ICP
For TCP flows identified as P2P
if (current_util > util_threshold) &
($FTPCount_{dst}$= 0) & ($BTThrput_{dst}$< (BWShare –$VoIPThrput_{dst}$)||
($FTPCount_{dst}$>0) & ($BTThrput_{dst}$< ($FTPThrput_{dst}$/$FTPCount_{dst}$)
        then
    for (j = 1; j++; $j \leq 3$) )
        if $CHV_i < CHVThreshold_j$
            $CHV_i$ ++
            DSCP = ICP + j
    done
else
        DSCP = ICP
```

Figure 1: CAPS Optimisation Algorithm

Testing and Validation

CAPS Implementation

As presented in [13] the CAPS algorithm was implemented in the *ns*2 network simulator environment [14], using the Nortel Networks Differentiated Services Implementation Framework [19]. Also previously highlighted, the calculation of R-values for VoIP flows does not necessarily have to be performed by edge routers; instead CAPS could make use of QoS reports sent by real-time protocols. For example, the extended RTCP report calculates and distributes the R-value between participants [20]. Furthermore, how traffic is identified as a particular application type has not been explored within this paper as it is beyond the scope of this work.

Simulation Design

The network topology shown in Fig. 3 was designed to represent a subsection of a hierarchical Internet topology, with end-users connected to regional ISP networks which are interconnected via an Internet backbone, which in turn has a number of connected content providers. Each ISP network consists of 100 clients, two edge routers and a single core router. The links between these edge routers and the ISP core network will provide the bottlenecks within the topology. This assumption is based upon the belief that the ISP access network is the most likely location for a bottleneck in this topology given the aggregating end-user traffic. It is also noted that routing protocols were not implemented in the simulations since routing algorithms are not the focus of this research. Clients are connected to the ISPs via an asymmetric link with up- and down-stream capacities of 400Kbps and 4Mbps respectively, modelling residential Internet connectivity [21].

For the purposes of testing the CAPS algorithm the simulations featured a simulated VoIP traffic source, a long-lived FTP download, HTTP traffic and a P2P-like application. These traffic sources represent a real-time service, a long-lived congestion controlled TCP connection, interactive traffic and a multi-streamed Peer-to-Peer application, the combinations of which are deemed suitable to test the effectiveness of the proposed algorithm, and encompass popular Internet services.

The VoIP call traffic was simulated between hosts from different ISP networks using a Constant Bit-Rate (CBR) traffic source, with a sending rate of 64Kbps and packetsize of 160 bytes, this traffic source closely models the G.711 codec [22]. The FTP downloads were simulated between the FTP servers and ISP clients. This traffic source was used to represent the rising trend of one-click-hosting services (such as RapidShare) where users download large files via FTP and HTTP [23].

When simulating P2P traffic in *ns*2, a number of modular solutions are available, [24] and [25] are two examples that provide P2P and BitTorrent (a P2P protocol) functionality.

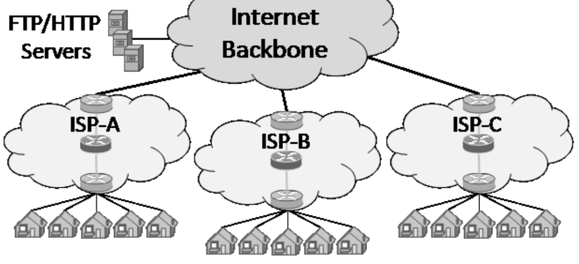


Figure 2: Simulated Network Topology

Since the evaluation of CAPS only requires the establishment of connections and exchange of data between P2P peers it was decided to implement a P2P-like traffic source that would instruct a peer to establish a number of connections with a random selection of peers, and download data for a random period of time. This random nature of data exchange between peers is considered to emulate P2P-like traffic sufficiently enough to evaluate CAPS.

The simulations were conducted using five different "traffic profiles" which describe a number of combinations of services, which included VoIP, FTP and P2P traffic. The aim was to prove the CAPS algorithm can provide an improvement to the overall QoS delivered to a user, irrespective of the applications or services in use. The simulation was run using four traffic management schemes, Best effort, traditional Diffserv (TDS), WRED and the CAPS architecture. The TDS configuration featured three traffic classes, EF, AF and BE. VoIP was assigned to the EF group providing explicit precedence over other traffic types. The FTP and HTTP flows were assigned to the AF group, guaranteeing a predetermined level of service providing each flow stayed within its CIR. The P2P-like traffic was assigned to the BE group, the justification for this was based on the perceived attitudes of ISPs towards P2P, as per the discussion in the introduction.

The WRED configuration featured three weighted RED-queues, VoIP flows are assigned to the highest priority queue with the most conservative RED parameters. HTTP and FTP traffic was assigned to a second queue with marginally more aggressive RED parameters. The remaining P2P-like traffic was assigned to the third queue, which was configured with the most aggressive RED parameters of the three queues. The Best Effort configuration featured a single queue, to which all traffic was assigned. The CAPS configuration featured a single traffic class with four precedence queues within that class. No single traffic type was given explicit priority prior to the simulation beginning, neither were any CIR values declared for any of the flows. The four traffic management configurations were simulated using ns2, with each simulation lasting 5minutes.

DISCUSSION OF SIMULATION RESULTS

To avoid repeating the discussion for each separate ISP, only the results for a single ISP (ISPA) are presented, with supporting figures on the following pages. It should be noted that due to the symmetrical design of the simulations, the results for each ISP are very similar.

The average R-value for users only using VoIP is given in Fig. 3. Unsurprisingly, the highest rated VoIP quality was achieved using TDS, which provided VoIP with explicit priority. However, between 60 and 240 seconds, when the network became heavily congested, CAPS successfully maintained the R-value of the VoIP flows above 75, averaging 78.4 during this period, an improvement of 10% and 17% over WRED and Best effort respectively.

Fig. 4 shows the average throughput achieved by FTP over time, for users only using this service. The impact of giving VoIP traffic explicit priority under TDS resulted in concurrent traffic being starved of resources, achieving an average throughput of 0.15Mbps, only 40% of what the users' fair share of the bottleneck would have been (0.38Mbps). Under CAPS the average throughput was 0.78Mbps, which despite being more than the fair share can be seen to be policed back on a number of occasions, whereas the flows under WRED and Best effort achieve average throughputs of more than 300% greater than the user's fair share.

For users solely using P2P services the average throughput shown in Fig. 5. The static traffic precedence of TDS again limits the performance for less-favoured services, in this case restricting the total P2P throughput per user to less than 50% of their fair share of the bandwidth. While CAPS maintained the total P2P throughput within 10% of the user fair share. Both WRED and Best effort configurations allowed for P2P services to consume far beyond the fair share. It should be noted that the drop in throughput across all scenarios from 150 seconds is due to the number of active peers declining over time.

Fig. 6, 7 and 8 represent the respective performance for users engaged in VoIP, P2P and FTP services. CAPS manages to maintain the R-value of the VoIP flow within 4% of the target 75, while also balancing the throughput achievable by FTP and P2P services, ensuring P2P does not obtain an unfair proportion of resource. Whereas TDS provides an improvement of almost 10% over CAPS for VoIP traffic, FTP and P2P traffic achieved only 14% and 50% of the throughput achieved using CAPS.

The final simulated traffic type was HTTP, for which CAPS achieved the lowest average request-response time of 170ms, an improvement of 34%, 30% and 24% on TDS, Best effort and WRED respectively. Fig. 9 shows a CDF plot for HTTP response times for each of the four scenarios.

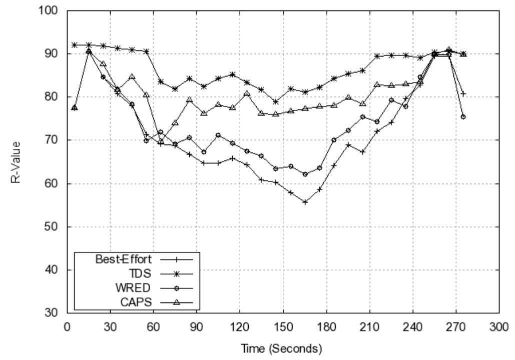


Figure 3: Average VoIP performance for users only using VoIP

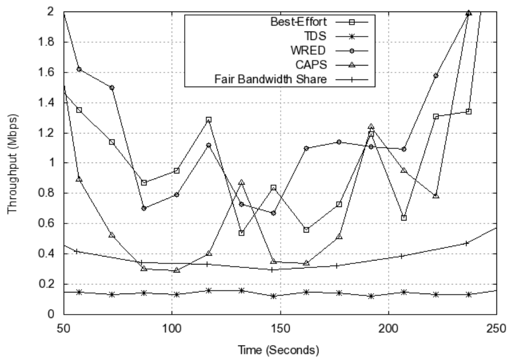


Figure 4: Average FTP throughput for users only using FTP

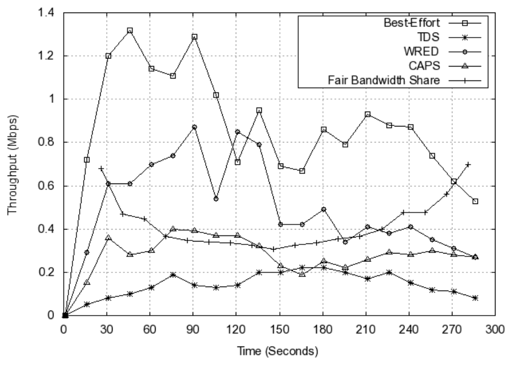


Figure 5: Average P2P throughputfor users using only P2P

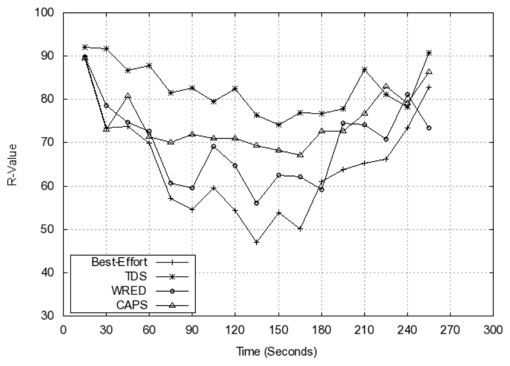


Figure 6: Average VoIP performance for users using VoIP, FTP & P2P

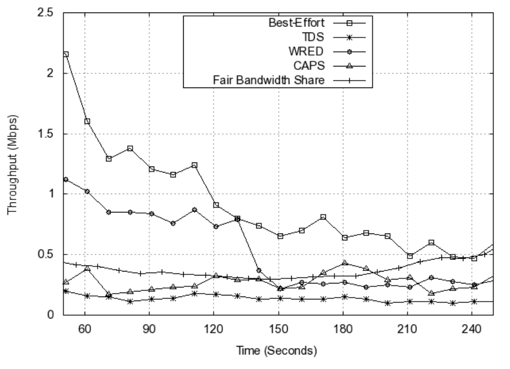


Figure 7: Average P2P throughput for users using VoIP, FTP & P2P

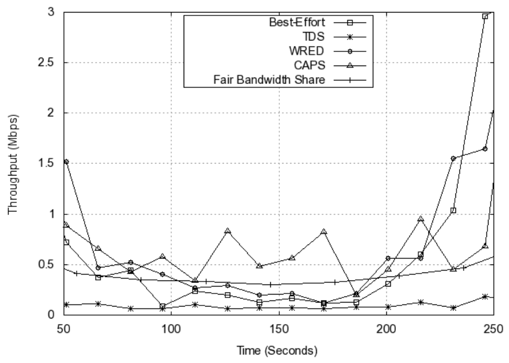


Figure 8: Average FTP throughput for users using VoIP, FTP & P2P

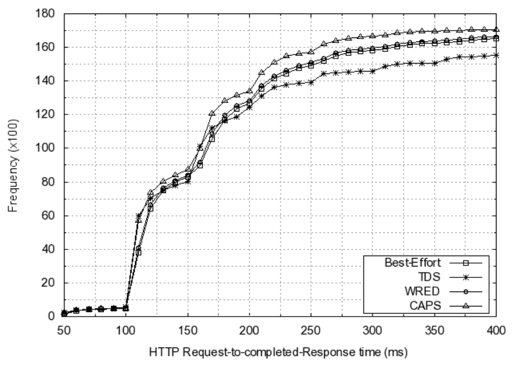


Figure 9: Cumulative Distribution of HTTP Request-Response times (ms)

CONCLUSIONS

This paper has presented a novel architecture for providing QoS within IP networks from a user-centric perspective. The CAPS architecture aims to introduce a dynamic resource management system within Diffserv based infrastructures that can equilibrate QoS with the fairness of resource allocation among users and application traffic.

The CAPS algorithm has been developed to meter, police and schedule user traffic, ensuring each user receives, at minimum, a fair share of the available bandwidth, irrespective of the services they are using. Within a user's share of bandwidth, CAPS will maintain the QoS of VoIP traffic (if present) above a threshold rather than explicitly reserving bandwidth for individual services.

Through extensive simulations using large scale topologies and realistic Internet traffic the CAPS algorithm has been validated to show that for a number of different traffic profiles the aggregate QoS for a user is improved using CAPS compared with traditional Diffserv configuration, a Weighted-RED and Best-Effort scenario. CAPS ensures that the aggregate throughput of "greedy" applications, such as P2P, is maintained within a user's fair share and if necessary restricts this throughput to allow an equal distribution of resources between a user's applications, irrespective of how many connections an application establishes.

It is noted that controlling the throughput of TCP can be quite difficult, and while the throughput under CAPS is maintained closer towards the fair share threshold, the fast recovery mechanisms of TCP Reno make the protocol quick to claim any unused bandwidth, Therefore, further work will be conducted to identify the optimal method of controlling TCP.

REFERENCES

[1] Miniwatts-Marketing-Group, "World Internet usage statistics news and population stats," 2010, http://www.internetworldstats.com/stats.htm, accessed 02-02-2010

[2] R. Braden, D. Clark, and S. Shenker, "Integrated services in the Internet architecture," IETF RFC1633, 1994

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," IETF RFC 2475, 1998

[4] B. Davie, A. Charny, J. C. R. Bennett, K. Benson, J. Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis, "An expedited forwarding PHB (per-hop behavior)," IETF RFC 3246, 2002

[5] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured forwarding PHB group," IETF RFC 2597, June 1999

[6] H. Y. Kung and F. W. Kuo, "Dynamic user-oriented QoS specification and mapping for multimedia differentiation services," *IJCSNS,* vol. 6, p. 116, 2006.

[7] K. Molnar and V. Vrba, "DiffServ-based user-manageable quality of service control system," in *Proc. of the 7th Int. Conf. on Networking*, 2008, pp. 485-490.

[8] J. Elias, F. Martignon, A. Capone, and G. Pujolle, "A new approach to dynamic bandwidth allocation in quality of service networks: performance and bounds," *Computer Networks,* vol. 51, pp. 2833-2853, 2007.

[9] T. Ahmed, R. Boutaba, and A. Mehaoua, "A measurement-based approach for dynamic QoS adaptation in DiffServ networks," *Computer Communications,* vol. 28, pp. 2020-2033, 2005.

[10] H. Schulze and K. Mochalski, "Ipoque Internet Study 2008/2009," January 2010.

[11] ISPReview, "ISPReview - Top 10 UK ISPs," 2010, http://www.ispreview.co.uk/review/top10.php, accessed January 2010

[12] Vuze, "A list of ISPs reported to implement a form of traffic management against P2P protocols.," 2010, http://wiki.vuze.com/, accessed January 2010

[13] M. E. Culverhouse, B. V. Ghita, P. Reynolds, and X. Wang, "Optimising Quality of Service through the controlled aggregation of traffic," 2010, Unpublished

[14] ns2 Network Simulator, "Available from: http://isi.edu/nsnam/ns/," 2010

[15] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," IETF RFC 2581, 1999

[16] ITU-T., "The E-Model, a computational model for use in transmission planning," ITU-T Recommendation G.107, 2000.

[17] R. G. Cole and J. H. Rosenbluth, "Voice over IP performance monitoring," *ACM SIGCOMM Computer Communication Review,* vol. 31, pp. 9-24, 2001.

[18] J. J. Martin and J. M. Westall, "Assessing the impact of BitTorrent on DOCSIS networks," in *IEEE Broadband Communications, Networks and Systems 2007*, 2007, pp. 423-432.

[19] P. Pieda, J. Ethridge, M. Baines, and F. Shallwani, "A network simulator differentiated services implementation open IP, Nortel Networks," 2000.

[20] T. Friedman, R. Caceres, and A. Clark, "RTP control protocol extended reports (RTCP XR)," IETF RFC 3611, 2003

[21] Ofcom, "UK broadband speeds: initial findings of consumer experience of broadband performance," 2008.

[22] ITU-T., "Pulse Code Modulation (PCM) of voice frequencies," ITU-T Recommendation G.711, 1988.

[23] H. Schulze and K. Mochalski, "Ipoque Internet Study 2008/2009," 2009, http://www.ipoque.com/resources/internet-studies/internet-study-2008_2009, accessed

[24] K. Eger, T. Hoßfeld, A. Binzenhöfer, and G. Kunzmann, "BitTorrent Implementation for *ns*2 Network Simulator," 2007, http://www.tu-harburg.de/et6/research/bittorrentsim/index.html, accessed

[25] Q. He, "Packet-level peer-to-peer simulation framework and gnutellsim," 2003, http://www.cc.gatech.edu/computing/compass/gnutella/, accessed January 2010