Université d'Ottawa · University of Ottawa

Université d'Ottawa · University of Ottawa

FACULTÉ DES ÉTUDES SUPÉRIEURES
ET POSTDOCTORALES

FACULTY OF GRADUATE AND
POSTDOCTORAL STUDIES

Zhao CHEN

AUTEUR DE LA THÈSE - AUTHOR OF THESIS

M. A. Sc. (Electrical Engineering)

GRADE - DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT - FACULTY, SCHOOL, DEPARTMENT

TITRE DE LA THÈSE - TITLE OF THE THESIS

Differentiated Services in an Integrated Broadband Network

D. Makrakis

DIRECTEUR DE LA THÈSE - THESIS SUPERVISOR

CO-DIRECTEUR DE LA THÈSE - THESIS CO-SUPERVISOR

EXAMINATEURS DE LA THÈSE - THESIS EXAMINERS

I. Lambadaris

O. Yang

J.-M. De Koninck, Ph.D.

LE DOYEN DE LA FACULTÉ DES ÉTUDES
SUPÉRIEURES ET POSTDOCTORALES

DEAN OF THE FACULTY OF GRADUATE
AND POSTDOCTORAL STUDIES

DIFFERENTIATED SERVICES IN AN INTEGRATED BROADBAND NETWORK

Zhao Chen

A thesis submitted to Faculty of Graduate and Postdoctoral Studies in partial fulfillment of the requirements for the degree of **Master of Applied Science**, Electrical Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario, Canada

Your file Votre référence
ISBN: 0-494-01437-7
Our file Notre référence
ISBN: 0-494-01437-7

ABSTRACT

This thesis evaluates the Differentiated Services (diffserv) mechanism proposed by IETF to provide QoS and resource reservation under bursty web traffic. Particularly, the performance of Weighted Fair Queuing (WFQ) algorithm is studied, using two-node and multiple link network configuration. A self-similar realistic web workload model generating aggregation of web-accessing traffic flows is developed and used to evaluate the performance, as well as other multimedia applications. Our analysis reveals that due to the burstiness of web traffic, diffserv cannot achieve all the desired QoS guarantees, especially packet delay and jitter. Three mainstream multimedia applications are modeled and analyzed in the Diffserv network with our realistic web traffic model as background traffic. Finally, our performance analysis also demonstrates that in more realistic network environment, only bi-directional resource reservation can provide to the customers QoS guarantees in Diffserv network. Several scenarios of aggregations of applications traffic flows are used to evaluate the performance, such as shot-lived web traffic and non-adaptive UDP traffic.

ACKNOWLEDGEMENT

The author wishes to express deepest gratitude to my supervisor, Professor Dimitrios Makrakis, for his continuous support, interesting discussion and direction throughout the research period. I would also like to thank the other member of my thesis committee for their valuable comments and suggestions.

I am very thankful to the Broadband Wireless Internetworking Research Lab team for sharing uncountable hours, ideas and fruitful hints.

Finally, special thanks to my parents and sister for their continued support and encouragement.

TABLE OF CONTENTS

LIST OF FIGURE

LIST OF TABLES

LIST OF ACRONYMS

AF	Assured Forwarding.
DiffServ	Differentiated Service.
EF	Expedited Forwarding.
FIFO	First In first Out.
ISP	Internet Service Providers.
PHB	Per-hop Behaviour.
RED	Random Early Detection.
RIO	RED with In and Out.
RTT	Round Trip Time.
TCP	Transport Control Protocol.
UDP	User Datagram Protocol.
WFQ	Weighted Fair Queuing.

Chapter 1 Introduction

The Internet is currently based on the best-effort model that treats all traffic steams in the same way. The best-effort model has been successful till now because a large proportion of the traffic in the Internet is TCP-based. The TCP end-to-end congestion control mechanisms will force the TCP sources to back off whenever congestion is detected in the network [1]. However, such a dependence on the end systems' cooperation is becoming increasingly unrealistic. Given the current best-effort model with FIFO (First in first out) queuing inside the network (i.e. routers), it is relatively easy for non-adaptive sources to gain greater shares of network bandwidth and thereby starve other, well-behaved, TCP sources. The best-effort model is also inadequate for multimedia applications, such as real time audio and video, which require explicit bandwidth and delay guarantees. Moreover, the best-effort model treats all packets equally, once they have been injected into the network. Therefore, it is difficult for Internet Service Providers (ISPs) to provide services that are commensurate with the expectations of consumers who are willing to pay more for a better class of service.

The above issues have led to a number of proposals for providing differentiated services (diffserv) [2] in the Internet. The differentiated service approach allows service providers to offer different levels of service to a number of classes of aggregated traffic flows in a differentiated service domain. For example, an ISP may offer two levels of services – a premium service [3] for customers who are willing to pay more and a best-effort service at a lower price. By using this approach, Quality of Service (QoS) sensitive traffic streams like video and audio can get better services, such as low delay, low packet losses. Currently, two types of services have been proposed in the differentiated services by IETF (Internet Engineering Task Force) [4]. One is the Assured Forwarding (AF) service [5], the second is called Expedited Forwarding (EF) service [6]. AF service provides different levels of forwarding assurances to IP packets while EF service provides low delay, low loss (should the traffic streams remain within their agreed upon contract) and low jitter.

1.1 Motivations and Objectives

With the introduction of diffserv service, a number of issues arise. These questions/issues include: (1) The network traffic model's influence on the performance of a diffserv network; (2) The right metrics of the quality of the service provided to customers in diffserv-capable networks; (3) Is diffserv capable to support the real-time applications with low delay, low packet loss ratio and low jitter; (4) The allocation and reservation for different customers in diffserv-capable networks; (5) The impact of round trip time (RTT) and the impact of non-adaptive traffic on diffserv-capable networks; (1) The effect of ACK loss on TCP traffic in a diffserv-capable network.

The main objective of this thesis is to evaluate the traffic performance of different applications (e.g. WWW, real-time Video application and Voice) in a diffserv-capable network under different constraints. Here, traffic performance means the quality of service (QoS) [7] different kinds of traffic can get from the Internet Service Provider (ISP), such as the amount of bandwidth a WWW user can get from the ISP, the number of packets that will be dropped when there is congestion in the bottleneck link and the packet delay of Telephony over IP. First of all, we aim to evaluate the DiffServ network using self-similar bursty web traffic model described in chapter 3 which can capture the bursty nature of real web traffic. Secondly, the critical multimedia applications like video, voice and 3-D virtual environment are examined in DiffServ network. Another goal of this thesis work is to evaluate the DiffServ network under various constraints, such as ACK loss, UDP traffic, RTT and multiple bottlenecks. By analyzing the traffic performance under different conditions, we should be able to determine the behavior of applications in a diffserv-capable network and then provide answers to the issues mentioned above.

In this thesis work, a number of simulations have been performed to evaluate diffserv on a single bottleneck network and multi-bottleneck network. All the simulations have been performed using OPNET 6.0 [8], a network simulation tool developed by Mil3 Inc.

1.2 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 presents the concepts of differentiated service, Assured Forwarding service and Expedited Forwarding Service. The basic ideas of RIO, Priority Queuing and WFQ algorithms that have been used for the queue management of diffserv routers are also introduced in chapter 2. Chapter 3 depicts the traffic models used in our simulation. Chapter 4 describes the details of the realistic bursty traffic modeling and its impact on the Differentiated Services network; simulation results and their analysis are also presented. Additionally in chapter 4 we investigate multimedia applications over DiffServ networks (voice, video and distributed virtual environment applications). Chapter 5 presents the resource reservation scheme and analyzes the Differentiated Services network under different conditions. Chapter 6 concludes the thesis and provides directions for future research.

Chapter 2 Overview of Differentiated Services

This chapter presents an overview of the concepts of Differentiated Services, Assured Forwarding service and Expedited Forwarding service. These concepts form the basis of the simulations that will be presented in the following chapters.

2.1 Differentiated Services

2.1.1 Definition of Differentiated Service

The differentiated services (DS) [2] architecture is based on a simple model where traffic entering a network is classified and possibly conditioned at the boundaries of the network, and assigned to different behavior aggregations. Each behavior aggregate is identified by a single Differentiated Services CodePoint (DSCP) [9]. Within the core of the network, packets are forwarded according to the per-hop behavior (PHB) [2] associated with their DSCP. The ultimate goal of the differentiated services is to provide network support for end-user service level agreements. The per-hop behavior is the externally observable forwarding behavior applied at a DS-compliant node to a DS behavior aggregate. By supporting differentiated service in Internet, the network service providers could offer different types or grades of service to different customers (e.g. video and audio require low delay and jitter while file transfers require high throughput). This means that the customers who want to pay more could get better traffic performance from the networks.

In order to provide different classes of services to different customers, the TOS (Type of Service) field in the headers of Ipv4 packets may be used [9]. Different values in this field indicate different types of services the customer may get. A value in this field is also called a DS codepoint. The TOS fields of the packets could be set by the traffic sources or by the boundary node of every DS domain the packets enter.

2.1.2 Differentiated Services Domain

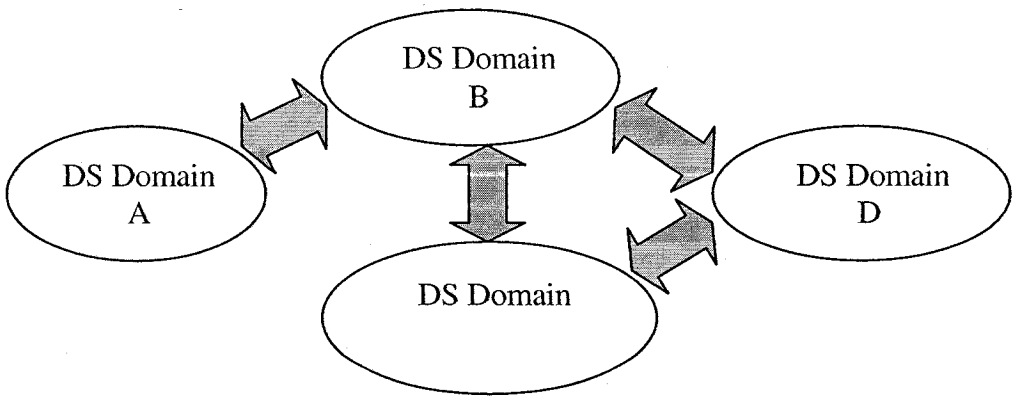
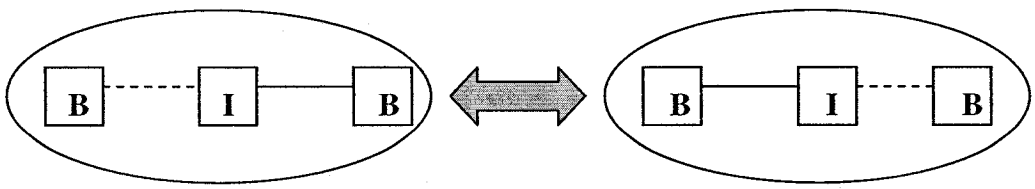


Figure 2-1 A number of interconnected DS domains

A DS domain [2] is a contiguous set of DS nodes that operate with the same service provisioning policy on each node. Figure 2-1 shows a number of interconnected DS domains.



Domain 1 **Domain 2**

Figure 2-2 A number of DS boundary nodes and DS interior nodes

A DS domain consists of DS boundary nodes and DS interior nodes [2]. DS boundary nodes interconnect the DS domain to other DS or non-DS-capable domains, while DS interior nodes only connect to other DS interior or boundary nodes within the same DS domain. Figure 2-2 shows a number of DS boundary nodes and DS interior nodes (B is DS boundary node; I is DS interior node). One of the important features of DS domains (i.e., DS-capable networks) is that most of the complexity, related to the support of differentiated services, is

located at the boundary nodes while interior nodes are kept simple [10]. This means that core nodes offer services only for aggregated traffic rather than on a per-flow basis. It is the boundary nodes' responsibility to classify the packets into several behavior aggregates, meter the traffic against profiles, mark/remark packets, shape or drop packets, etc [11]. (a description of these operations is presented in the next section). Interior nodes will just forward the aggregated traffic according to the DSCP (set by boundary nodes or flow sources) in the packets' headers.

2.1.3 Boundary Nodes: Traffic Classification and Conditioning

The packet classification policy identifies the subset of traffic that may receive a certain type of service treatment from the network. Packets are being conditioned or mapped to one or more behavior aggregates (i.e., codepoint re-marking) within the DS domain. Traffic conditioning consists of metering, shaping, policing and/or re-marking to ensure that traffic entering the DS domain conforms to the rules specified in the profile, in accordance with the domain's service provisioning policy [2].

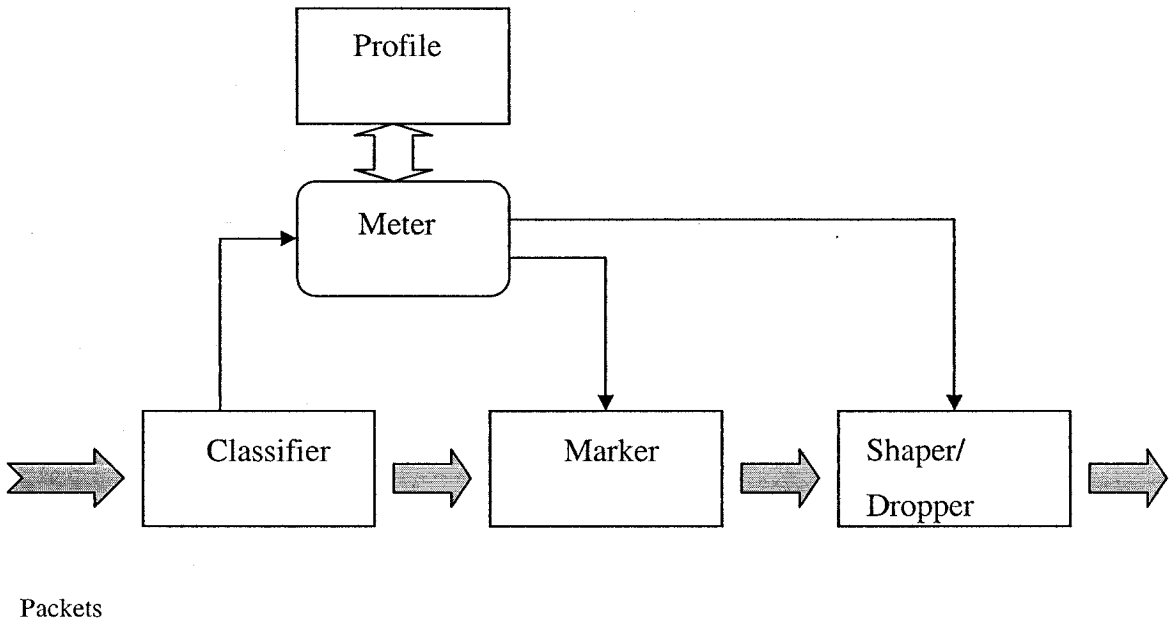


Figure 2-3 Logical View of a Packet Classifier and Traffic Conditioner

Figure 2-3 displays the processes performed by a packet classifer and traffic conditioner at a DS boundary node. Definitions and descriptrions of the classifier, profile, meter, marker, shaper and dropper follow..

In a boundary node of a DS domain, a flow of packets will be classified into several classes by a classifier [2] [11]. The classifier classifies the traffic by reading the codepoint in the packet's header if this packet has already been pre-marked; otherwise, it will read other fields in the packet's header, such as source/destination IP address and protocol identifier. For instance, if we want to safeguard the real-time video traffic from web traffic, we should mark the video traffic with low drop precedence and mark the web traffic with high drop precedence at the boundary nodes. Thus, when congestion occurs in the network, web packets will be dropped first by core nodes. The meter at the boundary nodes will then measure the traffic stream against a traffic profile, which specifies the temporal properties of this stream. A profile based on a token bucket may look like "codepoint = X, use token bucket rate R, buffer B". This profile means that all the packets with codepoint X will be measured against a token bucket meter with data rate R and burst size B. Out of profile packets are those packets that arrive when there aren't tokens available in the bucket. These packets along their way through the network may be remarked with a lower priority by a marker, shaped by a shaper or dropped by a dropper at the boundary nodes. Packet markers will set the DS field of a packet to a particular codepoint, adding the marked packet to a particular DS behavior aggregate. Shapers will delay some of the packets in a traffic stream in order to bring the stream into compliance with a traffic profile. Droppers will discard some of the packets in a traffic stream in order to bring the stream into compliance with a traffic profile, also known as "policing" the stream.

There are two kinds of differentiated services that have been proposed by IETF: (1) Assued Forwarding (AF) service; and (2) Expedited Forwarding (EF) service. A detailed depiction of these two services is presented in the follwing sections.

2.2 Assured Forwarding Service

Assured Forwarding (AF) service [6] is a service that allows the Internet Service Provider (ISP) to offer different levels of forwarding assurances for IP packets received from a

customer. The idea behind AF is to give the customer the assurance of a minimum throughput, even during periods of congestion, while allowing customers to consume more bandwidth when the network utilization is low. Thus, a connection using the AF service should achieve a throughput equal to the subscribed minimum rate, also called target rate, plus some share of the remaining bandwidth gained by competing with all the active connections. In a typical application, a company uses the Internet to interconnect its geographically distributed sites and wants an assurance that IP packets within this "Intranet" are forwarded with high probability as long as the aggregate traffic from each site does not exceed the subscribed information rate in the profile. It is desirable that a site may exceed the subscribed profile with the understanding that the excess traffic is not delivered with the same chance as the traffic that is within the profile.

Four AF sub-classes are defined in the AF service [6]. In each AF sub-class, IP packets can be marked with one of three possible drop precedences. In case of congestion, the drop precedence of a packet determines the relative importance of the packet within the AF sub-class. A congested DS node tries to protect packets with lower drop precedence from being lost by discarding packets with higher drop precedence. By using the drop precedence, we can effectively control non-adaptive sources (e.g. UDP sources) from getting more than their fair share of network resources. However, it has been reported in [12] that using three drop precedences or two drop precedences in an AF service class has almost no impact on the performance of the system.

In some earlier works, a marking algorithm (Three Color Marker (TCM) [13] or Time Sliding Window (TSW) [14]) and RIO (RED with IN and OUT [15]) algorithm have been used to implement the AF service. Description of these algorithms is presented in following sections. The marking algorithms (TCM or TSW) have been implemented at the boundary nodes of an AF service-capable network in order to mark the packets with different drop precedences. The RIO algorithm has been implemented in the interior nodes of an AF service-capable network in order to manage packets differently, according to the packet's drop precedence.

The performance of the simpler RIO algorithm is much less predictable according to several studies [19] [20] and it is hard to support the premium service, the most valuable service of the ISPs. Since we already studied the performance of RIO scheme and its obvious limit in our previous work [20] [21], in this thesis we choose WFQ as the scheduling scheme for AF service.

2.3 Expedited Forwarding Service

Expedited Forwarding (EF) service [5] provides low loss, low latency, low jitter, and bandwidth guarantees on end-to-end basis though the DS domain. This service sometimes is also called Premium service. Examples of applications that might use this service are IP telephony, video conferencing, or Virtual Private Networks, since they require low jitter and low delay in fore the applications to be of acceptable quality. Loss, latency and jitter are all due to size of the queues through which the traffic traverses while transiting the network. Therefore, providing low loss, latency and jitter for some traffic aggregate means ensuring that the traffic perceives no (or very small) queues; this can be achieved by ensuring that, at any time, the output capacity is higher or equal to the input capacity of a given queue. Several types of queue scheduling mechanisms may be employed to implement the EF service; examples are priority queuing (PQ), weighted round robin (WRR), weighted fair queuing (WFQ) and RIO. In this thesis, we used the priority queuing model to support the EF service. Using this model requires the integration of some procedures to limit the damage EF traffic could inflict on other traffic, such as AF or best-effort traffic. A token bucket or a rate estimator must be implemented at the boundary nodes and police the EF traffic streams to ensure that traffic, which exceeds its subscribed limit will be discarded. By doing so, delay is not introduced to the traffic.

2.4 Random Early Detection (RED) Algorithm

The RED algorithm [17] is a congestion avoidance and congestion control algorithm that detects incipient congestion by computing the average queue size. The RED algorithm reacts to congestion either by dropping packets arriving at the router or setting an appropriate bit in the packet headers. More specifically, when the average queue size exceeds a preset threshold, the router drops or marks each arriving packet with a certain probability, where the probability is a function of the average queue size. In our simulations, when congestion occurs, packets are dropped upon their arrival. This choice is based on the fact that we just use the TCP end-to-end congestion control mechanism to close the congestion window upon each packet loss and do not have to change the TCP protocol.

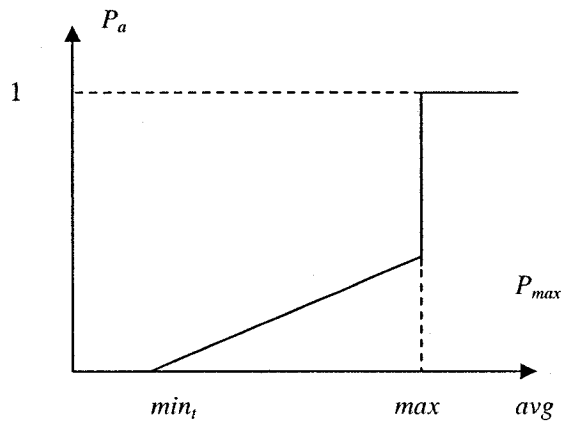


Figure 2-4 RED Algorithm

When a packet enters a network node (e.g., router), the RED gateway first calculates the average queue size using a low-pass filter, which has as input the instantaneous queue size, thus filtering out transient bursts occurring in the router. The average queue size is compared to two thresholds, a minimum threshold min_{th} and a maximum threshold max_{th}. When the average queue size is less than the minimum threshold, no packets are dropped; this is the normal phase [14] of RED algorithm. When the average queue size is greater than the maximum threshold, every arriving packet is dropped; this is the congestion control phase [14] of RED algorithm. This behavior ensures that the average queue size does not significantly exceed the maximum threshold. When the average queue size is between the

minimum and the maximum threshold, each arriving packet is dropped with probability P_a, where P_a is a function of the average queue size avg; this is the congestion avoidance phase [14] of RED algorithm. Each packet drop serves the purpose of indirectly notifying the (source) end host's transport layer to reduce its sending rate.

A RED algorithm is configured with the following parameters: min_{th}, max_{th}, and P_{max}. It works as illustrated in Figure 2-1; the x axis indicates the average queue size, avg, which is updated upon each packet arrival. The y axis indicates the probability of dropping an arriving packet P_a. The three phases of RED algorithm are shown in Figure 2-4.

2.5 RED with IN and OUT (RIO) Algorithm

RIO [14] [15] stands for RED algorithm with IN/OUT bit. RIO uses twin RED algorithms for dropping packets, one for IN packets and one for OUT packets. By choosing the parameters for both algorithms differently, RIO is able to discriminate against OUT packets.

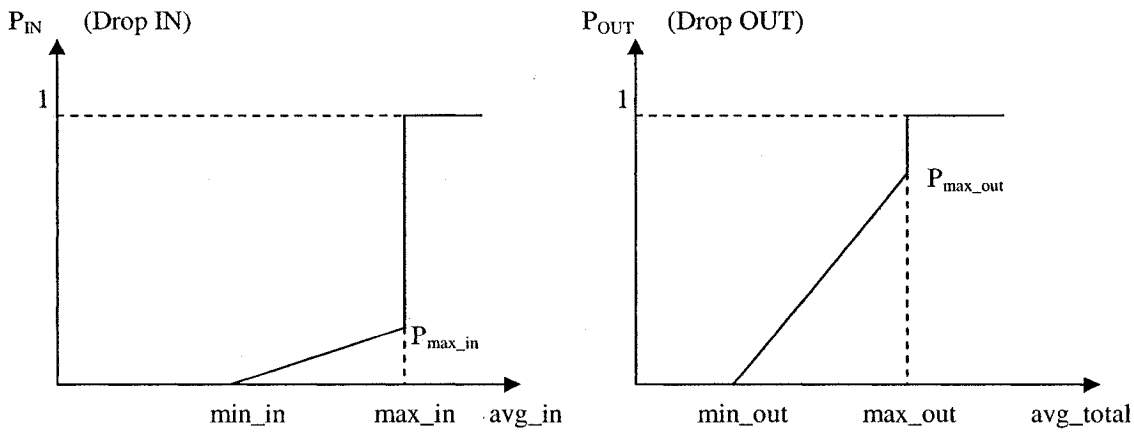


Figure 2-5 RIO Algorithm

RIO uses the same mechanisms of RED algorithm; however, it is configured with two sets of parameters, one for IN packets, the other for OUT packets. Upon each packet arrival, the router checks whether the packet is an IN packet or an OUT packet. If it is an IN packet, the average queue for the IN packets, avg_in, is computed. If it is an OUT packet, the average total queue size, avg_total, for all (both IN and OUT) arriving packets is computed. The probability of dropping an IN packet depends on avg_in. The probability of dropping an OUT packet depends on avg_total.

As shown in the Figure 2-5, there are three parameters for each of the twin algorithms: min_in, max_in, and P_{max_in} are for IN packets while min_out, max_out, and P_{max_out} are for OUT packets.

The discrimination against OUT packets in RIO is realized by carefully choosing the parameters (min_in, max_in, P_{max_in}) and (min_out, max_out, P_{max_out}). Figure 2-5 shows that RIO is more aggressive in dropping OUT packets on three ways. First, it drops OUT packets much earlier than it drops IN packets, which is done by choosing min_out smaller than min_in. Second, in the congestion avoidance phase, it drops OUT packets with a higher probability by setting P_{max_out} higher than P_{max_in}. Third, it enters the congestion control phase for the OUT packets much earlier than for IN packets; this is done by choosing max_out much smaller than max_in.

Figure 2-5 shows that the total average queue size, avg_total, is used to determine the probability of dropping OUT packets. This allows routers to maintain short queue length and high throughput no matter what kind of traffic mix is present. In fact, when avg_in is high, avg_total is high too; this means that the router will drop many OUT packets to protect IN packets. It is worth noting that if we use the average OUT packet queue size, avg_out, to control the dropping of OUT packets, the choice for the corresponding three parameters will be difficult and will have no direct intuitive correlation with the three parameters for IN packets. A pseudo-code for RIO algorithm [14] follows.

```
For each packet arrival
        If  IN packet
                Calculate the average IN queue size avg_in;
        else
                Calculate the average queue size avg_total;
If  IN packet
        If min_in < avg_in < max_in
                Calculate probability P_in;
                Drop this packet with probability P_in;
        Else if max_in < avg_in
                Drop this packet;
If  OUT packet
        If min_out < avg_total < max_out
                Calculate probability P_out;
                Drop this packet with probability P_out;
```

Figure 2-6 Pseudo-code of RIO

2.6 Weighted Fair Queuing (WFQ) Algorithm

Weighted Fair Queuing (WFQ) is a more sophisticated control mechanism, which can provide proportional fair sharing of the resources of the routers used in the diffserv domain [18]. Comparing with RED and RIO algorithms, WFQ can support bandwidth allocation with the most stringent fairness guarantees at the added cost of implementing more complex per-flow computation at the network core.

In WFQ scheduling, each flow has its own queue. When a packet of sequence number n of flow i arrives, it is tagged with virtual service start time $S_{i,n}$ and finish time $f_{i,n}$. More specifically

$$S_{i,n} = \max\{v(t), f_{i,n-1}\};$$

$$f_{i,n} = S_{i,n} + L_{i,n} / r_i$$

where

$L_{i,n}$	packet size of the arrived packet;
V(t)	system virtual time defined in WFQ;
r_i	service rate allocated to flow i.

Virtual time V(t) is defined to be zero for all flows when the server or scheduler is idle. V(t) of flow i evolves as follows:

$$V(0) = 0$$

$$V(t_{j-1} + \Delta t) = V(t_{j-1}) + \frac{\Delta t}{r_j},$$

$$\Delta t \le t_j - t_{j-1}, j = 2,3,...$$

Given this mechanism for updating the virtual time, WFQ is defined as follows: upon a packet arriving, the virtual time is updated and the packet is stamped with its finishing time. The server or router is always serving packets in an increasing order of timestamps. Packets are stored in this increasing order of finish times in each queue. The packet with the smallest finish time is chosen to be served each time.

The advantage of WFQ scheme is that it can ensure that the bandwidth allocated to a flow is commensurate with predefined metric of profile at any point in the network, even when the network is congested or underutilized. The major disadvantage of WFQ is the increased complexity of the router in the network core and the concern that such complexity may not be able to scale. On the other hand, the performance of the simpler RIO algorithm is much less predictable according to several studies [19] [20] and it is hard to support the premium service, the most valuable service of the ISPs. Since we already studied the performance of RIO scheme and its obvious limit in our previous work [20] [21], in this thesis we choose WFQ as the scheduling scheme.

Chapter 3 Models of Traffic Sources

The phenomenal growth and popularity of Internet leads to the expectation that in the future, significant amount of traffic going through networks will be related to Internet applications, such as, WWW, FTP, voice over IP and videoconferencing. Use of realistic traffic models in the performance analysis of these networks is absolutely essential for the accuracy of results and a successful design of new technologies. A number of proposals for providing differentiated services have been presented, such as the RIO (Random Early Detection with In/Out) scheme by D. Clark *et al* [14] and User-Share Differentiation scheme (actually is Weighted Fair Queuing scheme) by Z. Wang [19]. Results reported in several papers [14,19] have shown that the above schemes perform well when the bottleneck bandwidth is matching the aggregated expected bandwidth profiles. But in those works, the packet and connection arrivals are often assumed Poisson processes, and data transmission rate is often the only concern. However, in a real network, the properties of network traffic obviously differ from exponential process [22]. In this thesis, we investigate the performance of a diffserv structure using self-similar bursty traffic model. This model is capable of capturing bursty behaviour of web traffic, which has a significant negative impact on the performance of the diffserv architecture.

Moreover, we paid great attention to the traffic models of real time applications, in order to obtain accurate results for the increasingly popular multimedia data over diffserv. These models (voice, video and Virtual Environment) are based on either our measurements or the findings of the earlier works. First, voice traffic, produced by a slow speech activity detector, is modeled by a two-state Marcovian model as described in [26]. Secondly, for video traffic we have used actual H.263 video streams. Finally, we will introduce the Distributed Interactive Virtual Environment (DIVE) application model that can simulate the traffic behaviour generated by the real-time interaction among multiple users joined in a shared three-dimensional (3-D) virtual world. More information will be provided in following sections. In this chapter, we concentrate our effort to make the reader familiar with these relatively new traffic models. Chapter 4 and 5 will look into the application and performance of these applications in the diffserv network.

3.1 Web User Equivalents (WUE) Model

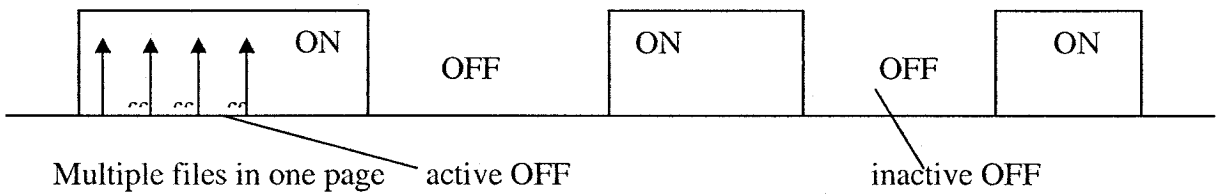


Figure 3-1 Web User Equivalents (WUE) model

The WUE model is used to generate web-access traffic in our simulations. The idea behind this model is that the workload generated by the model corresponds to that generated by a population of known number of users. Typically, when a user sends a request for a new web page, multiple objects embedding in the page are transmitted back to the user, which corresponds to an ON period. After the burst of files, a user will spend some think (OFF) time to study the downloaded web page without action. A new ON period starts when the user requests a new page. This access behavior is completely by the distribution of the requested file sizes, the embedded references in the ON periods and the OFF periods. Figurer 3-2 lists the distributions and parameters used in this model [23].

Table 3-1 Distribution and parameters of WUE model

Request Sizes	Pareto	$p(x) = ak^{a}x^{-(a+1)}$	k = 1000; a = 1.0
Active OFF Times	Weibull	$p(x) = \frac{bx^{b-1}}{a^{b}}e^{-(x/a)^{b}}$	a = 1.46; b= 0.382
Inactive OFF Times	Pareto	$p(x) = ak^{a}x^{-(a+1)}$	k = 1; a = 1.5
Embedded References	Pareto	$p(x) = ak^{a}x^{-(a+1)}$	k = 1; a = 2.43

In Figure 3-2, "request sizes" refers to the sizes of files transferred over the Internet from the web server. There are two kinds of OFF time, as shown in Figure 3-1. "Active OFF Time" refers to the time that elapses from the termination of a file and the start of a new one, when

both files belong to the same web page; it corresponds to the processing time spent by the browser parsing Web files. "Inactive OFF Time" refers to the time that elapses between the completion of transfer of a web page and the beginning of the transmission of the next. This corresponds to the user's "think time". Lastly, "Embedded References" capture the distribution of the number of objects or files contained in a web page.

The average transmission rate of each individual user is set close to 3.3 Kbps. A single source produces 0.3447 files per second on average, while the average file size is 9.2 Kbits. One important feature of this model is that when the traffic of many users is aggregated, it produces aggregate traffic that exhibits self-similar behaviour. This means that traffic bursts appear on a wide range of time scales. From figure 3-2, we can see that three important parameters have Pareto distribution, a well-known heavy-tailed distribution. Random variables with Pareto distribution exhibit very high variability; in fact, their variance is infinite. The self-similarity has serious implications on the design and analysis of a QoS capable network as will be shown in chapter 4 and 5.

3.2 Video Traffic

A video stream is generated by sampling still images (frames) of a scene at a certain sampling frequency rate. A high enough sampling frequency, typically between 20 and 30 images per second, will make the playback to appear as a continuous motion picture. Each frame consists of a grid of pixels (picture element), each pixel having a discrete value, or a set of discrete values, giving a measure for the intensity or color of the small square it represents. For grayscale images, the pixel value is typically represented using eight bits (one byte), giving possible values between 0 and 255 inclusively. The value usually represents the amount of light within the pixel; 0 is black and 255 is white, while the values in-between give the various shades of gray. For color images, pixels normally consist of three bytes, describing a color in a certain color model. For example, the three values represent the red, green and blue color components.

A video stream tends to be quite demanding when it comes to storage requirements and network resource requirements for electronic transfer. It is thus necessary to compress the data before being stored or transferred. Compression algorithms take advantage of

similarities between nearby frames, since usually there are small changes from frame to frame within a video sequence. To take advantage of the temporal redundancy, the pixel values in a block may be predicted based on blocks in nearby frames. When such prediction is used, the block is represented not by the actual pixel values, but rather by the differences from the matching pixel values in the frame used for prediction. To make prediction better, motion compensation is often used. A displacement vector may be associated with a block, describing how the block has moved relatively to the frame used for prediction. There are two video compression standards in ITU-T recommendations for video conferencing: H.261 [24] and H.263 [25].

H.261: This standard specifies a video encoding and decoding scheme for videophone, videoconferencing and other audio-visual services. This recommendation is designed for ISDN-lines or other media and is based on transfer rates that are multiples of 64 Kbps. Frames are partitioned in blocks of 8×8 pixels, each of which are transformed, quantified and Huffman-coded separately. A macro block is defined as four neighboring luminance blocks, and one block from each of the chrominance components, making up a 16×16 sub-image. Two types of frames are defined; intra-coded frames and inter-coded frames. Intra-coded frames are coded as stand-alone frames, while inter-coded frames use prediction errors with respect to the previous frame. The sender may decide not to send blocks that haven't changed since the previous frame.

H.263: This standard works much like H.261, but there are several extensions, and some modifications. Extensions to H.261 include "PB-frames mode", where two frames are coded as one unit. The latter frame is coded as an intra-frame, while the former frame is coded in inter mode, possibly using bi-directional prediction between the previously seen frame, and the intra-coded frame of the same unit.

In our simulations, we use the real H.263 video stream with 20 frames per second, captured by the network analyzer InterWatch 95000 manufactured by GN-Nettest.

3.3 Voice Traffic

Voice signals are composed of periods with voice activity (talk spurts) and silence periods (gaps). These periods correspond to the talking, pausing, and listening pattern of conversations. To better utilize network resources, a user's device should halt transmission during silence periods where there is no signal to be carried through. Slow speech activity detectors can be used to separate talk spurt and silence periods. The voice coder of the terminal samples and digitizes the voice signal during talk spurts with a sampling rate of 8,000 samples per second. The sample is digitized and a voice packet is produced. On the other hand, during the silence periods, it suppresses the signal. Two-state Markovian models can be used to model the traffic generated by a user equipped with a slow speech activity detector [26]. All spurts and gaps have exponentially distributed duration. The mean value of the talkspurt (ON) period is t_1 (equal to1 second) and the mean value of silence period is t_2 (equal to1.35 seconds). This is shown in the following figure.

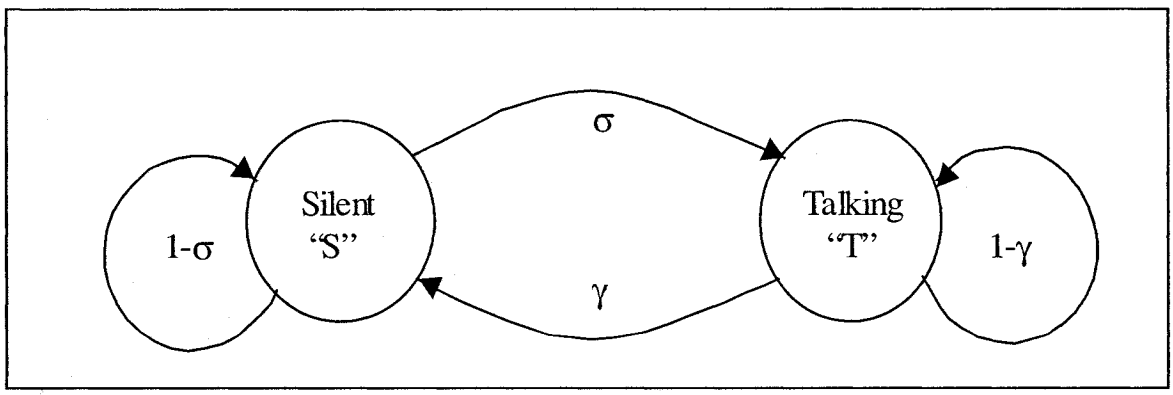


Figure 3-2 Slow speech activity detector modeling

As shown in Figure 3-3, the probability of transition from principal silent to principal talking is

$$\sigma = 1 - \exp(-\tau / t_2)$$

The probability of transition from principal talking to principal silent is

$$\gamma = 1 - \exp(-\tau / t_1)$$

Measured values for t_1 and t_2 are 1.00 sec and 1.35 sec, each with exponential distribution. This results in an average of 36% talk spurts and 64% silence gaps for each voice conversation.

3.4 Distributed Interactive Virtual Environment (DIVE) Model

DIVE is an evolving field of applications that are distributed in nature, where different parties interact within the virtual world concurrently. This is a new kind of multi-user applications including online training, teleconferencing, telemedicine, and electronic commerce in a 3-D virtual environment. Internet could potentially be deployed as the communication and message distribution mechanism for these applications. Internet-based Virtual Environments can bring a large number of participants together in a simulated 3-D space, let users explore virtual worlds and interact with each other. Standards for texture and 3-D information exchange between the viewer and application have already been published, and the software tools to browse information on different platforms are publicly available.

Extensive research has been conducted in the past, related to the characterization of the traffic encountered in modern telecommunication networks. However, to the best of our knowledge, little work has been done dealing with the traffic characterization of virtual reality applications. In [27], the traffic produced by a user to invoke changes in the virtual environment is monitored. The traffic coming to the user notifying of the changes that occurred in the virtual environment will be the aggregation of messages generated on all other users. In what follows, a more quantitative traffic analysis is performed. The outcome of this task is an empirical traffic model for the virtual reality application [27].

It can be observed that there is a bimodal behavior for the packet inter-arrival time, in the sense that the values tend to accumulate around two different values, which correspond to approximately 10 and 280 ms. The authors in [23] have proposed a bursty (ON/OFF) model for the generation of packets. In this model, the shorter packet inter-arrival times correspond to the distance between packets within the same burst, while the longer packet inter-arrival times correspond to the distance between packets corresponding to different bursts.

The burst-interarrival-time empirical pdf (probability density function) can be approximated using a Gaussian distribution with mean 0.282 and variance 1.5×10^{-4}, whereas the burst-size pmf (probability mass function, since it is a discrete random variable) can be approximated using the sum of 1.0 plus a Poisson random variable with parameter λ equal to 0.68. The empirical and approximated curves are compared in figures 4.4 and 4.5 demonstrating how closely the artificial trace resembles the measured data.

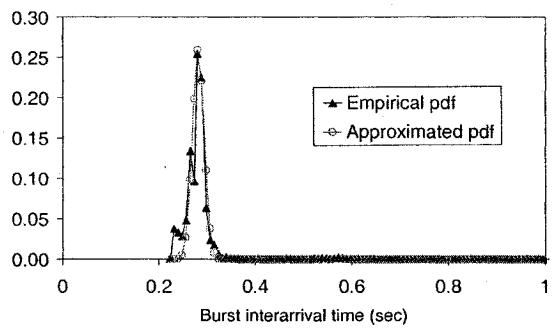


Figure 3-3 Burst-interarrival-time pdf

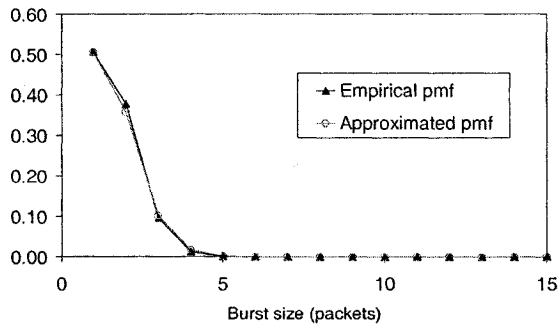


Figure 3-4 Burst size pmf

We developed sources in our simulators generating the statistics described by the empirical distributions obtained from the corresponding approximated analytic expressions.

A general conclusion that can be drawn from the model is that the source has an ON/OFF behavior. However, the distribution of the time, the source remains in the ON state does not appear to have long-tailed characteristics. The same applies for the OFF state. This observation leads us to the conclusion that aggregation of the traffic produced by such sources should have short-range dependence characteristics. This is in contrast to the models describing other types of traffic (*e.g.* www, etc.), which are governed by long-tailed distributions [22] and, when aggregated, they produce long-range dependence, as explained in section 3.1. This also suggests that this type of traffic will be friendlier to the network and the competing applications.

Chapter 4 Realistic Bursty Traffic Modeling for Differentiated Services Network

A number of proposals for providing differentiated services have been presented, such as the RIO (Random Early Detection with In/Out) scheme by D. Clark *et al* [14] and User-Share Differentiation scheme (actually is Weighted Fair Queuing scheme) by Z. Wang [19]. Results reported in several papers [14,19] have shown that the above schemes perform well when the bottleneck bandwidth is matching the aggregated expected bandwidth profiles. But in those works, the packet and connection arrivals are often assumed Poisson processes, and data transmission rate is often the only concern. However, in a real network, the properties of network traffic obviously differ from Poisson or exponential process [22]. To the best of our knowledge, no prior work dealing with evaluation of Diffserv under self-similar traffic models has been reported. In this chapter, we investigate the performance of a Diffserv structure using the self-similar bursty traffic model described in section 3.1. The model is capable of capturing the bursty and long-range dependent (self-similar) behaviour of aggregate web traffic. The results collected through our simulations indicate that this type of traffic has a significant negative impact on the performance of the Diffserv architecture. Taking into consideration that this type of traffic is the one that has been observed in modern networks, it is evident that these results should raise concern for the abilities of existing Diffserv architectures to effectively accommodate modern web based traffic.

The rest of the section is organized as follows. Section 4.1 describes our network configuration and structure. In section 4.2, we study in detail whether diffserv can effectively support QoS for bursty web traffic. Section 4.3 evaluates the performance of voice traffic with web background traffic in a diffserv-capable network. Section 4.4 evaluates the H.263 video traffic with web background traffic in diffserv-capable network. Section 4.5 presents the performance of Distributed Interactive Virtual Environments (DIVE) applications in a diffserv-capable network under web background traffic. Finally, section 4.6 concludes those results.

4.1 Network Configuration

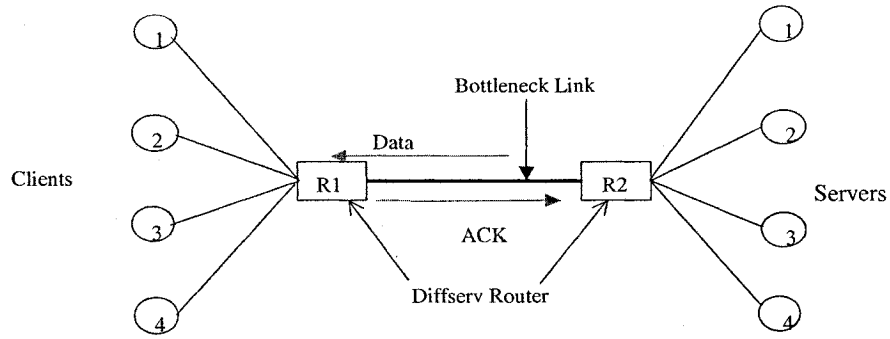


Figure 4-1 Network Topology

Figure 4-1 shows our simulation network configuration. The thick line represents the bottleneck link, with limited bandwidth while all other links have 10 Mbps bandwidth, which is much larger than the traffic on it. R1 and R2 are diffserv-capable routers and implement Weighted Fair Queuing (WFQ) to support multiple service classes. We partition the subscribers into two groups. The first is named "high class", the second "low class". Client groups 3 and 4 belong to the high-class group while client groups 1 and 2 belong to the low class. Low class group is assigned 1/6 of the bottleneck bandwidth, while the high class one is assigned double the bandwidth of the low class.

Two types of web traffic model are used to analyze the effect of burstiness of traffic on diffserv network. The first is from OPNET simulation package used. It models a session in which the workstation can establish multiple connections to servers, sends multiple request commands for HTML pages and inline objects, and processes the request responses. The distributions of object size and inter-arrivals times are Poisson. OPNET web traffic configuration of each server and client group is: each server serves 100 users, each user browses 30 pages per minute, each page has 10 objects, and object size is 100 bytes.

The second model, WUE, was derived from studies on WWW traffic [23]. There are 4 servers and 4 client groups in the network shown in Figure 4-1. Each client group consists of 100 workstations and each workstation in client group i will be served by server i. ($1 \leq i \leq 4$). Therefore, the traffic traversing each thin line is the aggregation of 100 individual flows, and

the traffic on the bottleneck is the aggregation of all 400 flows. In section 3.1 we already looked into the detail of this web traffic model.

For both models, each client group generates traffic at a average rate equal to 330 Kbps and the total generated traffic rate is 1.3 Mbps when all the network links are congestion free. TCP NewReno [30] is employed for all sessions, which can recover fast from a few packet losses. Following configurations are used throughout this chapter.

Basic Simulation Configurations:

Simulation Duration: 1000 sec Buffer size of each connection group: 100 packets

IP Packet Size: 576 bytes TCP Segment Size: 536 bytes

4.2 Simulation Results and Analysis

We choose four different bottleneck capacities: 44 Mbps, 1.5 Mbps, 1.2 Mbps and 0.9 Mbps.

In the first set of experiments, the capacity of the bottleneck is placed at 44 Mbps, much higher than traffic passing through the link. We run this set of simulations, in order to determine the performance level and observe the traces of both models when there is no congestion. Since the bottleneck link has abundance of bandwidth, all user groups perceive the transmission rate of 330 kbps and low packet delay, no matter what models are used in simulation. Table 4-1 shows the throughputs of two models.

Table 4-1 Throughput, bottleneck = 44 Mbps

User group	Network layer throughput, Poisson (kbps)	Application layer throughput, Poisson (kbps)	Network layer throughput, WUE (kbps)	Application layer throughput, WUE (kbps)	Share of bottleneck (Mbps)
Low class 1	326	314	339	326	7.3
High class 3	321	306	339	327	14.6

Table 4-2 Throughput, bottleneck = 1.5 Mbps

User group	Throughput, Poisson (kbps)		Throughput, WUE (kbps)		Share of bottleneck (kbps)
	Network layer	Application layer	Network layer	Application layer	
Low class 1	293	282	276	266	250
High class 3	332	320	334	321	500

In the next set of simulations, the bottleneck capacity is set at 1.5 Mbps, which is larger than the average aggregation (generating average rate of 1.3 Mbps). Table 4-2 summarizes the amount of bandwidth received by each class, for both models. From the reported values it is evident that the high-class user group receives more bandwidth than the low-class one. Also, the low class users receive higher average throughput when the Poisson model is used. The bandwidth received by high-class groups is almost the same in both cases and remains close to the traffic they produce (330Kbps). Figure 4-2 displays the application end-to-end delay,

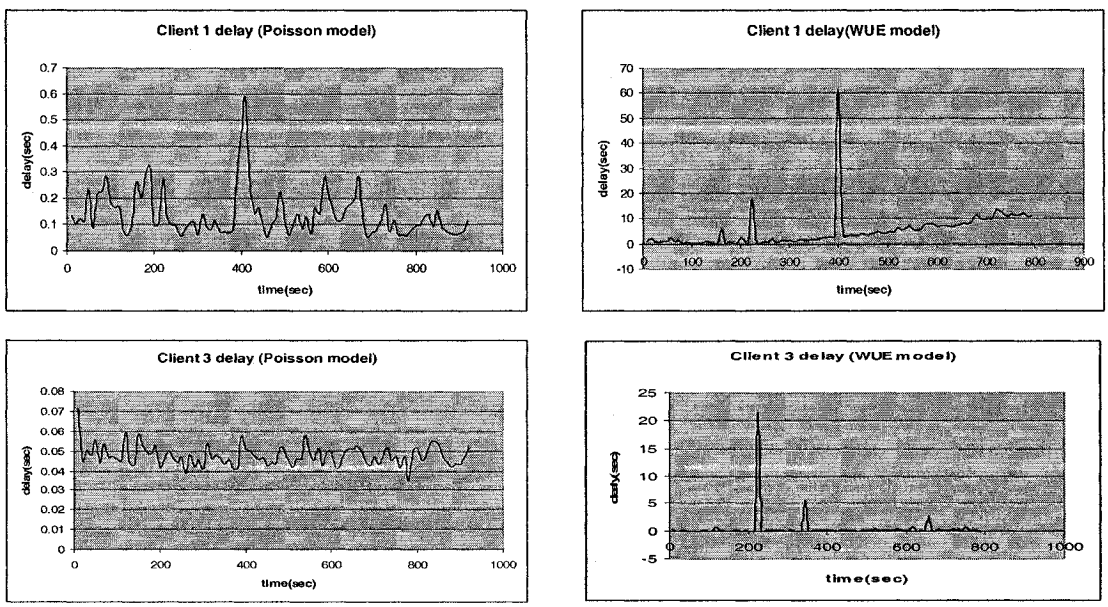


Figure 4-2 Application Layer Packet delay (bottleneck = 1.5 Mbps)

experienced by high-class and low-class user groups, under Poisson and WUE traffic models. From the figures, it is evident that under the WUE model, users experience noticeably higher delay (as high as ten times longer than the delay experienced by Poisson model). In both cases, the delay of client 3 (high-class group) is considerably lower than that of Client 1 (low class group). It is also evident that under the WUE model, users experience considerably higher jitter as compared to when the Poisson traffic model is used.

Table 4-3 Throughput, bottleneck = 1.2 Mbps

User group	Throughput, Poisson (kbps)		Throughput, WUE (kbps)		Share of bottleneck (kbps)
	Network layer	Application layer	Network layer	Application layer	
Low class 1	215	207	225	216	200
High class 3	323	311	306	294	400

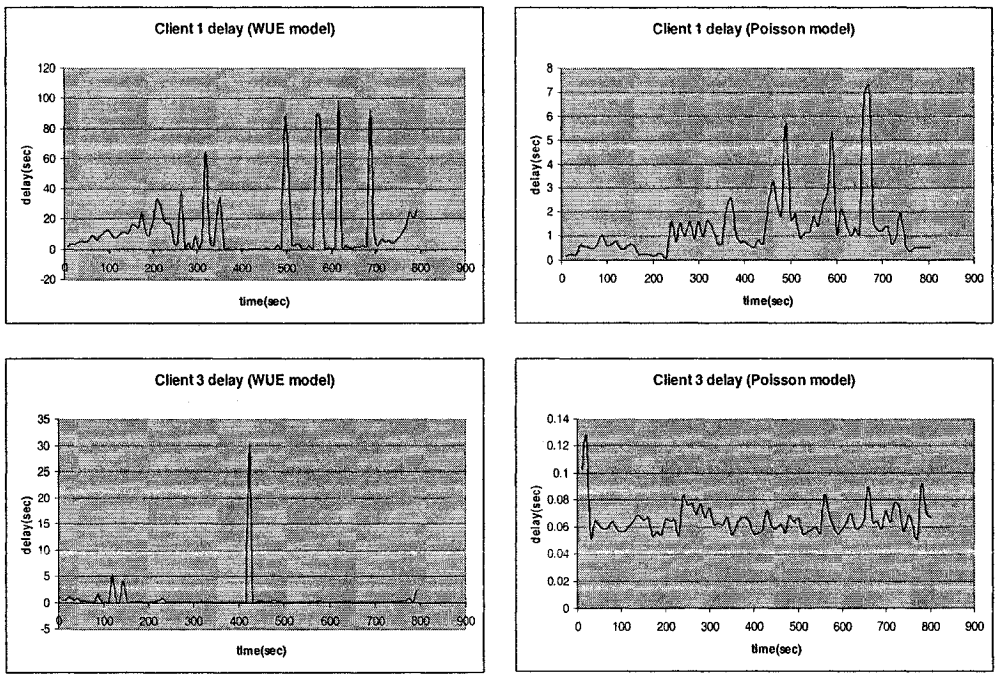


Figure 4-3 Application Layer Packet delay (bottleneck = 1.2 Mbps)

Then we set the bottleneck capacity to 1.2 Mbps, slightly less than the total generating traffic (1.3 Mbps). Based on the results summarized in Table 4-3, we can conclude that the high-class traffic is protected within its contracted profile envelope once more. From the throughput's point of view, the two models seem to have quite close performance. Based on

the delay results, displayed in Figure 4-3, we found that Client 3's end-to-end delay is considerably lower than Client 1's, and it is only slightly higher than Client 3' delay when the bottleneck capacity was set to 1.5 Mbps. However, yet again we find the delay users experiencing under WUE model is about 10 times higher than the delay under Poisson model. It is interesting that several spikes shown in the results of Client 1 delay nearly reaches 100 seconds.

Table 4-4 Throughput, bottleneck = 0.9 Mbps

User group	Throughput, Poisson (kbps)		Throughput, WUE (kbps)		Share of bottleneck (kbps)
	Network Layer	Application layer	Network Layer	Application layer	
Low class 1	145	138	125	120	150
High class 3	272	262	258	246	300

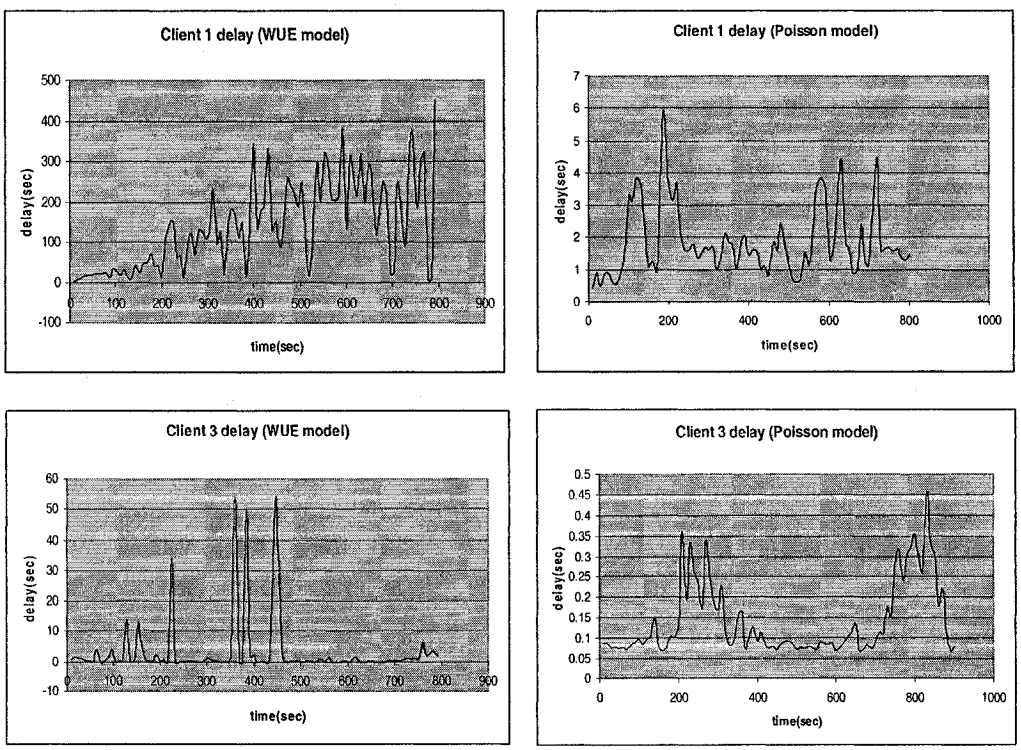


Figure 4-4 Application Layer Packet delay (bottleneck = 0.9 Mbps)

Finally, we set the bottleneck link's rate to 0.9 Mbps, which indicates fairly heavy congestion along the data path. The results summarizing the bandwidth received by each group under the two models are displayed in table 4-4. Under both models, users meet the requirement of bandwidth reservation, while the throughput of the Poisson model is a little higher. Similarly with previous results, under the WUE model, users from the low priority group experience end-to-end delay that is sometimes dozens of times higher from the delay experienced under the Poisson model from table 4-5. On the other hand, the delay experienced by users of the low priority group under Poisson traffic is surprisingly short, given its contracted service profile is merely 150 Kbps. This is evidently "too good ideal to be true". Also, the results for the Client 3 group give another proof that the traffic produced by WUE model raises the delay a lot: there are some huge spikes (50 seconds) in Client 3's delay, even though Client 3's contracted share is only slightly less than its average data generating rate.

Table 4-5 Average End-to-end delay and delay jitter for all scenarios

User group	Delay (sec), bottleneck = 0.9 Mbps	Delay (sec), Bottleneck = 1.2 Mbps	Delay (sec), Bottleneck = 1.5 Mbps
Low class, Poisson	1.55	1.14	0.139
Low class, WUE	138.9	12.69	5.047
High class, Poisson	0.137	0.065	0.052
High class, WUE	3.31	0.55	0.385

User group	Jitter(sec), Bottleneck = 0.9 Mbps	Jitter (sec), Bottleneck = 1.2 Mbps	Jitter (sec), Bottleneck = 1.5 Mbps
Low class, Poisson	0.986	1.15	0.087
Low class, WUE	110.7	21.36	6.98
High class, Poisson	0.088	0.0089	0.0049
High class, WUE	10.11	3.09	2.214

We collect all the delay results and calculate the average end-to-end delay and the jitter (standard deviation of delay) for all cases and report them in table 4-5. It is obvious that

under the WUE traffic model, the delay and jitter experienced by users in all scenarios are significantly higher than those under the traditional Poisson model.

To find out the reason of such considerable difference, we compare the total packets dropped in congested routers R1 and R2 when bottleneck bandwidth is 0.9Mbps. The results are shown in figure 4-6. Here we can see that under the same network configuration, the number of dropped packets under WUE traffic is significantly higher than under Poisson traffic. This means that TCP will face higher packet loss in the WUE case and its performance will be adversely affected, especially its delay and jitter. The Poisson model undoubtedly underestimates the occurrence of network congestion, hence gives over-optimistic results for the performance of the network.

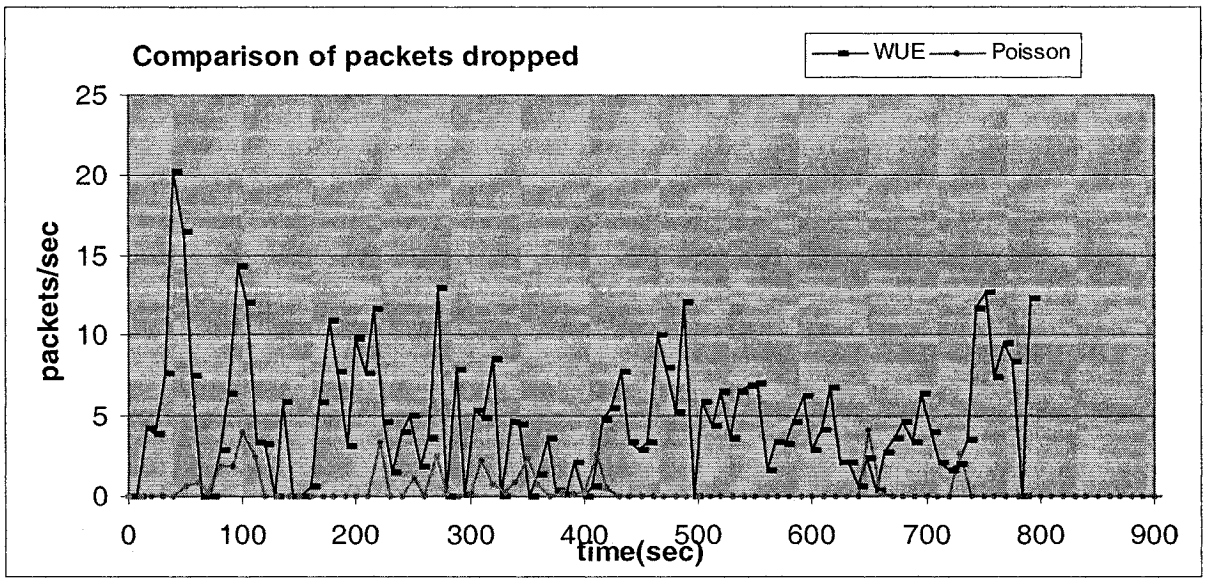


Figure 4-5 Comparison of Packet Dropped

4.3 Voice Traffic in Diffserv Networks

From this section, we present the performance analysis we conducted to evaluate the ability of diffserv-capable networks in terms of supporting different types of real time multimedia traffic. Each section presents the simulation environment (e.g., values of simulation parameters), simulation results and analysis for a given application.

The network architecture used in these simulations is shown in Figure 4-1. The thick line represents the bottleneck link, with limited bandwidth, while all other links have 10 Mbps link capacity, which is much larger than the traffic on it. There are 4 servers and 4 clients groups in the network shown in Figure 4-1. Each clients group consists of a number of users. Each user in client group i will be served by server i ($1 \leq i \leq 4$). Therefore, the traffic traversing each thin line is the aggregation of fixed number users traffic, and the traffic on the bottleneck is aggregation of all users' flows. R1 and R2 are diffserv-capable routers that implement Weighted Fair Queuing (WFQ) to support multiple service classes. In this chapter, users in client group 1 are multimedia application users, while all other users in client groups 2, 3 and 4 are using WWW application. The Web traffic model here are Web User Equivalents (WUE) model described in section 3.1, which is more representative of the actual traffic behavior as compared to Poisson. It is meaningful to analyze the results of multimedia under the web based background traffic due to the dominant role of web traffic in today's Internet. TCP NewReno [30] is employed for all sessions of WWW users. All multimedia (voice, video and DIVE) applications use UDP as transport protocol.
Since multimedia traffic is critical and delay-sensitive, and we expect this application will have a quite high price tag comparing to routine web traffic, multimedia traffic is a good candidate for Expedite Service (EF) or Premium Service. To evaluate the behavior of multimedia traffic when serviced through EF, we applied Priority Queuing as means of supporting EF service in our network. There are two queues in routers in Figure 4-1, one high priority queue and one low priority queue. Packets in the high priority queue are forwarded always before the packets in the low priority queue. The packets in the low priority queue are forwarded only when there are no packets left in high priority queue (i.e., the high priority queue is empty). When a multimedia packet arrives, it is placed into the high priority queue. Packets from web service are placed into low priority queue.

4.3.1 Voice Simulation Configurations

Table 4-6 Parameters and Configurations of Voice Traffic

User group	Average source generating data rate (Kbps)	Number of users	Queue Priority
Voice	687	20	High
Web	896	300	Low

We have 20 telephony connections between server 1 and client 1, generating traffic at a rate equal to about 687 Kbps at the application layer and use UDP as transport. The voice traffic model described in section 3.3 is utilized in this test. The voice source-sampling rate is 8,000 samples per second. There are 300 web browsing users, generating about 900 Kbps traffic on application layer if network layer has unlimited bandwidth, sent from server 2, 3, 4 to clients groups 2,3,4 respectively.

Table 4-7 Transmission Data Rate and Profile of Voice Application

Bottleneck (bps)	4M	2M	1.8M	1.5M
Voice application layer throughput (bps)	687K	687K	687K	686K
Voice netrwork layer throughput (bps)	713K	713K	713K	713K
Voice delay (second)	0.0052	0.0056	0.0059	0.0063
Voice Jitter (second)	0.0012	0.0014	0.0017	0.0017
Voice Packet Loss Rate	0	0	0	1.2×10^{-4}

4.3.2 Simulation Results and Analysis

We choose four different bottleneck capacities: 4 Mbps, 2 Mbps, 1.8 Mbps and 1.5 Mbps. Table 4-7 shows the results of the delay, jitter, throughput and packet loss the voice user experiences; most data are collected at the application layer. The throughput is quite stable and adequate for the voice application. The throughput achieved by voice users is constant 687 under all bottleneck link capacities. When we look into the delays and jitter, again the performance of the voice application is pretty good under all bottlenecks. The jitter of the voice user remains below 0.2 ms. Decrease of bandwidth at the bottleneck link pushes the delay upwards a little (from 5.2 ms to 6.3ms). The packet loss performance of voice is very good even when the bottleneck is 1.5 Mbps, which means average traffic loading is near full. The number of total voice IP packets sent is about 2.13×10^5.

Table 4-8 Web Traffic Data Rate and Delay

Bottleneck (bps)	4M	2M	1.8M	1.5M
Web traffic application layer throughput (kbps)	892	856	778	642
Web packet application layer delay (sec)	0.0081	0.176	2.21	16.3

Table 4-8 displays the corresponding metrics of the web users. It is evident that the good quality of the voice comes at the cost of background web traffic, especially when the bottleneck is congested. With the decrease of available bandwidth, the delay of web traffic packets increases dramatically with lower web traffic throughput. Most importantly (since web browsing is a throughput sensitive application), the throughput is reduced as well amid the 1.5 Mbps bottleneck, the delay of web traffic reaches 16.3 seconds while the web traffic goodput is reduced to 642 Kbps.

In summary, the loss and delay sensitive voice application can be well supported by diffserv-capable networks even under the bursty web style traffic, given that the voice application is assigned to EF class. This is good news for ISP and network gear vendors,

since telephony is still an important cash flow of telecommunication provider. The above results imply that in the future voice service can be shifted from legacy circuit-switched system to diffserv-capable data network.

4.4 H.263 Video Stream in Diffserv network

Table 4-9 Parameters and Configurations of Video Traffic

User group	Average source generating data rate (Kbps)	Number of users	Queue Priority
Video	874	10	high
Web	896	300	low

The configuration of this set of simulations is shown in Figure 4-1. This time, the clients group 1 consists of ten H.263 video application users via ten UDP sessions to video server 1. The video source (server) generates traffic at a combined rate of 874kbps, or 87.4 kbps per video client, and uses 20 frames per second for each session. We captured the real H.263 video stream and use this stream as source in our traffic model. In the clients groups 2,3 and 4, there are 300 web users, which produce altogether 900kbps traffic.

Table 4-10 Video Application Performance

Bottleneck (bps)	4M	2M	1.8M	1.5M
Video application layer throughput (bps)	874K	874K	874K	874K
Video network layer throughput (bps)	909K	909K	909K	909K
Video Delay (sec)	0.0063	0.0064	0.0068	0.0071
Video jitter (sec)	0.00032	0.00031	0.0014	0.0026
Video packet loss rate	0	0	0.0	2.2×10^{-4}

Table 4-11 Web Traffic Delay and Data Rate

Bottleneck (bps)	4M	2M	1.8M	1.5M
Web traffic application layer throughput (kbps)	896	846	721	534
Web packet delay (sec)	0.0076	1.64	3.78	14.5

Tables 4-10 and 4-11 summarize the performance results for both video and web applications. The first conclusion is that video application is working well in the diffserv-capable network, as long as the video traffic is under the protection of its EF service. With the bandwidth of all the bottleneck links, the delay and jitter are insignificant, and the video data application throughput is steady as table 4-10 shows. The total number of video traffic IP packets sent is 2.66×10^5. In table 4-11, the different trend is observed for web traffic: as the bottleneck link becomes lower, the web traffic users experience considerably longer delay as well as lower throughput.

Briefly speaking, a diffserv-capable architecture can support real time video application effectively if the video applications are assigned to EF class; even video streams are mangled with high bursty web style traffic.

4.5 Distributed Interactive Virtual Environments (DIVE) Application in Diffserv Network

This is a new kind of multi-user applications including online training, teleconferencing, telemedicine, graphic gaming and electronic commerce in a 3-D virtual environment. With the fast deployment and evolution of broadband Internet, this value-added service will become critical and lucrative service for all ISPs. In this section, we will investigate the deployment of these new applications over Internet.

In figure 4-1, Clients group 1 is composed of 30 Distributed Virtual Environments (DIVE) users. The corresponding DIVE server 1 will generate 87 kbps at the application layer with the DIVE traffic model illustrated in section 3.4. Each DIVE user will generate about 4.2 kbps traffic at the application layer and use UDP as transport protocol. DIVE application is studied on a real test bed and it is found that DIVE application is very sensitive to delay and jitter [27]. Since DIVE traffic is pretty thin and delay-sensitive, and we expect this application will have a quite high price tag comparing to routine web traffic, DIVE is a good candidate for Expedite Service (EF) or Premium Service. To evaluate the behavior of DIVE when serviced through EF, we applied Priority Queuing as means of supporting EF service in our network. When a DIVE packet arrives, it is placed into the high priority queue. Packets from web service are placed into low priority queue. Clients groups 2, 3 and 4 are web users, the overall number of web users being 100 users, which will produce about 310 kbps traffic at the application layer. We configure DIVE and web traffic parameters as following table 4-12.

Table 4-12 DIVE traffic WFQ parameters

User group	Average source generating data rate (Kbps)	Number of users	Queue Priority
DIVE	87	25	High
Web	310	75	Low

Four different bottleneck capacities are chosen to assess the DIVE traffic: 800kbps, 600kbps, 500kbps and 400kbps.

Table 4-13 DIVE Application Performances

Bottleneck (kbps)	800	600	500	400
DIVE application layer throughput (kbps)	86.6	86.6	86.6	86.6
DIVE network layer throughput (kbps)	90.1	90.1	90.1	90.1
DIVE Delay (sec)	0.011	0.014	0.016	0.022
DIVE jitter (sec)	0.0014	0.0023	0.0026	0.0026
DIVE packet loss rate	0	0	0	0

Table 4-14 Web Traffic Delay and Data Rate

Bottleneck (kbps)	800	600	500	400
Web traffic application layer throughput (kbps)	310	301	279	262
Web packet delay (sec)	0.022	0.077	0.48	5.6

Table 4-13 shows that the performance of DIVE traffic, successfully received by the users. This gives us clear evidence that under Priority Queuing, the DIVE application using EF service is achieving very satisfied results. For all four-bottleneck capacities, the application layer throughput of DIVE traffic keeps unaffected, and the delay and jitter of DIVE are always negligible even as the bandwidth is 400kbps, close to the sum of both DIVE traffic and web traffic generating rate. And the packet losses of DIVE applications are zero throughout the simulations. Total number of DIVE application packets sent is 3.02×10^4.

The performance of web application is in contrast to that of DIVE application judging from table 4-14. The throughput, delay and jitter of web application are deteriorated as we reduce the bandwidth of bottleneck.

Based on these observations, we can state that the DIVE traffic has been protected in diffserv-capable network from the web traffic. It is reasonable to assign this thin but critical application to EF service, and the performance of DIVE EF traffic is excellent from our simulations.

4.6 Conclusion

In this chapter, we studied the performance of the diffserv architecture in terms providing several service classes to typical WWW users. Both traditional Poisson and self-similar WUE traffic models are considered. From the results it is evident that due to the burstiness of realistic web traffic, packet scheduling and dropping algorithms like WFQ can provide QoS commitment only to a limited extent, even when the sharing of link bandwidth is roughly equal to the user's average data generating rate. From the standpoint of throughput, it looks like WFQ can effectively allocate the link bandwidth to a user according to his contracted profile, no matter what traffic model is used. A high-class client is able to perceive the desired data transmission rate within his profile envelope in both models. However, the comparable average throughputs from both models do not imply that the transmission delay and jitter are similar. In fact, the Poisson traffic model appears to be too optimistic in its assessment of packet delay and jitter. The delay under Poisson traffic is merely a tenth or less of that under WUE traffic, in the presence of intense congestions. The self-similarity and high burstiness of the WUE model generates heavy congestion, which results to higher packet loss. Moreover, the long-range dependence of actual traffic suggest that the impact of this persistent burstiness can not be alleviated by the aggregation of large number of connections, thus, the core routers in the ISP center will face the similar behaviors as the edge router of the enterprise intranet.

In summary: we should be extremely careful when we use Poisson traffic model to investigate the performance of diffserv-based networks. The throughput should not be the

only performance measure to evaluate the network. The performance of applications should be considered and thoroughly evaluated as well. Self-similar traffic should be used instead of Poisson traffic in simulations considering QoS commitment and resources scheduling.

In the other hand, based on the findings of last three sections in this chapter, multimedia applications such as voice, video and DIVE applications can be well supported by diffserv-capable networks even in the presence of self-similar bursty web traffic if one application is protected by EF class. However when heterogeneous real time traffic is all put into the same EF class, the loss sensitive application like DIVE may experience some loss as [27] described. The EF class is a good candidate for these delay sensitive multimedia traffic like voice and video to protect the main cash flow of service providers when the real time traffic is homogeneous, but may not true when it is heterogeneous.

Chapter 5 Resource Reservation Analysis for Differentiated Services Network

IETF has proposed some service models and mechanisms to meet the demand of QoS, such as Integrated Services/RSVP and Differentiated Services. In those architectures, QoS and resource reservation is shown to work in one direction, while availability of bandwidth at the reverse path is taken for granted. This condition may not be the case in a real network. In this chapter, we will investigate the effect of both, unidirectional and bi-directional reservation. Our performance analysis demonstrates that in more a realistic network environment, only bi-directional resource reservation can provide to the customers QoS guarantees. Several scenarios of aggregations of applications traffic flows are used to evaluate the performance, such as self-similar bursty web traffic we proposed in chapter 3 and non-adaptive UDP traffic. Eventually, a bi-directional resources allocation scheme is described and evaluated.

Results reported in several papers [14,19] have shown that differentiated services perform well when the bottleneck bandwidth is matching the aggregated expected bandwidth profiles. But in those architectures, QoS and resources reservation are conducted only in one direction, while an abundance of bandwidth at the reverse path is assumed. That means the returning ACKs are supposed to never be lost along the reverse path. However, in a real network, this condition might not be satisfied. In [31] and [32], the effect of asymmetry and lossy link (wireless) on ACKs and TCP performance are well studied in best-effort TCP/IP network. In this chapter, we will investigate the impact of ACK loss on DiffServ-capable network and the performance of a diffserv structure under bi-directional heavy traffic conditions with more realistic self-similar web traffic model. We can see that use of unidirectional reservation does not work; only bi-directional reservation can really differentiate the users' services according to their subscribed profiles.

The rest of the chapter is organized as follows: Section 5.1 introduces the resource reservation and some terminology. A complete example for delivering end-to-end unidirectional service and related reservation is presented here as well. Section 5.2 describes

our network configuration and reports the performance evaluation results for unidirectional reservation, under different traffic types and link conditions. In section 5.3 we propose a possible bi-directional resource reservation scheme and compares it with the unidirectional scheme under two-way heavy traffic. Section 5.4 studies above reservation with multiple network congestions. Finally section 5.5 concludes our work and discusses some relevant issues.

5.1 Current Resource Reservation Schemes for Diffserv

5.1.1 Useful Terminology

The following terms that are used throughout the paper are defined here for clarity. Some of the terms defined in the Diffserv (DS) architecture [2] are repeated here for completeness.

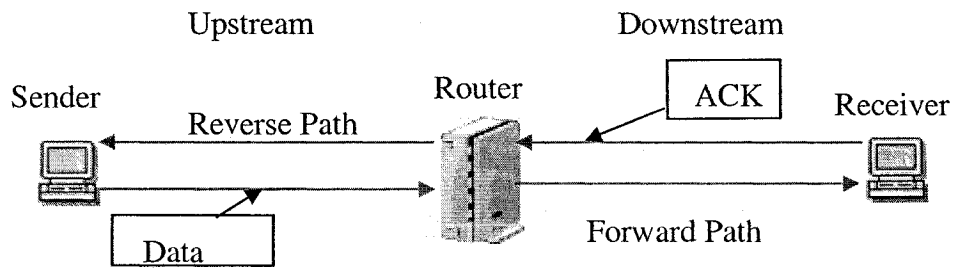


Figure 5-1 Direction of Data and ACK streams

DS boundary node or Edge Router: A DS node that connects one DS domain to a node in another DS domain or in a domain that is not DS-capable.

DS domain: A DS-capable domain; it consists of a contiguous set of nodes, which operate with a common set of service provisioning policies and PHB definitions.

Bandwidth Broker (BB): A bandwidth broker (BB) manages the network resources for a single domain, in order to support the QoS requirements of the customers

Forward Path: Along this path, data originated from a sender are transported to its destination, the appropriate receiving station.

Reverse Path: Used to transport the ACK messages generated by the receiving station back to sender.

Service Level Agreement (SLA): A service contract between a customer and a provider that specifies the forwarding service a customer should receive. A customer may be a user organization (source domain) or another DS domain (upstream domain).

5.1.2 A Unidirectional End-to-end Reservation Example

The work reported in [28] proposed a two-tier resource management model for diffserv networks. The AS or domain can still be the basic resource agent to control bandwidth allocation. Moreover, we assume that a Bandwidth Broker (BB), presented by Van Jacobson [3], takes this responsibility for each domain. Adjacent domains with related aggregation of border-crossing traffic reach bilateral SLA. Meanwhile, each domain may choose its own resource reservation protocol for its internal QoS need. [3] and [28] also give us some details describing the bandwidth allocation architecture and its feasibility. Their conclusion is that end-to-end QoS support can be implemented through the concatenation of inter- and intra-domain resource allocations, which is similar to current IP two-level routing architecture.

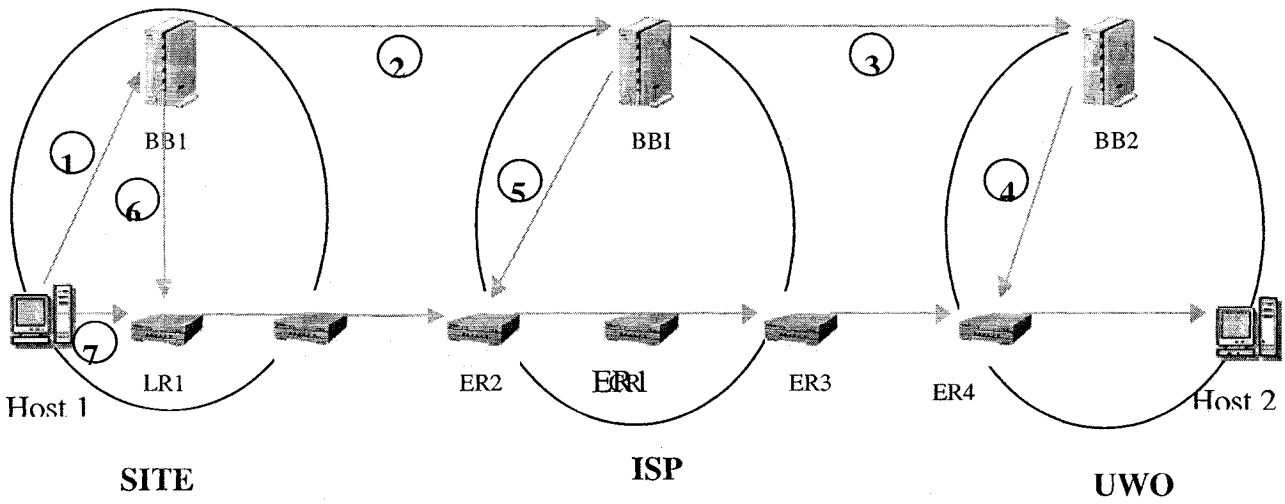


Figure 5-2 End-to-end resource allocation, with unidirectional reservation

Here we give an end-to-end allocation example to illustrate the ideas mentioned above. In figure 5-2, there are dynamic SLAs between domain SITE, its ISP domain and domain UWO, and the BBs existing in each domain are using RSVP as their signaling protocol. Host 1 (in domain SITE) needs to send 100 kbps data to host 2 (in domain UWO) using Assured Service class. The reservation procedure is described below.

1. Host 1 sends a RSVP PATH request to its Bandwidth Broker BB1.

2. BB1 makes an admission control decision to check if it can accept this request. If the request is denied, the reservation process has failed. If accepted, BB1 sends a PATH message to its neighbor BBI.

3. BBI then makes its admission control decision and in turn sends its PATH to BB2, if the request is authenticated.

4. BB2 checks this request and gives it permission. Then, BB2 will use RSVP or another local signaling protocol to configure the edge router ER4 to allocate resource for this request. Meanwhile, BB2 returns an RSVP RESV message to BBI.

5. After receiving RESV, BBI will configure its edge router ER2 to support the reservation. As next action, it sends RESV back to BB1.

6. After receiving RESV, BB1 will configure the leaf router LR1 to correctly classify and shape the admitted data from H1. BB1 then sends RESV to host 1.

7. All BBs will also set the policing and reshaping rules on the egress routers like ER1 and ER3. Now reservation is completed and host 1 starts to send data.

From the above example, we can observe that this reservation scheme is only working in one direction, the data-forwarding path. It assumes implicitly that the reverse path has enough bandwidth available, so that ACKs can traverse it without loss and congestion. However, with the deployment of a QoS-capable network like Diffserv and MPLS, even if ACKs are smaller than data packets, if marked as low priority or best effort class, they may experience congestion in the reverse path and be discarded or be delayed considerably, as the network takes action to protect and accommodate higher priority class traffic. This will affect the performance of data forwarding path in a negative manner, despite the fact that reservation has already been made along the forward path. In the next section will see the effect of this lossy reverse link under different network conditions.

5.2 Simulation Settings and Results

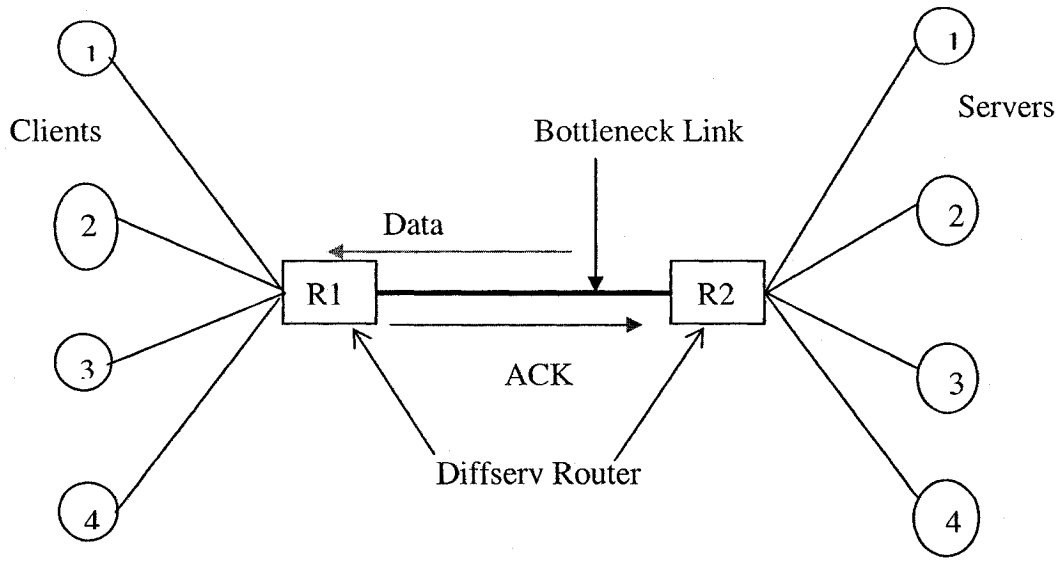


Figure 5-3 Network Topology with one bottleneck

Figure 5-3 shows our simulation network configuration. The thick line represents the bottleneck link, with 3 Mbps bandwidth, while other links have 10 Mbps bandwidth, which is much larger than the traffic traversing through the link. There are 4 servers and 4 client groups in the other side in Figure 5.3. Each client group is made up of a number of workstations and each workstation in client group ***i*** is served by server ***i.*** Therefore the traffic traversing each thin line is the aggregation of many individual flows The traffic on the bottleneck link is the aggregation of all flows. R1 and R2 are diffserv-capable routers implementing Weighted Fair Queuing (WFQ) to support multiple service classes. We partition the subscribers into two groups. The first is named "high class", the second "low class". Client groups 3 and 4 belong to the high class group while client groups 1 and 2 belong to low class. Low class is assigned 1/6 of the bottleneck bandwidth, while the high class is assigned 1/3 of the bottleneck bandwidth. Since Web traffic constitutes the largest portion of current Internet, we select the accurate web traffic (WUE) model described in chapter 3 since this model can realistically capture the self-similar characteristics of web traffic. TCP NewReno [29] is employed for the adaptive sources, which can fast recover from packet losses. Following configurations are used throughout this chapter.

Basic Simulation Configurations

Simulation Duration: 1000 sec Buffer size of each client group: 100 packets

IP Packet Size: 576 bytes TCP Segment Size: 536 bytes

Number of users in each client group: 300 (generating data rate at 900 kbps)

5.2.1 One Way Reservation without ACK Disturbing on Reverse Path

First, we consider the case where traffic is sent from server i (i = 1,2,3,4) to client group i, and ACKs are traveling at the opposite direction. In the scenario examined in the previous section 4.2, we provide considerably high bandwidth to the reverse path, so that ACKs will not be delayed or lost due to queuing and congestion. From section 4.2, when WFQ is used, for three different bandwidth allocations made to the bottleneck path (1.5 Mbps, 1.2 Mbps, 0.9 Mbps), the high class clients group 3 always get higher throughput than the low class clients group 1. Under these conditions, our results from section 4.2 confirm the findings by Z. Wang in [19]: The throughputs of connections in diffserv network are matching their expected bandwidth allocation.

Table 5-1 Configuration and Results of non-adaptive traffic

Client group	1	2	3	4
Traffic type	UDP	WWW	UDP	WWW
Allocated Bandwidth (WFQ)	0.5 Mbps	0.5 Mbps	1 Mbps	1 Mbps
Goodput (WFQ)	443 Kbps	396 Kbps	904 Kbps	884 Kbps
Network Layer Throughput (WFQ)	459 Kbps	407 Kbps	940 Kbps	915 Kbps
Goodput (Best Effort)	904 Kbps	386 Kbps	907 Kbps	374 Kbps
Network Layer Throughput (Best Effort)	936 Kbps	401 Kbps	941 Kbps	388 Kbps

Next, we consider a scenario where a mixture of TCP and non-adaptive UDP traffic compete for the bottleneck capacity. Non-adaptive traffic is increasingly becoming a problem in Internet, undermining the fairness of the network [17]. In our simulations, the UDP client group is flooding the network and is not responsive to congestion and ACK loss. Table 5-1 shows the assignment of profiles to TCP and UDP traffic. Each UDP or web client group is generating packets at the average rate around 900Kbps if the network layer has unlimited

bandwidth. This time, the bottleneck bandwidth is 3 Mbps. We notice from the results displayed in Table 5-1 that UDP traffic from clients 1 and 3 takes most of the link capacity when competing with the TCP traffic in a best effort network. This is due to the fact that malicious UDP sources are sending at free, without backing off in the face of congestion. When WFQ is employed, the bandwidth allocations are much closer to the expected bandwidth profiles. Low profile clients in groups 1 and 2 achieve about 400 Kbps goodput, while their profile was 500 Kbps. High profile clients in groups 3 and 4 achieve about 900 Kbps throughputs, while their profile is 1 Mbps. In each group, UDP sources still get slightly higher throughput than their TCP counterparts but the difference is acceptable. This implies that diffserv is effective when dealing with non-responsive traffic.

Table 5-2 Configuration and Results of Longer RTT source

Client group	1	2	3	4
RTT (second)	0.5	1	0.5	1
Allocated Bandwidth (WFQ)	0.5 Mbps	0.5 Mbps	1 Mbps	1 Mbps
Goodput (WFQ)	526 Kbps	353 Kbps	907 Kbps	673 Kbps
Network Layer Throughput (WFQ)	547 Kbps	367 Kbps	939 Kbps	699 Kbps
Goodput (Best Effort)	879 Kbps	422 Kbps	847 Kbps	430 Kbps
Network Layer Throughput (Best Effort)	912 Kbps	435 Kbps	880 Kbps	447 Kbps

It is well known that long round-trip time (RTT) can affect TCP connections in a negative manner. Large RTT connections are not be able to get as much bandwidth as shorter RTT connections, and TCP goodput is inversely proportional to the RTT [14]. To see its effect on a diffserv capable environment, we set up four web client groups and add extra delay to effectively double the RTT of the links from clients in groups 2 and 4 (to servers 2 and 4) respectively. From table 5-1 we see that in a best effort network, shorter RTT connections have considerable advantage as compared to longer RTT links. In our case, clients from

groups 1 and 3 have throughput that is almost twice the throughput of clients from groups 2 and 4.

In the same table, we provide the throughput when diffserv is used. We assign clients from groups 1 and 2 to low class and clients from groups 3 and 4 to high class. Obviously, the throughputs of the different RTT connections are roughly matching their bandwidth allocation under diffserv, even though clients with shorter RTT still receive more bandwidth at the expense of clients with longer RTT. While the profile of clients from the low class group is 500 Kbps, the throughput of client 1 with short RTT is 547 Kbps, while client 2 with long RTT obtains 367 Kbps. Regarding the high profile groups, the throughput of clients from group 3 with short RTT is 939 Kbps while the throughput of long RTT clients from group 4 is 700 Kbps. We conclude that the long RTT connections are still quite assured of their expected bandwidth allocation in diffserv network.

In summary, our findings reported in this section, lead to the following conclusions. Assuming that the ACK path is not experiencing losses and delays, diffserv network architectures are able to render rate guarantee to subscribers under different network contexts. In the following section, we will investigate the validity of this conclusion when a realistic approach regarding the effect of network on ACK packets is adopted.

5.2.2 Impact of ACK Loss on Reverse Path

TCP uses the ACK clock to estimate the conditions within the path. Congestion in the reverse path, carrying the ACK messages, increases the RTT of the connection and causes loss of ACKs. Longer RTT reduces throughput and increases end-to-end delay. Furthermore, multiple loss of ACK slows the growth of congestion window fast, which results in poor performance for TCP connections. To see the impact of ACK loss, we intentionally introduce packet losses on the ACK path for the network configurations presented above (See figure 5-3). Table 5-3 is the bandwidth allocation of four clients groups, and all of them use web TCP traffic model described in chapter 3 generating data around 900 Kbps if network layer has unlimited bandwidth. ACK packets over the reverse path are randomly discarded according to the probability specified by loss rate.

Table 5-3 Bandwidth Allocation of Different Groups

Clients group	1	2	3	4
Allocated Bandwidth	0.5 Mbps	0.5 Mbps	1 Mbps	1 Mbps

As Figure 5-4 indicates, web sources using the "NewReno" version of TCP [29] are fairly sensitive to ACK loss. Since network layer throughput observes the same trend as application layer goodput and goodput is a closer metric of performance to end user, we display the goodput here as our results. When ACK loss rate is above 2%, all TCP connections suffer considerable degradation, to the point that we cannot differentiate the performance of high class clients in groups 3 and 4 from the performance of low profile clients in groups 1 and 2. The reason is that WWW sources will set up multiple short duration TCP connections for each Web page and TCP Reno can only recover one ACK packet loss. As result, many lost ACK cannot be recovered. Moreover, when multiple ACK losses occur, TCP is forced to enter slow start status, with only one segment congestion window size. Although we have made reservation on the data-forwarding path, the performance of all clients is heavily impaired, regardless of the class they belong, due to the ACK losses. Their goodput is considerably below their allocated bandwidth, and it is almost the same for all of them (thus practically eliminating differentiation of resource reservation). Figure 5-5 displays the utilization at the bottleneck link as a function of ACK loss rate, which is a good indicator of the network performance. Note that the utilization is less than 40% when ACK loss rate becomes 5%. It is evident that a considerable amount of resource in the forward link is wasted.

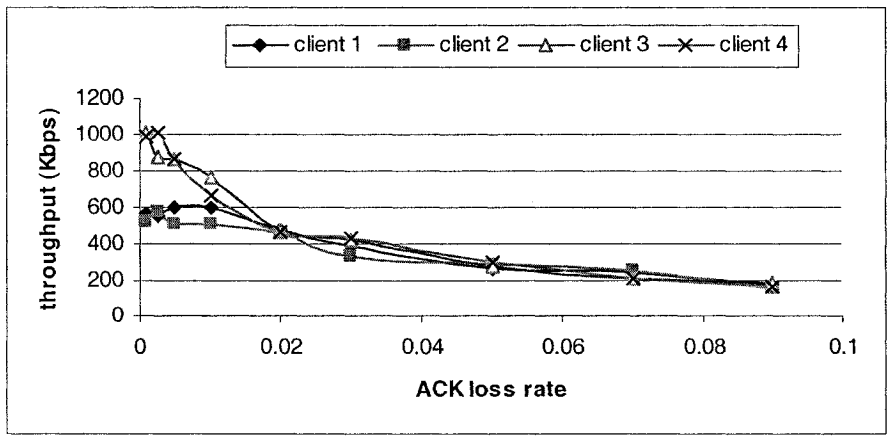


Figure 5-4 Goodput vs ACK loss

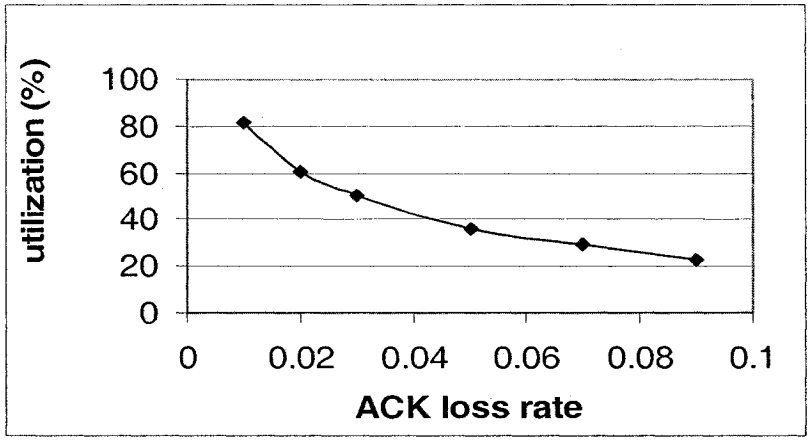


Figure 5-5 Utilization vs ACK loss

Next we take the non-adaptive sources into account. Instead of having all clients making use of TCP, clients in groups 1 and 3 are producing traffic running over the UDP protocol. We assign again clients from groups 1 and 2 to low class and clients from groups 3 and 4 to high class as table 5-3 shows. From the results in Figure 5-6 we see that UDP clients from groups 1 and 3 are immune to the increase in ACK loss rate, but TCP clients from groups 2 and 4 suffer considerable deterioration as the ACK loss rate increases. Also, clients from groups 2 and 4 receive almost the same throughput with ACK loss rates of 2% and above. Obviously, the TCP links' goodput is much lower than its expected profile. , Since UDP does not react to ACK loss, while TCP is backing off when experiencing loss of ACK, this behavior is expected. It is clear that one-way reservation does not make sense when there is the possibility (and it definitely exists in most cases) of facing heavy ACK loss if there is a large amount of UDP traffic.

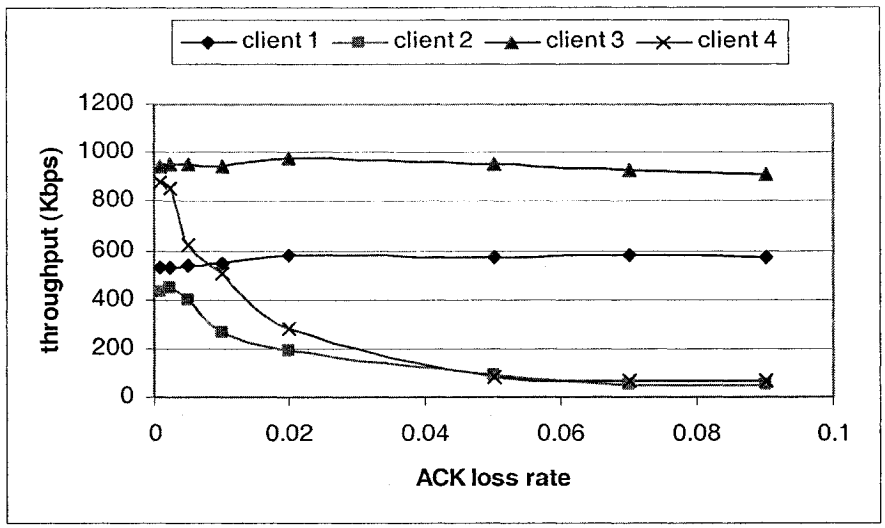


Figure 5-6 Mix of UDP and TCP with ACK loss

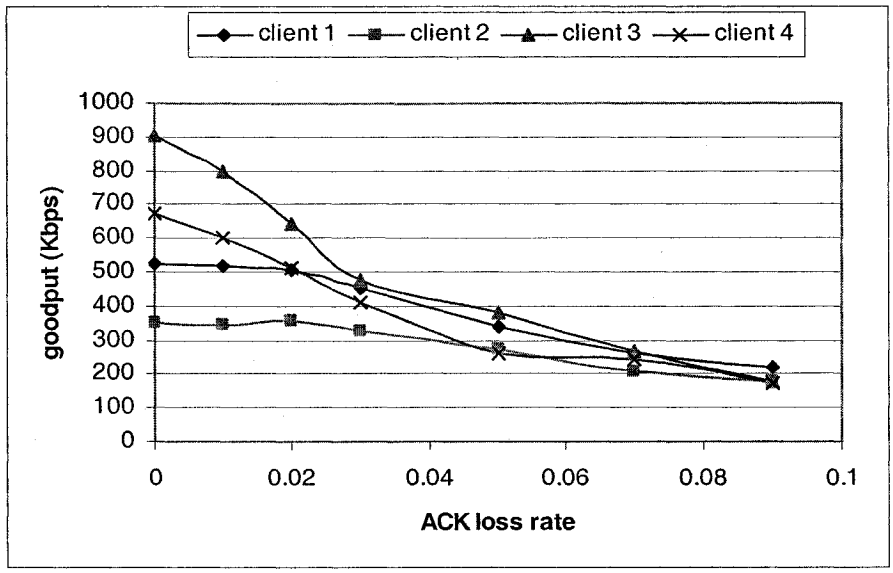


Figure 5-7 Impact of different RTT with ACK loss

Finally, we examine the combined effect of round-trip time delay and ACK losses on the performance of TCP connections. We use the same configuration as in table 5-2 and vary the ACK loss rate on the reverse direction. In agreement with earlier results in this section, the curves shown in Figure 5-7 indicate that the reservation at the forward path does not work when experiencing ACK losses at the reverse link. While for ACK loss rates higher than 6%, all TCP connections appear to be reaching the same goodput level, TCP connections with shorter RTT (clients from groups 1 and 3) have no advantage over TCP

connections with longer RTT (clients from groups 2 and 4). At the same time, high class reservation can not have any advantage to clients from groups with the low class reservation. This leads to the conclusion that under a loss ACKs scenario, ACK loss has a more significant impact on the performance than bandwidth reservation and RTT. According to our results shown in Figure 4-10, when ACK loss rate exceeds 6%, the difference between clients that were supposed to receive different treatment diminishes.

The findings in this section also mean the performance of Diffserv network suffers significant degradation under heavy ACK loss, similar to degradation of the traditional TCP/IP network in lossy wireless network link [32]. It is found that TCP transmission is almost stalled during heavy packet loss in wireless network (in the order of 10^{-2}) in [32].

All the above results imply that the unidirectional reservation does not provide advantage under the realistic scenario of bi-directional network traffic loading. Should we wise to provide differential treatment to Different Internet users, it is imperative that provision of some form of protection to ACK messages is required. In the next section we will give a feasible modification to the scheme described in Figure 5-2 to make bi-directional reservation.

5.3 Bi-directional Reservation and Its Effect

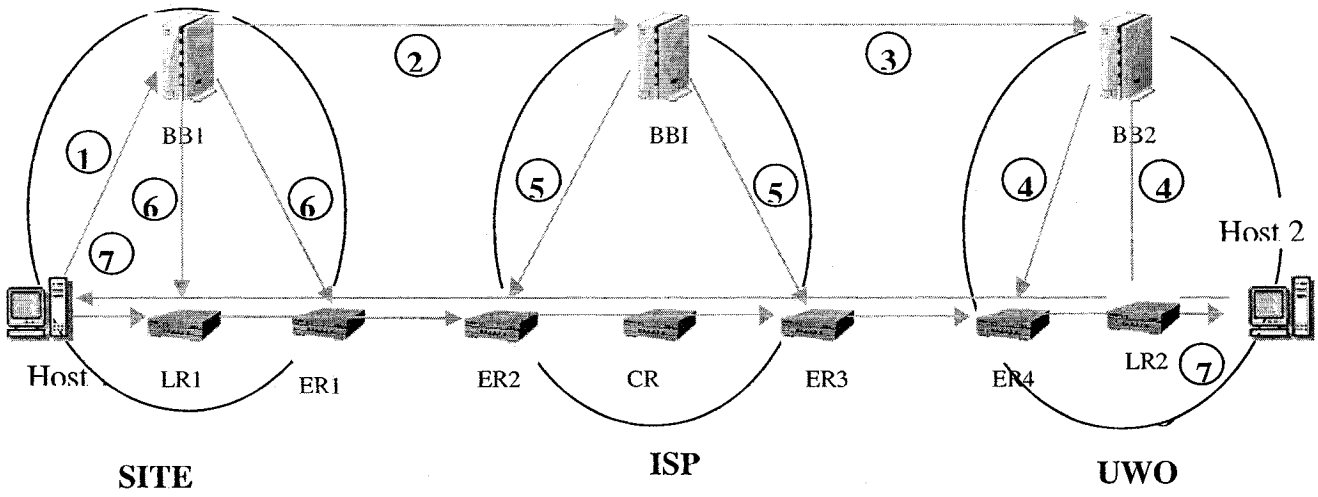


Figure 5-8 Bi-directional reservation example

5.3.1 Example of Bi-directional Reservation

The results presented in Section 3 suggest the failure of unidirectional reservation due to the possible ACK losses. Hence, we argue that we need to make reservation not only on the data-forwarding path but also on the reverse path, in order to avoid ACK loss. In the present section we modify the end-to-end allocation architecture that was described in section 2, in order to introduce bi-directional reservation. In figure 4-11, Host 1 (in domain SITE) needs to send data to host 2 (in domain UWO) at an average rate of 100 kbps using Assured Service class. At the same time, we assume that host 2 returns 20 kbps of ACK traffic to host 1. The bi-directional reservation procedure is described below.

1. Host 1 sends a RSVP PATH request to its Bandwidth Broker BB1. The request now includes 100 Kbps on forward path and 20 Kbps on reverse path.

2. BB1 makes an admission control decision if it can accept this request. If the request is denied, the reservation process has failed. If accepted, BB1 sends a PATH message to its neighbor BBI.

3. BBI then processes the request, makes its admission control decision and in turn sends its PATH message to BB2 if the request is authenticated.

4. BB2 checks this request and gives its permission. BB2 will use RSVP or another local signaling protocol to configure the edge leaf router LR2 to mark and shape ACK traffic sent from host 2, and ER4 is configured to support 100 Kbps data from host 1. Meanwhile BB2 returns RSVP RESV message to BBI.

5. Receiving RESV, BBI will configure its edge router ER2 and ER3 to support the reservation at both directions. Next, it sends RESV back to BB1.

6. After receiving RESV, BB1 will configure the leaf router LR1 to correctly classify and shape the admitted data from H1, and ER1 is set to accommodate ACK traffic from host 2. BB1 then sends RESV to host 1.

7. Now reservation is completed. Host 1 starts to send data and host 2 returns ACK with their reservation envelops.

Notice the difference from the unidirectional reservation scheme described in section 2. First, the reservation is made on both directions to avoid ACK loss and facilitate interaction between both ends. Sometimes, we must set up two-way allocation. Examples are interactive applications like videoconferencing and virtual private network service, which requires that both, incoming and outgoing traffic be guaranteed. Second, a BB may aggregate multiple requests and pass a single request to its neighbor. This can reduce the overhead of signaling messages. Since a signaling message may contain two-way reservation and multiple request aggregation, we can say that this reservation scales well with the increase of connections.

5.3.2 Simulation Results of Bi-directional Reservation

To evaluate the performance of our reservation schemes in a realistic network, we change our network settings in Figure 5-3. We are allowing both ends to be sending web traffic to the other side, i.e. data traffic is sent not only from server ***i*** to client ***i*** but also from client ***i*** to server ***i*** at the same time. On both directions, ACK packets are competing with data traffic for bandwidth, and the overall traffic rate exceeds the bottleneck capacity of 3 Mbps. Surely ACK packets are not immune to bottleneck congestion and can be delayed and lost. There have four connections groups in this section:

1. Connections group 1: Traffic is traversing from server 1 to client 1 and reverse.
2. Connections group 2: Traffic is traversing from server 2 to client 2 and reverse.
3. Connections group 3: Traffic is traversing from server 3 to client 3 and reverse.
4. Connections group 4: Traffic is traversing from server 4 to client 4 and reverse.

We run two sets of simulations under two scenarios, one using unidirectional reservation and the other bi-directional reservation. Under the unidirectional reservation, only data traffic on its direction is protected by the allocated bandwidth profile, while corresponding ACK packets on the return path are treated as best effort traffic that can only get the available bandwidth left over by data traffic running on that direction. Table 5-4 displays the bandwidth allocation to the data traffic of different connections groups. We still have two classes of service: low class connections groups 1 and 2 have 0.5 Mbps bandwidth reserved on both directions, and high class connections groups 3 and 4 have 1 Mbps bandwidth reserved on both directions. But there is no bandwidth allocated to ACK packets for any class connections groups.

Table 5-4 Bandwidth Allocation for Unidirectional Reservation

Connections group	1	2	3	4
Allocated Bandwidth to Data Traffic	0.5 Mbps	0.5 Mbps	1 Mbps	1 Mbps

Under the bi-directional reservation, ACK packets are protected on the return path as the packets of its acknowledged data on its forwarding path. That is: if data traffic is assigned to

the high class service on the forwarding path, then it's ACK will be assigned to the high class service on the return path too. If data traffic is assigned to the low class service on the forwarding path, then it's ACK will be assigned to the low class service on the return path too. Table 5-5 shows the bandwidth allocation under bi-directional reservation, and here both data and ACK traffic are protected by their class service profile.

Table 5-5 Bandwidth Allocation for Bi-directional Reservation

Connections group	1	2	3	4
Allocated Bandwidth to Data Traffic and ACK Traffic	0.5 Mbps	0.5 Mbps	1 Mbps	1 Mbps

We use a fairness index defined in [30] to evaluate the fairness of the shared bandwidth among users:

$$Fairness = \frac{\left[\sum_i x_i\right]^2}{n \bullet \sum_i x_i^2}$$

Where n is the number of links sharing the network resources, and x_i is the ratio of the actual goodput of connections to the fair share of the available bandwidth for the connections. This index reaches its maximum value 1 when the connections receive the allocation they subscribed for and is an increasing function of fairness.

Table 5-6 Case 1: Unidirectional vs. Bi-directional Reservation Results

Connections group	1 (low class)	2 (low class)	3 (high class)	4 (high class)
Goodput (Unidirectional Reservation)	531 Kbps	542 Kbps	804 Kbps	843 Kbps
Network Layer Throughput (Unidirectional Reservation)	552 Kbps	564 Kbps	836 Kbps	877 Kbps
Goodput (Bi-directional Reservation)	455 Kbps	488 Kbps	924 Kbps	889 Kbps
Network Layer Throughput (Bi-directional Reservation)	470 Kbps	507 Kbps	961 Kbps	913 Kbps

Table 5-7 Case 2: Mix of UDP and TCP Sources with Two Reservations

Connections group	1 (low class)	2 (low class)	3 (high class)	4 (high class)
Traffic Type	UDP	TCP	UDP	TCP
Goodput (Unidirectional Reservation)	668 Kbps	416 Kbps	921 Kbps	774 Kbps
Network Layer Throughput (Unidirectional Reservation)	701 Kbps	435 Kbps	960 Kbps	802 Kbps
Goodput (Bi-directional Reservation)	503 Kbps	346 Kbps	916 Kbps	877 Kbps
Network Layer Throughput (Bi-directional Reservation)	520 Kbps	362 Kbps	951 Kbps	910 Kbps

Table 5-8 Case 3: Different RTT TCP Sources with Two Reservations

Connections group	1 (low class)	2 (low class)	3 (high class)	4 (high class)
RTT (second)	0.5	1.0	0.5	1.0
Goodput (Unidirectional Reservation)	692 Kbps	398 Kbps	920 Kbps	646 Kbps
Network Layer Throughput (Unidirectional Reservation)	722 Kbps	420 Kbps	959 Kbps	672 Kbps
Goodput (Bi-directional Reservation)	543 Kbps	378 Kbps	917 Kbps	751 Kbps
Network Layer Throughput (Bi-directional Reservation)	565 Kbps	400 Kbps	955 Kbps	780 Kbps

Table 5-9 Fairness Comparison of Two Reservations

Case	1	2	3
Fairness Index (Unidirectional Reservation)	0.9820	0.9489	0.9207
Fairness Index (Bi-directional Reservation)	0.9996	0.9841	0.9769

In case 1, realistic web TCP traffic sources described in section 3.1 are used in all connections groups. On both directions, each connections group generates about 900 Kbps data if the network layer has unlimited bandwidth. In case 2, connections group 1 and 3 are CBR UDP sources, so that we can investigate the effect of UDP on web TCP traffic running

on connections group 2 and 4. Again in case 2, both UDP and TCP connections groups will generate 900 Kbps data if the network has unlimited bandwidth. Case 3 shows the impact of RTT on four web TCP connections groups. Here the connections groups 2 and 4 have double RTT as that of connections groups 1 and 3, and web traffic sources generating 900 Kbps data are used in all connections groups. Tables 5-6, 5-7 and 5-8 provide us with the average goodput and network throughput received by clients group in three different cases. As can be seen from the figures, in all cases, bi-directional reservation improves the throughput of high class connections groups and lowers the throughput of low class connections groups, when compared with unidirectional reservation. In case 1, the bi-directional reservation's allocation of bandwidth is very close to the subscribers' profile. The impact of UDP traffic is shown in table 5-7. The allocation here is worse than in case 1 but still is under control with bi-directional reservation. UDP sources have advantage over their TCP counterparts in the same class; nonetheless the bi-directional reservation is able to improve the goodput of high class TCP connections group 4 so that its goodput is comparable to high class UDP connections group 3. In case 3, RTT acts as predominant factor for bandwidth share in unidirectional reservation, since short RTT connections group 1 gets more throughput even than high class connections group 4, which has a longer RTT link. Once more, bi-directional reservation again performs better than the unidirectional scheme: the goodput of long RTT high class connections group 4 is higher than that of low class short RTT group 1.

The fairness index shown in table 5-9 is a clear proof of the advantage of bi-directional reservation against unidirectional one. In all cases, the fairness index of the bi-directional scheme is better than that of its unidirectional counterpart. The increase of the fairness is 1.8% for the first case, 3.7% for the second case and 6.1% for the last case. Judging from our results, we can make claim safely that bi-directional reservation provides a definite advantage in a diffserv capable network under different conditions.

5.4 Simulation Results of Reservation over Multiple Bottlenecks

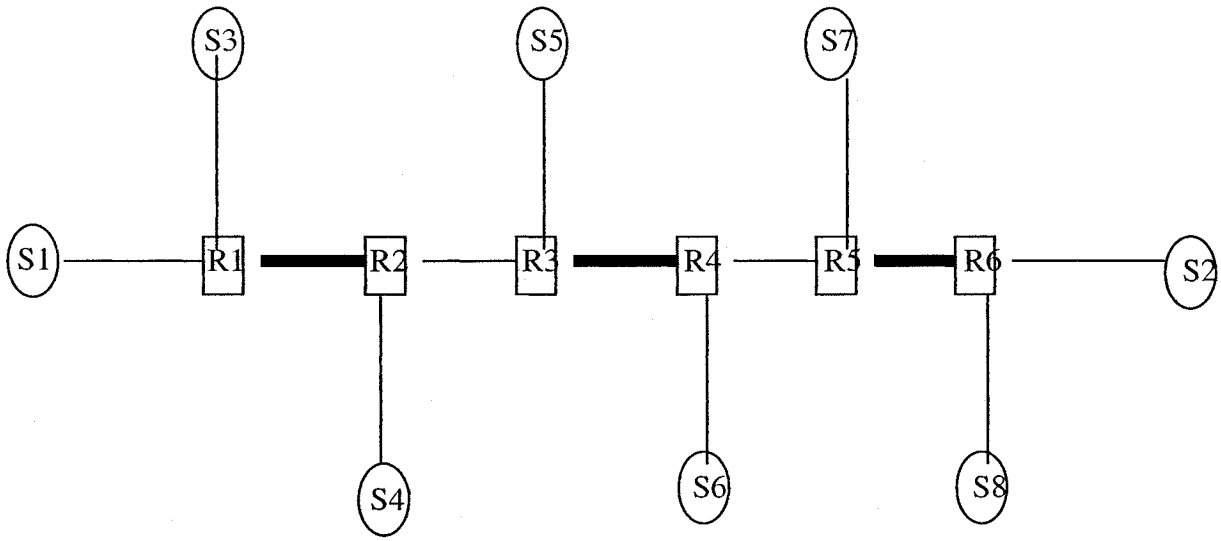


Figure 5-9 Network with multiple bottlenecks

This section considers the network configuration with multiple bottlenecks. In Figure 5-9, the thick lines are bottlenecks representing 1.0 Mbps link. The other links represent 44 Mbps links. R1, R2, R3 and R4 are diffserv-capable routers and implement Weighted Fair Queuing (WFQ) to support multiple service classes. There are four TCP/IP connections groups:

1. Connections group 1: Traffic is traversing from S2 to S1 and reverse.
2. Connections group 2: Traffic is from S4 to S3 and reverse.
3. Connections group 3: Traffic is from S6 to S5 and reverse.
4. Connections group 4: Traffic is from S8 to S7 and reverse.

Each connections group has 300 users and generates web TCP traffic at a rate close to 900 Kbps if the network has unlimited bandwidth. The WUE web traffic model discussed in chapter 3 and used earlier is used here as well to generate TCP traffic. Congestion exists in the three 1 Mbps bottleneck links since more than one connections groups' traffic tries to traverse through them. Connections group 1 competes with all three other connections groups along its path.

Four scenarios are configured to test the effect of multiple congestions and different resource reservation schemes described in previous section:

1. Best effort: All connections groups have traffic traveling in one direction from S_{i+1} to S_i (i = 1,3,5,7) are best-effort traffic competing with each other.

2. WFQ: All connections groups have traffic traversing in one direction from S_{i+1} to S_i (i = 1,3,5,7) with protection of their allocated bandwidth using WFQ. Connections group 1 is assigned with 0.5 Mbps bandwidth on each of three bottleneck links, while all remaining connections groups each get 0.5 Mbps bandwidth share on the bottleneck they pass. ACKs travel from S_i to S_{i+1} and are immune to loss in this case, since they do not compete with data flow and their traffic is only about 5% of their data traffic and far less than the bottleneck capacity.

3. 1D Reservation: All connections groups have traffic traversing in both directions from S_{i+1} to Si (i = 1,3,5,7) and reverse, but their ACKs are treated as best effort class without any bandwidth allocation. This corresponds to unidirectional reservation in section 5.3.2. Connections group 1 is assigned with 0.5 Mbps bandwidth on each of three bottleneck links, while all remaining connections groups each get 0.5 Mbps bandwidth share on the bottleneck they pass.

4. 2D Reservation: The same traffic configuration as scenario 3, but this time ACKs are assigned to the same class as the data stream they correspond to and are protected by bandwidth allocation. This corresponds to bi-directional reservation in section 5.3.2. Connections group 1 is assigned 0.5 Mbps bandwidth for both its data and ACK traffic on each of three bottleneck links, while the rest of the connections groups get 0.5 Mbps bandwidth share for both data and ACK traffic on the bottleneck link they pass.

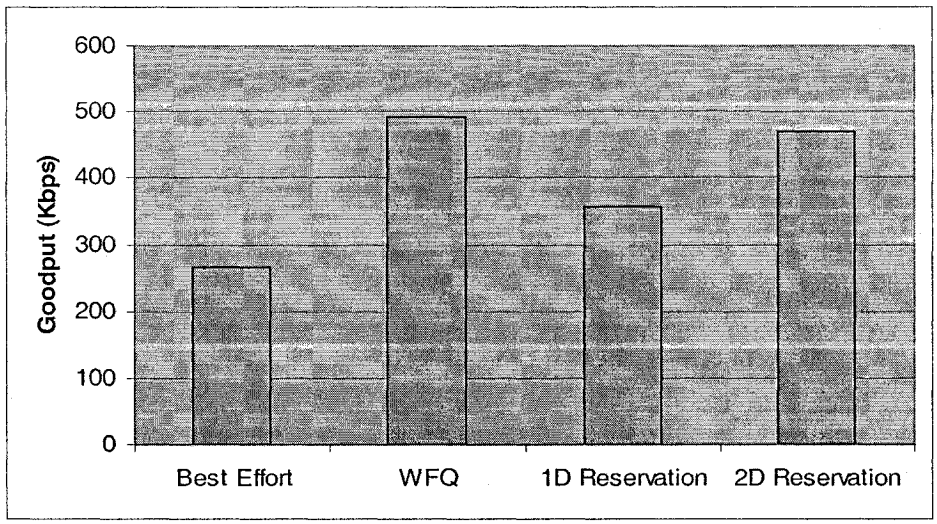


Figure 5-10 Goodput of connections group 1 in four scenarios

Figure 5-10 shows the results of four scenarios. In best effort case, the goodput of connection 1 is only 265 kbps after 3 bottlenecks. When ACKs are free from loss in scenario 2 (WFQ), diffserv works well and the goodput of connections group 1 is nearly 500 Kbps, equal to its allocated bandwidth. But this scenario is assuming ACKs are never lost along multiple bottlenecks, which is too optimistic. In scenario 3 (1D reservation), ACKs are treated as the best effort class and can only get the bandwidth left over by the data traffic that is protected by bandwidth allocation. In this scenario the goodput of connections group 1 is reduced to 358 kbps and far from its 500 kbps allocated bandwidth but still better than best effort case. In the last scenario, where ACKs are protected as their corresponding data traffic (via bi-directional reservation), the goodput of connections group 1 is 470 Kbps and close to 500 kbps allocated bandwidth.

We can see that when experiencing multiple congestions, the connections group 1's goodput is reduced to 265 Kbps, only half of its allocated bandwidth 500 Kbps in a best effort network. Using reservations with diffserv clearly improves the performance of traffic over multiple bottlenecks. Bi-directional reservation can achieve the expected bandwidth allocation. Unidirectional reservation cannot guarantee the bandwidth level it committed but is still performing much better than best effort scenario.

5.5 Conclusion

In this chapter, we have evaluated the performance of two different bandwidth allocation schemes for differentiated services capable environment, under realistic network conditions (with presence of UDP, different RTT and multiple bottlenecks). The first scheme is based on unidirectional reservation, while the second scheme on bi-directional reservation. The two different schemes have been evaluated under several different scenarios and traffic loading. The self-similar web traffic model discussed in chapter 3 is used throughout this chapter to evaluate the performance under realistic traffic model. A key part of our analysis, is the assessment of the effect, loss of ACK messages has on the performance of TCP connections running through the differentiated services capable network and on the bandwidth allocation. The results show that: (1) diffserv-capable networks perform well without ACK loss; (2) ACK loss may destructively impair the performance of TCP connections even when bandwidth reservation has been made on the data forwarding path; (3) bi-directional reservation performs better than unidirectional reservation in all examined scenarios. We also outline a possible reservation solution that protects from ACK losses that are occurring at the return path. Taking the realities of current Internet and the traffic loading levels into consideration, we must make bi-directional reservation in order to be able to provide the QoS guarantee to users. Also, under bi-directional reservation, the diffserv network can provide class service differentiation even under complex network conditions, something that is not possible under unidirectional reservation.

Chapter 6 Summary and Future Work

In this chapter, we conclude our work and propose ideas for future work.

6.1 Summary

In this thesis, we evaluated the traffic performance of different types of traffic in Diffserv-capable networks and reservation schemes of Diffserv structure. Following are three main contributions of this thesis work:

1. **Evaluation of realistic bursty web traffic model and its impact on DiffServ network**: A realistic self-similar web traffic model is implemented. From the standpoint of data rate, Diffserv architecture can effectively allocate the link bandwidth to a user according to his contracted profile no matter what traffic model is used. However, the comparable average data rates do not imply that the end-to-end delay and jitter are similar. In fact, the Poisson traffic model may be too optimistic in its assessment of packet delay and jitter, when compared to the more realistic WUE model. The delay under Poisson traffic is merely one tenth or less of that under WUE traffic, in the presence of intense congestions. Self-similar traffic should be used instead of Poisson traffic in simulations considering QoS commitment and resources scheduling, plus more performance metrics should be taken into account in analysis and design of diffserv networks.

2. **Evaluation of multimedia applications in Diffserv networks under realistic bursty web traffic**: Three mainstream multimedia applications are modeled and analyzed. Specially, a new ON/OFF model of next generation DIVE is developed. Based on the findings of last three sections in chapter 4, multimedia applications such as voice, video and DIVE applications can be well supported by diffserv-capable networks even in the presence of self-similar bursty web traffic. The EF class is a good candidate for these delay sensitive traffic like video and voice to protect the main cash flow of service providers. It is reasonable to assign these critical multimedia applications to EF service..

However when heterogeneous real time traffic is all put into the same EF class, the loss sensitive application like DIVE may experience some loss as [27] described. The performance of multimedia EF traffic is very well from our simulations when the real time traffic is homogeneous, but may not true when it is heterogeneous.

3. **Evaluation of Diffserv bandwidth allocation schemes**: The first is based on unidirectional reservation, while the second on bi-directional reservation. The two different schemes have been evaluated under several different scenarios and traffic loading. A key part of our analysis, is the assessment of the effect, loss of ACK messages has on the performance of TCP connections running through the differentiated services capable network. The results show that: (1) diffserv-capable networks perform well without ACK loss; (2) ACK loss may destructively impair the performance of TCP connections even when bandwidth reservation has been made on data forwarding path; (3) bi-directional reservation performs better than unidirectional reservation in all examined scenarios: in the presence of UDP, different RTT as well as multiple bottlenecks. We also outline a possible reservation solution that protects from ACK losses that are occurring at the return path. Under bi-directional reservation, the diffserv network can provide class service differentiation even under complex and highly loaded network conditions, something that is not capable of achieving under unidirectional reservation.

From network engineering's point of view, this thesis work shows that when the new generation QoS-capable network is designed the solution must consider the harsh nature of network traffic. The realistic self-similar traffic model must be used instead of traditional Poisson model. Both data packet loss and ACK loss must be carefully studied, not just the loss over data forwarding path.

6.2 Future Work

Possible future research, related to this thesis, can be summarized as follows:

We already saw the destructive impact of ACK loss in chapter 5.With Diffserv and other QoS mechanism reasonable expansion into wireless network, the lossy nature of wireless link is worth study to investigate its influence and propose suitable scheduling or reservation mechanism for wireless network.

The other interesting topic is to apply our rich application models (WUE, Voice, Video and DIVE) to new MPLS protocol network. We need to evaluate bursty web traffic and other critical multimedia models in more connection-oriented MPLS network. In addition, the bi-direction nature of TCP/IP is need to be carefully considered in unidirectional LSP context.

REFERENCES

[1] Stevens, W. Richard. "TCP/IP illustrated: the protocols", Addison Wesley, pp310 – pp311, 1994.

[2] D. Black, S. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, " An architecture for Differentiated Services", IETF RFC 2475, December 1998.

[3] K. Nichols, V. Jacobson, and L. Zhang, " A Two-bit Differentiated Services Architecture for the Internet", RFC 2638, November 1997.

[4] IETF, http://www.ietf.org.

[5] Juha Heinanen, Fred Baker, Walter Weiss and John Wroclawski, " Assured Forwarding PHB Group", IETF RFC 2597, June 1999.

[6] Van Jacobson, Kathleen Nichols and Kedarnath Poduri, "An Expedited Forwarding PHB", IETF RFC 2598, June 1999.

[7] P. Ferguson and G. Huston, "Quality of Service: Delivering QoS on the Internet and in Corporate Networks", Wiley, 1998, 320 pp.

[8] OPNET, http://www.mil3.com.

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

[10] Y. Bernet, et al, " A Framework for Differentiated Services", RFC 2998, November 2000.

[11] Y. Bernet, D. Durham and F. Reichmeyer, " Requirements of Diff-Serv Boundary Routers", IETF Internet Draft <draft-bernet-diffedge-01.txt>, November 1998.

[12] Mukul Goyal, Padmini Misra and Raj Jain, " Effect of Number of Drop precedence in Assured Forwarding", IETF Internet Draft <draft-goyal-dpstdy-diffserv-01.txt>, March 1999.

[13] J. Heinanen, et al., " A Single Rate Three color Marker", RFC 2697, September 1999.

[14] D. Clark and W. Fang, " Explicit Allocation of Best Effort Delivery Service", IEEE/ACM Trans. on Networking, vol. 6, no. 4, pp. 362-373, August 1998.

[15] D. Clark and J. Wroclawski, " An approach to Service Allocation in the Internet", IETF Internet Draft <draft-clark-diff-svc-alloc-00.txt>, July 1997.

[16] Duke Hong and Tatsuya Suda, " Congestion Control and Prevention in ATM Networks", IEEE Network, Vol. 5, No. 1, pp10 – pp16, July 1991.

[17] S. Floyd and V. Jacobson, " Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, Vol. 1, No. 4, pp397 – 413, August 1993.

[18] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control – the Single Node Case", IEEE/ACM Transactions on Networking, Vol. 1, No. 3, June 1993

[19] Anindya Basu and Zheng Wang, " A Comparative Study of Schemes for Differentiated Services", Proceedings of the IFIP Workshop on Protocols for High Speed Networks (PfHSN), June 1999.

[20] T. Yang, Z. Chen, A. Hafid and D. Makrakis, "The Performance of AF Service in Multi-DS Domain Networks", The 3rd International Workshop on Multimedia Network Systems (MNS2001), April 2001

[21] T. Yang, Z. Chen, A. Hafid and D. Makrakis, "An Evaluation of Different Marking Algorithms in AF service Network", MMNS 2000, September 2000

[22] Vern Paxson and Sally Floyd, "Wide-Area Traffic: The Failure of Poisson Modeling", IEEE/ACM Transactions on Networking, 3(3), pp. 226-224, June 1995.

[23] P. Barford and M. Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation", SIGMETRICS '98/PERFORMANCE'98, Performance Evaluation Review, v. 26, June 1998

[24] Recommendation H.261, ITU-T Standard, March 1993

[25] Recommendation H.262, ITU-T Standard, February 2000

[26] S. Nanda, D. J. Goodman, and U. Timor, "*Performance of PRMA: A packet voice protocol for cellular systems*," IEEE Trans. Veh. Technol., vol. 40, pp. 584--598, Aug.1991.

[27] H. Yu, Q. Zhou, D. Makrakis, et al, "Quality of Service Support of Distributed Interactive Virtual Environment Applications in IP Networks", Proc. IEEE Pacific RIM Conference, Victoria, August 2001

[28] A. Terzis, L. Wang, J. Ogawa, L. Zhang, "A Two-Tier Resource Management Model for the Internet", Global Internet 99, Dec 1999

[29] K. Fall and S. Floyd, "Simulation-Based Comparisons of Tahoe, Reno, and SACK TCP", Comp. Comm. REV., July 1996

[30] ATM-Forum, "ATM Traffic Management Specification Version 4.0", April 1996.

[31] H. Balakrishnan, V. N. Padmanabhan and R. H. Katz, "The Effects of Asymmetry on TCP Performance", 3rd ACM/IEEE Intl. Conference on Mobile Computing and Networking (MobilCom), September 1997

[32] B.S. Bakshi, P. Krishna, et al, "Improving Performance of TCP over Wireless Networks", International Conference on Distributed Computing Systems, May 1997

VITA

Zhao Chen was born in Xi'an, China, 1972. He studied at University of Science and Technology of China from 1988 to 1993, earning a B.Sc in Electrical Engineering. After working several years as a hardware engineer at Zhongxing Telecom Ltd. (ZTE) in China, he continued with graduate study at Department of Electrical and Computer Engineering, University of Western Ontario in January 1999. After one year he transferred to School of Information Technology and Engineering (SITE) of University of Ottawa for Master program.

He will be graduating with a M.A.Sc in Electrical Engineering in 2002 from University of Ottawa. He also start working as a software designer for Alcatel Canada in Ottawa from January 2001.