2002

# Performance analysis of QoS mechanisms in IP networks

Dix Xiaodong Jia
*University of Wollongong*

# Performance Analysis of QoS Mechanisms in IP Networks

A Thesis submitted in fulfilment of the requirements for the award of the degree

Master of Engineering (Honors)

From

**University of Wollongong**

By

Dix Xiaodong Jia

Department of Electrical, Computer and Telecommunication Engineering

January 2002

# Acknowledgement

I would like to express my thanks to my supervisor Dr. Tony Eyers and Professor Joe Chicharo for their advice, support and assistance in this study.

I would also like to thank Dr. Chuntong Chou for his advice and helpful discussions in Chapter 4.

Finally, I would like to thank my wife Dr. Jing Guan and my daughter Nancy Jia for their patience, understanding and support.

# Table of Contents

# List of Table and Figures

# Abstract

In this thesis a performance analysis of Quality of Service (QoS) mechanisms in IP (Internet Protocol) networks is presented. IP QoS mechanisms are surveyed and classified as either non-fractional service rate reserved or fractional service rate reserved. We show that most mechanisms supporting Differentiated Services (DiffServ) are non-fractional service rate reserved, while most supporting Integrated Service (IntServ) are fractional service rate reserved.

Among the non-fractional service rate reserved mechanisms, the focus is on two fundamental ones -- Threshold Dropping (TD) and Priority Scheduling (PS), from which many others, including Random Early Detection (RED) and RED with In and Out (RIO), are derived. Among fractional service rate reserved mechanisms, we specifically examine the loss behaviors of Latency Rate Servers (LR Servers), to which most well known mechanisms such as Generalized Processor Sharing (GPS), Weighted Fair Queuing (WFQ) or Packet-by-packet version GPS (PGPS) and Worst-case Faire Weighted Faire Queuing ($WF^2Q$) belong.

There are two issues addressed in this dissertation. One is how well do the scheduling mechanisms which support Diffserv provide various QoS levels. The second is the performance behavior, particularly the loss rate, of the various QoS mechanisms under a worst case scenario, when the input buffer of the server is finite. We also determine the arrival process that will result in the maximal average loss rate. The first issue is addressed with a performance analysis and subsequent comparison of TD and PS. A method for approximating the mean delay and loss for PS is proposed, and the results are verified with simulations. The second issue has been addressed by analyzing the loss behavior of LR servers. In particular, the arrival process that results in the maximal average loss rate for individual sessions of LR server is determined. Formulae for calculating the average loss rate are then derived, and zero loss buffer requirements for LR servers are obtained.

# Glossary of Acronyms and Symbols

## ACRONYMS

| | |
|---|---|
| ALQD | Approximated Longest Queue Drop |
| AF | Assured Forwarding |
| BOL | Burst Over Latency |
| CBQ | Class Based Queuing |
| Diffserv | Differentiated Services |
| EF | Expedited Forwarding |
| FCFS | First Come First Served |
| GPS | General Processor Sharing |
| IntServ | Integrated Services |
| IETF | Internet Engineering Task Force |
| LR Servers | Latency Rate Servers. QoS mechanisms belong to fractional service rate reserved scheduling mechanisms, where packets from individual applications are guaranteed a minimum service rate. |
| PHBs | Per Hop Behaviors |
| PS | Priority Scheduling |
| QoS | Quality of Service |
| RSVP | Resource Reservation Protocol |
| RED | Random Early Detection |

| | |
|---|---|
| SCFQ | Self-Clocked Fair Queuing |
| SEFF | Smallest Eligible Virtual Finish time First |
| SCFQ | Self Clock Fair Queue |
| TD | Threshold Dropping |
| WFQ | Weighted Fair Queue |
| $WF^2Q$ | Worst case Fair WFQ |
| WRR | Weighted Round Robin |

## SYMBOLS

| | |
|---|---|
| $(a)^+$ | $max\{a, 0\}$ |
| $A_j(\tau,t)$ | the amount of traffic of session $j$ that leaves the bucket and enters the network during $(\tau,t]$ |
| $b_i(t)$ | the token bucket state at time $t$ |
| $b_s$ | token bucket state (number of tokens in the bucket) |
| $B_i$ | the set of sessions that are busy in the time interval $(t_{i-1},t_i)$ |
| $B(t)$ | the set of backlogged sessions at time $t$ |
| $C_j$ | the maximum rate at which the bits of session j can leave the bucket |
| $D_j^*$ | the maximum delay for session $j$ |
| $F_p$ | the time at which packet p will finish service under GPS |
| $F_p^w$ | the time at which packet p will finish service under WFQ |
| $g$ | average reward per transaction of the system if it started from state $(b_s,q_s)$ and the number of transitions $m$ is large |
| $gr_j$ | guaranteed service rate for session $j$ |

| | |
|---|---|
| $h_j(t)$ | the sequence number of the packet at the head of the session $j$'s queue |
| $h(b_s,q_s)$ | relative value of the policy. It initially represents the intercepts at $m=0$ of the asymptotes of $v_s(m)$ |
| $k_i$ | the input buffer size of session $i$ at the server |
| $K_i$ | input buffer size, it also means the zero loss buffer size for Bang Bang policy |
| $L_{max}$ | the maximum packet size |
| $n$ | the number of the session busy period |
| $N$ | the number of states that the Markov process may have |
| $P_{b_s j}$ | the transition probability from state $b_s$ to state $j$ in next transition |
| $P_d$ | the transition probability from state $(1,k_i)$ to state $(1,k_i)$ in next transition. The probability equal to $\frac{\theta}{\theta + 1/\rho_i}$ |
| $p(Q)$ | Packets drop probability |
| $q_s$ | the input buffer queue |
| $Q_j(a_j^i-)$ | the queue length of session j just before time $a_j^i$ |
| $Q_j^*$ | the maximum backlog for session $j$ |
| $Q_i(t)$ | the amount of session $i$ traffic queued in the server at time $t$ |
| $Q$ | average queue length |
| $r_{b_s j}$ | the reward obtained when the Markov process make a transition from state $b_s$ to state $j$ |

| | |
|---|---|
| $R(s)$ | the reward to be expected in the next transition out of state $b_s$ |
| $S_j(\tau,t)$ | the amount of traffic served in an interval $(\tau,t]$ for sessions $j$ |
| $S_{i,j}(\tau,t)$ | the service received by the traffic of session j that arrived during time interval of $(\tau,t]$ |
| $S_j^{h_j(t)}$ | the virtual start time of the packet at the head of the session $j$ queue |
| $Td_i$ | threshold for flow $i$ |
| $\mu$ | service rate of an server |
| $\lambda_i$ | mean arrival rate for flow $i$ |
| $\mu_i$ | the allocated service rate for session $i$ at the server |
| $u_{stp}(t)$ | the unit step function |
| $\mu_i^0$ | the guaranteed service rate for session $i$ |
| $u_i(t)$ | the function of how the tokens in leaky bucket being used at time $t$ It is also the rate of increasing of Arrivals $A_i(0,t)$ |
| $u_s$. | the decision to use tokens when system is in state $(b_s, q_s)$ at time $s$ |
| $v_s(m)$ | the sum of the expected total earnings in the next $m$ transitions if the system is now in state $b_s$ |
| $V(t)$ | System virtual time |
| $W_i(\tau,t)$ | the amount of service received by session $i$ during $(\tau,t)$ |
| $\alpha$ | a constant which determines how fast the mechanism will respond to changes in the queue length |
| $\theta$ | latency of the LR servers |

| | |
|---|---|
| $\phi_j$ | a positive number which can be interpreted as the weight by which the service rate is assigned to session $j$ |
| $r$ | service rate of the server |
| $(\sigma_j, \rho_j, C_j)$ | stand for leaky bucket |
| $\sigma_j$ | the leaky bucket capacity for session $i$ |
| $\rho_j$ | the token generation rate |
| $\delta(t)$ | the impulse function |
| $t_d$ | is the time at which the system is in state $(\sigma_i, 0)$ |
| $tp_i$ | the token bucket filled up every $tp=\sigma_i/\rho$ time units |
| $T_{ieb}$. | time required to empty the token bucket |
| $\tau_{tpi}$ | the set of start times of the periodic arrivals of session $i$ |
| $\beta$ | token interarrival time ($\beta=1/\rho_i$) |
| $\vartheta_d$ | the latency of each session $d$ busy period $(d=1,2,...n)$ |
| $u_{stp}(t)$ | the unit step function |
| $\Re_i$ | zero loss buffer requirement (denoted with $\Re_i$ ) for BOL |
| $\eta$ | the number of arrival period that follows Bang Bang Policy and BOL policy. $\eta \geq 0$, $\eta < \infty$. |

# Chapter 1 Introduction

## 1.1 BACKGROUND

The rapid growth of new Internet applications make IP networks mission critical and will provide a future demand for Quality of Service (QoS) provision. Two efforts have been made by the IP community to develop standards that support a variety of scalable QoS capabilities for IP networks. These standards are called Integrated Services (IntServ) and Differentiated Services (Diffserv).

The IntServ model is characterised by resource reservation, using the Resource Reservation Protocol (RSVP) to manage QoS requirements for individual application sessions. IntServ makes it possible for an application to request QoS with a high level of granularity and the best guarantees of service delivery. This model, however, faces some important difficulties such as the deployment and scalability of RSVP and the requirement for an inter-domain policy.

On the other hand, by grouping traffic with similar QoS requirements into an aggregate and providing consistent treatment for the aggregate, Diffserv is able to provide scalable QoS capabilities. Diffserv defines configurable types of packet forwarding (Per–Hop Behaviours), that can provide local (per-hop) service differentiation for large network traffic aggregates. Diffserv compliments IntServ to provide a new service model, which enables end to end QoS more effectively [55].

For either IntServ or Diffserv, packets of different sessions belonging to different service classes will interact with each other when they are multiplexed at the edge network switches (corresponding to IntServ) and forwarded in the core network switches (corresponding to Diffserv). Therefore, QoS scheduling mechanisms at switching nodes play a critical role in providing agreed QoS to applications when controlling the interactions among the different traffic streams and different service classes.

The many different QoS scheduling mechanisms proposed in the literature can be classified into non-fractional service rate reserved and fractional service rate reserved scheduling mechanisms. For example, Threshold Dropping (TD) [14] and Priority Scheduling (PS) [31], proposed for the Diffserv Assured Forwarding (AF) and Expedited Forwarding (EF) Per Hop Behaviours, belong to non-fractional service rate reserved mechanisms, where there is no service rate guarantee for the packets of lower priority sessions. Latency Rate Servers (LR Servers) [13] belong to fractional service rate reserved scheduling mechanisms, where packets from individual applications are guaranteed a minimum service rate. A key example is Weighted Fair Queue (WFQ) [3], which is potential QoS scheduling mechanism for IntServ and an alternate means to PS for Diffserv EF implementation. WFQ can be proved to be an LR server [13]. It also has been proved that General Processor Sharing (GPS) [3], Worst case Fair WFQ ($WF^2Q$) [28], Self Clock Fair Queue (SCFQ) [18] and Weighted Round Robin (WRR) [37] are all examples of LR Servers [51].

The Quality of Service (QoS) of a packet network is indicated by a combination of criteria that include loss probability, delay and delay jitter. To provide a guaranteed QoS network requires the determination of whether it has sufficient resources to meet the required service level. A key issue is a quantitative understanding of the performance arising from the various proposed QoS mechanisms, in terms of packet delay and packet loss rate. Future networks are likely to be heterogeneous in terms of scheduling mechanisms used in servers. Therefore another significant issue is the performance behavior, particularly loss behaviors under the worst case scenario for this broad range of scheduling mechanisms.

This thesis addresses these two issues in IP networks. The initial approach to analyze the performance of two Diffserv mechanisms -- Threshold Dropping (used in Assured Forwarding) and Priority Scheduling (used in Expedited Forwarding) in terms of packet loss and mean packet delay. To further examine the per hop behavior of PS, an analytical model is developed based on non-preemptive priority queues. The focus then shifts to the most widely used scheduling mechanisms, LR servers. The traffic arrival process that will result in the maximum average loss rate is determined. The upper bound of the average loss rate and the zero loss buffer requirement of a corresponding session are derived. Numerical results are verified with simulations. The significance of this study is that it provides a better understanding of network per hop behaviors and the resources requirements for the worst case scenario at a switch, where a broad range of scheduling mechanisms may be deployed. This would be helpful to network providers designing and provisioning future IP networks.

## 1.2 CONTRIBUTIONS RESULTING FROM THESIS

This dissertation has classified the many QoS mechanisms into fractional service rate reserved and non-fractional service rate reserved mechanisms. Performance analysis of these QoS mechanisms has been done to determine how well the DiffServ supported QoS mechanisms, most of which are non-fractional service rate reserved type, perform in providing various levels of QoS requirements. The focus has been on the two basic Diffserv mechanisms - TD and PS. Another important issue, the loss behaviours of a broad range of scheduling mechanisms called LR servers, has also been addressed in this dissertation.

The contributions of the dissertation are as follows:

1 Determinations of packet loss and mean packet delay of the TD mechanism with two arrival traffic flows.

2 Observation of the impact of the variable threshold of the non-preferred flow on packet loss and mean packet delay.

3. A clear tradeoff between packet loss and mean packet delay for preferred and non-preferred flows is observed in PS when buffer allocation changes.

4. Presentation of an approximate method for calculating the packet loss and the mean packet delay of non-preemptive Priority Scheduling with a finite

buffer and two traffic flows. The accuracy of this method has been confirmed with simulations.

5. Development of simulation models for PS with three traffic flows. Results show that highest priority flows can meet the requirements of Expedited Forwarding.

6. Extension of applications of the theory of LR servers to include packet loss rate evaluation for LR servers. Derivation of the maximal average loss rate for Latency Rate Servers for the worst case scenario.

7. Observation of zero loss buffer requirements in LR server systems and comparison between different arrival processes.

8. Determination and proof of arrival processes that will result in the maximal average loss rate for an individual session of an LR server when the traffic is leaky bucket smoothed.

## 1.3 DISSERTATION OVERVIEW

This dissertation is organized as follows:

**Chapter 2** reviews the literature on QoS scheduling mechanisms in IP networks. QoS scheduling mechanisms are classified into non-fractional service rate reserved and fractional service rate reserved scheduling mechanisms. This chapter also presents the key issues arising from the literature, which are addressed in the thesis.

**Chapter 3** presents the analysis of non-fractional service reserved mechanisms. In this part, two Diffserv mechanisms, TD and PS, are examined. Packet loss and mean packet delay in TD and PS are compared, based on the same level of packet loss for the preferred flow. Further analytical results for PS mechanisms are presented.

**Chapter 4** analyses the fractional service rate reserved QoS mechanisms. A broad range of scheduling mechanisms that belong to the fractional service rate reserved class (i.e. LR servers) are described. Using the LR server framework, the maximum average loss rate of an LR server is derived for the worst case scenario. The arrival process that results in the maximal average loss rate is determined and proof is provided. Numerical results are then verified with simulations.

**Chapter 5** presents a case study to extend the study of the arrival processes that follows the Bang Bang policy to LR servers. Simulations and numerical results are presented to verify the calculation of maximal average loss rate of a session in an LR server under the worst case scenario and to compare the zero loss buffer requirements for the arrival processes that follow Burst Over Latency (BOL) and Bang Bang policy.

**Chapter 6** provides summary of this dissertation, conclusions and future research work.

**Appendix**

Appendix provides the full text of the publications based on the dissertation.

## 1.4 PUBLICATIONS BASED ON THE THESIS

I. D. Jia, E. Dutkiewicz and J. Chicharo "Performance Analysis of QoS Mechanisms in IP Networks", ISCC2000

II. D. Jia, J. Chicharo E. Dutkiewicz "Understanding Traffic Behavior – An Approximation Method For Computing Packet Loss and Mean Packet delay in Non-preemptive Finite Priority Queues", PAM2000

# Chapter 2

# QoS Scheduling Mechanisms in IP Networks

## 2.1 INTRODUCTION

Current IP networks (such as the Internet) mostly offer a best effort service, where the performance of a session can degrade significantly when the network is overloaded. This is because packets from different connections interact with each other at switches where they are multiplexed. There is an urgent need to provide network services with performance guarantees, and hence mechanisms to support these guarantees. One of the key issues in providing guaranteed performance service is the choice of packet scheduling mechanisms.

This chapter surveys the scheduling mechanisms proposed for Diffserv and IntServ and classifies them based on the fractional service rate reservation. This survey and classification identify key issues in the performance of QoS mechanisms that will be addressed in this study. Section 2.2 discusses the QoS requirements for future IP networks. In section 2.3 after a survey of QoS mechanisms, a classification of scheduling mechanisms based on service rate reservation is proposed for all work conserving schedulers. This classification outlines the methodology that has been adopted in the analysis of QoS mechanisms in the thesis. Based on the survey and classification of the QoS mechanisms, section 2.4 discusses the key issues in the performance of QoS mechanisms that will be addressed in the following chapters.

## 2.2 QOS REQUIREMENTS FOR FUTURE IP NETWORKS

The Internet is making the transition from a best effort service model, where traffic is processed as quickly as possible without guarantees of delivery, to one that can provide differentiated predictable service levels for specific QoS requirements. This transition is driven by the rapid transformation of the Internet into a commercial infrastructure and rapidly developed demands for service quality [8] [15] [21] [41]. Introduction of these services implies that future IP networks need to discriminate between different packets, in contrast to existing best effort networks, which treat all packets equally.

Service quality in the Internet can be expressed as the combination of network imposed delay, jitter, bandwidth and reliability [22]. Greater delay places higher stress on the operating efficiency of the Transport Control Protocol (TCP). This is because increasing delay can result in deterioration of the sensitivity of the protocol to short term dynamic changes in network load. High levels of jitter can cause very conservative round trip time estimates to be made by the TCP protocol which result in inefficiency in re-establishing a data flow connection.

For multi-medium applications such as interactive video, the introduction of delay can causes the system to appear unresponsive. Bandwidth shortage may lead to overflow of the input buffer at a switch and packets that find the buffer is full are lost [11]. The levels of services required by applications may vary and the service providers are willing to not waste resources while provide guaranteed QoS to their customers.

According to these measurements of service quality, it is required that future IP networks provide differentiated and guaranteed services. It will thus be essential for future IP networks to provide QoS applications in a customer specific manner such as gold, silver and bronze services and to guaranteed service for applications requiring fixed delay and loss rate.

In practice, the combination of IntServ (RSVP) and Diffserv will be required in providing end-to-end and top to bottom QoS [10][19][55]. One example of this idea is illustrated with Figure 2.1. Of course, the national wide or international wide combination of IntServe and Diffserv requires the co-operation among different network operators.

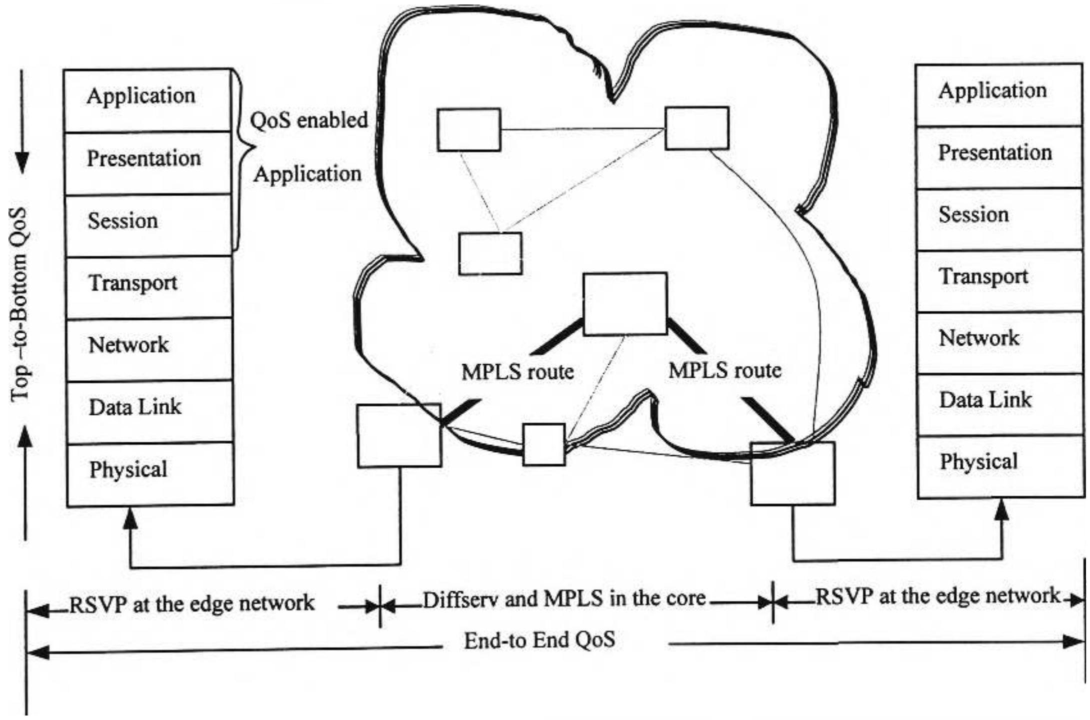



FIGURE 2.1 ILLUSTRATION OF THE CONBINATION OF INTSERV MECHANISMS (RSVP) WITH DIFFSERV

Efficient support of the requirements in providing differentiated service and end to end performance will however require implementation of various QoS mechanisms in different parts of the network. It is widely accepted in the literature and evidenced by industrial vendors that mechanisms will still be needed to provide QoS to applications [1] [4] [5] [29] [30] [33] [36] [39] [50]. Various mechanisms and protocols proposed by the Internet Engineering Task Force (IETF) for integrated and differentiated services seek to provide interoperable, customisable solutions in this area [10] [19] [24] [42] [53]. The next section will survey and classify these various mechanisms.

## 2.3 CLASSIFICATION OF QOS SCHEDULING MECHANISMS

As stated in section 2.2, service differentiation and guaranteed performance are the QoS requirements of future IP networks. In this section, the various QoS scheduling mechanisms proposed in the literature for IntServ and Diffserv will be surveyed and categorized for further analysis.

There are different Classifications of QoS mechanisms such as Packet Dropping Policy (PDP) [56], Traffic Management Algorithms and Packet Service Disciplines (PSD) [57] have been reported in the literature. Packet Dropping Policy is to drop packets to reduce traffic congestion and maintain the guaranteed QoS according to certain policy. In this class, packets are normally served at the server with same priority. Packet Service Discipline allocates resources according to the reservation during data transfer, bandwidth-promptness and buffer space. It characterized with separate queues and service policies for guaranteed service and other packets. A service discipline can be further classified as either work-conserving or non-work conserving.

Although the classification could be done in various ways, in this dissertation, the classification is made on the basis of service rate reservations which have significant impact on the two main factors of a QoS mechanism---the packet loss rate and delay. In this classification, all QoS mechanisms are categorized as fractional service rate reserved and non-fractional service rate reserved mechanisms. Work-conserving PSD falls into the fractional service rate reserved category while PDP and PS belong to the class of non-fractional service rate reserved.

Introducing a new terminology rather than simply using existing categorization such as Latency Rate Servers (LR Servers)* and non-LR Servers has the advantages that it makes the dissertation present a clear, simpler structure and issue focusing.

It is interesting to observe, in the following sections, that most scheduling mechanisms suggested for Diffserv are non-fractional service rate reserved and those for IntServ are mainly fractional service rate reserved scheduling mechanisms.

** Please note LR that LR server is fully defined and elaborated in chapter 4.*

### 2.3.1 NON-FRACTIONAL SERVICE RATE RESERVED MECHANISMS

Non-fractional service rate reserved scheduling mechanisms are defined as follows:

*For a server attending N packet flows, there are no minimum bandwidth guarantees for flows with lower priority. The output bandwidth is fully shared by all flows that have been backlogged, in the order of higher to lower priority or in a First Come First Served (FCFS) manner.*

TD [14], PS [31], RED [46], RIO [7] and buffer management schedulers [43] fall into the non-fractional service reserved scheduling mechanisms category. A concept model is illustrated in Figure 2.2. There are N packet flows are attended by a server with service rate of μ. If flow 1 has the highest priority, there is no minimum bandwidth guarantee for flows 2 to N.

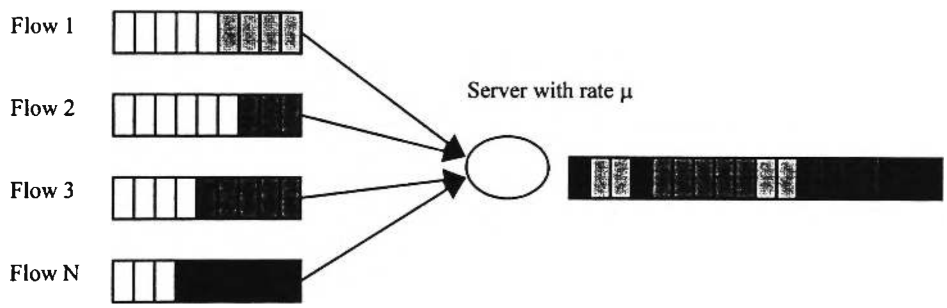



FIGURE 2.2 NON-FRACTIONAL SERVICE RATE RESERVED SCHEDULER

#### 2.3.1.1 FIRST COME FIRST SERVED

First Come First Served (FCFS) is the simplest scheduling mechanism, whose principle is that packets are served in the order in which they have arrived. Its implementation is simple and no per flow state maintenance is required. This mechanism on its own does not provide delay or rate guarantees. Delay guarantees are proportional to buffer size, and there are no bandwidth guarantees or flow isolation, which means that FCFS itself does not support service differentiation. For this reason, FCFS usually works as a default scheduler in buffer management schemes such as Threshold Dropping (TD) [11] [14] and Random Early Detection (RED) [46].

#### 2.3.1.2 TD, RED AND RIO

TD, RED (Random Early Detection), RIO (RED with In and Out packet) and Approximated Longest Queue Drop (ALQD) are schemes for deciding which packets can be stored as they wait for transmission, while the scheduler controlling the actual transmission of packets is FCFS. We may describe them as conditional dropping schedulers, and they are schematically illustrated in Figure 2.3.

In the Threshold Dropping mechanism, the decision to accept or discard a packet is based on the current buffer usage of the flow from the source of the packet. A packet that reached its threshold in the buffer is dropped. Service discrimination is supported by assigning different thresholds to packet flows.

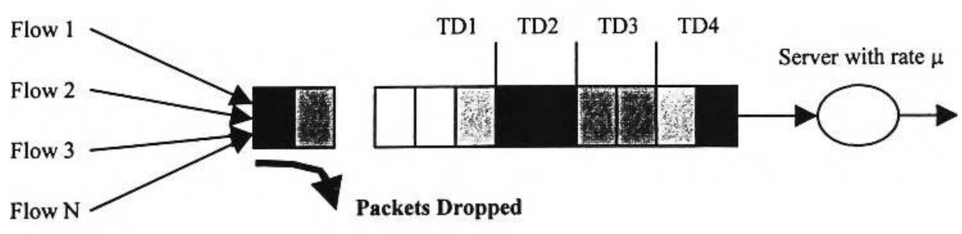



FIGURE 2.3 CONDITIONAL DROPPING SCHEDULERS

There are some problems that arise in TD. For instance, it can not effectively prevent the onset of congestion and, on the other hand, it may always cause packets to be dropped from same flow. This problem motivated the development of RED mechanisms. Based on TD, RED randomly drops packets when the buffer content exceeds a given threshold, so that heavy flows experience a large number of dropped packets in case of congestion. RED scheduling mechanism can be described using the following components [17].

- Computation of average queue length $Q$

  $Q_n=(1-\beta)\ Q_{n-1}+\beta q_n,\ 0\le\beta\le 1$

  Where $Q_n$ is the current average queue length and $q_n$ is the current queue length. $\beta$ is a constant value that determines how fast the mechanism will respond to changes in the queue length.

- Probabilistic packets dropping according to the average queue length

  If Q is less than minimum threshold $TD_{min}$, there is no packet dropped.

  If Q is exceeds the maximum threshold $TD_{max}$, all packets are dropped.

  Packets are dropped with probability p(Q) where $TD_{min} \le Q \le TD_{max}$.

  $p(Q)=p_{max}(Q-TD_{min})/(TD_{max}-TD_{min})$ where $0\le p_{max}\le 1$

- A counter is used to track the number of packets accepted in the queue since the last drop and to avoid the global synchronization problem, i.e. always dropping packets from same flow. This ensures that packets are dropped in a random manner.

According to Floyd, setting $p_{max}$ to 1 is not recommended and simulations show that $p_{max}$=0.1 is a suitable setting [17].

RIO is a variant of RED and inherits all the features of RED. In addition, by tagging packets that conform to the connection contract as In packets and those that don't as Out packets, it is able to differentiate and penalize packet flows using more resources than have been contracted with the network. Service discrimination between In and Out packets can be achieved in RIO in different ways. One way is to use two thresholds to decide when to start dropping packets. The threshold for In packets is set higher than the threshold for Out packets. Another way is by using the same threshold for both In and Out packets but selecting higher dropping probabilities for Out packets, i.e., $p_{max\_out} > p_{max\_in}$. As in RED, the average queue length Q determines in which region the scheduling mechanism takes dropping actions. The regions are congestion control (Q is above the highest threshold), congestion avoidance (Q is between the thresholds) and normal operation (Q is below the low threshold). Note that when calculating Q for In packets, the counter will only count the In packets, while both type of packets should be counted for Out packets.

The implementation of Assured Forwarding (AF) [24], which is one of the recent proposed Diffserv Per Hop Behaviors recently proposed by IETF, requires an active scheduling mechanism to minimize long-term congestion within each service class while allowing transient congestion resulting from bursts. A resent study [38] [49] has shown that TD, RED and RIO are suitable AF implementations. As discussed above, TD is a fundamental Scheduling mechanism, and buffer capacity has a significant impact on its QoS. Thus, further analysis on the QoS performance of TD and the impact of buffer threshold and buffer size is necessary and will be conducted in Chapter 3.

#### 2.3.1.3 PRIORITY SCHEDULING MECHANISM

The Priority Scheduling mechanism provides the ability to support different levels of QoS with a rather coarse granularity. Packet flows are classified using a number of static priorities, and each flow is assigned to an individual queue. Packets from lower priority queues are served only if all higher priority queues are empty. Within each queue, however, packets are served according to the FCFS rule. Although PS does not readily allow end-to-end performance guarantees to be provided on a per flow basis, it does offer a certain amount of service differentiation capability and provides better QoS with low loss, low latency, low jitter and assured bandwidth for the highest priority flow. This highest priority flow is independent of other traffic flows. These features exactly conform to the requirements of the Expedited Forwarding PHB of Diffserv proposed by the IETF [53]. Like FCFS, PS has a simple implementation. More importantly, PS is a basic scheduling mechanism in EF from which many other scheduling mechanisms are derived.

Some examples are Class Based Queuing (CBQ), which aims to solve the starvation problem of PS [34] and Weighted Fair Queuing (WFQ) which, apart from solving the starvation problem, improves the fairness and granularity of PS.

Because of this, it is important to conduct a performance analysis of PS and a further study on this will be presented in Chapter 3.

### 2.3.2 FRACTIONAL SERVICE RATE RESERVED MECHANISMS

Fractional service rate reserved scheduling mechanisms can be defined as follows: *If there are N flows attended by a single server, the bandwidth allocated to each flow is guaranteed with a minimum bandwidth based on its assigned fraction of the service rate of the server. Output bandwidth is fully shared by all backlogged flows in proportion to their assigned fraction.*

According to this definition, a separate queue needs to be maintained for each packet flow. Some well known scheduling mechanisms, such as GPS, PGPS (or WFQ), Worst case Fair Weighted Fair Queueing ($WF^2Q$), Self-Clocked Fair Queuing (SCFQ) [18] and Waited Round Ribbon (WRR) belong to the fractional service reserved scheduling mechanism category. A general model of scheduling mechanisms called Latency Rate Server (LR Servers) has recently been proposed [13]. LR Servers have some new properties but still meet the same definition of fractional service rate reserved scheduling mechanism [51]. A more detailed discussion of this general model and its properties will be presented in Chapter 4.

#### 2.3.2.1 GENERAL PROCESSOR SHARING

General Processor Sharing (GPS) is generalization of Uniform Processor Sharing as described in [35], and its packet based version PGPS (or WFQ) is developed by Parekh [3] and Demers [2]. Based on GPS, PGPS was combined with Leaky

Bucket [52] rate control [3] to provide flexible, efficient and fair use of the output link of a single node.

The GPS scheduling mechanism is defined with the assumption that the server is work conserving (i.e. the server is never idle if there is work in the system) and operates at a fixed rate. Let $S_j(\tau,t)$ and $S_k(\tau,t)$ denote the amount of traffic served in an interval $(\tau,t]$ for sessions j and k respectively. A session backlog time period $(\tau,T]$ is defined in [3] as that within the time period (session backlog period) the session queue is not empty at any time $t\in(\tau,T]$. A GPS server is then further defined with use of the concept of session backlog period. For any session j that is continuously backlogged in $(\tau,t]$, GPS server has the following properties:

$$\frac{S_j(\tau,t)}{S_k(\tau,t)} \geq \frac{\phi_j}{\phi_k} \text{ where } j,k = 1,2,...,N \text{ and } \phi_1,...\phi_N \text{ are positive numbers} \qquad (2\text{-}1)$$

Where k can be any session from 1 to N.

The positive number $\phi_j$ can be interpreted as the weight by which the service rate is assigned to each session. If the service rate of the server is *r* and a summation is done for all *k* of (2-1), it can be found that

$$S_j(\tau,t)\sum_k \phi_k \geq (t-\tau)r\phi_j$$

and session j is guaranteed a rate of

$$gr_j = \frac{\phi_j}{\sum_k \phi_k} r \qquad (2\text{-}2)$$

If $B(\tau)$ is the set of backlogged sessions at time $\tau$, the service rate of a non-backlogged session j will be

$$r_j = \frac{\phi_j}{\sum_{k \in B(\tau)} \phi_k} r \geq g r_j \qquad (2\text{-}3)$$

Together with using a leaky bucket to constrain incoming traffic, the GPS scheduling mechanism guarantees applications with a worst case delay due to the guaranteed service rate stated in (2-2). The constraint imposed by leaky bucket $(\sigma_j, \rho_j, C_j)$ to the traffic of session j that enters the network is

$$A_j(\tau,t) \leq \min\{(t-\tau)C_j, \sigma_j + \rho_j(t-\tau)\} \quad \forall t \geq \tau \geq 0 \qquad (2\text{-}4)$$

where $A_j(\tau,t)$ is the amount of traffic of session $j$ that leaves the bucket and enters the network during $(\tau,t]$, $\sigma_j$ is the leaky bucket capacity, $\rho_j$ is the token generation rate and $C_j$ is the maximum rate at which the bits of session j can leave the bucket $(C_j > \rho_j.)$.

The worst case packet delay for session $j$ is determined by the maximum queue length and guaranteed service rate for the session. Clearly both queue length and guaranteed service rate are determined by the arrival process of all sessions. It has been shown in [3] that the upper bound of the session $j$ delay $D_j^*$ and queue length $Q_j^*$ are achieved for GPS server when $C_j \geq r$ and every session is greedy starts at the beginning of a system busy period. If $D_j^*$ is the maximum delay and $Q_j^*$ is the maximum backlog for session $j$, then

$$Q_j^* = \max_{(A1,\ldots,AN)} \max_{\tau \geq 0} Q_j(\tau) \qquad where \quad Q_j(\tau) = A_j(0,\tau) - S_j(0,\tau) \qquad (2\text{-}5)$$

$$D_j^* = \max_{(A1,\ldots,AN)} \max_{\tau \geq 0} D_j(\tau) \quad where\, D_j(\tau) = \inf\{t \geq \tau : S_j(0,t) = A_j(0,\tau)\} - \tau \quad (2\text{-}6)$$

where $S_j(\tau,t)$ is the amount of session *j* traffic served in the interval $(\tau,t]$ and $D_j(\tau)$ is the session *j* delay at time $\tau$. $A_k$ is arrival function of session *k* where $k=1,2,...,N$. The system busy period is defined as a maximal interval B such that

$$\sum_{j=1}^{N} S_j(\tau,t) = (t-\tau)r \qquad \text{for any } \tau \leq t \text{ and } t \in B \qquad (2\text{-}7)$$

Note that the definition of system busy period here is identical to that given for LR servers in Chapter 4.

As stated above, GPS is an attractive scheduling mechanism due to its following features:

- Flexibility in treating application sessions differently by varying the $\phi_j$'s without degrading service to other sessions to which different $\phi_j$ have been assigned.
- Better and fairer utilization of output bandwidth is achieved in GPS due to its work conserving characteristic. The guaranteed service rate and actual service rate of each session is proportional to its assigned positive number $\phi_k$ where $k \in B(\tau)$.
- The delay bound of an arriving session *j* bit only relates to its own queue length and is independent of the arrivals and queues of other sessions.
- Worst-case network queuing delay guarantees (upper bound) can be provided if the traffic sources are leaky bucket constrained. This upper bound is given by (2-7)

Although there are many attractive advantages in GPS, there are also some drawbacks to it. The significant drawbacks of GPS are that it can not transmit packet as entities, it assumes that backlogged sessions can be served simultaneously and that traffic is infinitely divisible. These drawbacks make GPS impractical. Therefore, in next section, some scheduling mechanisms that approximate GPS proposed are reviewed.

#### 2.3.2.2 WEIGHTED FAIR QUEUE (WFQ), WORST CASE FAIR WEIGHTED FAIR QUEUING ($WF^2Q$) AND $WF^2Q+$

The problem of approximating GPS in packet switched networks has attracted considerable attention in the literature, and many approaches have been proposed [2] [3] [12] [18] [28] [40] [48] Among them, the one that is best known is WFQ and its variations $WF^2Q$ and $WF^2Q+$.

At a work-conserving server of a realistic packet system, only one session at a time can receive service, and a packet can be served only after the previous packet has been served. WFQ is a work-conserving server, due to its property of serving packets from all backlogged sessions when the server is idle. In WFQ the finish time of packets in the corresponding GPS system is used to decide the packet service order. If there are $N_\tau$ sessions are backlogged at time $\tau$ and the server is ready to transmit the next packet, then from all backlogged sessions, the packet with the smallest finish time will be served. Let $F_p$ and $F_p^w$ be the time at which packet p will finish service under GPS and WFQ respectively, an important result established by Parekh [3] is that the delay bound provided by WFQ is within one packet transmission time difference of that by GPS. It can be presented as

$$F_p^W - F_p \leq L_{max}/r \quad (2\text{-}8)$$

Where $L_{max}$ is the maximum packet size and $r$ is the rate of the server. This feature makes WFQ a reference server model for the guaranteed service class in IntServ [47]. However, this does not mean that the WFQ scheduling mechanism and GPS provide almost identical service with only a difference of one packet. In a GPS system, there may exist N maximum size packets that finish service simultaneously at time $\tau$ and no matter how perfectly the GPS system is tracked, there is the possibility, due to the arbitrary service order in packet based WFQ systems, that

$$F_p - F_p^W = (N_\tau - 1)L_{max}/r + L_{max}/(r*\phi)$$

where $\phi$ is the weight of the session concerned. That is

$$F_p - F_p^W \geq (N_\tau - 1)L_{max}/r \quad (2\text{-}9)$$

This means that the time at which a packet departs from WFQ may actually $(N_\tau-1)$ maximum packet transmission times earlier than from a GPS system. This inaccuracy of WFQ in approximating GPS can have a significant negative impact on the QoS of real time service in terms of delay variance when a link is shared by a large number of backlogged sessions. Consider the example where 2000 backlogged sessions sharing a 100Mbps link with a maximum packet size of 1500 bytes. For a real time session reserving 20% ($\phi$=20%) of the link bandwidth, according to (2-9) the packet of this session may have a delay variance of 155ms at one switch node.

Implementation complexity is another drawback of WFQ, because, implement WFQ, it is necessary to track the progress of GPS. The concept of system virtual time and virtual time is proposed for this purpose [3]. There are three virtual times are used in tracking the progress of packets being served in GPS. System virtual time V(t), virtual start time of S and finish time F of a packet. S is the time packet begins to be served if the system is GPS and F is the time that the service is completed. System virtual time is used to update virtual start and finish time of a packet in the system when there is an event of packet arrival or departure to occur. System virtual time V(t) is defined in [3] as

$$V(t)=\begin{cases} 0 & \textit{when server is idle} \\ V(t_{i-1})+\dfrac{\tau}{\sum_{k\in B_i}\phi_k} & \tau = t-t_{i-1} \textit{ and } \tau \le t_i - t_{i-1}, i=2,3,\ldots \end{cases} \tag{2-10}$$

where $B_i$ is the set of sessions that are busy in the time interval $(t_{i-1},t_i)$ when the event of the $i^{th}$ arrival to or departure[1] from GPS occurs. Based on (2-10), the calculation of virtual finish time of a packet is given in as follows:

if the $i^{th}$ packet of session j arrives at time $t_j^i$,

$$F_j^i = S_j^i + \frac{L_j^i}{\phi_j} \quad \textit{where } S_j^i = \max\{F_j^{i-1}, V(t_j^i)\} \tag{2-11}$$

The implementation of WFQ is based on the virtual time function (2-11). When a packet arrives, the system virtual time is updated and the virtual finish time is stamped to it. The packets in the system are sorted based on their virtual finish

[1] The convention that a packet has arrived or left only after its last bit has arrived or left has been adopted in this dissertation

time and picked up by the server in an increasing order of time stamp, i.e. the packet with the smallest virtual finish time is to be served first. This has a complexity of O(N)[2] [3] [56] where N is the number of sessions sharing the link. Under the worst case the scheduling mechanism needs to process N events (arrivals or departures) for a single scheduling decision, which makes WFQ difficult to be implemented at high speed.

To diminish the inaccuracy and complexity of WFQ, $WF^2Q$ and a further refinement $WF^2Q$+ are proposed by Bennett and Zhang [28][26]. $WF^2Q$ uses the Smallest Eligible Virtual Finish time First (SEFF) policy to schedule packets in the session queue. $WF^2Q$ selects the next packet to transmit at time $\tau$ only from packets that have started service in the corresponding GPS system. A packet is said to be eligible at time $\tau$ if its virtual start time is no greater than the current system virtual time. The $i^{th}$ packet of session $j$ is eligible at time $\tau$, if only if

$$S_j^i \le V(t_j^i + \tau) \tag{2-12}$$

By the use of both virtual start time and virtual finish time, the $WF^2Q$ scheduling mechanism achieves a more accurate emulation of GPS. During any time interval, the difference between the amount of traffic transmitted by GPS and $WF^2Q$ is within one packet size. Like WFQ, $WF^2Q$ still possesses a implementing complexity of O(N).

$WF^2Q$+ further improves the performance of WFQ, by using a new virtual time function and a simplification of virtual start and finish time, to reduce its implementing complexity from O(N) to an overall complexity of O(logN) [27].

The major task associated with $WF^2Q+$ implementation is computing the system virtual time function and maintaining the set of eligible sessions sorted by virtual finish time. Differing from $WF^2Q$, $WF^2Q+$ uses a new virtual time function which is given in [27] as

$$V_{WF^2Q+}(t+\tau) = \max\{V_{WF^2Q+}(t)+\tau,\ \min_{j \in B(t)}(S_j^{h_j(t)})\} \quad (2\text{-}13)$$

where $B(t)$ is the set of backlogged sessions at time $t$, $h_j(t)$ is the sequence number of the packet at the head of the session $j$'s queue and $S_j^{h_j(t)}$ is the virtual start time of the packet at the head of the session $j$ queue.

In both WFQ and $WF^2Q$, virtual start and finish times need to be maintained on a per packet basis. In $WF^2Q+$, however, only one pair of virtual start and finish times is maintained and is updated whenever a new packet reaches the head of the queue. The updating of the virtual start time and virtual finish time is also given in [27] as

$$S_j = \begin{cases} F_j & Q_j(a_j^i-) \neq 0 \\ \max(F_j, V(a_j^i)) & Q_j(a_j^i-) = 0 \end{cases}$$
$$F_j = S_j + \frac{L_j^i}{r\phi_j} \quad (2\text{-}14)$$

where $a_j^i$ is the arrival time of the $i^{th}$ packet of session $j$ and $Q_j(a_j^i-)$ is the queue length of session j just before time $a_j^i$. $L_j^i$ is the packet length of the packet at the head of the session j queue, and $\phi_j$ is the weight of session $j$. The work of updating system virtual time and sorting virtual finish times among eligible sessions in $WF^2Q+$ has a complexity of O(logN) [23]. It should be noted that $WF^2Q+$ has not only the same properties as $WF^2Q$ in terms of fairness and delay bound, but also has significantly lower complexity.

#### 2.3.2.3 LATENCY RATE SERVER (LR-SERVERS) FRAMEWORK

Future IP networks are likely to be heterogeneous in terms of switches (routers, gateways), and hence a variety of scheduling mechanisms may be employed in these switches.

In this section, a general model for scheduling mechanisms, the Latency Rate Server, is introduced. The Latency Rate (LR) Server, or simply LR servers, is not an individual scheduling mechanism but a class (or a category) of scheduling mechanisms. LR servers were first proposed by Stiliadis [13] as a general model for the analysis of a broad range of scheduling mechanisms employed in a network and, in particular, for studying the worst case delay behaviour of individual sessions.

The key feature of the theory of LR servers is it uses session busy period to measure service received by the session and identify scheduling mechanisms that belong to LR servers by comparing the average service rate received with the rate reserved for the session during the session busy period. Please note that the session busy period used in LR Server theory by Stiliadis is different from that used in [3]. A session *i* busy period, as defined in [13], can be interpreted as the maximum time interval during which the session is continuously backlogged assuming it only receives reserved service rate. It can be taken as the worst case if a session can only receive its reserved service rate, as far as the packet loss and delay of the session is concerned, when there are more than one backlogged sessions attended by a server in a work conserving manner. It is also pointed out in [13] that when same traffic distribution is applied to two different scheduling mechanisms, the session busy period remains constant if the service rate reserved for the session is identical. This is the fundamental reason for introducing the session busy period which makes it possible to analyse the performance of different scheduling mechanisms. The theory of LR server is based on the session busy period, with LR servers defined as follows:

*A scheduling mechanism is an LR server if there exists an non negative number $\theta$ such that the following inequality hold for all times t from the start of the $j^{th}$ busy period of session i till all packets that arrived during this period are served and vice versa. That is*

$$S_{i,j}(\tau,t) \geq \max(0, \mu_i * (t-\tau-\theta))$$

*where $\theta$ is the minimum non-negative number that satisfies the above inequality and $S_{i,j}(\tau,t)$ is the service received by the traffic of session j that arrived during time interval of $(\tau,t]$. $\mu_i$ is the service rate reserved for session i. $\theta$ is also called latency of the LR server.*

It can be proved that the θ in the above inequality is the worst case delay seen by the first packet of each backlogged session of the LR server.

Proof:

If there exists a non-negative number $\theta^*(\theta^*>\theta)$ such that the first packet of a backlogged queue of session *v* is served on or after time $\theta^*$after the arrival time $t^*$ of the packet. Since the server is LR server, then from the definition the LR server above we have

$$S_v(\tau, t^* + \theta) \geq \mu_v(t^* - \tau)$$

$$S_v(\tau, t^* + \theta^*) \geq \mu_v(t^* - \tau) + \mu_v(\theta^* - \theta)$$

We also know that the packets that served of session $v$ at time $t^*+\theta$ are identical to that at $t^*+\theta$ (please note that the queue of session $v$ is empty right before $t^*$ and no packet is served after $t^*$). Therefore it is also observable that

$0 \geq \mu_v(\theta^* - \theta)$ which contradicts with the assumption of $\theta^*>\theta$.

There are two important findings regarding LR servers by Stiliadis [13]. One is the derivation of upper bounds on end to end delay which extended the work of [3][57] to accommodate a broad range of scheduling algorithms in an arbitrary ways. This derivation is based on the assumption of leaky bucket $(\rho_j, \sigma_j, \infty)$ constrained traffic arrivals. The other is the derivation of zero loss buffer requirements are derived for the worst case. Within a single LR server, these two findings can be described as:

$$D_j \leq \frac{\sigma_j}{r_j} + \theta_j \qquad (2\text{-}15)$$

$$B_j \leq \sigma_j + \rho_j \theta_j \qquad (2\text{-}16)$$

where $D_j$ is the maximum delay of any packet of session $j$ in the LR server, $B_j$ is the zero loss buffer requirements for session $j$. $\theta_j$ is the latency of session $j$ at the LR server. With the use of (2-15) and (2-16), one can have a good understanding of the worst case delay behaviour of an individual session in a network with heterogeneous scheduling mechanisms.

From the above introduction, it is clear that LR servers are a general representation of fractional service rate reserved mechanisms. Chapter 4 addresses key LR server issues, in particular the worst case average loss rate.

## 2.4 PERFORMANCE OF QOS MECHANISMS: KEY ISSUES

This section summarises the previous ones, and outlines key issues not yet addressed in the literature.

The Internet is evolving rapidly with an increasing number of applications with diverse requirements. As discussed in section 2.2, service differentiation and QoS guarantees are the basic requirements for future IP networks. QoS mechanisms will play an important role in controlling the amount of network resources that each service class can consume, and will provide guaranteed QoS by minimising the packet loss rate and end-to-end delay. While there is significant work on the analysis of various scheduling mechanisms, especially for delay and delay variation, there are still issues regarding the comparative merits and loss behaviour of the various scheduling mechanisms that are not fully understood. In particular, two key issues are considered in the following chapters. One is how well do the scheduling mechanisms which support Diffserv providing various QoS levels. This issue will be addressed in Chapter 3 via a quantitative comparison, which shows the comparative merits of the scheduling mechanisms that support Diffserv. In particular, the work will focus on their suitability in the DiffServ environment and will develop analysis techniques. The other issue is the performance behaviour of the various QoS mechanisms, particularly the loss rate under the worst case scenario when the input buffer of the server is finite and determining the arrival process that will cause the maximum average loss rate. This issue is a critical dimensioning issue which is not well understood in the literature. To address this issue, chapter 4 focuses on the performance analysis of a general model of QoS mechanisms -- LR servers.

# Chapter 3

# Performance Analysis of Non-Fractional Service Rate Reserved Scheduling Mechanisms--Threshold Dropping and Priority Scheduling

## 3.1 INTRODUCTION

According to the classification of the QoS scheduling mechanisms in Chapter 2, most Differentiated Services (DiffServ) mechanisms are non-fractional service rate reserved scheduling mechanisms. DiffServ was proposed as an alternative for Integrated Service (IntServ) with simplified scheduling mechanisms and protocols. The DiffServ QoS architecture relies on the definition of a limited set of local behaviors which are referred to as Per Hop Behaviors (PHBs). Best effort is the default PHB in DiffServ. The most recent IETF DiffServ working group focuses mainly on two PHBs, Assured Forwarding (AF)[24] and Expedited Forwarding (EF)[53].

In Assured Forwarding, IP packets are classified as belonging to one of N traffic classes (e.g. N=4). Within a traffic class, a packet is assigned with a level of drop precedence such as green, or yellow or red. In case of congestion, packets with lower precedence, e.g. red, will be dropped first. AF PHBs can thus provide different levels of forwarding assurance for IP packets. AF packet can expect to be forwarded with a high probability, as long as the traffic does not exceed its service profile (subscribed information rate). It is proposed that the implementation of AF uses an active queue management mechanism, such as TD or RED [24].

Expedited Forwarding, also called the Premium Service Scheme, improves on the current best effort service with low loss, low latency and jitter, and assured bandwidth. EF traffic should, as suggested in [53], receive a predefined service rate independent of the intensity of any other traffic attempting to transit the node. Priority scheduling is suggested for implementing EF.

Although there are some other QoS mechanisms proposed in the literature that can be used in implementing AF and EF, TD and PS are the two fundamental QoS mechanisms from which the others are derived. Therefore, in this chapter, the focus has been on the performance analysis and subsequent comparison of these two mechanisms.

## 3.2 Analysis of Threshold Dropping

The Threshold Dropping mechanism forms the basis of QoS mechanisms such as RED and RIO. It provides differential service to applications by assigning different dropping precedences (discard thresholds) to traffic flows. A threshold dropping mechanism is depicted in Figure 3-1.

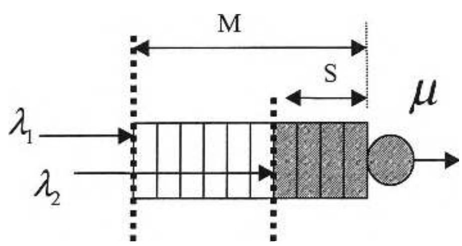



**FIGURE 3-1. THRESHOLD DROPPING MECHANISM WITH TWO PACKET FLOWS**

In Figure 3-1, two arrival flows are considered: preferred flow and non-preferred flow. The preferred flow consists of packets tagged as in profile (i.e. which do not violate their traffic contract) and the non-preferred flow consists of packets tagged as out of profile. Preferred flow should receive preferential treatment with respect to the non-preferred flow. This is achieved in TD by assigning a threshold S. Non-preferred flow packets which arrive to the system when the queue length exceeds S are dropped. On the other hand, preferred flow packets are only discarded when the queue length reaches the buffer size M.

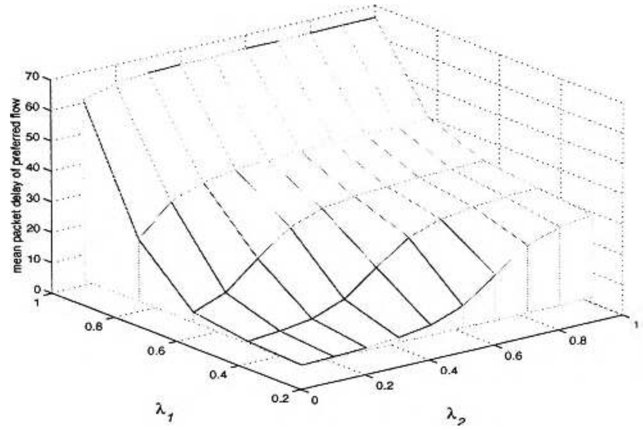



FIGURE 3-2-1 MEAN PACKET DELAY BEHAVIOURS OF PREFERED FLOW IN TD MECHANISM UNDER VARIOUS LOADS FROM BOTH FLOWS. (BUFFER SETTINGS: M=100, S=30) $\lambda_1$ IS FOR PREFERRED FLOW AND $\lambda_2$ IS FOR NON-PREFERRED FLOW.

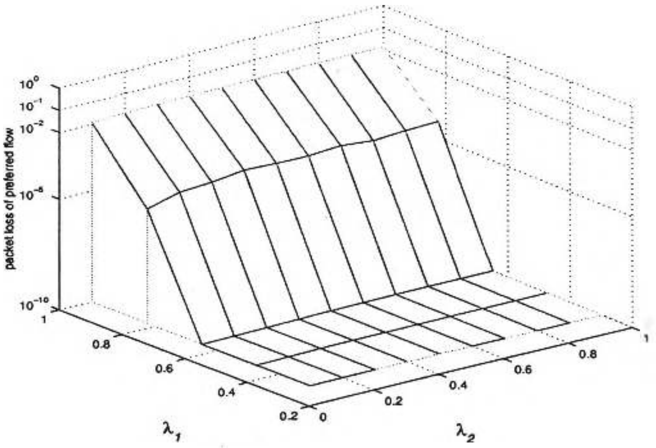



FIGURE 3-2-2 PACKET LOSS BEHAVIOURS OF PREFERED FLOW IN TD MECHANISM UNDER VARIOUS LOADS FROM BOTH FLOWS. (BUFFER SETTINGS: M=100, S=30) $\lambda_1$ IS FOR PREFERRED FLOW AND $\lambda_2$ IS FOR NON-PREFERRED FLOW.

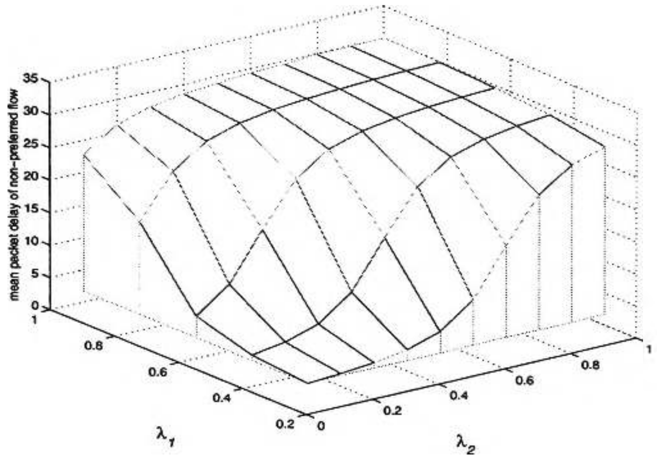



FIGURE 3-2-3 MEAN PACKET DELAY BEHAVIOURS OF NON-PREFERED FLOW IN TD MECHANISM UNDER VARIOUS LOADS FROM BOTH FLOWS. (BUFFER SETTINGS: M=100, S=30) $\lambda_1$ IS FOR PREFERRED FLOW AND $\lambda_2$ IS FOR NON-PREFERRED FLOW.

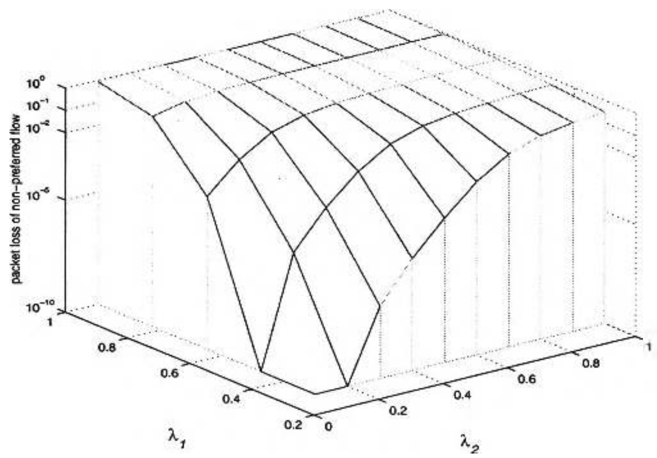



FIGURE 3-2-4 PACKET LOSS BEHAVIOURS OF NON-PREFERED FLOW IN TD MECHANISM UNDER VARIOUS LOADS FROM BOTH FLOWS. (BUFFER SETTINGS: M=100, S=30) $\lambda_1$ IS FOR PREFERRED FLOW AND $\lambda_2$ IS FOR NON-PREFERRED FLOW.'

A key consideration is the loss and delay arising from various differential loads and discard thresholds.

Simulations are designed in ARENA®to look at the loss and delay behaviors of both preferred and non-preferred flows. The module contains a single server with two input queues, preferred and non-preferred queues. Packets arrived to the queues according to Poisson and the packet service time at the server is exponentially distributed. The arrival rates for both flows are varied using different rate to generate packet in the simulation module. Packets arrived in the queue is counted until the queue is full. The simulation module in ARENA can be depicted in figure3-2-5.

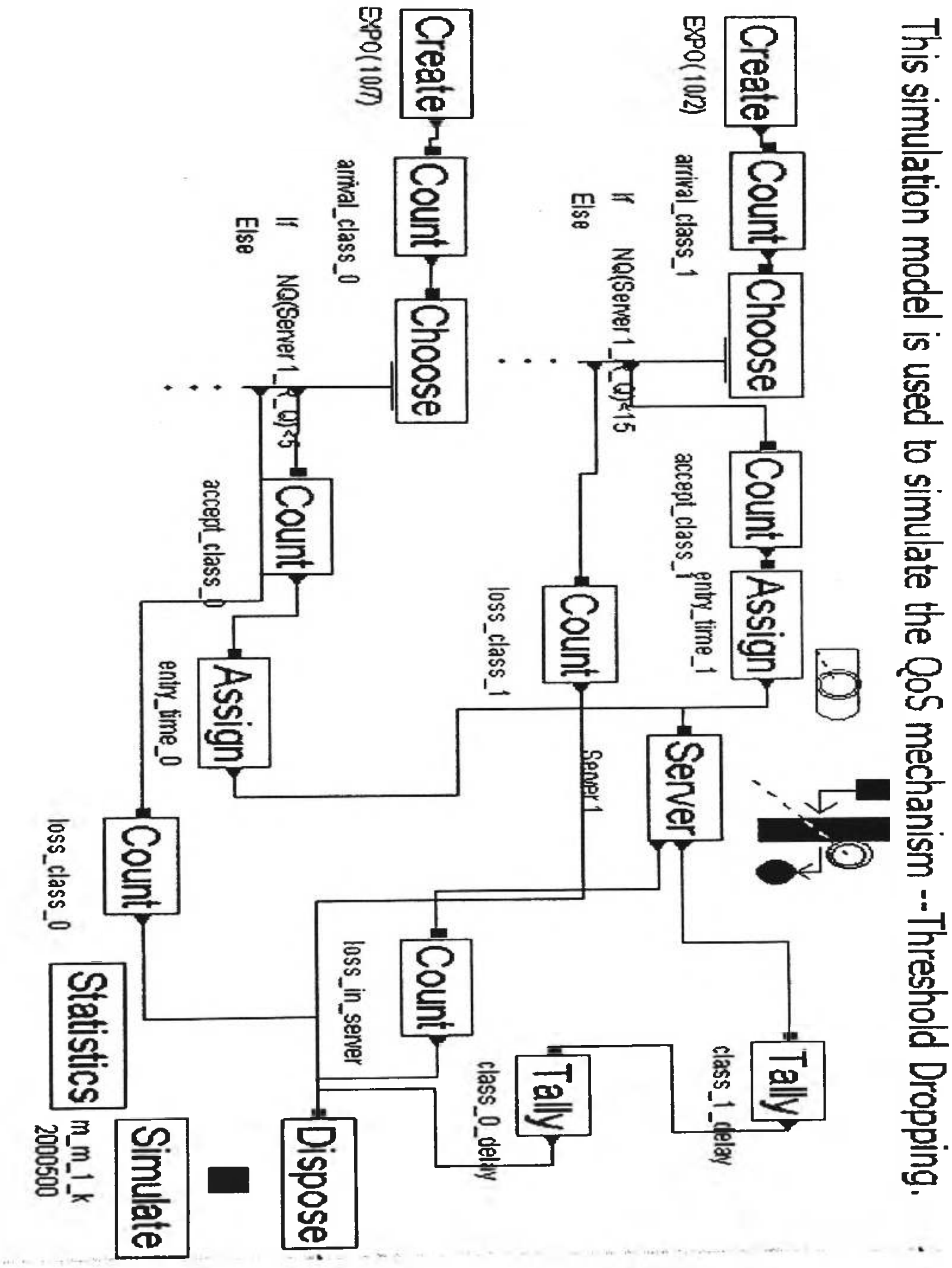
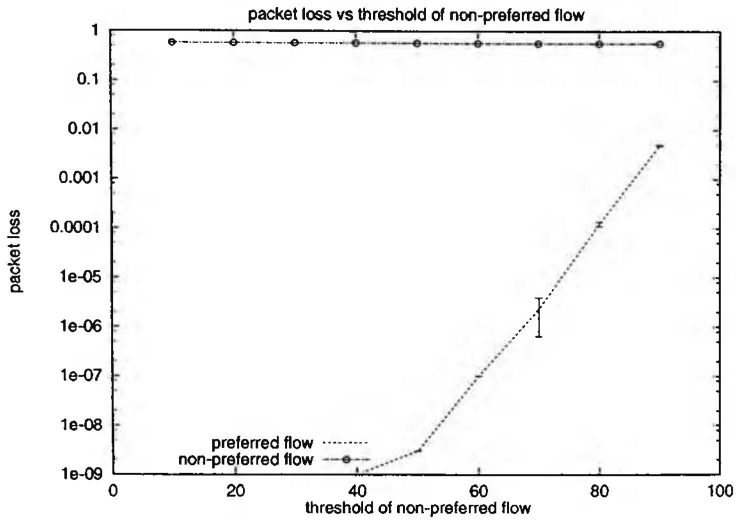


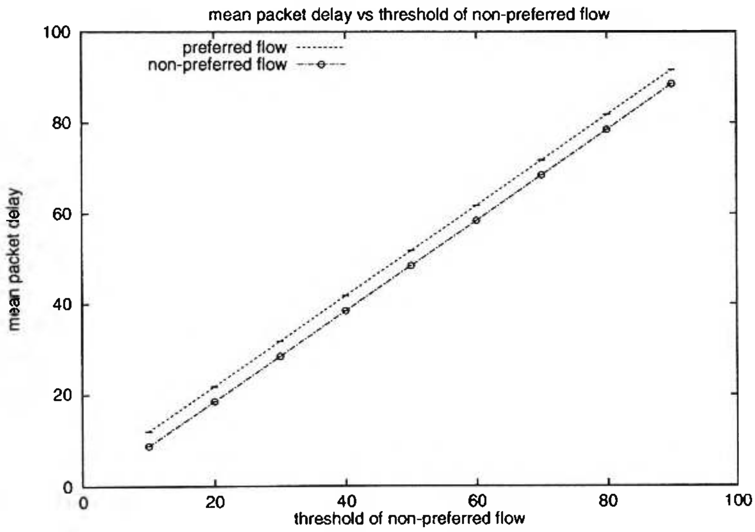

**Figure 3-2-5** Simulation model of Threshold Dropping Mechanism

Figures 3-2-1 to 3-2-4 showed simulation results for the TD mechanism under various loads. These results were obtained assuming that preferred and non-preferred flows followed a Poisson distribution with mean arrival rates $\lambda_1$ and $\lambda_2$ (both arrival rate have been normalised with respect to service rate), respectively. The packet service time was assumed to be exponential. The mean packet delay is normalised with respect to service time. In Figure 3-2, packet loss and mean packet delay are shown as a function of $\lambda_1$ and $\lambda_2$. In this figure the buffer size is set to M = 100 and the threshold is set to S = 30. As expected, increasing the load of the non-preferred flow has little effect on packet loss experienced by the preferred flow. The mean packet delays of both flows are bounded by their respective discard thresholds.

Figure 3-3 shows the impact of threshold S on packet loss and mean packet delay of the preferred and non-preferred flows. In this figure both flows had a fixed load of 0.7, the total buffer size was set to M = 100 and the threshold value S was varied from 10 to 90. Under the above conditions, increasing the threshold value results in little improvement in packet loss of the non-preferred flow. However, packet loss of the preferred flow increases sharply as the threshold is increased beyond 50. Increasing the threshold leads to a linear increase in the mean packet delay for both flows.




(a)




(b)

**FIGURE 3-3. IMPACT OF THRESHOLD OF NON-PREFERRED FLOW ON PACKET DELAY AND LOSS**

## 3.3 ANALYSIS OF PRIORITY SCHEDULING

Priority Scheduling (PS) is a QoS mechanism which could potentially form the basis of a Diffserv EF implementation. This however requires that the highest priority flow in PS receives a guaranteed forwarding rate, independent of the intensity of other flows (a key requirement for EF [53]). This section examines PS performance, to determine its suitability in an EF environment.

A Priority Scheduling mechanism handling two packet flows is depicted in Figure 3-4. Packets belonging to the preferred flow receive non-preemptive priority over packets belonging to the non-preferred flow. Buffer sizes for the preferred and non-preferred flows are set to K and L, respectively.

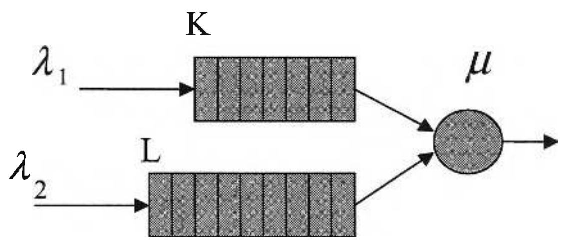
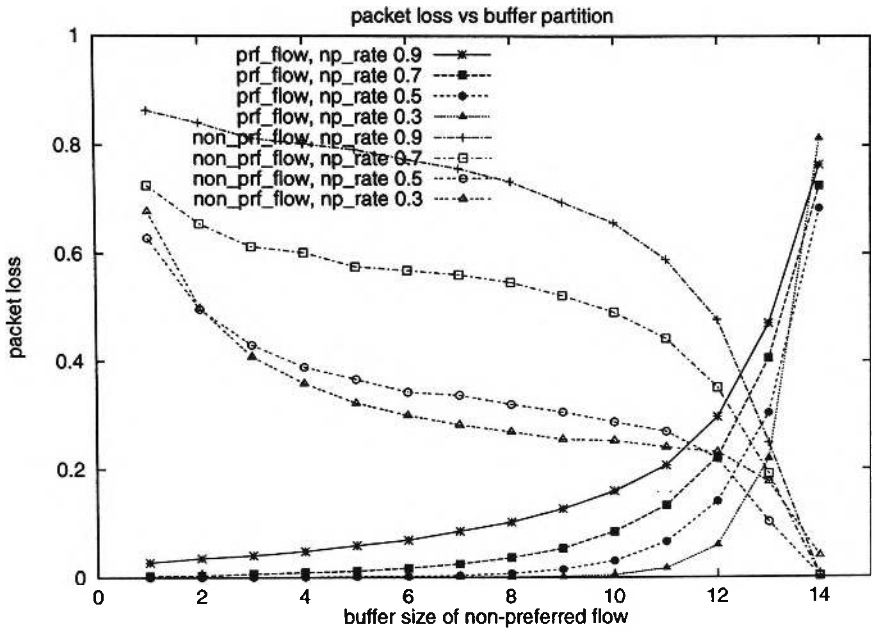



**FIGURE 3- 4 PRIORITY FINITE QUEUES**

We will first investigate the impact of buffer partitioning between preferred and non-preferred flows while keeping the overall buffer size constant, assuming that preferred and non-preferred flows are Poisson distributed with mean arrival rates $\lambda_1$ and $\lambda_2$, respectively. Packet service time is assumed to be exponential. The mean packet delay is normalised with respect to service time. The total buffer size is set at 15.

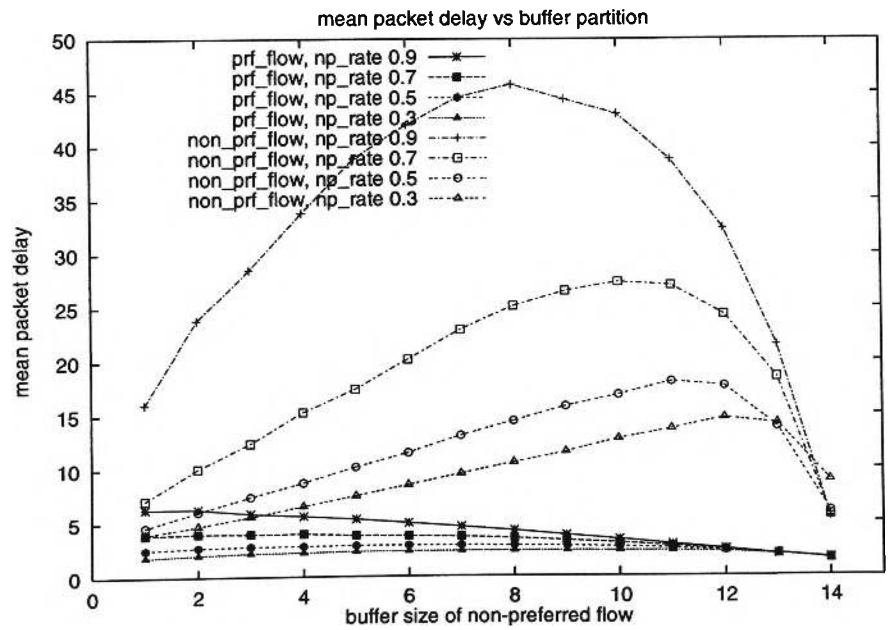
Figure 3-5 shows typical packet loss and mean packet delay behaviour for preferred and non-preferred flows as a function of buffer space allocated for non-preferred traffic. The results show a clear trade-off between packet loss and mean packet delay for preferred and non-preferred flows when the buffer allocation is changed.

The cause of the trade-off is the way of changing buffer size. The over all buffer size is fixed and the changing of the buffer size for one flow will automatically change the buffer allocation to the other. When buffer size for non-preferred flow is increased, the buffer size for preferred flow is automatically reduced. So the packet loss for preferred flow is increased accordingly when the packet loss rate for non-preferred flow declined.




(A)

**FIGURE 3-5 (A) : PACKET LOSS VS BUFFER PARTITION FOR VARIOUS VALUES OF NORMALISED ARRIVAL RATE $\lambda_2$ (np_rate in the figure) OF NON-PREFERRED PACKETS. NORMALISED ARRIVAL RATE OF PREFERRED PACKETS $\lambda_1$ IS SET TO 0.7**




(B)

**FIGURE 3-5(B): MEAN PACKET DELAY VS BUFFER PARTITION FOR VARIOUS VALUES OF NORMALISED ARRIVAL RATE $\lambda_2$ (np_rate in the figure) OF NON-PREFERRED PACKETS. NORMALISED ARRIVAL RATE OF PREFERRED PACKETS $\lambda_1$ IS SET TO 0.7**

In Figure 3-5, mean packet delay curves for non-preferred flow show interesting behavior when the buffer space allocated to non-preferred traffic is varied. The mean packet delay for non-preferred flow is small when the buffer space allocation is either small (less than 2) or large (more than 12). This is because when the allocated buffer size is small, the mean delay is bounded by the small buffer size. When more buffer space is allocated to non-preferred flow, however, the buffer space left for preferred flow will be decreased due to the constant total buffer size. Under this scenario, packets from the non-preferred flow will spend less time waiting for the queue of the preferred flow to become empty. This behavior is due to the fact that we ignore packet re-transmission in our simulation and only consider the mean delay of those packets that were not dropped from the queue.

## 3.4 COMPARISON OF THRESHOLD DROPPING AND PRIORITY SCHEDULING MECHANISMS

As discussed in the introductory section of this chapter, TD and PS can be regarded as basic scheduling mechanisms from which the other mechanisms have been derived. Hence the comparative performance of these two mechanisms is an important issue. TD and PS have been analyzed in the literature [49], but the comparison is based on packet loss probability for the non-preferred flow. However our performance comparison of the TD and PS mechanisms aims to provide a constant packet loss to the preferred flow. Our comparison allows us to determine the associated loss rate for the non-preferred flow and the mean packet delay for both the preferred and non-preferred flows.

We set the two mechanisms with the same total buffer space of 15 packets and the link capacity. As in earlier tests the preferred and non-preferred flows were modeled as Poisson processes. For given arrival rates of both flows, we varied the threshold $S$ in the TD mechanism and the buffer size $K$ in the PS mechanism until the same level (precise to $10^{-5}$) of loss probability for the preferred flow was obtained from both mechanisms. We then compared the resulting packet loss of the non-preferred flow and the mean packet delay of both flows between these two mechanisms.

The packet loss and mean packet delay results are shown in Figure 3-6 and Figure 3-7, respectively. The mean packet delay is normalized with respect to service rate. The normalized arrival rate of the non-preferred flow in both figures is 0.7.

The results of Figure 3-6 indicate that the TD mechanism has better performance in terms of packet loss for the non-preferred flow when the load of the preferred flow is light. When the load is heavy the difference in packet loss between the two mechanisms is negligible. The results of Figure 3-7 indicate that as the load of the preferred flow changes, the PS mechanism provides a smaller mean delay to the referred flow than does the TD mechanism. However, the TD mechanism results in a smaller mean delay for the non-preferred flow.

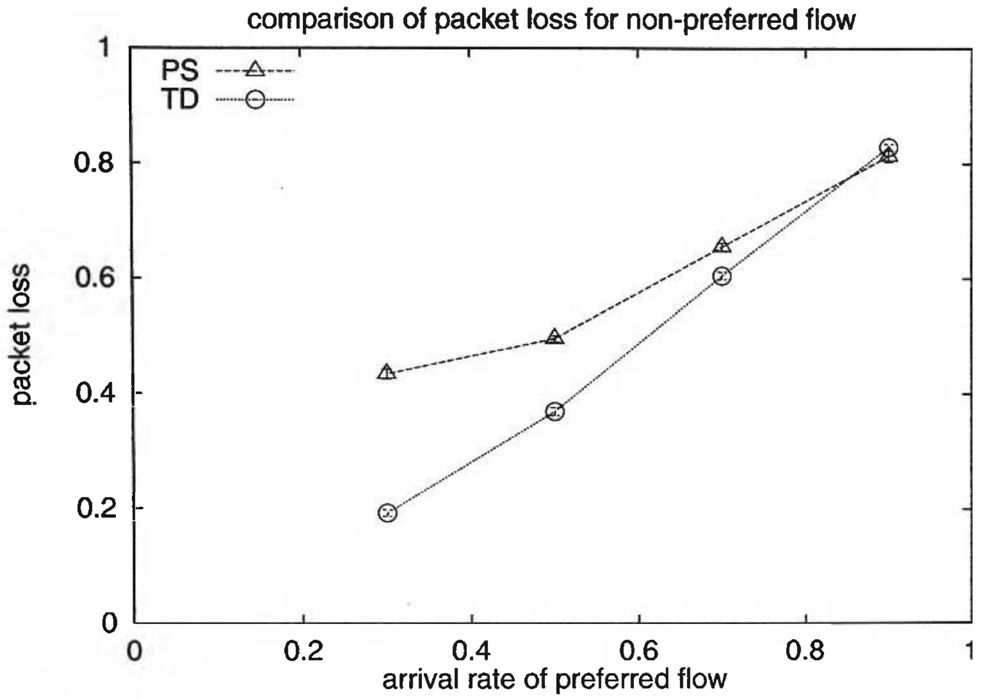



**FIGURE 3-6 PACKET LOSS COMPARISON**

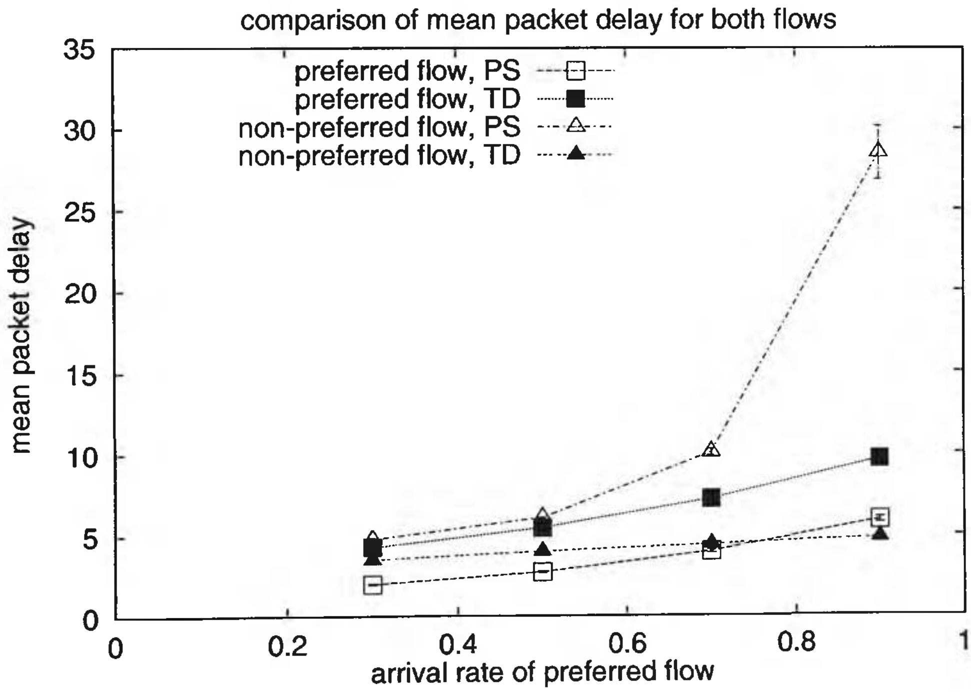


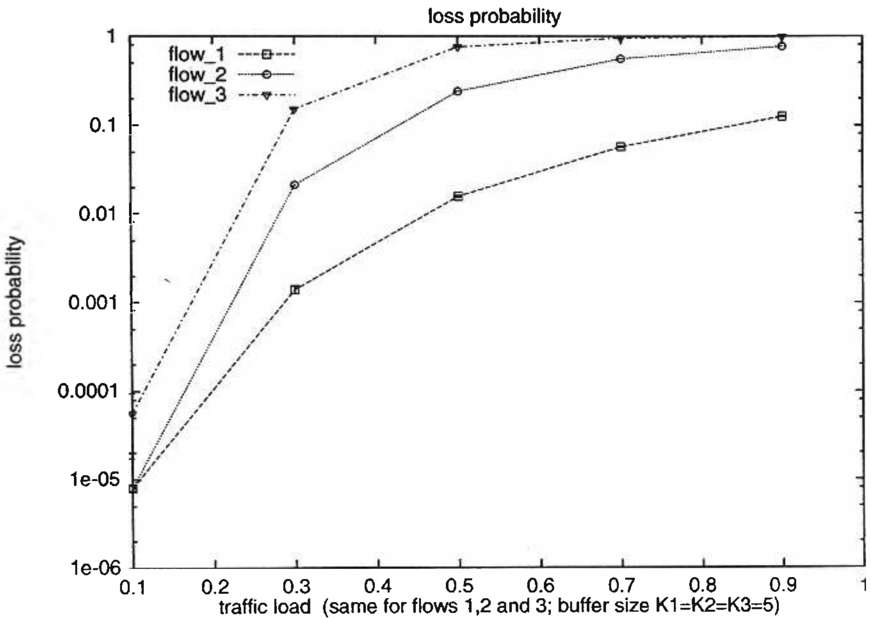

**FIGURE 3-7 MEAN PACKET DELAY COMPARISON (THE MEAN PACKET DELAY IN THE FIGURE IS NORMALISED WITH RESPECT TO SERVICE RATE)**

## 3.5 PERFORMANCE OF PS WITH THREE TRAFFIC FLOWS

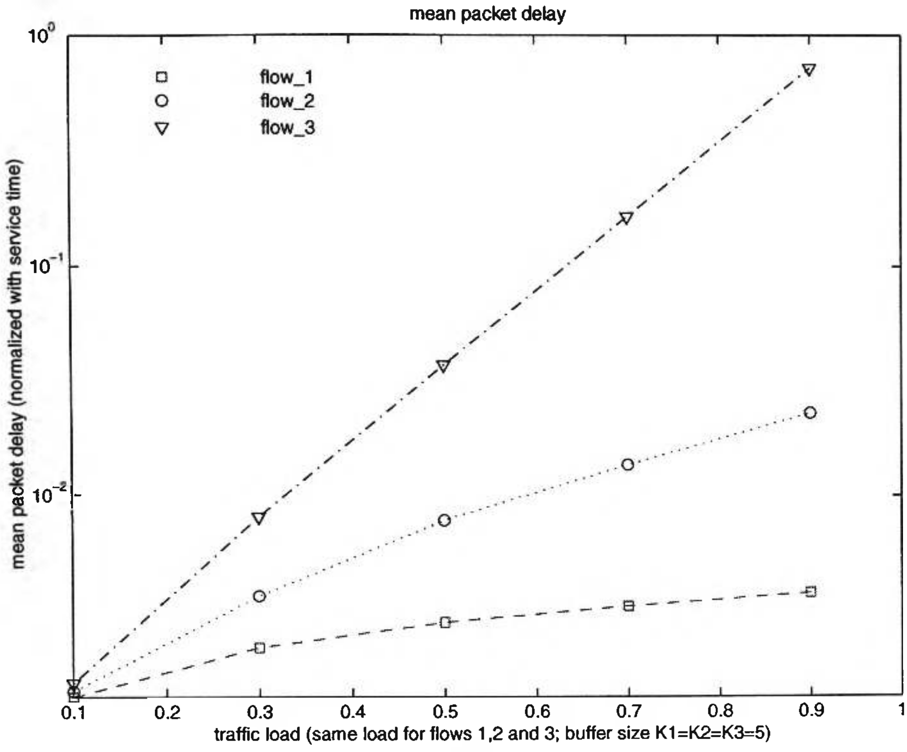
A key requirement for an EF flow is to maintain a specified departure rate from a Diffserv node, regardless of the intensity of other traffic flows [53]. Our previous analysis of TD (refer to figure 3-2-1) indicated that the mean packet delay of preferred flow increases when the load of the non-preferred flow increases. This TD feature does not therefore guarantee an EF traffic departure rate independent of other traffic flows' intensity. To investigate the effectiveness of PS in this regard, we have extended our PS simulation model to three traffic classes. A non-preemptive policy is used in the simulation. The order of priorities assigned to traffic flows, from high to low, is flow 1, flow 2 and flow 3. Packet service time was assumed to be exponential with mean of 1000 packets per service time unit. The traffic loads and mean packet delay are normalised with respect to service rate and service time respectively.

The simulation results, as shown in figure 3-8, indicate that the performance of the lowest priority flow deteriorates greatly as the load of the other flows increases. (Note the input buffers for flow 1, 2 and 3 are K1, K2 and K3 respectively).

Figure 3-9 indicates that changing the traffic load for an individual flow (such as flow 2) has no impact on the highest priority flow in terms of packet loss probability and mean packet delay. However, on the other hand, the impact on the flows with lower priority (e.g. flow 3) is significant. Figure 3-9 shows a higher packet loss rate of flow 2 than flow 3. The reason for this is the increased load for flow 2, compared to flow 3. Figure 3-10 shows that increasing the buffer size for all traffic flows will improve the performance of flows with higher priority in terms of packet loss. However, it degrades the performance of the lowest priority flow, for both packet loss and mean packet delay. From the simulation result we can conclude that flow 1 will meet the requirements for EF, whereas flow 3 would provide a best effort service only.

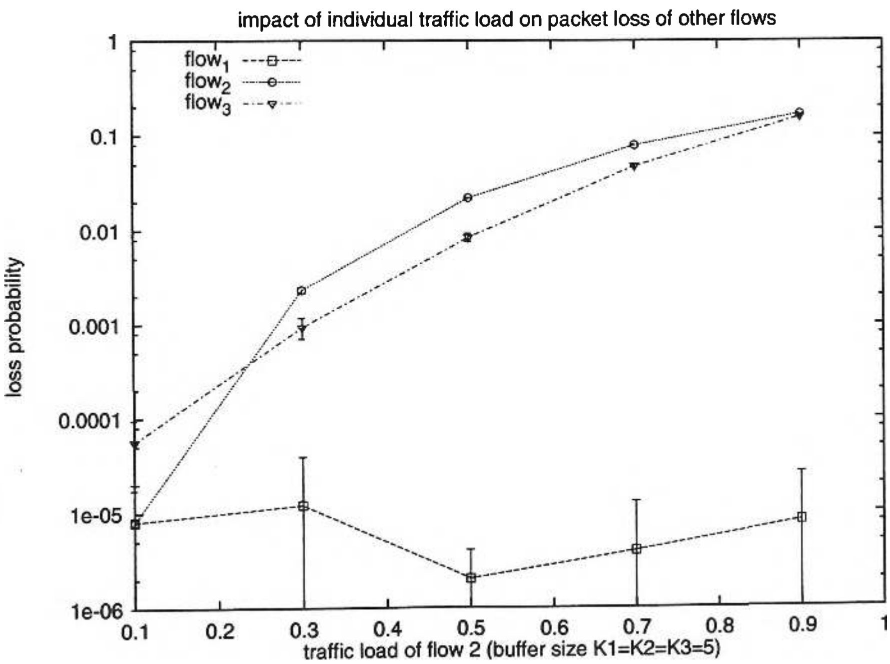



**(A)**

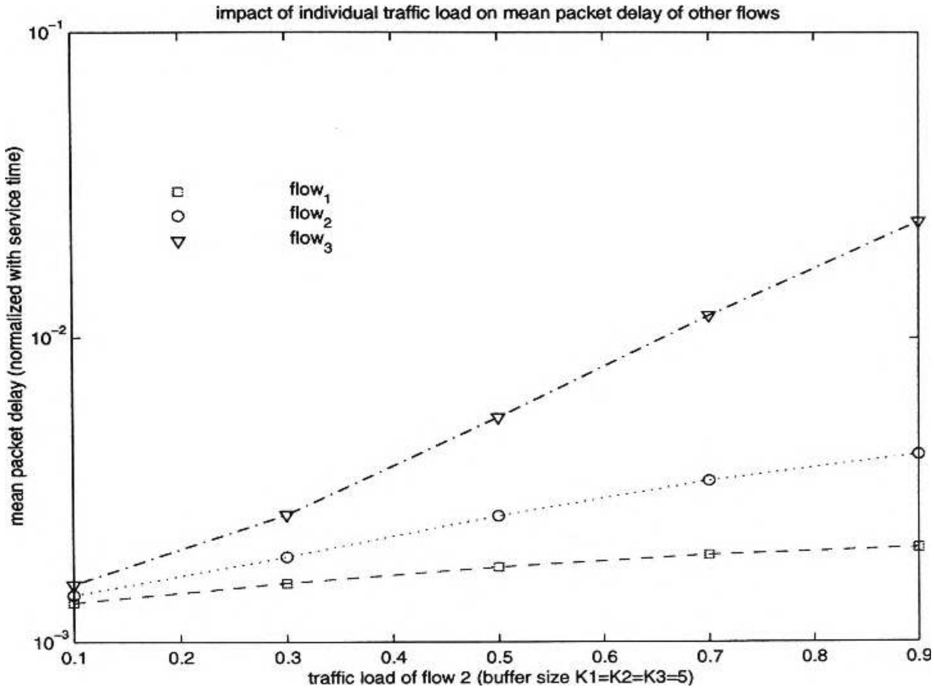



(B)

**FIGURE 3-8 PACKET LOSS AND MEAN PACKET DELAY OF PS MECHANISM VS TRAFFIC LOAD**

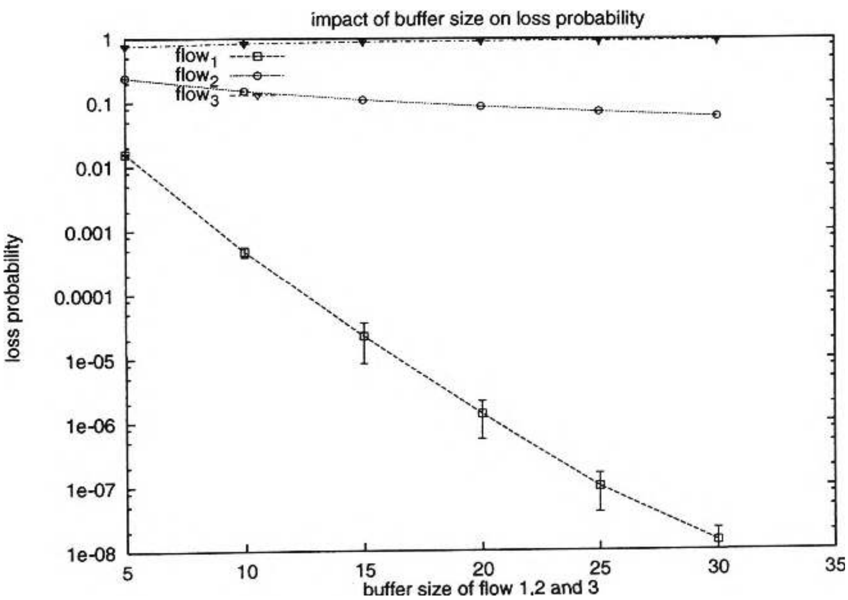



(A)

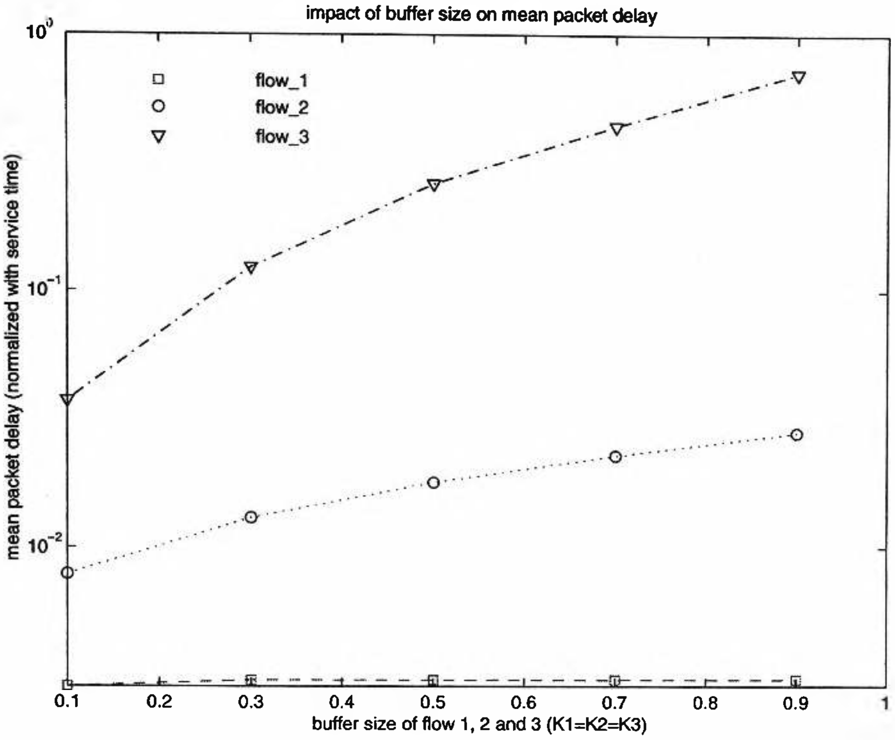



(B)

**FIGURE 3-9** IMPACT OF INCREASING TRAFFIC LOAD OF FLOW2 ON OTHER FLOWS (THE LOAD OF 0.1 IS FOR BOTH FLOW 1 AND 3)




(A)




(B)

**FIGURE 3-10. IMPACT OF INCREASING BUFFER SIZE ON THE PACKET LOSS AND MEAN PACKET DELAY**

Low loss, low latency and low jitter are key characteristics of EF. Therefore, the key issue in determining the scalability of a scheduling mechanism for EF in DiffServ is to examine the queue length of corresponding traffic flows. To address this issue, the next section continues investigations into PS performance based on non-preemptive priority queues, by developing an analytical PS model.

## 3.6 AN APPROXIMATE PS PERFORMANCE ANALYSIS

A key requirement for an EF implementation is to engineer mechanisms to provide high priority traffic with low loss, low delay and jitter [53]. In this section we continued our investigations into PS performance with an approximate method. As discussed in the previous section, Priority Scheduling (PS) is a potential mechanism for Expedited Forwarding (EF). The queue length of traffic flows will determine the performance of PS in terms of loss and delay. Non-preemptive finite priority queues arise naturally in practical networks. With the non-preemptive priority policy, a packet receiving service is allowed to complete its service without being interrupted, even if a packet of higher priority arrives in the meantime.

In [16], Bertsekas and Gallager derive the mean packet delay for non-preemptive priority queues with infinite buffers. A solution by Sahu *et al* [49] requires a knowledge of the service rate for each individual queue, which may not be known in practice. May *et al* [38] only provide a solution to the high priority queue. Both [49] and [38] assume a preemptive service policy. Blondia [6] provides a method to calculate the queue length distribution and waiting time distribution. However, this method is complicated due to the recursive formulas for computing the Laplace Transform of the busy period of the preferred flow and the blocking time of the non-preferred flow.

We propose an approximate method for obtaining packet loss and the mean packet delay for two traffic classes, using a non-preemptive priority finite queue mechanism. The basic idea of our approximation method is to decompose the joint queues in Figure 3-4 into two equivalent individual queues with derivable equivalent service rates. The results from the approximation method are then verified with simulations.

## APPROXIMATION METHOD FOR A PS MECHANISM WITH TWO SERVICE CLASSES

Consider a router deployed with non-preemptive finite priority queues sharing a single processor. Also assume that there are 2 classes of traffic (packet flows) with different service preferences where class one has non-preemptive priority over class two. Both two flows arrive at the router according to the Poisson process with an exponential distributed service time. The mean arrival rates of classes one and two are $\lambda_1$ and $\lambda_2$ respectively. A separate queue is maintained for each class. Since the buffers for both queues are finite (assuming buffer size $K$ is assigned to the queue with high priority and buffer size $L$ is assigned to the queue with low priority), all the packets that find queues full are dropped. Packet retransmission is not considered in this study. We intend to work out the packet loss probability and the mean packet delay experienced by both flows. The queuing model is illustrated in Figure 3-4.

We use the following notation:

$K$ buffer size of preferred flow.

$L$ buffer size of non-preferred flow.

$K_0$ a selected queue length on purpose.

$P_{i,j}$ probability that $j$ packets are found in queue $i$, where $i$=1,2.

$\mu$ service rate of the processor which is normalised to *1*

$\mu_i$ service rate for queue *i*

$\lambda_i$ mean arrival rate of flow *i*.

$\rho_i = \dfrac{\lambda_i}{\mu_i}$ utilisation factor for queue *i*

$\alpha_i$ rate that packets are accepted by queue *i*.

$NQ_i$ the average number of packets in queue *i*.

$R$ the mean residual time of the packet in the server

When the queue with high priority is K (buffer queue is full), all following packets from the flow that find the queue is full will be dropped and hence $\lambda_i = 0$ for the time when the queue is full. This situation also applies to the low priority queue. So after some time, the inequation $\lambda_i/\mu_i<1$ hold. Therefore, all states in this process will be ergodic [35] and the equilibrium probabilities $\{P_i\}$ exist.

Under steady state conditions, the probability that *j* packets are found in queue is given by $P_j = p_0 \prod_{i=0}^{j-1} \frac{\lambda_i}{\mu_{i+1}}$ where $p_0 = \dfrac{1}{1 + \sum_{j=1}^{\infty} \prod_{i=0}^{j-1} \frac{\lambda_i}{\mu_{i+1}}}$ [35]. The mean arrival rate is $\lambda_0$ when there is 0 packet in the queue. Hence we have

$$P_{1,0} = \frac{1}{1 + \sum_{n=1}^{K} \left( \frac{\lambda_1}{\mu_1} \right)^n} = \frac{1 - \frac{\lambda_1}{\mu_1}}{1 - \left( \frac{\lambda_1}{\mu_1} \right)^{K+1}} \tag{3-1}$$

$$P_{2,0} = \frac{1}{1+\sum_{n=1}^{L}\left(\frac{\lambda_2}{\mu_2}\right)^n} = \frac{1-\frac{\lambda_2}{\mu_2}}{1-\left(\frac{\lambda_2}{\mu_2}\right)^{L+1}} \tag{3-2}$$

$$P_{1,K} = P_{1,0}\left(\frac{\lambda_1}{\mu_1}\right)^K = \frac{\left(1-\frac{\lambda_1}{\mu_1}\right)\left(\frac{\lambda_1}{\mu_1}\right)^K}{1-\left(\frac{\lambda_1}{\mu_1}\right)^{K+1}} \tag{3-3}$$

$$P_{2,L} = P_{2,0}\left(\frac{\lambda_2}{\mu_2}\right)^L = \frac{\left(1-\frac{\lambda_2}{\mu_2}\right)\left(\frac{\lambda_2}{\mu_2}\right)^L}{1-\left(\frac{\lambda_2}{\mu_{21}}\right)^{L+1}} \tag{3-4}$$

The basic idea of this approximation is to decompose the joint queues in figure 1 into two equivalent individual queues with derivable equivalent service rates. This will make the above probabilities obtainable. Since the priority is given to flow one (preferred flow), flow two (non-preferred flow) can only get serviced while the processor is idle and queue one is empty. If we approximate the service rate of class one $\mu_1$ with $\mu$, then the service rate for class two can be derived as

$$\mu_2 = \left(1-\frac{\alpha_1}{\mu}\right)\mu = \mu - \lambda_1\left[1-\frac{\left(1-\frac{\lambda_1}{\mu}\right)\left(\frac{\lambda_1}{\mu}\right)^K}{1-\left(\frac{\lambda_1}{\mu}\right)^{K+1}}\right] \tag{3-5}$$

From now on, the system can be equivalently decomposed into two individual queues with

$$\alpha_1 = \lambda_1 P_{1,0} + \lambda_1 P_{1,1} + \ldots + \lambda_1 P_{1,K-1} = \lambda_1(1-P_{1,K}) \tag{3-6}$$

$$\alpha_2 = \lambda_2 P_{2,0} + \lambda_2 P_{2,1,} + ... + \lambda_2 P_{2,L-1} = \lambda_2 (1 - P_{2,L}) \qquad (3\text{-}7)$$

In equilibrium, the average number of packets in both queues are obtained, according to [16], as

$$NQ_1 = \sum_{n=0}^{K} nP_{1,n} = \frac{\rho_1(1-(K+1)\rho_1^K + K\rho_1^{K+1}}{(1-\rho_1)(1-\rho_1^{K+1})} \qquad (3\text{-}8)$$

$$NQ_2 = \sum_{n=0}^{L} nP_{2,n} = \frac{\rho_2(1-(L+1)\rho_2^L + L\rho_2^{L+1}}{(1-\rho_2)(1-\rho_2^{L+1})} \qquad (3\text{-}9)$$

According to Little's formula, the average waiting time for the packets in both queues are

$\frac{NQ_1}{\alpha_1}$ and $\frac{NQ_2}{\alpha_2}$

The mean residual time in the server is [16]

$$R = \frac{1}{2}\sum_{i=1}^{2} \alpha_i \overline{\chi_i^2} \qquad (3\text{-}10)$$

where $\overline{\chi_i^2}$ is the second moment of the service time. When service times are exponentially distributed [35], $\overline{\chi_i^2} = \frac{2}{\mu_i^2}$.

Since a packets in the processor is served at the same rate $\mu$ no matter which flow the packet belongs to, the mean residual time can be derived as

$$R = \frac{1}{2}\sum_{i=1}^{2} \alpha_i \overline{\chi_i^2} = \frac{1}{\mu^2}\sum_{i=1}^{2} \alpha_i \qquad (3\text{-}11)$$

So the mean packet delays for flow one and flow two (preferred flow and non-preferred flow) are

$$Delay_1 = R + \frac{NQ_1}{\alpha_1} \tag{3-12}$$

$$Delay_2 = R + \frac{NQ_2}{\alpha_2} \tag{3-13}$$

## NUMERICAL RESULTS

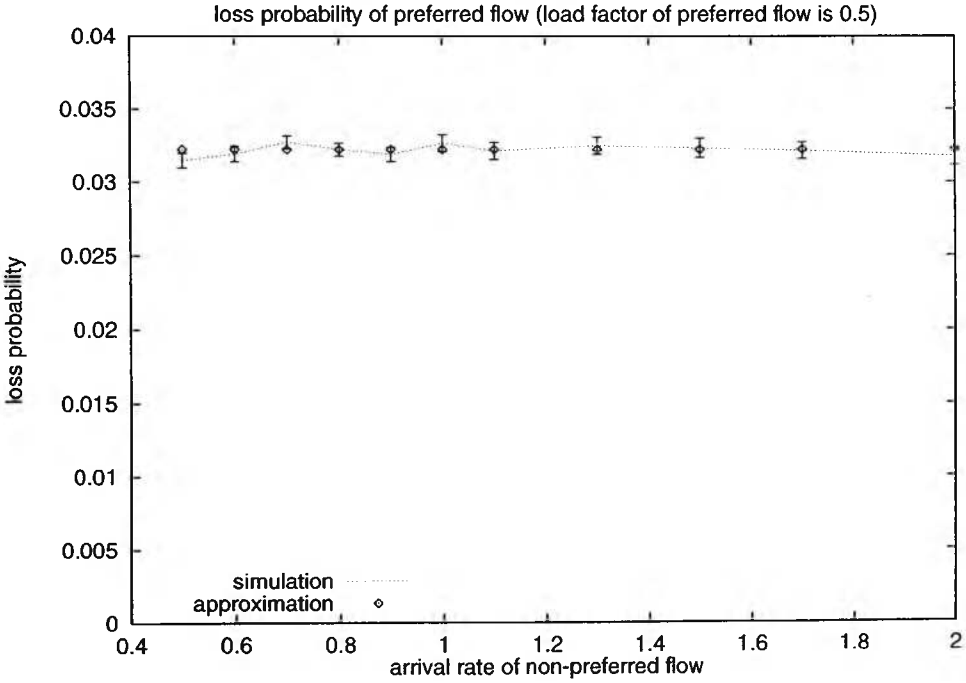
The accuracy of the approximation will be affected by the non-preemptive service rule when packets from preferred flow find the server is processing packets of non-preferred flow. This can take place particularly when the possibility that the server attends the packets of non-preferred flow increases. That is

1) The buffer size of non-preferred flow is large (compared with preferred flow) or
2) The load of non-preferred flow is heavy (for example when the load factor of the non-preferred flow is equal to 2.0).

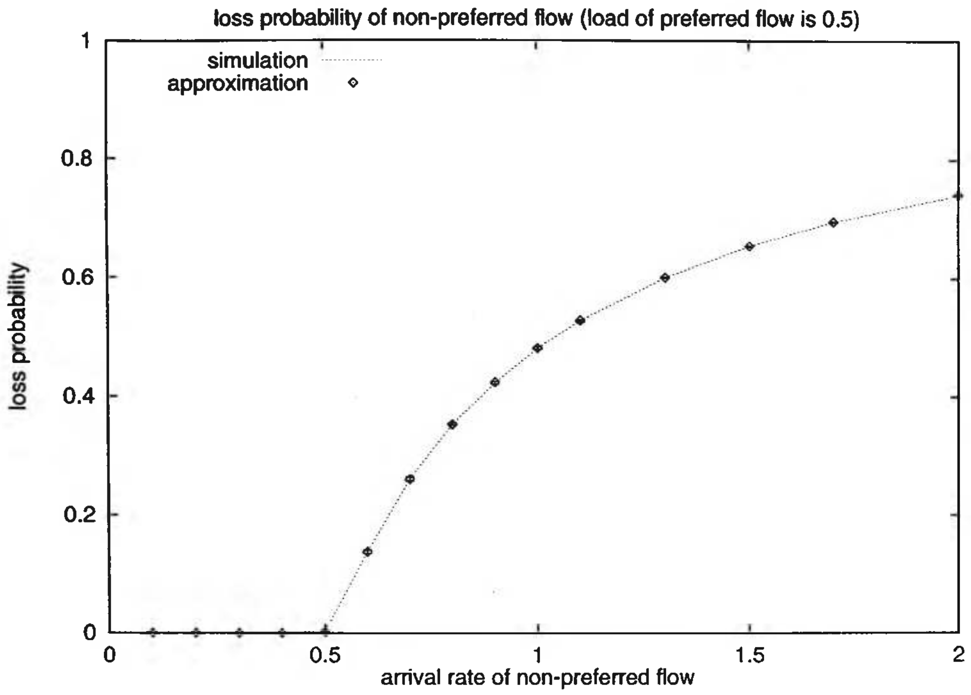
In order to verify the accuracy of the approximation method, simulation has been carried out. Figures 3-11 and 3-12 show the results arising from our approximation method.

In figure 3-11 and 3-12, the buffer size of the non-preferred flow is large (100 packets in comparison with 4 packets of preferred flow) and the load of the preferred flow be moderate (the load factor is 0.5). Figure 3-11 and 3-12 shows the packet loss and mean packet delay for both flows while varying the load of the non-preferred flow, and indicates a close agreement between simulation results and analytical ones. The mean packet delay is normalised with respect to service time.

We conducted additional simulation experiments, where the arrival rate of the preferred flow was varied with a constant non-preferred load factor of 2.0, and where the non-preferred arrival rate was varied with a constant preferred load factor of 1.1. In all cases, close agreement was observed between simulation and analytical results.

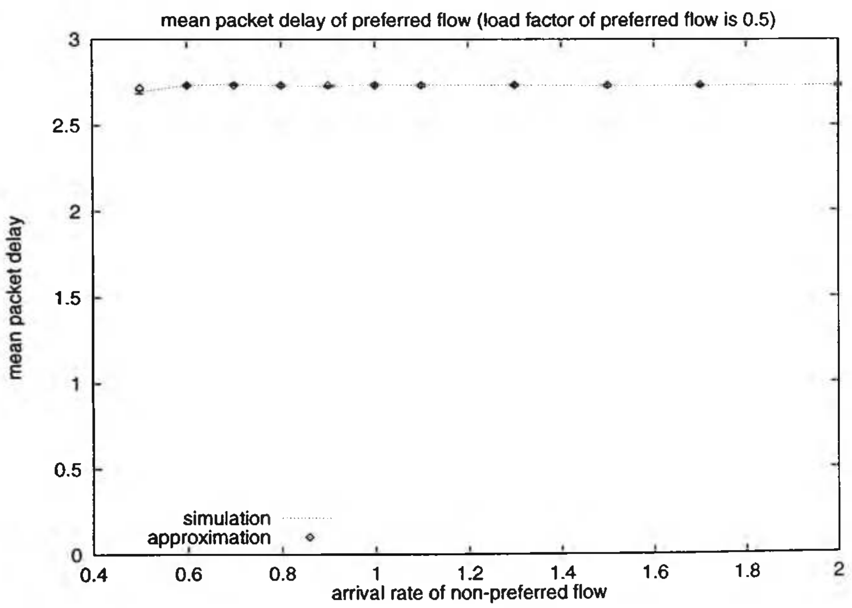



(A)

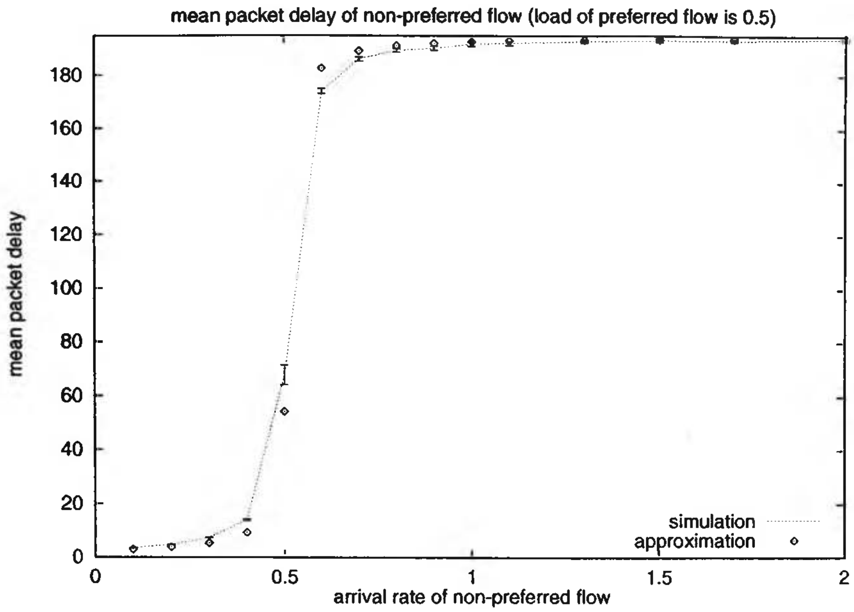



(B)

**FIGURE 3-11 PACKET LOSS COMPARISON OF THE RESULTS FROM SIMULATION AND ANALYTICAL APPROXIMATION METHOD WHEN THE BUFFER SIZE OF THE NON-PREFERRED FLOW IS LARGE**




(A)




(B)

**FIGURE 3-12 MEAN PACKET DELAY COMPARISON OF THE RESULTS FROM SIMULATION AND ANALYTICAL APPROXIMATION METHOD WHEN THE BUFFER SIZE OF THE NON-PREFERRED FLOW IS LARGE**

## 3.7 CONCLUSIONS

In this chapter, an analysis of non-fractional service rate reserved QoS mechanisms has been carried out. The focus of this study has been on the performance of two basic DiffServ mechanisms: Threshold Dropping (TD) and Priority Scheduling (PS), which are two fundamental mechanisms for Diffserv QoS provision.

Our performance investigation of the TD mechanism has indicated that changing the threshold of the non-preferred flow has a minimal effect on the packet loss of the preferred flow. With a fixed total buffer size and the same arrival rate for both flows, there is a minimal improvement in the loss for the non-preferred flow when its threshold is increased. The mean packet delays for both flows are bounded by their thresholds. A clear trade-off between packet loss and mean packet delay for the preferred and non-preferred flows is observed in the PS mechanism when the buffer allocation is changed. The TD mechanism provides lower packet loss and low mean packet delay to the non-preferred flow than PS. However the PS mechanism has the advantage over the TD mechanism in providing a lower mean delay to the preferred flow when the two mechanisms are engineered so as to provide the same level of packet loss for the preferred flow. As this would be a key requirement for an EF Diffserv implementation, we have continued our investigations into PS performance.

A simulation study has been undertaken to evaluate the suitability of PS for a Diffserv Expedited Forwarding (EF) implementation by looking at the performance of PS with three traffic flows. The simulation results show that the highest priority flow in PS will meet the requirements of EF, while the lower priority flows provide a best effort service. These results have motivated the development of an analytical technique for PS performance modelling. By using this analytical model, the packet loss and the mean packet delay of two traffic class flows can be easily approximated. The accuracy of this approximation method has been verified with simulations. This approximation method provides a simple way to understand the EF PHBs of DiffServ where PS mechanisms are deployed.

# Chapter 4

# Performance Analysis of Fractional Service Rate Reserved QoS Mechanisms—Latency Rate Servers

## 4.1 Introduction

The quality of service (QoS) of a packet network is indicated by a combination of criteria including loss probability, delay and delay jitter. A guaranteed QoS network requires a determination of whether there are sufficient resources to meet the needs of the required service level. It is necessary to have a good understanding of the performance behavior of QoS mechanisms, to guide bandwidth allocation and buffer dimensioning policies. In particular, future networks are likely to use multiple scheduling mechanism types. Hence the performance of broad range of scheduling mechanisms, particularly in terms of packet loss under the worst case scenario, needs to be determined.

Some related work on this issue has been presented by [3], [13], [25] and [44]. Parekh [3] has determined the worst case session backlogs for the GPS system, with the assumption of an infinite buffer size. More comprehensive analysis work has been done by Cruz [44] [45] on end to end delay and buffer requirements of sessions in an arbitrary topology network where all sources are leaky bucket controlled. A general model, called Latency-Rate (LR) server, developed by Stiliadis [13] has been used to derive the buffer requirements for an individual session of a broad range of scheduling algorithms. Stiliadis' study also gives an upper bound on the requirement for to guarantee a zero packet loss at the server. However, both Cruz's and Stiliadis' work assume an infinite buffer at the server, and hence loss behaviors are not addressed. The packet loss rate of a GPS server system with a finite buffer has been considered by Yee [25], but this work is limited to GPS only. Therefore further study is needed on the worst case loss behaviors of servers where buffer size is finite and a broad range of scheduling mechanisms employed.

Latency Rate servers, according to the classification of QoS mechanisms in section 2.3 of Chapter 2, belongs to the category of fractional service rate reserved QoS mechanisms. As LR servers describe a variety of QoS mechanisms, it is therefore possible to analyze the worst case performance in network with arbitrary QoS mechanisms. Accordingly the objective of this chapter is to analyze the packet loss behaviors of Latency Rate servers, calculate the upper bound on the average packet loss rate, and to determine the arrival processes which causes the maximum average loss rate.

Our approach extends the theory of LR servers to consider packet loss behaviors. We also extend the work in [25] to determine the arrival processes which causes the maximal average loss rate for LR servers with finite buffers (i.e. a worst case scenario).

By the worst case scenario, we mean that if a server attends $N$ sessions, the service rate that session $i$ can receive is only its reserved (or guaranteed) rate, i.e. all arrival processes of other sessions are selected to be backlogged when session $i$ is backlogged. It is assumed that there is a set of arrival processes U and there are countable arrival processes $A_m$ ($m$=1, 2, ...) in U, i.e. U={$A_m$} ($A_m$ is an arrival process and $m$=1, 2, ...). For any arrival process, let $\overline{L}_i(A_j)$ denote the time average traffic loss from session $i$ during [0,∞). By the maximal average loss rate, we mean that for session $i$ of the LR server, there exists $\eta$ and $A_\eta$ satisfying the leaky bucket and rate constraints of (4-4) (in section 4.3.1) such that $\overline{L}_i(A_\eta) = \max_{A_j \in U} \overline{L}_i(A_j)$.

The rest of this chapter is organized as follows: In section 4.2, we introduce the general analysis model of LR servers. In section 4.3 we present the arrival processes that result in the maximal average loss rate for LR servers. A proposition concerning Burst Over Latency (BOL) process which will result in the maximum average loss rate for the LR server is proposed. In section 4.4, an analysis of LR Servers with an arrival process which follows the Burst Over Latency (BOL) policy is presented. Two useful theorems are presented and proved. By using these theorems, formulae are derived for calculating the maximal average loss rate for LR servers. The proof of the proposition that BOL process results in the maximal average loss rate is provided in section 4.5.

In chapter 5, a case study for these two arrival processes, Bang Bang policy [25] and BOL policy and the comparison of zero buffer requirements are provided. This chapter also presents simulation results from a single LR server where WFQ is employed as the scheduling mechanism. Simulations are designed for verifying the maximal average loss rate and the impact of latency. Chapter 6 summaries the dissertation and provides directions for future work.

## 4.2 Latency Rate Server (LR server) Model

### 4.2.1 LR-SERVERS

The Latency Rate Server or LR-Server developed by Stiliadis and Verma [13] comprises a class of scheduling mechanisms (or schedulers). These scheduling mechanisms form the general model for studying the worst case behavior of individual sessions in a heterogeneous networks, where a broad range of scheduling mechanisms are used. According to the definition of LR server by Stiliadis, a scheduling mechanism can be said to be an LR server, if the average rate of service received by a busy session during any time interval starting at θ and within the session busy period is not less than its reserved rate. The parameter $\theta$ is called the latency of the server. Figure 4-1 presents two session busy periods (t1,t2], (t3,t4] and the latency $\theta$.

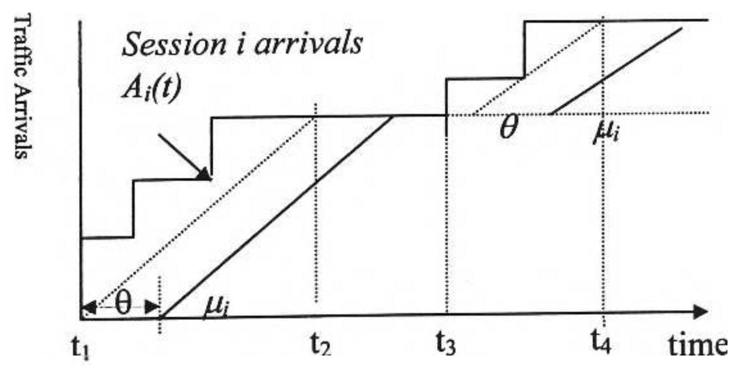



FIGURE 4-1. SESSION BUSY PERIODS ($t_1$, $t_2$] AND ($t_3$, $t_4$]. θ IS THE LATENCY OF THE LR SERVER.

(THE SOLID LINE INDICATES SERVICE STARS AT A GUARANTEED RATE AFTER THE LATENCY $\theta$)

Assume a packet switch where N sessions share the same output link. Denote by $\mu_i$

the rate allocated to session *i*, and by $A_i(\tau,t)$ the arrivals from session *i* during the interval $(\tau,t]$. $W_i(\tau,t)$ is the amount of service received by session *i* during the same interval. In the packet-by-packet model, it is assumed that $A_i(\tau,t)$ increases when a packet is completely received by the server, and $W_i(\tau,t)$ increases when a packet in service has completely departed from the server. Some definitions in [13] related to LR servers are revisited here.

**A session *i* backlogged period** is any time interval when packets of the session are continuously queued in the system. If $Q_i(t)$ denotes the amount of session i traffic queued in the server at time *t*, then $Q_i(t) = A_i(0,t) - W_i(0,t)$, and session *i* is said to be backlogged at time *t* if $Q_i(t) > 0$.

**A session *i* busy period** is the maximal time interval $(\tau_1,\tau_2]$ that for any given time $t \in (\tau_1,\tau_2]$,

$$A_i(\tau,t) \geq \mu_i(t-\tau_1) \quad (4\text{-}1)$$

Where $\mu_i$ is the reserved service rate for session *i* at the server.

The session busy period is defined in relation to a hypothetical system where a backlogged session *i* is served at a constant rate $\mu_i$ such as is illustrated as Figure 4-1. The key point about the session busy period is that it only depends on the arrival function $A_i(\tau,t)$ and the reserved service rate $\mu_i$. If the same traffic distribution is applied to different scheduling mechanisms with an identical service rate reservation, the session busy periods of the mechanisms are identical [13], even though their session backlog periods may vary. This is the fundamental reason for defining an LR server by the service received over a session busy period. This feature of the session busy period has proven to be an effective tool in LR server theory for analyzing the delay behavior in the LR server system. We will show in following sections that it can also be used in analyzing the loss behavior of the system.

The definition of the LR server [13] is revisited here due to the frequently reference made in the following sections of this Chapter. A scheduling mechanism is an LR server if there exists an non negative number $\theta$ such that the following inequality hold for all times $t$ from the start of the j[th] busy period of session $i$ till all packets that arrived during this period are served and vice versa.

$$W_{i,j}(\tau,t) \geq \max(0, \mu_i(t-\tau-\theta)) \qquad (4\text{-}2)$$

Where $\tau$ is the starting time of the j[th] busy period of session $i$, $\theta$ is the minimum non-negative number that satisfies the above inequality and $W_{i,j}(\tau,t)$ is the service received by the traffic of session j that arrived during time interval of $(\tau,t]$. $\theta$ is also called latency of the LR server. As shown in chapter 2, the $\theta$ in the above inequality is the worst case delay seen by the first packet of each backlogged session of the LR server. The latency $\theta$ of an LR server depends on the scheduling mechanism used as well as the service rate reserved for the session and the relative traffic parameters.

### 4.2.2. PROPERTIES OF LR SERVERS

In this sub-section, a summary of some important properties of LR servers defined by Stiliadis [13] is presented. For LR servers, if source traffic is leaky bucket constrained with a token bucket depth of $\sigma_i$ for session $i$, $Q_i(t)$ is the queue length of session $i$ at time $t$ and $\rho_i$ is the token arrival rate for the session, then the queue length is bounded and

$$Q_i(t) \le \sigma_i + \rho_i \theta_i \qquad (4\text{-}3)$$

The queuing delay is also bound by $D_j \le \frac{\sigma_j}{r_j} + \theta_j$ where $D_j$ is the maximum delay of any packet of session j in the LR server and $\theta_j$ is the latency of session j at the server. It is also proved in [13] that some well-known schedulers such as GPS, WFQ(PGPS), SCFQ belong to the LR server class. The latencies of these scheduling mechanisms are listed in table 4-1

TABLE 4-1. LATENCY OF GPS, WFQ (PGPS), SCFQ

| Scheduling mechanisms | Latency |
|---|---|
| GPS | 0 |
| WFQ(PGPS) | $L_i / r_i + L_{\max} / r$ |
| SCFQ | $L_i / r_i + L_{\max} (N-1) / r$ |

$L_i$ is the maximum packet size of session $i$ and $L_{max}$ is the maximum packet size of all sessions. $r_i$ is the reserved service rate of session $i$, $r$ is the service rate of the server and N is the connection number in SCFQ.

## 4.3. Arrival process that results in the maximal average loss rate for LR server

### 4.3.1. LEAKY BUCKET CONSTRAINED SOURCES

For a single LR server system with a total service rate of μ, let $N$ be the set of sessions in the system with each session policed by a leaky bucket ($\sigma_i$, $\rho_i$, $C_i$), as shown in Figure 4-2. For each unit of session $i$ traffic into the network, a session $i$ token is required. The rate at which session $i$ traffic can be input to the network is further constrained by a peak rate parameter $C_i$. Assume that $C_i > \rho_i$. $\rho_i$ is the token generating rate for session $i$. $\sigma_i$ is the capacity of the token bucket of session $i$, and $k_i$ is the input buffer size of session $i$ at the server. Traffic that arrives when the buffer is full is lost. The leaky bucket constraints can be described by

$$A_i(\tau,t) \le \min\{(t-\tau)C_i, \sigma_i + \rho_i(t-\tau)\} \quad \forall i\in N \text{ and } \forall \tau, t>\tau. \quad (4\text{-}4)$$

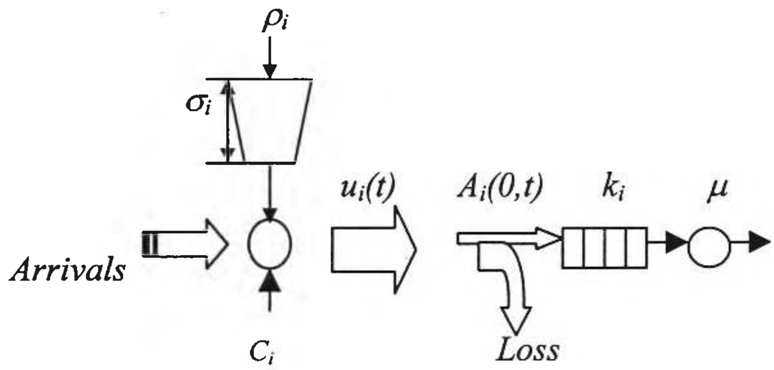



**FIGURE 4-2 LEAKY BUCKET CONSTRAINED SOURCES**

Let $u_i(t)$ be the number of packets instantaneous arrived after constraining of the leaky bucket at time $t$. It is the increasing rate of actual arrivals $A_i(0,t)$ at time $t$ which indicates how tokens in the leaky bucket are used at time $t$. The analysis of loss behaviors of the LR server in the following sections is carried out on the assumption of leaky bucket smoothed traffic arrivals.

### 4.3.2. ARRIVAL PROCESS THAT RESULTS IN THE MAXIMAL AVERAGE PACKET LOSS RATE

It has been proven [25] that the essential properties of the arrival processes that result in the worst case average loss rates for a GPS system are (i) the inputs occur in bursts, and (ii) they are periodic. The process can also be described as a Bang Bang control policy. To extend this work to a more general situation where the schedulers are LR servers, we introduce the following policy to control the use of tokens in the bucket. This policy maximizes average loss rate for the LR servers and can be stated as:

Whenever the token bucket for session $i$ is full, that is $b_i(t)=\sigma_i$ (where $b_i(t)$ denotes the token bucket state at time $t$ ), then all a batch of $\sigma_i$ traffic is input to the network. This assumes that $C_i=\infty$. Thereafter, the available tokens are continuously used until the time $\tau+\theta$, where $\tau$ is the start time of this session busy period. A packet that finds the input buffer full is lost. Tokens are to be accumulated after the server starts to provide service to the traffic of the session ($t=\tau+\theta$) until the token bucket fills up again. Hence for $C_i=\infty$, the token control policy is presented as

$$u_i(t)=\begin{cases}\sigma_i \sum_{n=0}^{\infty}\delta(t-n(\theta+tp))+\rho_i(t-n(\theta+tp)) & n(\theta+tp)\le t\le\theta+n(\theta+tp)\\ 0 & \theta+n(\theta+tp)<t<(n+1)(\theta+tp)\end{cases} \quad (4\text{-}5)$$

Where $\delta(t)$ is the impulse function and $tp=\sigma_i/\rho_i$, which means that the token bucket is emptied of $\sigma_i$ tokens every $tp$ time units. $b_i(t)$ is the token bucket state at time $t$.

We call an arrival process that follows (4-5) as BOL (burst over latency) arrival process or BOL policy.

Observe that the arrival process given in (4-5) is periodic with a period of $\theta+tp$, assuming $\mu_i>\rho_i$ and $\sigma_i>k_i$. The BOL policy can be interpreted as implying that whenever the token bucket of session $i$ becomes full, a burst of $\sigma_i$ packets (if each token represents a packet) is input to the network. The network input buffer becomes full with $k_i$ packets and $\sigma_i$-$k_i$ packets are lost. During the period until the session traffic receives service (i.e. the LR server latency), tokens are continuously used at the rate of $\rho_i$, the token generating rate. Tokens are accumulated under other circumstances until token bucket becomes full again. The input buffer (queue) is be emptied every $k_i/\mu_i$ time units.

Since $\frac{k_i}{\mu_i} < \frac{\sigma_i}{\rho_i}$, the queue will be empty by the time $\theta+tp$, so that the process repeats, beginning at time $\theta+tp$. Therefore, the token bucket state $b_i(t)$ and the state of input buffer queue $q_i(t)$ are both periodic with period $\theta+tp$. The average loss rate L(0,t) is

$$\lim_{t \to \infty} \frac{L_i(0,t)}{t} = \frac{\sigma_i + \rho_i \theta - k_i}{\theta + \frac{\sigma_i}{\rho_i}} \tag{4-6}$$

**Proposition** The BOL token control policy given by (4-5) is optimal with respect to maximizing the average loss rate for session $i$ with latency θ of a LR server. The worst case based maximal average loss rate of session $i$ is given by (4-6). The proof of this proposition is given in section 4.5.

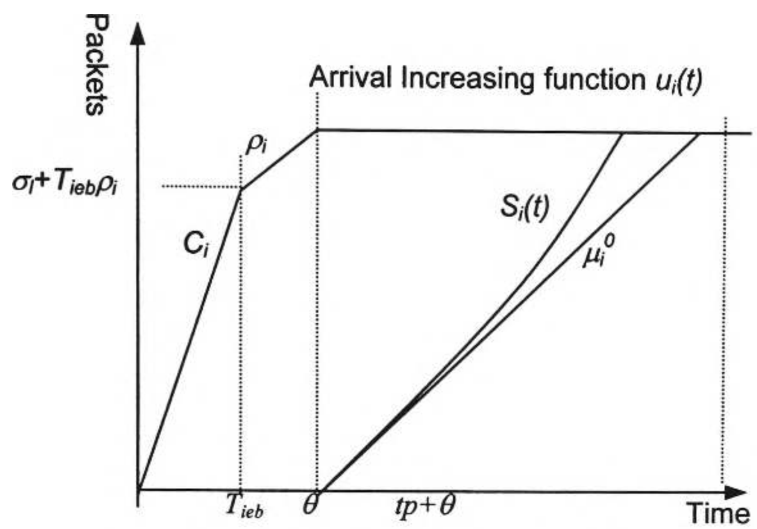



**FIGURE 4-3. ARRIVAL PROCESS AND SERVICE FUNCTION**

Figure 4-3 illustrates the arrival process and the service function of BOL policy (assuming $C_i > \rho_i$).

In figure 4-3, $\mu_i^0$ is the guaranteed service rate for session $i$ and $S_i(t)$ is actual service received by session $i$. The traffic of session $i$ enters the network at the rate of $C_i$ until the token bucket becomes empty at time $T_{ieb}$. After that the arrival rate equals to the token generation rate $\rho_i$ and the number of total tokens used at the end of latency period $\theta$ is

$$C_i T_{ieb} + \rho(\theta - T_{ieb})_i = \sigma_i + \rho_i \theta \qquad T_{ieb} < \theta$$

$$C_i T_{ieb} = \sigma_i + \rho_i T_{ieb} \qquad T_{ieb} \geq \theta$$

where $T_{ieb} = \dfrac{\sigma_i}{C_i - \rho_i}$

With the token control policy BOL, the arrival process for LR servers can be presented as

$$u_i(t)=\begin{cases}\rho_i(t-n(\theta+tp)-T_{ieb}) & n(\theta+tp)+T_{ieb}\le t<\theta+n(\theta+tp) \quad \theta>T_{ieb}\\ 0 & n(\theta+tp)+\theta\le t<(n+1)(\theta+tp) \quad \theta>T_{ieb}\\ C_i\sum\limits_{n=0}^{\infty}[u_{stp}(t-n(T_{ieb}+tp))-u_{stp}(t-n(T_{ieb}+tp)-T_{ieb})] & when \quad \theta\le T_{ieb}\end{cases} \tag{4-7}$$

where $u_{stp}(t)$ is the unit step function and $n$ is the number of the session busy period associated with the time $t$. If $k_i$ is the buffer allocation for session $i$, in each busy period cycle, the amount of traffic lost due to buffer overflow is

$max[0,\ C_i{*}T_{ieb}+\rho_i(\theta\text{-}T_{ieb})\ \text{-}S_i(0,T_{ieb})\text{-}k_i]$

where $\rho_i(\theta\text{-}T_{ieb})=0$ when $\theta< T_{ieb}$ and $S_i(0,T_{ieb})$ indicates the minimum service received by session $i$ during $(0,\ T_{ieb})$. Within the interval of $(0,T_{ieb})$, tokens are used at the maximum rate of $C_i$. If $\theta\geq T_{ieb}$, tokens will be continuously used at the rate of $\rho_i$ when $T_{ieb}\leq t<\theta$ in each period, and no tokens are used otherwise. Since only the worst case scenario is considered, the service rate of session $i$ is $\mu_i^0$. So $S_i(0,T_{ieb})=\mu_i^0(T_{ieb}\text{-}\theta)$ if $\theta< T_{ieb}$. The worst case based maximal average loss rate for LR servers when $C_i<\infty$ is then

$$\frac{\max\{0,C_iT_{ieb}+\rho_i(\lambda-T_{ieb})-\mu_i^0(\lambda-\theta)-k_i\}}{tp+\lambda} \tag{4-8}$$

$$\text{where } \lambda=\begin{cases}T_{ieb} & \theta<T_{ieb}\\ \theta & \theta\geq T_{ieb}\end{cases} \tag{4-9}$$

## 4.4. Analysis of LR servers with an arrival process which follows BOL under the worst case scenario

This section applies the theory of LR servers to an arrival process that follows the

BOL policy. Under the worst case scenario, two theorems are derived. By using these theorems one can calculate the maximal average loss rate for any individual arrival session (refer to section 4.6 for a case study). Apart from these, the zero loss buffer requirement of an LR server is also derived.

Consider a single LR-server system attending N sessions with each session having a finite input buffer size. For session *i*, the source traffic is leaky bucket constrained with parameters $\sigma_i$, $\rho_i$, $C_i$. Assuming $\mu_i^0$ is the minimum service rate allocated to session *i* and $\mu_i^0 > \rho_i$. Note that $\mu_i^0$ is also the maximum service rate under the worst case scenario due to the fact that all other sessions are backlogged. Assuming the arrival process is periodic according to (4-5) so that it follows the BOL control policy. In this part, we show that no session busy period exceeds the cycle length of the periodical arrival process and has the same start time as the periodic cycle.

**Lemma 1**: If $C_i = \infty$, assume $\tau_1$ is the starting time for both a session *i* busy period and the periodical cycles of arrival processes (4-5). If the session busy period is $(\tau_1, \tau_2]$ and $\mu_i^0$ is the constant service rate of the session as in (4-1), then

$\tau_2 - \tau_1 < \theta + tp$ and $\sigma_i/\rho_i$ is the token bucket *i* fill up time.

***Proof***: This lemma is proved by contradiction.

Suppose $\tau_2 - \tau_1 \geq \theta + tp$ and let $\tau_2 - \tau_1 = \theta + tp + x$ $(x \geq 0)$, then there exists a *t*, where $t \in (\tau_1, \tau_2]$ and $t - \tau_1 = \theta + tp$.

From the definition of the session busy period and (4-1), it is the case that

$$A_i(\tau_1, t) \geq \mu_i^0 (t-\tau_1) \tag{4-10}$$

Since $\mu_i^0 > \rho_i$ is assumed and $\boldsymbol{tp} = \sigma_i/\rho_i$, it is easy to derive that

$\mu_i^0 (t-\tau_1) = \mu_i^0 (\theta+\boldsymbol{tp}) > \rho_i\theta + \sigma_i$, that is

$$A_i(\tau_1, t) > \rho_i\theta + \sigma_i \tag{4-11}$$

According to the worst case arrival process which follows the BOL control policy, $A_i(\tau_1, t) = \rho_i\theta + \sigma_i$ which is conflict with (4-11). Therefore the assumption of

$\tau_2 - \tau_1 \geq \theta + tp$ does not hold and there must be $\tau_2 - \tau_1 < \theta + tp$.

**Lemma 2:** For any busy period $(t_{si}, t_f]$ of session $i$, the starting time $t_{si} \in \{\tau_{tpi}\}$, where $\tau_{tpi}$ is the set of start times of the periodic arrivals of session $i$.

***Proof:*** Contradictorily supposing the busy period starting time $t_{si} \notin \{\tau_{tpi}\}$ and there exists $\tau_{tpi+1}$, with $\tau_{tpi}$ such that $\tau_{tpi+1} > t_{si} > \tau_{tpi}$. $\tau_{tpi+1}$ is the starting time of a periodic arrival that follows $\tau_{tpi}$. Let $t_{si} = \tau_{tpi} + x$ $(0 < x < t_f - t_{si})$. Since $t_{si}$ is the starting time of a session $i$ busy period, for any $\xi \in (t_{si}, t_f] (\xi > \theta)$, there is

$$A_i(t_{si}, \xi) \geq \mu_i^0 (\xi - t_{si}) > \rho_i(\theta - t_{si}) \tag{4-12}$$

According to the arrival process stated in (4-5), at time $\tau_{tpi}$, the volume of traffic arrivals is $\sigma_i$. After $\tau_{tpi}$, the traffic is input to the network at the rate $\rho_i$ until time $\theta + \tau_{tpi}$. After time $\theta + \tau_{tpi}$, the traffic input into the network is zero. That means there exist an $0 < \varepsilon \leq x$ and $A_i(\tau_{tpi} + \varepsilon, t_f) = \rho_i(\theta - \varepsilon)$. Since the arrival function $A_i(t)$ is an increasing positive function in $t$, we obtain the result that

$A_i(t_{si}, \xi) \leq A_i(\tau_{tpi}+\varepsilon, t_f) = \rho_i(\theta-\varepsilon)$ which contradicts (4-12).

So $t_{si} \in \{\tau_{tpi}\}$ is shown to be valid.

**Theorem 1:** For an LR server, when $C_i = \infty$ and the arrival process follows the BOL control policy, there is only one session busy period within each time interval between $\theta+tp$ and the starting time of the busy period $t_{si} \in \{\tau_{tpi}\}$.

*Proof*:

Suppose there is another session busy period $(t_s', t_f']$ within the busy period $(t_s, t_f]$, it is easy to prove from lemma 1 and lemma 2 that $t_s' = t_s$ and $t_f' = t_f$. Again from lemma 2, we have $t_{si} \in \{\tau_{tpi}\}$.

**Lemma 3:** If $C_i < \infty$, assume $\tau_1$ is the starting time for both a session $i$ busy period and the periodical cycles of the arrival processes (4-7). If the session busy period is $(\tau_1, \tau_2]$ and $\mu_i^0$ is the constant service rate of the session as in (4-1), then

$\tau_2 - \tau_1 < tp+\xi$ (Note that $\xi$ is defined as in (4-9), $tp=\sigma_i/\rho_i$, $T_{ieb} = \dfrac{\sigma_i}{C_i - \rho_i}$ and $tp+\xi$ is the cycle length).

***Proof***: In the same way as lemma 1, we contradictorily assume that

$\tau_2 - \tau_1 \geq tp+\xi$ and we further assume $\tau_2 - \tau_1 = tp+\xi+x$ $(x \geq 0)$. Then there exists a

$t \in (\tau_1, \tau_2]$ where $t-\tau_1 = tp+\xi$

From the definition of session busy period, it is the case that

$A_i(\tau_1, t) \geq \mu_i^0 (t-\tau_1)$

Since $\mu_i^0 > \rho_i$, it is easy to derive that

$\mu_i^0(t-\tau_1) > \sigma_i + \rho_i\xi$, so that

$$A_i(\tau_1, t) \geq \mu_i^0(t-\tau_1) > C_iT_{ieb} - \rho_iT_{ieb} + \rho_i\xi \tag{4-13}$$

According to the worst case arrival process (6),

$$A_i(\tau_1, t) = C_iT_{ieb} + \rho_i(\theta - T_{ieb})^+ \tag{4-14}$$

$(a)^+ = max\{0, a\}$.

This is contradictory to (4-13) for any $\theta$. So $\tau_2 - \tau_1 < tp + \xi$ is valid.

**Lemma 4**: If $C_i < \infty$, for any busy session period $(t_{si}, t_f]$ of session $i$ with $\mu_i^0$ as the constant service rate of the session as in (4-1), the starting time $t_{si} \in \{\tau_{tpi}\}$, where $\tau_{tpi}$ is the set of start times of the periodic arrivals of session $i$.

***Proof***: Suppose there is a session $i$ busy period $(t_{si}, t_f]$ with starting time $t_{si} \notin \{\tau_{tpi}\}$ and $T_{ieb} > t_{si} > \tau_{tpi}$. We will prove that $(t_{si}, t_f]$ is not the maximum time interval such that there exists a number $\zeta$ where the inequality $A_i(t_{si}, \zeta) \geq \mu_i^0(\zeta - t_{si})$ holds for any $\zeta \in (t_{si}, t_f]$. Then it is easy to derive that $(t_{si}, t_f]$ is not a session $i$ busy period which is in conflict with our assumption. So there must be $t_{si} \in \{\tau_{tpi}\}$.

According to the definition of session busy period in section 4.2.1 (refer to Figure 4-3), there is

$$A_i(t_{si}, \zeta) \geq \mu_i^0(\zeta - t_{si}) \qquad \zeta \leq T_{ieb} \tag{4-15}$$

From (4-15) and the arrival process, we have $C_i(\zeta - t_{si}) \geq \mu_i^0(\zeta - t_{si})$ and therefore

$C_i \geq \mu_i^0$ (4-16)

On the other hand, the accumulated arrivals of session $i$ in the time interval $(\tau_{tpi}, t_{si}]$ is $A_i(\tau_{tpi}, t_{si}) = C_i(t_{si} - \tau_{tpi})$. From (4-16), it is easy to obtain the result that

$C_i(t_{si} - \tau_{tpi}) \geq \mu_i^0(\tau_{tpi} - t_{si})$. (4-17)

Inequality (4-17) indicates that the time interval $(t_{si}, t_f]$ is not the maximum time interval that makes the inequality $A_i(t_{si}, \zeta) \geq \mu_i^0(\zeta - t_{si})$ hold for any $\zeta \in (t_{si}, t_f]$. Therefore $(t_{si}, t_f]$ is not a session busy period. This is in conflict with the assumption of the lemma. So the $t_{si} \in \{\tau_{tpi}\}$ must hold.

**Theorem 2:** For a LR server, when $C_i < \infty$ and the arrival process follows the BOL control policy, there only one session busy period within each time interval between $tp+\xi$ and the starting time of the busy period $t_{si} \in \{\tau_{tpi}\}$. ($\xi$ is defined as in (4-9). $tp = \sigma_i/\rho_i$, $T_{ieb} = \dfrac{\sigma_i}{C_i - \rho_i}$ and $tp+\xi$ is the cycle length).

***Proof*:**

Similar to lemma 2, after $\xi$, there are no more packets arriving until the next starting point of the cycle period. Therefore, there is no session busy period starting after time $\xi$. From (4-16), it is easy to show that $(\tau_{tpi}, T_{ieb}]$ must be contained in the same session busy period as $(t_{si}, t_f]$. Lemma 4 indicates that each start time of session busy period $t_{si} \in \{\tau_{tpi}\}$. Lemma 3 indicates that the length of session busy period will not exceed $tp+\xi$. This ends our proof.

Based on the theorems 1 and 2, the BOL process for session $i$ and the associated service behavior can be illustrated as in Figure 4-4 for $C_i$=∞ and figure 4-5 for $Ci<\infty$.

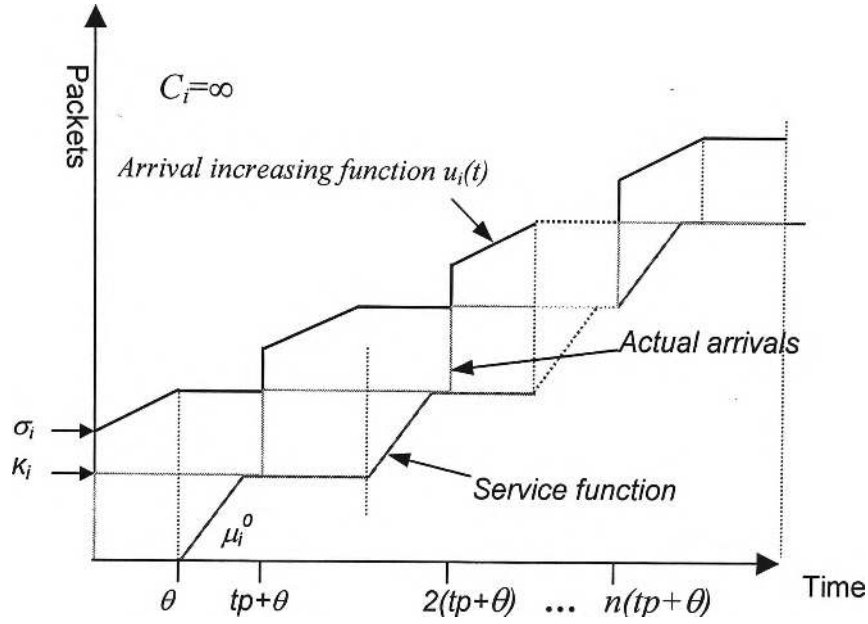



FIGURE 4-4. BOL PROCESS AND SERVICE FUNCTION WHEN $C_i$=∞

In Figure 4-4, the token bucket is emptied at the start time of each period which will cause a $\sigma_i$-$k_i$ traffic loss for session *i*. When the input buffer queue of the session is full, tokens are continuously used as long as there are any available tokens. The maximum rate is identical to the token generating rate. The use of tokens will cease when the server starts to serve the traffic of the session, due to the BOL policy. During the time interval $(\theta, tp+\theta)$, all the traffic in the input buffer will be emptied since $\mu_i^0 tp > \sigma_i$ according to (4-2). The maximal packet loss in each period is $\sigma_i + \rho_i\theta - k_i$. It is obvious that, under the worst case scenario, the buffer requirement to guarantee zero loss from a session of an LR server is

$$k_i = \sigma_i + \rho_i\theta \tag{4-18}$$

and the maximal average loss rate is give by (4-6). For convenience, we simply rewrite it here as $\overline{L}_i = \dfrac{\sigma_i + \rho_i\theta - k_i}{\theta + \dfrac{\sigma_i}{\rho_i}}$. Note that (4-18) is identical to the results for a single LR server [13].

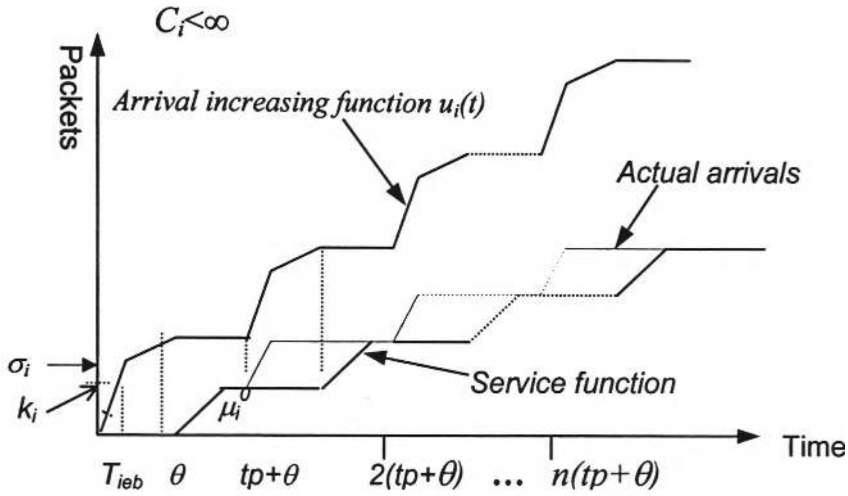



FIGURE4 5. BOL PROCESS AND SERVICE FUNCTION WHEN $C_i<\infty$ AND $\theta>T_{IEB}$

Similar to Figure 4-4, Figure 4-5 presents the arrival process according to (4-7). For an arbitrary latency θ, the packet loss in each period of session $i$ is

given by

$$\begin{cases} C_iT_{ieb} + \rho_i(\theta - T_{ieb}) - k_i & \theta \ge T_{ieb} \\ C_iT_{ieb} - \mu_i^0(T_{ieb} - \theta) - k_i & \theta < T_{ieb} \end{cases} \qquad (4\text{-}19)$$

From (4-19) we see that the buffer requirement to guarantee zero loss from session $i$ under the worst case scenario is

$$k_i = \begin{cases} \sigma_i + \rho_i\theta_i & \theta \ge T_{ieb} \\ \sigma_i + \mu_i^0\theta - (\mu_i^0 - \rho_i)T_{ieb} & \theta < T_{ieb} \end{cases} \qquad (4\text{-}20)$$

# 4.5. The proof of the arrival process that results in the maximal average loss rate for LR servers

## 4.5.1. PROPOSITION

The problem of determining the arrival process that results in the maximal average loss rate for LR servers is to determine the optimal control policy for using tokens in the bucket. This section shows that the BOL policy introduced in section 4.3.2 produces the maximum average loss rate for a general case where the schedulers are LR servers. Consider individual traffic sessions that can only receive reserved service rate at LR servers, then the maximal average loss rate of session $i$ is first derived based on the following proposition.

**Proposition:**

**If we let $A_i(0,t)$ be the amount of session $i$ traffic input to the network at a rate of $C_i$=∞, and all traffic admitted into the network is leaky bucket constrained, the BOL (Burst Over Latency) arrival process given by**

$$u_i(t) = \begin{cases} \sigma_i \sum_{d=1}^{n} \delta(t - t_d) + \rho_i t_d & t_d \le t < t_d + \theta \\ 0 & t_d + \theta \le t < t_{d+1} \end{cases} \qquad (4\text{-}21)$$

**is optimal in terms of maximizing the average packet loss rate for session $i$ with latency $\theta$ of a LR server.**

$u_i(t)$ is the number of tokens in the token bucket $i$ that are used at time $t$. This is just equivalent in value to the traffic arrived at time $t$ or the increasing rate of $A_i(0,t)$ at time $t$, where $t_d = t_{d-1} + \theta + \sigma_i / \rho_i$ ($t_0 = 0$) and $n$ tends towards infinity, while $t$ has infinite length.

**The maximal average packet loss rate, based on per token interarrival time, of session *i* under the worst case scenario is given by**

$$\overline{L_i} = \frac{\sigma_i + \rho_i \theta - k_i}{\rho_i \theta + \sigma_i} \qquad (4\text{-}22)$$

where $k_i$ is input buffer size for session $i$ in the system, $\rho_i$ is the token generating rate and $\sigma_i$ is the token bucket capacity of session $i$.

### 4.5.2. BACKGROUND

In an LR server, if session $i$ is considered to be worst case, then the traffic from the session will only receive its reserved service rate $\mu_i^0$ ($\mu_i^0 > \rho_i$), i.e. all other sessions of the server are continuously backlogged. The packets of the session that arrive to find that the input buffer queue is full are dropped. Whenever there is a packet in the input buffer queue, the queue will be serviced at the rate $\mu_i^0$ after the latency period. The first packet arriving during a session busy period will wait an interval $\vartheta$ ($0 \le \vartheta \le \theta$) before service begins.

To determine the arrival process that maximizes the average packet loss rate for session $i$ of an LR server, we must find an optimal control policy for using tokens in token bucket $i$, in order to maximize the packet loss. We naturally assume that the input buffer and token bucket are finite.

Based on assumption that $C_i = \infty$, the approach used to prove our proposition is dynamic programming, a method for the solution of a sequential decision process.

Taken the number of packets lost as the reward of the policy for using tokens, the problem of determining the optimal policy to maximize packet loss can be formulated as a problem of sequential decision making in a Markov Process with Reward [20]. If the traffic arrives on a packet-by-packet basis or byte-by-byte basis, it is more realistic to assume that the token bucket capacity $\sigma$ and the bucket state $b(t)$ take only integer values. Then we can simplify the problem by transforming the continuous time, continuous state representation into discrete time and state one.

Let $\beta$ be the token interarrival time ($\beta=1/\rho_i$) for session $i$. Then the instances of token arrivals will be $\beta$, $2\beta$, $3\beta$,.... The decision to use the tokens is made immediately after their respective arrivals at times $\beta^+$, $2\beta^+$, $3\beta^+$,.... The states of the token bucket and the input buffer queue are written as $b_s$ and $q_s$ respectively. The decision to use tokens when system is in state $(b_s, q_s)$ at time $s$ is denoted by $u_s$. So $u_s$ can be any integer between 0 and $b_s$ inclusively.

In this approach, let R(s) be the expected immediate reward for the state $(b_s,q_s)$ when decision $u_s$ is made. Denote the set of the decisions with $U=\{u_s\geq 0$ where $u_s\leq b_s$ for any $s\}$. With a given control policy, a Markov process with rewards is specified. If the process is to operate for $m$ transactions, the total expected reward that the system will obtain is stated as (4-23), starting from state $b_s$ under the given policy [20].

$$v_s(m) = R(s) + \sum_{j=1}^{N} p_{b_s j} v_j(m-1) \quad s = 1,2,\ldots,N \quad m = 1,2,3,\ldots \tag{4-23}$$

$v_s(m)$ is the sum of the expected total earnings in the next $m$ transitions if the system is now in state $b_s$. The quantity R(s) can be interpreted as the reward to be expected in the next transition out of state $b_s$. $P_{b_s j}$ is the transition probability from state $b_s$ to state $j$ in next transition. $N$ is the number of states that the Markov process may have. $R(s)$ is defined by

$$R(s) = \sum_{j=1}^{N} P_{b_s j} r_{b_s j} \qquad s = 1,2,3,\ldots N \tag{4-24}$$

$r_{b_s j}$ is the reward obtained when the Markov process make a transition from state $b_s$ to state $j$.

When $m$ becomes large, $v_s(m)$ is approximated as [20]

$$v_s(m) = mg + h(b_s, q_s)$$

where $g$ is the average reward per transaction of the system if it started from state $(b_s, q_s)$ and the number of transitions $m$ is large. $h(b_s, q_s)$ is called the relative value of the policy. It initially represents the intercepts at $m=0$ of the asymptotes of $v_s(m)$. Then (4-23) is further derived [20] as

$$mg + h(b_s, q_s) = R(s) + \sum_{j=1}^{N} p_{b_s j}[(m-1)g + h(b_j, q_j)] \quad s = 1,2,\ldots,N \tag{4-25}$$

$$mg + h(b_s, q_s) = R(s) + (m-1)g\sum_{j=1}^{N} p_{b_s j} + \sum_{j=1}^{N} p_{b_s j} h(b_j, q_j) \tag{4-26}$$

Since $\sum_{j=1}^{N} p_{b_s j} = 1$, (4-25) and (4-26) can be written as follows

$$g + h(b_s, q_s) = R(s) + \sum_{j=1}^{N} p_{b_s j} h(b_j, q_j) \quad s = 1,2,\ldots, N \tag{4-27}$$

In our system, a transition happens at interval of $\beta$, due to the arrival of a new token. Hence the $g$ can be interpreted as the reward gained from the system per token interarrival time. If we let $g' = \max_{u \in U} g$, then $g'$ will be the maximal average loss rate of the system.

The approach to determining an optimal policy that maximizes the average loss rate for session $i$ is based on the following steps:

**Step 1** is called the value-determination operation. It uses equation (4-27) to determine the relative values $\{h(b_s, q_s)\}$ and g with the setting $h(b_N, q_N)=0$.

**Step 2** is policy-improvement [20]. The policy improvement phase will find the optimal policy for each state $b_s$ that maximizes the test quantity (4-28) using the relative values determined under the old policy. Note that the test quantity (4-28) is just the RHS (Right Hand Side ) of (4-27).

$$R(s) + \sum_{j=1}^{N} p_{b_s j} h(b_j, q_j) \quad s = 1,2,\ldots,N \tag{4-28}$$

For example, if $\chi$ tokens are used in policy $u$ when the system is in state $(b_s, q_s)$, the relative value in the next state will be

$$h(b_s - \chi + 1, q_{s+1})$$

With using the technology of Value-determination and Policy-improvement, in next section, we show that BOL policy is optimal with regard to the maximum average loss rate.

### 4.5.3. PROOF

To prove the proposition introduced in section 4.5.1, two steps of reward based techniques have been used. In step 1, we apply the BOL policy to the LR server to solve (4-27) for all relative values and $g$ by setting $h(b_{\sigma_i}, q_{\sigma_i})=0$ [20]. In step 2, the policy improvement phase, with using the relative values of the BOL policy obtained from step 1, we show that the BOL policy is optimal in terms of maximizing (4-28).

With applying the BOL policy (4-5) to the LR system, the state transition diagram of the system can be presented in Figure 4-6.

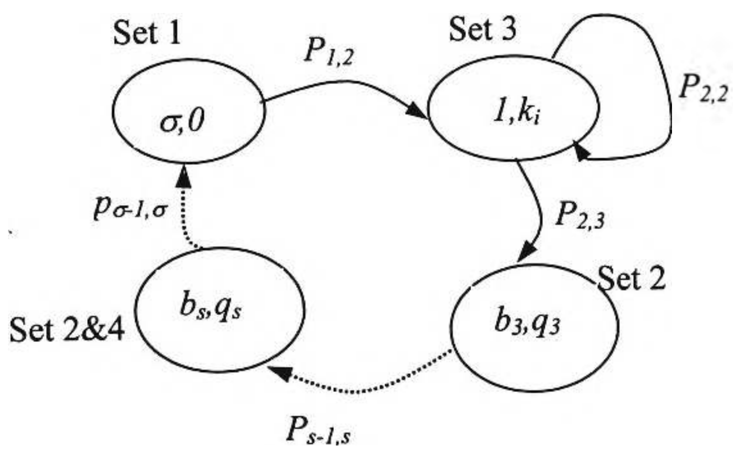



FIGURE 4-6. TRANSITION DIAGRAM OF BOL POLICY

WHERE ($SYMBOL_1$, $SYMBOL_2$) IS THE STATE OF THE SYSTEM AND $SYMBOL_1$ INDICATES THE TOKEN NUMBER IN THE BUCKET AND $SYMBOL_2$ INDICATES THE QUEUE LENGTH OF THE INPUT BUFFER FOR SESSION $i$. S = 1,2,..., N AND N IS THE NUMBER OF STATES THAT THE SYSTEM MAY HAVE.

The periodical transition starts from state ($\sigma$,$0$) when the token bucket for session $i$ is full ($\sigma$ *tokens in the bucket*) and input buffer is empty ($q_s=0$). With the probability of $P_{1,2}$, the state $(\sigma,0)$ transits to state $(1,k_i)$ where input buffer for session $i$ is full and the token bucket contains the only token which arrived during the transition interval. During the latency period, tokens are continuously used. So the state $(1,k_i)$ remains till the end of the latency period at the possibility of $P_{2,2}$. Then the service period starts and the BOL policy controls not to use any token till the token bucket filled up again. During the serving time interval, states are transit from $(b_3,q_3)$ to $(b_s,q_s)$.

According to the definition of BOL Policy and the states of transition, we can obtain the following matrixes:

The Transition probability matrix **P** and Reward matrix **R**

$$\mathbf{P}=\begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & p_d & 1-p_d & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \text{ and } \mathbf{R}=\begin{bmatrix} 0 & \sigma_i - k_i & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

Where $P_d = \dfrac{\theta}{\theta + 1/\rho_i}$ and there are $\sigma_i$ states in total for session $i$.

In the Figure 4-6, all the possible states fall into the following four sets:

- Set 1 $(\sigma_i, 0)$ is the start state of the periodical transitions.
- Set 3 $(b_s,k_i)$ are states of the system during latency period.
- Set 2 $(b_s,0)$ and set 4 $(b_s,q_s)$ are the states of the system during serving time period.

Corresponding to these four sets, the equation (4-27) can be presented as (4-29). Note that in this approach, if $b_s=\gamma$ (i.e. there are $\gamma$ token in the token bucket and $\gamma\leq\sigma_i$) then $b_{s+1}=\gamma+1$. Since $\beta^+$, $2\beta^+$, $3\beta^+$,...are the times that decision is to be made, $b_s\geq 1$.

$$\begin{array}{ll} g+h(\sigma_i,0)=(\sigma_i-k_i)+h(1,k_i) & \\ g+h(b_s,k_i)=1+h(b_s,k_i) & t_d\leq t\leq t_d+\theta \quad d=1,2,..n-1 \\ g+h(b_s,0)=0+h(b_{s+1},0) & \forall b=k_i+1,k_i+2,...,(\sigma_i-1) \\ \begin{cases} g+h(b_s,q_s)=0+h(b_{s+1},(q_s-\mu_i^0\beta)^+) \\ g+h(b_s,q_s)=0+h(b_{s+1},(q_s-\mu_i^0\beta)^+) \end{cases} & \begin{array}{l} \forall(b_s,q_s)\geq 0, b_s+q_s\leq k_i \\ t_d+\theta<t\leq t_{d+1} \quad d=1,2,..n-1 \end{array} \end{array} \tag{4-29}$$

where $(a)^+=max\{a, 0\}$ and $t_d$ is the time at which the system is in state $(\sigma_i,0)$. Note that queue length is relevant only when it reaches its maximum value $k_i$ for session $i$ and will be zero after time $\theta+k_i/\mu_i^0$ for the d$^{th}$ busy period. Apply matrix **P**, **R** to (4-27) in the step 1 of above approach, the average reward of the system per token interarrival time $g$ under the policy (4-5) is given as

$$g=\frac{\rho_i\theta+\sigma_i-k}{\rho_i\theta+\sigma_i} \tag{4-30}$$

And the some useful relative values are derived as follows

$$\begin{cases} h(b_s,q_s)=(b_s-\sigma_i)g \\ h(1,k_i)=g-(\sigma_i-k_i) \\ h(\sigma_i,0)=g \end{cases} \tag{4-31}$$

In the following part, it will be shown that (4-5) is optimal in terms of obtaining the maximum reward (i.e. loss), previously denoted as $g'$. To achieve this, in the policy improving phase (i.e. step 2), by using (4-30) and (4-31) we only need to show that the BOL policy provided in (4-5) will maximize the RHS of (4-27) [20] for each set in (4-29). As a matter of fact, if the BOL policy (4-5) is proved to be optimal, the g in (4-30) will be $g'$.

**Set 1: ($\sigma_i$, 0)**

Consider the decision of using $\chi$ tokens is made when system is in the state of ( *($\sigma$,0))*. The maximization of the RHS of (4-27) is

$$\max_{u\in U}\{R(s)+\sum_{j=1}^{N}p_{b_s j}h(b_j,q_j)\}$$

$$=\max_{0\le\chi\le\sigma_i}\{(\chi-k_i)^+ + h(b_2,q_2)\}$$

$$\max_{0\le\chi\le\sigma_i}\{(\chi-k_i)^+ + g-(\sigma_i-k_i)\} \quad (4\text{-}32)$$

where $\chi$ is the number of token to be used in this state. When $\chi=\sigma_i$, the value of (4-32A) is $g$. Under other conditions, we can observe that the term $(\chi-k_i)^+$ is a piecewise linear function of $\chi$ with a slope of 1 when $\chi > k_i$ and has the value of zero when $\chi \le k_i$.

Obviously, when $\chi=\sigma_i$ (4-32) reaches the maximum value. Therefore, it is optimal to maximize the reward by using all available tokens when the token bucket is full.

**Set 2: *($b_s$,0)***

For any state *($b_s$,0)* where $b_s=k+1, k+2,...,(\sigma_i-1)$ the maximization of RHS of (4-27) is

$$\max_{u\in U}\{R(s)+\sum_{j=1}^{N}p_{b_s j}h(b_j,q_j)\} \quad (4\text{-}33)$$

$$=\max_{0\le\chi\le\sigma_i-1}\{(\chi-k_i)^+ + (b_s-\chi+1-\sigma_i)g\}$$

Take $b_s=(\sigma_i-1)$ as a sample case for illustration and the remaining cases will have the same pattern. Substitute $b_s=(\sigma_i-1)$ in (4-33), so that

$$\max_{u\in U}\{R(s)+\sum_{j=1}^{N}p_{b_s j}h(b_j,q_j)\}$$
$$=\max_{0\le\chi\le\sigma_i-1}\{(\chi-k_i)^+-\chi g\}$$

Term $-\chi g$ has a negative slope for all $\chi$ $(0\le\chi\le\sigma_i-1)$. Hence, a linear convex function of $\chi$ is formed from summing up term $-\chi g$ and $(\chi-k_i)^+$. The maximum value is reached either when $\chi=0$ or $\chi=\sigma_i-1$. To substitute the values of $\chi$ we obtain a value of 0 when $\chi$=0 or a negative value $\frac{-\rho_i\theta-k}{\rho_i\theta+\sigma_i}$ when $\chi=\sigma_i-1$. This indicate that, in set 2, not to use any token is optimal.

**Set 3: $(b_s,k_i)$**

When the system is in state $(b_s,k_i)$ and $t_d+\theta+\sigma_i/\rho_i\le t\le t_{d+1}+\theta$, $d=1,2,\ldots n-1$, the RHS of equation (4-27) becomes

$$\max_{u\in U}\{R(s)+\sum_{j=1}^{N}p_{b_s j}h(b_j,q_j)\}$$
$$=\begin{cases}\max\limits_{0\le\chi\le b_s}\{\chi+g-(\sigma_i-k_i)\} & when\quad b_s-\chi+1=1 \qquad (4-34A)\\ \max\limits_{0\le\chi\le b_s-1}\{\chi+(b_s-\chi+1-\sigma_i)g\} & otherswise \qquad (4-34B)\end{cases}$$

When $b_s-\chi+1=1$, i.e.$\chi=b_s$, (4-34A) becomes $b_s+g-(\sigma_i-k_i)$. The equation (4-34B) is reaches its maximum value at either point $\chi=0$ or at $\chi=b_s-1$. We substitute in these two values of $\chi$ to obtain the values $(b_s+1-\sigma_i)g$ and $b_s-1+(2-\sigma_i)g$. Among these three values, the largest one is $b_s+g-(\sigma_i-k_i)$ when $\chi= b_s$. And thus the optimal policy at this state is to use all of the available tokens.

Note that the last state of set 3 should be $(1, k_i)$ due to the fact that the decision to use all available tokens was made in previous states, and there is only one new token arriving between decisions. When the state is $(1,k_i)$ and $t_d+\theta\leq t\leq t_{d+1}$ $d=1,2,...n-1$, the packets in the input buffer are being served at the rate of $\mu_i^0$, the RHS of equation (4-27) becomes

$$\max_{u\in U}\{R(s)+\sum_{j=1}^{N}p_{b_s j}h(b_j,q_j)\}$$
$$=\max_{0\leq\chi\leq b_s}\{(\chi-\mu_i^0\beta)^+-\chi g+(b_s+1-\sigma_i)g\} \tag{4-35}$$

The value of $\chi$ that maximizes this sum is at one of the extreme points $\chi=0$ or $\chi=1$. Because the value $(2-\sigma_i)g$ when $\chi=0$ is greater than $(1-\sigma_i)g$ when $\chi=1$. So at the state $(1,k_i)$ $t_d+\theta\leq t\leq t_{d+1}$ $d=1,2,\ldots n-1$, not using any token is optimal.

**Set 4: $(b_s,q_s)$, $(b_s,q_s)\geq 0$**

For any state $(b_s,q_s)$, where $(b_s,q_s)\geq 0$, and $(b_s+q_s)\leq k_i$, there will be no reward (no packet loss or $R(s)=0$ ) for any single transition of the process, and the value of RHS of (4-27) will be

$$\max_{u\in U}\{R(s)+\sum_{j=1}^{N}p_{b_s j}h(b_j,q_j)\}$$
$$=\max_{u\in U}\{0+(b_s-\chi+1-\sigma_i)g\}$$

For any decision *u*, the use of the zero token, $\chi=0$, will maximize the value of the above equation. So the decision to not use any token when the system state is $(b_s,q_s)$ is optimal.

If $(b_s+q_s)>k_i$ $b_s\neq\sigma_i$ and $t_d\leq t\leq t_d+\theta$, we need to consider the state sets previous to $(b_s,q_s)$ (set 2 and set 3). Since the packets in the input buffer are waiting for the server, $q_s$ has the value of $k_i$ or zero and the corresponding state would be $(b_s,k_i)$ or $(b_s,0)$. The optimal policy for using tokens will be the same as for set 2 or set 3.

If $(b_s+q_s)>k_i$ $b_s\neq\sigma_i$ and $t_d+\theta\leq t\leq t_{d+1})$, there exist two cases for the RHS of (4-27):

- ***when state $(b_s,q_s)$ is transited from set 2***

$\max_{u\in U}\{R(s)+\sum_{j=1}^{N}p_{b_sj}h(b_j,q_j)\}$ equals to

$$\begin{cases}\max\limits_{0\leq\chi\leq b_s}\{(\chi-\mu_i^0\beta-k_i)^+ + g-(\sigma_i-k_i) & for\begin{cases}b_s-\chi+1=1\,and\\ \chi-\mu_i^0\beta-k_i\geq 0\end{cases} & (4-36A)\\ \max\limits_{0\leq\chi\leq b_s-1}\{(\chi-\mu_i^0\beta-k_i)^+ +(b_s-\chi+1-\sigma_i)g\} & otherwise & (4-36B)\end{cases}$$

Since the value $(b_s+1-\sigma_i)g$ at $\chi=0$ is greater than $b_s-\mu_i^0\beta+g-\sigma_i$ at $\chi=b_s$, in a similar way to (4-35), not using any token is the optimal decision in this state. The ongoing state from here will have the same pattern till $b_s=\sigma_i-1$ as in set 2.

- ***when state $(b_s,q_s)$ is transited from set 3***

Since at the time of making the decision for state $(1,k_i)$, the number of tokens that can be used is $\chi=1$ and $\chi-\mu_i^0\beta<0$, we have

$$\max_{u \in U}\{R(s)+\sum_{j=1}^{N} p_{b_s j} h(b_j,q_j)\}$$
$$= \max_{0\le\chi\le b_s}\{(\chi+q_s-\mu_i^0\beta-k_i)^+ +(b_s-\chi+1-\sigma_i)g\} \qquad (4\text{-}37)$$
$$= \max_{0\le\chi\le b_s}\{(b_s-\chi+1-\sigma_i)g\}$$

Obviously, $\chi$=0 will maximize the value of $(b_s-\chi+1-\sigma_i)g$ and thus to not use any tokens is the best decision in terms of maximizing the value of (4-37).

Now let us look at the consequential states. If there are $b_s$ tokens available for use, the time elapsed after $b_s$ tokens have been used since $b_s$=1will be $b_s\beta$. The state after $b_s$ tokens are used will be ($b_{s+1}=1$, $q_{s+1}=k_i- b_s\mu_i^0\beta$). Since $1-\mu_i^0\beta<0$, the RHS of (4-27) can be written as

$$\max_{u \in U}\{R(s)+\sum_{j=1}^{N} p_{b_s j} h(b_j,q_j)\}$$
$$= \max_{0\le\chi\le b_s}\{(b_s-\chi+1-\sigma_i)g\}$$

Since $(-g\chi+(b_s+1-\sigma_i)g)$ has a negative slope for all $\chi$ ($0\le\chi\le b_s$), $\chi$=0 will thus maximize the value of the above equation.

From set 1 to set 4, by the use of policy improvement method, we have shown that the arrival process that follows the BOL policy will result in the maximal average packet loss for an LR server.

Under BOL, if the latency of each session $i$ busy period is variable $\vartheta_d$ $(d=1,2,\ldots n)$, then the average loss rate of session $i$ can be derived as

$$L_{\vartheta_d} = \lim_{t\to\infty} \frac{L_{\vartheta_d}(0,t)}{t} = \lim_{n\to\infty} \frac{n(\sigma_i - k) + \rho_i \sum_{d=1}^{n} \vartheta_d}{n(\frac{\sigma_i}{\rho_i}) + \sum_{d=1}^{n} \vartheta_d} \tag{4-38}$$

When latency has a constant value of θ for every busy period, the average loss rate, denoted by $\overline{L}_i$, can be directly derived from (4-38) as

$$\overline{L}_i = \lim_{n\to\infty} \frac{n(\sigma_i - k_i) + \rho_i \sum_{d=1}^{n} \theta}{n(\frac{\sigma_i}{\rho_i}) + \sum_{d=1}^{n} \theta} = \frac{(\sigma_i - k_i) + \rho_i\theta}{\rho_i\theta + \sigma_i}$$

Since $\theta \geq \vartheta_d$, it is obvious that $\overline{L}_i \geq L_{\vartheta_d}$. The proof of the proposition is now completed. It is also worth our a while to see that if we want $\overline{L}_i$=0, the only requirement is $(\sigma_i - k_i) + \rho_i\theta$=0. That is $k_i = \sigma_i + \rho_i\theta$. This is just the zero loss buffer requirement for LR servers given by (4-3) in [13]. Our derivation verified this important result from another angle.

## 4.6. Conclusions

This chapter considers the performance of fractional service rate reserved QoS mechanisms. Our focus has been on the best known QoS mechanisms which can be classified together as LR servers. After briefly introducing the LR server model, this chapter has determined and proved the arrival process that will result in the maximal average loss rate for an individual session of an LR server. This process, which we have called the BOL policy, bursts over the latency period of the server. The worst case based maximal average loss rate is shown to depend on latency and the service rate allocated to the session under consideration. This work has extended the results in [25] to a more general QoS mechanism for LR servers. This is because GPS, the scheduling mechanism considered by Yee in [25], is a special LR server case, with latency $\theta=0$.

The issue of packet loss behaviors in LR servers has also been addressed. This study suggests that for a guaranteed service level, particularly under the worst case scenario, it is important to select scheduling mechanisms with smaller latency. By using the derivation of (4-6) and (4-8), for any given maximum packet loss requirement for LR servers, one can calculate the maximum buffer required.

On the other hand, the zero loss buffer requirement of the LR server system is also derived and it is consistent with that in [13].

# Chapter 5
# Case Study and Comparison

In this case study section, the study of the arrival process that follows the Bang Bang policy [25] is extended to the LR server. By using theorems derived in section 4.4, the average loss rate of session *i* is calculated under the worst case scenario. Upper bounds on the requirement of buffers at the LR server that will guarantee zero loss are derived and compared with the results of (4-18) when the arrival process follows the BOL policy. This study is conducted based on the assumption of that the traffic is leaky bucket constrained and $C_i=\infty$.

## 5.1. BANG BANG POLICY CASE

Consider a LR-server system with a finite input buffer $K_i$ for session *i*. $\theta$ is the latency of the scheduling mechanism used in the server. In this section, we will derive the worst case based average loss rate for session *i* for the case when the peak rate at which session *i* traffic can be input to the network is $C_i=\infty$. The arrival process that follows a Bang Bang policy is restated as

$$u_i(t) = \sigma_i \sum_{n=0}^{\infty} \delta(t - n * tp) \qquad C_i=\infty \qquad (5\text{-}1)$$

When $C_i<\infty$, $\forall t$, for $0\le t\le T_{ieb}+\sigma_i/\rho_i$, and the arrival process is

$$u_i(t) = C_i[u_{stp}(t)-u_{stp}(t-T_{ieb})] \qquad (5\text{-}2)$$

where $u_{stp}(t)$ is the unit step function. In each cycle, the amount of traffic lost due to buffer overflow is

$max[0,\ C_i{*}T_{ieb}-S_i(0,T_{ieb})-K_i]$

## 5.2. BANG BANG POLICY WHEN THE PEAK RATE OF THE OUTPUT FROM THE LEAKY BUCKET IS INFINITE, $C_i=\infty$

Under the worst case scenario, the service rate for session $i$ is $\mu_i=\mu_i^0$. With periodic arrivals according to (5-1), the packets that find the input buffer full are lost and not taken as actual arrivals. So the session busy period and the latency $\theta$ are based on actual arrivals and the service received by the session after $\theta$ for the worst case scenario. By using Theorems[1] 1 and 2, the arrival process and service behavior can be illustrated as in Figures 5-1 and 5-2 for an arbitrary $\theta$. The dotted line indicates a repeat of non-dotted line part.

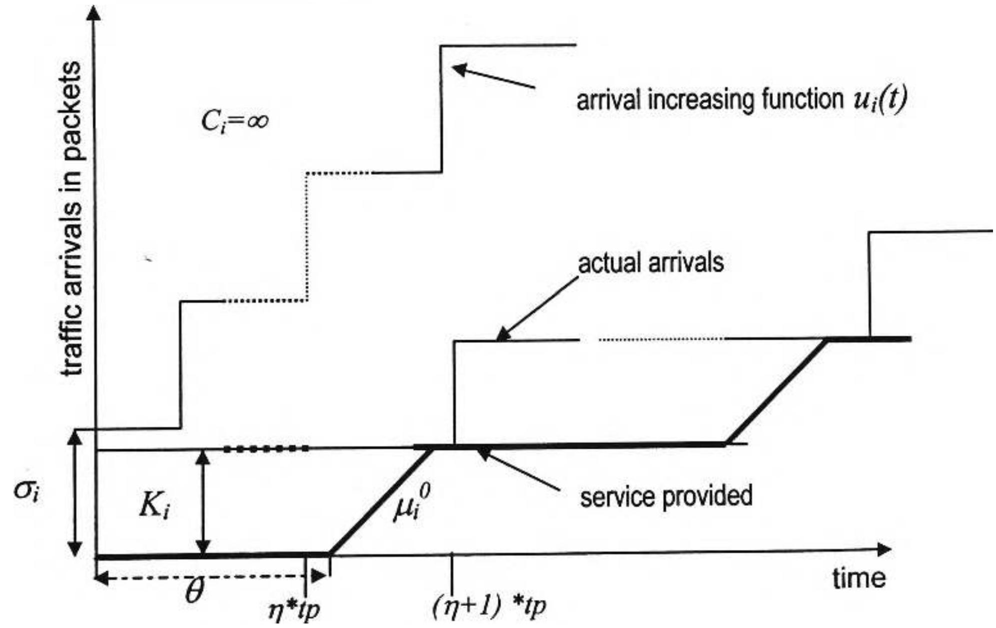



FIGURE 5-1. SESSION $i$ BUSY PERIODS WITH BANG BANG ARRIVALS AND SERVICE FUNCTION AT LR SERVERS WHEN $\eta{*}tp\le\theta\le(\eta+1)tp-K_i/\mu_i^0$, $\eta\ge0$.

[1] These two theorems are also applicable to (5-1) and (5-2). It is easy to prove these with a similar approach to that in section 4.4.

Figure 5-1 depicts the case when $C_i=\infty$ and the latency of the server lasts after the $\eta^{th}$ arrival periods, i.e. $\eta*tp\leq\theta\leq(\eta+1)tp-K_i/\mu_i^0$, $\eta\geq 0$. $K_i/\mu_i^0$ is the maximum time needed to empty the input buffer for session $i$. It is obvious that the input buffer will be emptied by $\theta+K_i/\mu_i^0$. So the packet loss caused by the finite buffer in any period of $(\eta+1)tp$ is $(\eta+1)\sigma_i-K_i$ and the average loss rate is

$$\frac{\sigma_i - \frac{K_i}{\eta+1}}{tp}$$

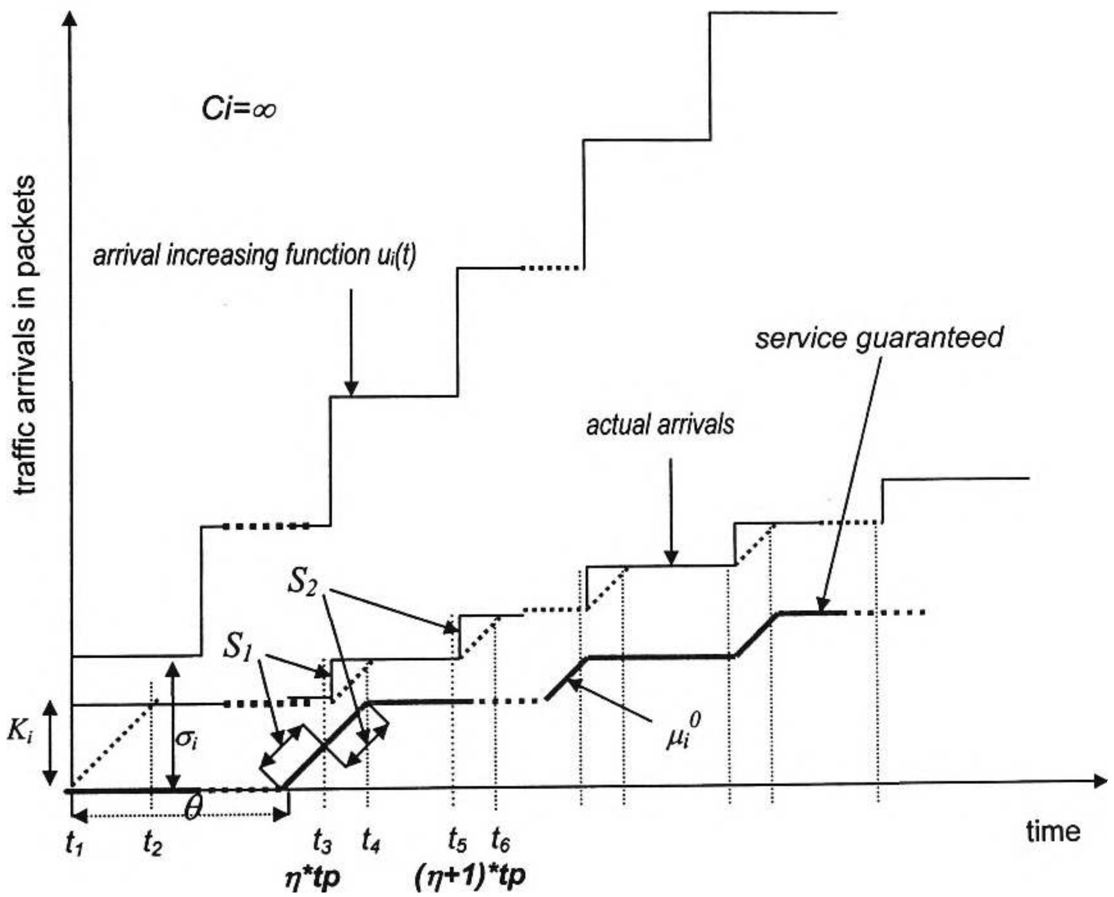



**FIGURE 5-2 SESSION $i$ BUSY PERIODS WITH BANG BANG ARRIVALS AND SERVICE FUNCTION AT LR SERVERS WHEN $\eta*tp>\theta\geq\eta*tp-K_i/\mu_i^0$, $\eta\geq 0$. ($T_1$,$T_2$), ($T_3$,$T_4$) AND ($T_5$,$T_6$) ARE ALL BUSY PERIODS OF SESSION $i$.**

Figure 5-2 shows the case when the latency of the server ends before the $\eta^{th}$ arrival periods, i.e. $\eta*tp>\theta_i \geq \eta*tp-K_i/\mu_i^0$. There are two scenarios need to be considered.

**Scenario A: When $S_2>S_1$ and $\eta\leq (m=1,2,3,...)$**

Let $mS_1 \geq S_2>(m-1)S_1$, $S_1$ and $S_2$ is the service received by session $i$ during time period $(\theta, \eta*tp)$ and $(\eta*tp, t^f)$respectively. From figure 5-2, it is obvious that where $K_i=S_1+S_2$. $S(\theta, \eta*tp)$ and $S(\eta*tp, t^f)$ is the minimum service received by session $i$ during $(\theta, \eta*tp)$ and $(\eta*tp, t^f)$ respectively. $t^f$ is the time when the last packet that arrived during the session busy period previous to t1 (refer to figure 5-2) left the server. For $\eta\leq m$ ($\eta\geq 0$), the packet loss in a time period of $(\eta+1)tp$ will be $(\eta+1)\sigma_i-(K_i+S_1)$ and the average loss rate is

$$\frac{\sigma_i - \frac{K_i+S_1}{\eta+1}}{tp}$$

**Scenario B: When $S_2>S_1$ and $\eta>m$ or $S_2\leq S_1$ (m=1,2,3,...)**

The average loss rate obtained in this scenario is

$$\frac{\sigma_i - \frac{K_i}{\eta}}{tp}$$

To summarize the above scenario, when the arrival process follows the Bang Bang Policy, the worst case based average loss rate of session $i$ in LR server system is

$$\overline{L} = \begin{cases} \dfrac{\sigma_i - \dfrac{K_i}{\eta+1}}{tp} & \eta * tp \le \theta < (\eta+1)tp - \dfrac{K_i}{\mu_i^0} \\ \dfrac{\sigma_i - \dfrac{K_i + S_1}{\eta+1}}{tp} & \eta * tp - \dfrac{K_i}{\mu_i^0} \le \theta < \eta * tp \text{ and } S_2 > S_1 \\ \dfrac{\sigma_i - \dfrac{K_i}{\eta}}{tp} & \eta * tp - \dfrac{K_i}{\mu_i^0} \le \theta < \eta * tp, \eta > 0 \text{ and } S_2 \le S_1 \\ \dfrac{\sigma_i - \dfrac{K_i + S_1}{2}}{tp} & tp - \dfrac{K_i}{\mu_i^0} \le \theta < tp \end{cases} \quad (5\text{-}3)$$

Compared with (4-18), a tighter bounds on the zero loss buffer requirement can be obtained from (5-3).

If $K_i$ is the upper bound of the zero loss buffer requirement for session $i$ with Bang Bang arrival process given by (5-1).

Since $\overline{L}$=0 when

$$K_i=(\eta+1)\sigma_i \qquad \text{for } \eta*tp\le\theta<(\eta+1)tp-K_i/\mu_i^0 \quad (5\text{-}4)$$

$$K_i=(\eta+1)\sigma_i-S_1 \qquad \text{for } \eta*tp-K_i/\mu_i^0\le\theta<\eta*tp \text{ and } S_2>S_1 \quad (5\text{-}5)$$

$$K_i=\sigma_i \qquad \text{for } tp-K_i/\mu_i^0\le\theta<tp \quad (5\text{-}6)$$

To substitute the $\theta$ in (5-4), with the smallest value, $\eta*tp-K_i/\mu_i^0$ which will make $S_1$ reach its maximum value, we immediately obtain the lower bounded $K_i$ that

$$K_i=(\eta+1)\sigma_i/2 \qquad \theta= \eta*tp-K_i/\mu_i^0 \quad (5\text{-}7)$$

The reason of choosing the smallest value of $\theta$ is to make sure that $K_i$ reaches its maximum value.

To substitute the $\theta$ in (4-18) with $\eta^*tp$, $tp\text{-}K_i/\mu_i^0$ and $\eta^*tp\text{-}K_i/\mu_i^0$, we have the following zero loss buffer requirement (denoted with $\Re_i$ ) respectively (Note that $\rho_i<\mu_i^0$):

$$\Re_i = (\eta+1)\,\sigma_i \qquad \theta=\eta^*tp \tag{5-8}$$

$$\Re_i = (\eta+1)\,\sigma_i - (\rho_i/\mu_i^0)\,K_i \qquad \theta=tp-K_i/\mu_i^0 \tag{5-9}$$

$$\Re_i = 2\sigma_i - (\rho_i/\mu_i^0)\,K_i \qquad \theta=\eta^*tp-K_i/\mu_i^0 \tag{5-10}$$

With comparing $K_i$ and $\Re_i$ in the pairs (5-4) and (5-8), (5-7) and (5-9), (5-6) and (5-10), it is found that $\Re_i \geq K_i$. This result indicated that the upper bound of the zero loss buffer requirements provided by (4-18) is conservative for the Bang Bang arrival process given by (5-1).

## 5.3. Simulations and numerical results

In chapter 4, we have determined that the BOL arrival process causes maximal average loss rate of a session in an LR server under the worst case scenario. The calculation of the maximal average loss rate is given by (4-6). In this section, we present simulations to verify this calculation. In addition, this section also presents numerical results to compare zero loss buffer requirement for arrival process that follows BOL and Bang Bang policy.

### 5.3.1. SIMULATIONS

Simulations are designed to verify the calculation of the worst case based maximal average loss rate for LR servers.

A single LR server employing the WFQ scheduling mechanism and variable length packets is assumed. There are three sessions attended by the server at a rate of 622Mbps. Service weights assigned to sessions 1, 2 and 3 are 33%, 21% and 46% respectively. The token generating rate for each session is 90% of its reserved service rate. The token bucket size is set at 51200 bytes while the input buffer size varies from 5120 to 54000 bytes. The IP packet length varies from 40 to 1500 bytes, according to the distribution stated in [32]. To generate the worst case for session 3, the arrival processes of the other two sessions are greedy (input packets to the maximal extent), so that session 3 only receives its reserved service rate. The arrival process of session 3 follows the BOL policy.

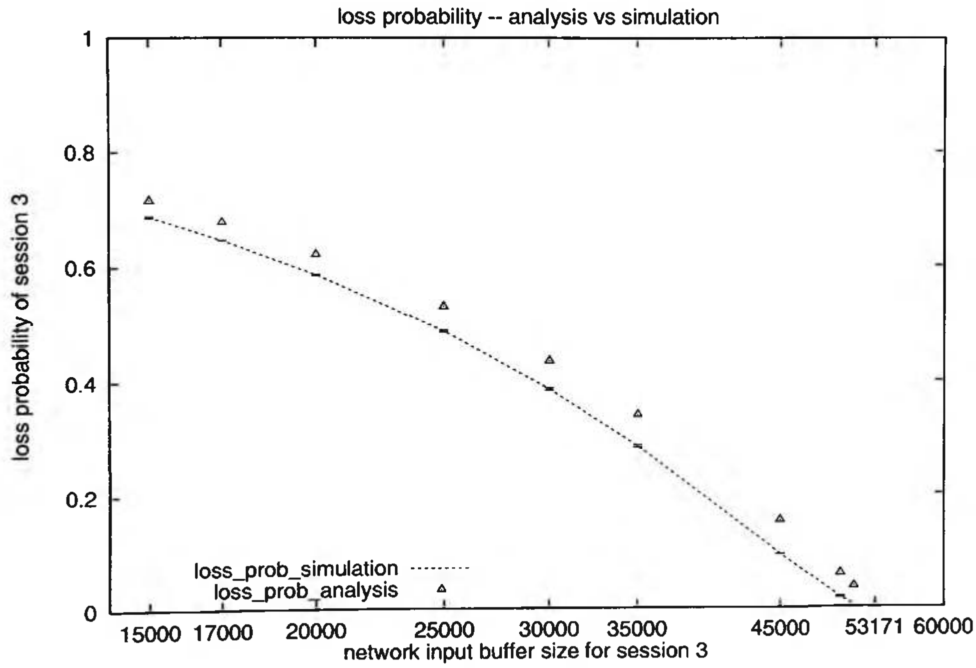



**FIGURE 5-3. THE WORST CASE BASED MAXIMAL AVERAGE LOSS RATE OF SESSION 3 WHEN ARRIVAL PROCESS IS BOL.**

Figure 5-3 shows the worst case based loss probability of session 3 against its input buffer size (using BOL policy). The figure indicates that the results from the simulation and the analytical calculation are quite close, and the simulation result is upper bounded by the analytical results.

### 5.3.2. NUMERICAL RESULTS

Having determined that the arrival processes which follow the BOL policy result in the maximal average loss rate, it is possible that, using the formulae derived from BOL policy in previous sections, the input buffer allocation can be made to ensure zero packet loss at an LR server. Because the zero loss buffer requirement given by the formulae for BOL arrivals upper bounds the requirements of other arrivals. However, it is necessary to know when to use this bound in dimensioning the input buffer for a network, how conservative it could be and what would be the savings if not use this bound when the arrival process does not follow the BOL.

This section, therefore, compares the BOL policy and Bang Bang policy to present

- How conservative the zero loss buffer allocation for an arrival process following BOL would be in comparing with the Bang Bang policy.
- The impact of packet length on the upper bound of the zero loss buffer requirement.

To apply the arrival processes follow BOL and Bang Bang policy to WFQ, a well known LR server, numerical data are collected against zero loss requirement.

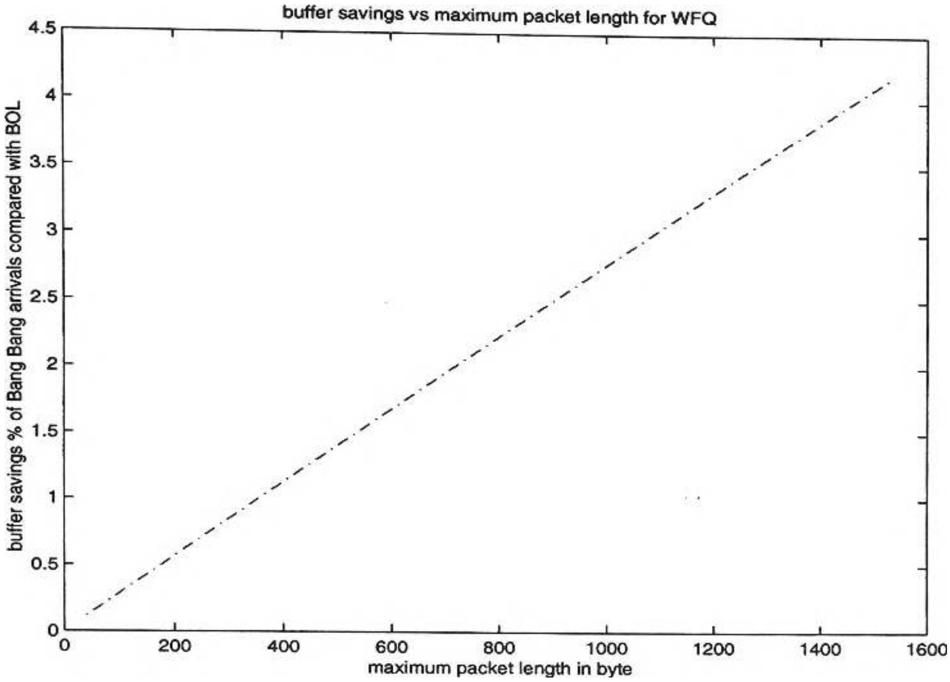



**FIGURE 5-4. BUFFER SAVINGS USING ARRIVAL PROCESS (5-1) COMPARED WITH USING PROCESS (4-5)**

Figure 5-4 compares the zero loss buffer requirement for the arrival processes of (4-5), i.e. BOL, and (5-1), i.e. the Bang Bang Policy. For Bang Bang arrival process, the buffer requirement for guaranteeing zero loss is less than that of for BOL arrival process. This also can be interpreted as buffer savings when dimensioning the input buffer for the network if arrival process is considered to follow Bang Bang policy other than BOL.

If the maximum packet length is 1500 byte, 5% more buffers required for the BOL process than the Bang Bang arrival process. This figure also implies that it may be too conservative to apply the zero loss buffer requirement for BOL to arrival process other than BOL.

## 5.4. Conclusions

This chapter, via case study, has extended the study of arrival process that follows the Bang Bang policy to the LR servers. The formulae used to calculate the maximal average loss rate for Bang Bang process is derived.

In particular, this chapter also, with simulations, verified the calculation of the maximal average loss rate of a session in an LR server under the worst case scenario. In addition, zero loss buffer requirements of the LR server system for the BOL and Bang Bang arrival processes are derived and numerically compared. From the comparison we find that the upper bound of the zero loss buffer requirement of BOL can be conservative when arrival process does not follow BOL policy.

# Chapter 6 Summary and Future Study

This dissertation has analysed the performance of typical QoS mechanisms in IP networks. These QoS mechanisms have been classified as either non-fractional service rate reserved scheduling mechanisms or fractional service rate reserved. With this classification, it is shown that most mechanisms supporting DiffServ are non-fractional service rate reserved mechanism, while most mechanisms supporting IntServ are fractional service rate reserved. The advantage of this classification is that it decomposes complex problems into separate comparatively simple problems. These simpler problems are in relation to the features of QoS that an IP network is supposed to provide to various kinds of applications. Among non-fractional service rate reserved mechanisms, the focus has been two fundamental scheduling mechanisms from which many others, including RED and RIO, are derived. Among fractional service rate reserved mechanisms, we specifically examine the loss behaviours of a broad range of QoS mechanisms called LR servers, to which most well known mechanisms such as GPS, WFQ (PGPS) and $WF^2Q$ belong.

Two key issues have been addressed in this dissertation. One is how well do the scheduling mechanisms which support Diffserv perform in providing various QoS levels. This issue has been addressed via a quantitative comparison of TD and PS, to examine the relative merits of these scheduling mechanisms in the DiffServ environment, and to develop suitable techniques in analysis modelling. The other issue is the performance behaviour, particularly the loss rate, of the various QoS mechanisms under the worst case scenario, when the input buffer of the server is finite. This performance issue has been addressed with analytical and simulation work to determine the worst case loss rate for LR servers and the arrival processes that result in the worst case.

## 6.1. SUMMARY OF THE DISSERTATION

As a result of the literature survey, this dissertation has identified the two key issues for performance analysis of QoS mechanisms in IP networks, namely the performance of DiffServ and IntServ. To address these issues, a classification is proposed which groups QoS mechanisms based on whether the service rate is reserved for different traffic classes.

Among the non-fractional service rate reserved QoS mechanisms, two basic DiffServ scheduling mechanisms, TD and PS, have been investigated. Our performance analysis of TD indicates that changing the load of the non-preferred flow has a minimal effect on packet loss of the preferred flow. Given a fixed total buffer size and identical arrival rate for both flows, there is a minimal improvement in loss for the non-preferred flow when its threshold increases. In PS, when the buffer allocation changes, a clear trade-off between packet loss and mean packet delay for the preferred and non-preferred flows is observed. From this comparison of TD and PS, based on the premise that both mechanisms provide the same level of packet loss for preferred flows, TD provides a lower packet loss and lower mean packet delay to the non-preferred flow than PS. However PS has the advantage over TD in providing a lower mean delay to the preferred flow. Simulations of PS with three traffic flows show that the flow with highest priority will meet the requirements of DiffServ Expedited Forwarding (EF). To extend our study, an approximate PS performance analysis is presented. It approximates packet loss and mean packet delay for non-preemptive PS. The accuracy of the approximation has been verified with simulations.

Among the fractional service rate reserved QoS mechanisms, a broad range of QoS mechanisms have been considered. Since future networks are more likely to be heterogeneous in deploying QoS mechanisms, a general model is required to analyse the performance of these mechanisms. The Latency Rate Server (LR Server) is just a such model that QoS mechanisms are characterised with only two parameters: Latency and Rate. The definition of LR server is based on the concept of session busy period, which depends only on the pattern of arrivals and the service rate reserved for the session. Because of this, the arrival process is a key factor affecting the packet loss rate of a session at an LR server.

After a brief introduction to LR Server and its theory and properties, this dissertation studied packet loss behaviours of the LR servers. In particular, the arrival process that results in the maximal average loss rate for individual sessions of LR server is determined. Formulae for calculating the loss rate are then derived and zero loss buffer requirements for LR servers are obtained.

With a case study, we also extended the work of [25] to a general case, where an LR server is employed (rather than only GPS), to look at the maximal average loss rate when the arrival process follows the Bang Bang policy. This study suggests that for guaranteed service level, it is important to select scheduling mechanisms with smaller latency. By using the derivation of (4-6) and (4-8), for any given maximum packet loss requirement at LR servers, one can calculate the maximum buffer required to guarantee a lower packet loss rate.

## 6.2. FUTURE STUDY

In addition to the work in this dissertation, it is important to look at maximal average loss rate of each session in the LR server if all sessions have the BOL arrivals. Analytical work to address this issue is left for future study. It is also noted that the end to end delay bound and zero loss buffer requirement for LR servers are derived with an assumption that the maximal rate at which traffic can be input to the network from leaky bucket is infinite [13][3][25], i.e.$C_i=\infty$. The case when $C_i<\infty$ is an open issue which needs to be addressed.

As discussed in the dissertation, it may in practice be necessary to combine the characteristics of IntServ and DiffServ in the future IP network. Some industry products have already been prepared for this transition. For example, Lucent has proposed its PacketStar 6400 series [36] to employ WFQ as a scheduling mechanism with buffer management techniques. Therefore, further investigation is required for the combined performance analysis of DiffServ mechanisms and IntServ Mechanisms.

# References

[1] 3Com, *Switches, http://www.3Com.com/products/switches.html.*

[2] Demers, A., Keshav, S. and Shenkar, S., *Analysis and Simulation of a Fair Queueing Algorithm.* Internet. Res. and Exper., 1990. **vol. 1**.

[3] Parekh, Abhay K. and Gallager, R.G., *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case.* IEEE/ACM Transaction on Networking, 1993. **Vol, 1**(No. 3).

[4] Ascend, *GRF Routers, http://ascend.com/300.html.*

[5] Betworks, B., *Accelar Routing Switches, http://business5.baynetworks.com/Mainbody.asp.*

[6] Blondia, C. *An M/G/1 Finit Capacity Queue with Vacation and Priorities. Performance'87.* 1987. Brussels, North Holland.

[7] Clark, D. and Fang, W. *Explicit Allocation of Best Effort Packet Delivery Service. Internet draft.* 1997.

[8] Comerford, R., *State of the Internet: Roundtable 4.0.* IEEE Spectrum, Oct. 1998.

[9] Cruz, R., *A Calculus for Network delay I: Network Elements in Isolation.* IEEE Transaction on Information Theory, 1991. **Vol 37**: p114-131.

[10] Awduche, D., Gan, D., Li, T., Swallow, G. and Srinivasan, V., *Extensions to RSVP for Traffic Engineering*, August 1998, IETF draft-swallow-mpls-rsvp-trafeng-00.txt.

[11] Jia, D.X., Dutkiewicz, E. and Chicharo, J., *Performance Analysis of QoS Mechanisms in IP networks. IEEE ISCC 2000, The Fifth International Symposium on Computers and Communications*. 2000. Paris, France.

[12] Stiliadis, D. and Verma, A., *Design and Analysis of Frame based Fair Queueing: A New Traffic Scheduling Algorithm for Packet-sweitched Network. ACM-SIGMETRICS'96*. 1996.

[13] Stiliadis, D. and Verma, A., *Latency-Rate Servers: A General Model for Analysis of Traffic Scheduling Algorithms. INFOCOM'96*. 1996.

[14] Clark, D.D. and Wroclaski, J., *An Approach to Service Allocation in the Internet. IETF Draf.* July 1997.

[15] Ferrari, D. and Delgrossi, L., *Charging for QoS. IWQOS'98*. May 1998. Napa, CA: IEEE/IFIP.

[16] Bertsekas, D. and Gallager, R., *Data networks*. 1992: Prentice Hall.

[17] Floyd, S., and Jacobson, V., *Random Early Detection gateways for Congestion Avoidance.* IEEE/ACM Transactions on Networking, August 1993. **Vol 1**(N.4): p397-413.

[18] Golestani, S.J. *A Self-Clocked Fair Queuing Scheme for Broadband Applications. IEEE INFOCOM'94*. 1994.

[19] Heinanen, J., *Differentiated Service in MPLS Networks*, . June 1999, IETF draft-heinenen-diffserv-mpls-00.txt.

[20] Howard, R.A., *Dynamic Programing and Markov Processes*, 1964, Cambridge, Massachusetts: The MIT Press.

[21] Ferguson, P. and Huston, G., *Quality of Service*. 1998: Wiley.

[22] Ferguson, P. and Huston, G., *Quality of Service in the Internet: Fact, Fiction, or Compromise? INET'98*. July 1998. Geneva, Switzerland.

[23] Stioica, I. and Abdel-Wahab, H., *Earlist Eligible Virtual deadline First: A flexible and Accurate Mechanism for Proportional Share Resources Allocation*, . 1995, Old Dominion University.

[24] Heinanen, J., Baker, F., Weiss, W. and Wroclawski, J., *Assured Forwarding PHB Group*, . June 1999, IETF RFC 2597.

[25] Yee, J.R. and Gaarder, N.T., *Maximal Average Loss Rate for a Single GPS Server System with Finite Buffers. IEEE INFOCOM'95*. 1995.

[26] Bennett, J.C.R. and Zhang, H., *Why WFQ Is Not Good Enough for Integrated Services Networks? Proceedings of NOSSDAV'96*. 1996.

[27] Bennett, J.C.R. and Zhang, H., *Hierarchical Packet Fair Queuing Algorithms. SIGCOMM '96*. 1996. Palo Alto, California.

[28] Bennett, J.C.R. and Zhang, H., *$WF^2Q$: Worst case Fair Weighted Fair Queueing. IEEE INFOCOM'96*. 1996. San Francisco, CA, USA.

[29] Jain, R. *Myths About Congestion Management in High-Speed Networks. Internetworking: Research and Experience*. 1992. Helsinki, Finland.

[30] Juniper, *M40 Series, http://juniper.net/products/default.htm*.

[31] Nicholos, K., Jacobson, V. and Zhang, L., *A Two-bit Differentiated Services Architecture for the Internet. IETF Draft*, Nov. 1997.

[32] Thompson, K., Miller, G.J. and Wilder, R., *Wide-Area Internet Traffic*

*Patterns and Characteristics*. IEEE Network, 1997: p10-23.

[33] Fendick, K.W., Kumar, V. P., Lakshman, T.V. and Stiliadis, D., *The PacketStar 6400 IP Switch- An IP Switch for the Converged Network*. Bell Labs Technical Journal, Oct-Dec. 1998. 4(Oct-Dec 1998): p32-45.

[34] Kilkki, K., *Differentiated Services for The Internet*. 1999: Macmillan Technical Publishing. p188-192.

[35] Kleinrock, L., *Queueing Systems*. Vol. I. 1975: John Wiley & Sons. 93-94.

[36] Lucent, *PacketStar 6400 Series*, *http://www.lucent.com:80/dns/products/ps6400.html.*

[37] Katevenis, M., Sidiropoulos, S. and Courcoubetis, S., *Weighted Round-Robin cell Multiplexing in a General-purpose ATM switch chip*. IEEE Journal on Selected Areas in Communications, 1991. **vol 9**: p1256-1279.

[38] May, M., Bolot, J.C., Marie, A.J. and Diot, C., *Simple Performance Models of Differentiated Services Schemes for the Internet. INFOCOM'99*. 1999. New York, USA.

[39] Xiao, X. and Ni, L.M., *Internet QoS: A Big Picture*. IEEE Network, March/April 1999: p8-18.

[40] Goyal, P., Vin, H.M. and Chen, H., *Start time Fair Queueing: A Scheduling algorithm for Intergrated Services*. *ACM-SIGCOMM 96*. 1996. Palo Alto, CA.

[41] Braden, R., Clark, D., and ShenkerR, S., *Integrated Service in the Internet Architecture: an Overview*, . June 1994, IETF.

[42] Callon, R., Freedman, N., Fredette, A., Swallow G., and Viswanathan, A., *A Framework for Multiprotocol Label Switching*, June 1999, IETF draft-ietf-mpls-framework-03.txt.

[43] Guerin, R. and PerisR, V., *Quality of Service in Packet Networks Basic Mechanisms and Directions.* Computer Networks, Feb. 1999. **vol. 31**(No. 3): p169-179.

[44] Cruz, R., *A Calculus for Network Delay I: Network Elements in Isolation.* IEEE Transaction on Information Theory, 1991A. **vol 37**: p114-131.

[45] Cruz, R., *A Calculus for Network Delay II: Network Analysis.* IEEE Transaction on Information Theory, 1991B. **Vol 37**: p132-141.

[46] Floyd, S. and Jacobson, V., *Random Early Detection Gateways for Congestion Avoidance. IEEEE/ACM Trans. Networking*, August 1993. **vol 1**(4): p397-413.

[47] Shenker, S., Braden, R. and ClarkS, D., *Integrated services in the Internet Architecture : an Overview. Internet RFC 1633.* 1994.

[48] Suri, S., Varghese, G. and ChandranmenonS, G., *Leap Forward Virtual Clock. INFOCOM 97.* Kobe, Japan.

[49] Sahu, S., Towsley, D. and Kurose, J., *A Quantitative Study of Differentiated Services for Internet*, 1999, Umass CMPSCI Technical Report.

[50] Sisco, *12000 Series, http://www.sisco.com/warp/public/733/12000.*

[51] Stiliadis, D., *Traffic Scheduling in Packet-Switch Networks: Analysis, Design, and Implementation, Computer Engineering Department.* 1996,

University of California: Santa Cruz.

[52] Turner, J., *New Directions in Communications, or Which way to the Information age?* IEEE Communication Magazine, 1986. **vol. 24**: p 8-15.

[53] Jacobson, V., Nichols, K. and Poduri, K., *An Expedited Forwarding PHB*, . June 1999, IETF RFC 2598.

[54] Kelton, W. D., Sadowski, R. P.and Sadowski, D. A., *Simulation with Arena*. International Editions 1998 ed. 1998, Boston: WCB Mc Graw Hill.

[55] Bernet, Y., Yavatkar, R., Ford, P., Baker, F., Zhang, L., Nichols, K., Speer, M.and Braden, R., *Interoperation of RSVP/Int-Serv and Diff-Serv Netrworks*. IETF draft, February 1999.

[56] M. A. Labrador and S. Banerjee, *Packet Dropping Policies for ATM and IP Networks*, IEEE Communications Surveys, Third Quarter 1999, vol 2 no. 3

[57] H. Zhang, *Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks*, Proceedings of IEEE. Vol 83, no 10, October 1995

# Appendix Publications

## TITLES OF PUBLICATIONS

**I. PERFORMANCE ANALYSIS OF QOS MECHANISMS IN IP NETWORKS**
ISCC2000

**II. UNDERSTANDING TRAFFIC BEHAVIOR – AN APPROXIMATION METHOD FOR COMPUTING PACKET LOSS AND MEAN PACKET DELAY IN NON-PREEMPTIVE FINITE PRIORITY QUEUES**
PAM2000

# Performance Analysis of QoS Mechanisms in IP Networks

Dix Jia, Eryk Dutkiewicz, Joe F. Chicharo

*The Switch Network Research Centre*
*University of Wollongong, Australia*
dix@snrc.uow.edu.au

## Abstract

*Integrated services IP networks are expected to provide a variety of services with differentiated QoS. This requires the implementation of mechanisms that can discriminate service classes in terms of QoS. The IETF has recently proposed a Differentiated Services (Diffserv) framework for provision of QoS. In this paper we analyse performance of two Diffserv mechanisms: Threshold Dropping and Priority Scheduling in terms of packet loss and mean packet delay. A comparison of the two mechanisms is carried out with the requirement that both mechanisms provide the same level of packet loss for the preferred flow. This comparison extends the results reported in the literature for these two mechanisms. In particular, in this paper we determine the impact of buffer threshold and buffer size on packet loss and mean packet delay in these mechanisms.*

*Keywords*— Diffserv, QoS, Threshold Dropping, Priority Scheduling.

## INTRODUCTION

Rapid growth of new applications and the need for differentiated Quality of Service (QoS) has increased the demand for better performance and flexibility of the Internet to support both existing and emerging applications. The current Internet offers best effort service to all users and is inadequate for those applications with more stringent QoS requirements. Differentiated Services (Diffserv) framework has been proposed by the IETF [6][7][8][9]. In Diffserv, packets are tagged with different priorities according to their service classes. Service differentiation is achieved when packets are processed and forwarded by Diffserv mechanisms according to packets' priorities. Efficient support of different QoS services, however, may require the implementation of different QoS mechanisms in different parts of a network.

A number of QoS mechanisms have been proposed in literature including Threshold Dropping (TD) [8], Priority Scheduling (PS) [9], Random Early Detection (RED) [11], RED with In and Out profile packets (RIO) [3] and Weighted Fair Queuing (WFQ) [1][2][10]. TD and PS can be regarded as basic mechanisms from which the other mechanisms have been derived. Hence comparative performance of these two mechanisms in providing required QoS is an important issue. The results can be used to choose the appropriate mechanism to provide the required QoS for particular applications in the most efficient manner. The above mechanisms have been analysed in the literature to a certain extent. These include the analysis of RIO in [4] and WFQ in [1] and TD and PS in [5]. However, the important issue of how to engineer these mechanisms for optimal performance still needs to be tackled. In this paper we carry out a performance comparison of the TD and PS mechanisms with the aim of providing the same level of packet loss to the preferred flow. Our comparison allows us to determine resultant packet loss for the non-preferred flow and mean packet delay for both the preferred and non-preferred flows as a function of various parameters of the two mechanisms.

The paper is structured as follows. Section 2 briefly describes the operation of the TD and PS mechanisms. Section 3 presents a performance comparison of the mechanisms in terms of packet loss and mean packet delay. The impact of the threshold setting and buffer partitioning on the relative performance of the two mechanisms is also examined in this section. Section 4 concludes the paper.

# Overview of TD and PS Mechanisms

## Threshold Dropping

A threshold dropping mechanism is depicted in Figure 1. Two arrival flows are considered: preferred flow and non-preferred flow. The preferred flow consists of packets which are tagged in profile (i.e. which do not violate their traffic contract) and the non-preferred flow consists of packets which are tagged out of profile. Preferred flow should receive preferential treatment with respect to the non-preferred flow. This is achieved in the TD mechanism by setting a threshold S. Non-preferred flow packets which arrive to the system when the queue length exceeds S are dropped. On the other hand preferred flow packets are only dropped when the queue length reaches the buffer size M.

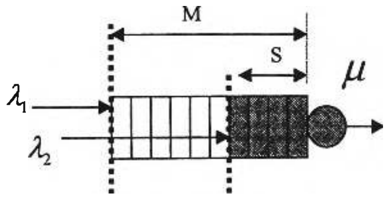
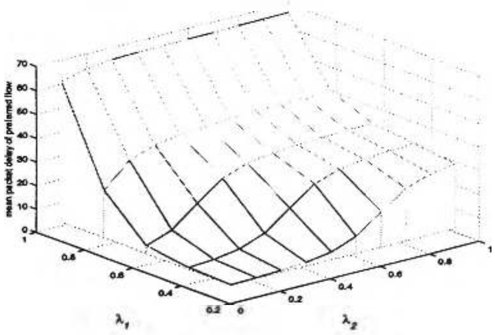
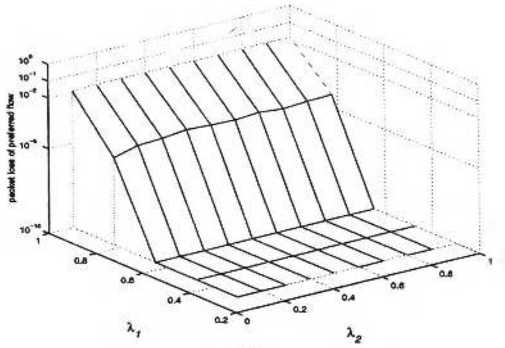



**Figure 1.** Threshold dropping mechanism with two packet flows

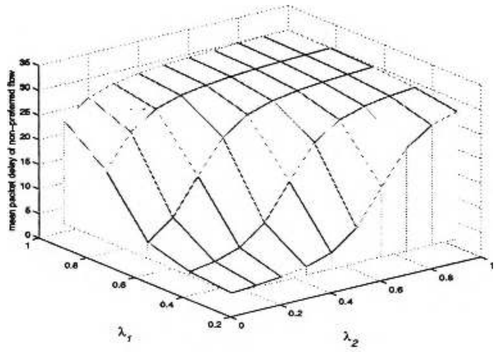
Figures 2 and 3 show simulation results for the TD mechanism under various load and threshold conditions. These results were obtained assuming that preferred and non-preferred flows were Poisson with mean arrival rate $\lambda_1$ and $\lambda_2$, respectively. Packet service time was assumed to be exponential. The mean packet delay is normalised with respect to service time. No flow control and packet re-transmission were considered



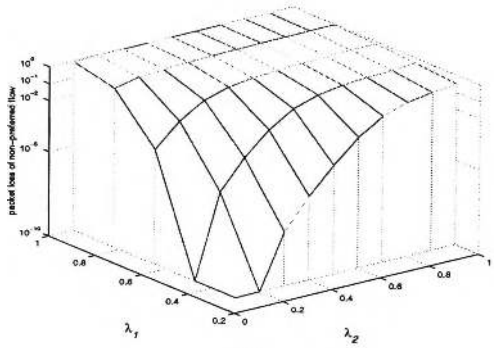
**(a)**



**(b)**



**(c)**

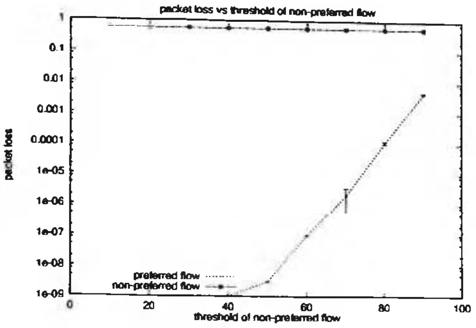


(d)

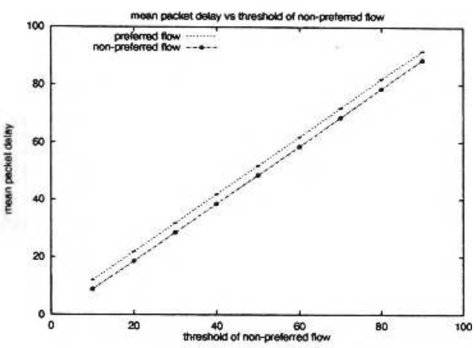
**Figure 2.** Loss and delay behaviors of TD mechanism under various load from both flows. (Buffer settings: M=100, S=30).

Figure 2 shows packet loss and mean packet delay as a function of $\lambda_1$ and $\lambda_2$ (normalised with respect to $\mu$). In this figure the buffer size was set to M = 100 and the threshold was set to S = 30. As expected, increasing the load of the non-preferred flow has little effect on packet loss experienced by the preferred flow. The mean packet delays of both flows are bounded by their thresholds.

Figure 3 shows the impact of threshold S on packet loss and mean packet delay of the preferred and non-preferred flows. In this figure both flows had a fixed load of 0.7, the total buffer size was set to M = 100 and the threshold value S was varied from 10 to 90. Under the above conditions increasing the threshold value results in little improvement in packet loss of the non-preferred flow. However, packet loss of the preferred flow increases sharply as the threshold is increased beyond approximately 40. Increasing the threshold leads to a linear increase in the mean packet delay for both flows.




(a)




(b)

**Figure 3.** Impact of threshold of non-preferred flow on packet delay and loss

# PRIORITY SCHEDULING

A priority scheduling mechanism handling to packet flows is depicted in Figure 4. Packets belonging to the preferred flow receive non-preemptive priority over packets belonging to the non-preferred flow. Buffer sizes for the preferred and non-preferred flows are set to K and L, respectively.

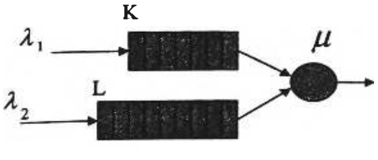
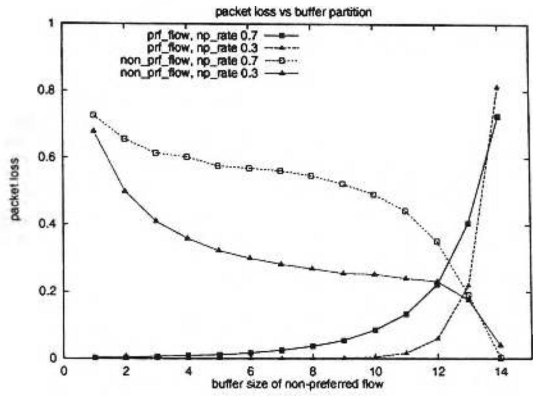
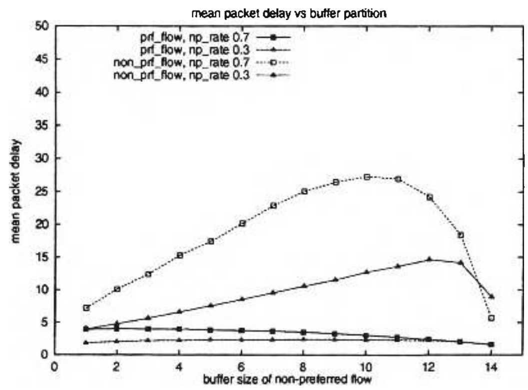



**Figure 4.** Priority Finite Queues

Figure 5 shows simulation results for packet loss and mean packet delay experienced by the preferred and non-preferred flows in the PS mechanism as a function of the buffer size L allocated to the non-preferred flow. The total buffer size (K+L) was set to 15 and preferred and non-preferred flows were Poisson with mean arrival rate $\lambda_1$ and $\lambda_2$, respectively. Packet service time was assumed to be exponential. The mean packet delay is normalised with respect to service time. No flow control and packet re-transmission was considered.




(a)




(b)

**Figure 5.** Packet loss and mean packet delay vs buffer partition for various of np_rate ($\lambda_2$). Normalized arrival rate of preferred flow ($\lambda_1$) is 0.7.

Figure 5 shows a clear trade-off between packet loss and mean packet when the buffer allocation is changed. Mean packet delay curves for non-preferred flow show interesting behavior when buffer space allocated to non-preferred traffic is varied. The mean packet delay for non-preferred flow is small when the buffer space allocation is either small (less than 2) or large (more than 12). This is because when the allocated buffer size is small, the mean delay is bounded by the small buffer size. When more buffer space is allocated to non-preferred flow, however, the buffer space left for preferred flow will be decreased due to the constant total buffer size. Under this scenario, packets from the non-preferred flow will spend less time waiting for the queue of the preferred flow to become empty. This behavior is due to the fact that we ignore packet re-transmission in our simulation and only consider the mean delay of those packets which were not dropped from the queue.

## Performance Comparison of TD and PS Mechanisms

In this section we present the results of a number of simulations carried out to obtain relative performance of the two mechanisms. We set the two mechanisms with the same total buffer space of 15 packets and the same link capacity (normalized to 1). As in earlier tests the preferred and non-preferred flows were modeled as Poisson processes. For given arrival rates of both flows, we varied the threshold $S$ in the TD mechanism and the buffer size $K$ in the PS mechanism until the same level of loss probability for the preferred flow was obtained from both mechanisms. We then compared the resulting packet loss of the non-preferred flow and the mean packet delay of both flows between these two mechanisms. The packet loss and mean packet delay results are shown in Figure 6 and Figure 7, respectively. The mean packet delay is normalized with respect to service time. Normalized arrival rate of non-preferred flow in both figures is 0.7.

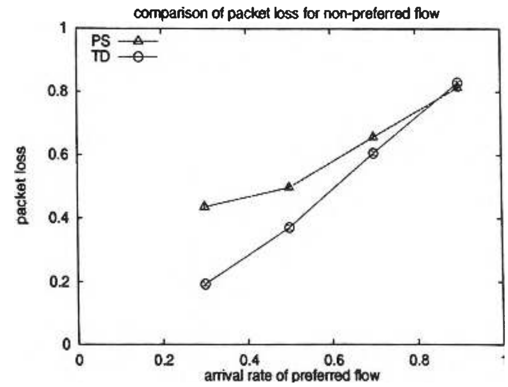



**Figure 6.** Packet loss Comparison

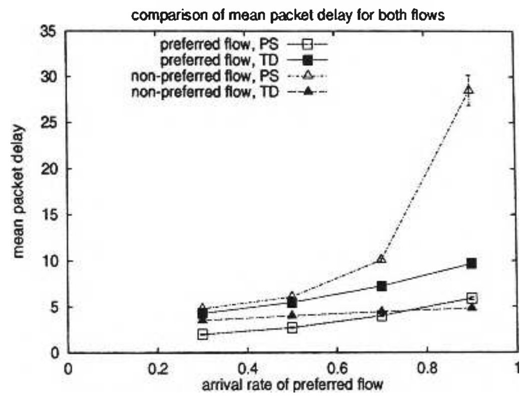



**Figure 7.** Mean Packet Delay Comparison

The results of Figure 6 indicate that the TD mechanism has better performance in terms of packed loss for the non-preferred flow when the load of the preferred flow is light. When the load is heavy the difference in packet loss between the two mechanisms is negligible. The results of Figure 7 indicate that as the load of the preferred flow changes, the PS mechanism provides a smaller mean delay to the preferred flow than does the TD mechanism. However, the TD mechanism results in a smaller mean delay for the non-preferred flow.

## Conclusion

Threshold dropping (TD) and priority scheduling (PS) are two fundamental mechanisms that can provide the ability to discriminate between QoS of traffic classes in Diffserv. Our performance investigation of the TD mechanism indicated that changing the load of the non-preferred flow has a minimal effect on packet loss of the preferred flow. With a fixed total buffer size and the same arrival rate of both flows, there is a minimal improvement in loss for the non-preferred flow when its threshold is increased. The mean packet delays for both flows are bounded by their thresholds. A clear trade-off between packet loss and mean packet delay for the preferred and non-preferred flows is observed in the PS mechanism when the buffer allocation is changed. The PS mechanism has the advantage over the TD mechanism in providing a lower mean delay to the preferred flow when the two mechanisms are engineered so as to provide the same level of packet loss for the preferred flow. However, under the same scenario, the TD mechanism provides lower packet loss and mean packet delay to the non-preferred flow.

## REFERENCES

[1] J.C.R. Bennett and H. Zhang, ``Why WFQ Is Not Good Enough for Integrated Services Networks?", *Proceedings of NOSSDAV'96*, Apr, 1996.

[2] J.C.R. Bennett and H. Zhang, "Hierarchical Packet Fair Queuing Algorithms," *Proceedings, SIGCOMM '96*, Palo Alto, California, August 1996.

[3] Clark, D., Fang, W., "Explicit Allocation of Best Effort Packet Delivery Service". *Internet draft*, Sept. 1997

[4] Martin May, J.C. Bolot, Alain J. Marie and C. Diot, "Simple Performance Models of Differentiated Services Schemes for the Internet", *Proceedings of INFOCOM'99*, New York, March 1999

[5] Sambit Sahu, Don Towsley, Jim Kurose, "A Quantitative Study of Differentiated Services for Internet", *Umass CMPSCI Technical Report* 99-09.

[6] S. Blake, D. Black, M. Davies, Z. Wang, W. Ewiss, "An Architecture for Differenciated Services", *RFC 2475*, December 1998.

[7] Y. Bernet, J. Biner, S. Blake, M. Carlson, B. E. Carpentyer, E. Davies, B. Ohlman, S. Keshav, D. Verma, , Z. Wang, W. Weiss, "A Framework for Differentiated Services", *Internet Draft*, February 1999.

[8] D.D. Clark, J. Wroclaski, "An Approach to Service Allocation in the Internet", *IETF Draft*, July 1997.

[9] K. Nicholos, V. Jacobson, L. Zhang, "A Two-bit Differentiated Services Architecture for the Internet", *IETF Draft*, Nov. 1997.

[10] Hui Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks", *Proceedings of The IEEE*, Vol. 83, NO. 10, October 1995.

[11] S. Floyd, V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", *IEEEE/ACM Trans. Networking*, 1(4):397-413, August 1993.

# Understanding Traffic Behavior –An Approximation Method for Computing PacketLoss and Mean Packet Delay in Non-Preemptive Finite Priority Queues

Dix Jia, Joe F. Chicharo, Eryk Dutkiewicz

*The Switch Network Research Centre*
*University of Wollongong, Australia*
dix@snrc.uow.edu.au

***Abstract*–*Forthcoming Internet will have the ability of providing differentiated Quality of Service (QoS) to different traffic classes. Understanding the performance of the models used in discriminating different service classes is an important issue. Priority Scheduling (PS) is a model that can discriminate traffic classes and engineer Expedited traffic Forwarding (EF) in Internet [8]. In this paper, we analyze the performance of Priority Scheduling (PS) and determine the impact of its buffer partition on the mean packet delay and packet loss. In particular, in order to simplify the approach of obtaining the mean packet delay and packet loss probability, an analytical approximation method is proposed for non-preemptive finite priority queues. Results from the approximation method under various scenarios are verified with simulation.*

*Index Terms-- Quality of Service (QoS), Priority Scheduling (PS), Expedited traffic Forwarding (EF).*

## I INTRODUCTION

Future Internet will be able to support service differentiation in terms of QoS. This requires sufficient models to be engineered in different parts of a network. A number of queuing models have been proposed in literature including Threshold Dropping (TD) [9], Priority Scheduling (PS) [10], Random Early Detection (RED) [12], RED with In and Out profile packets (RIO) [3] and Weighted Fair Queuing (WFQ) [1][2][11]. PS can be regarded as one of the basic mechanisms from which the other mechanisms have been derived. Hence understanding performance of PS in providing required QoS is an important issue. In this paper, we carry out simulations to analyze the performance of PS where the impact of buffer partition on the mean packet delay and packet loss are examined. On the other hand, non-preemptive finite priority queues arise naturally as models of communication systems [13]. In order to simulate practical situation, we look at traffic behaviors of non-preemptive priority queues in terms of packet loss and mean packet delay. With non-preemptive priority rules, a packet under service is allowed to complete service without interrupt even if a packet of higher priority arrives in the meantime. In [5], Bertsekas and Gallager gave a solution to packet loss probability distribution and mean packet delay for infinite non-preemptive priority queues that share a single server. But this solution assumes the service rate of both individual queues is known. However, these service rates are not known and still need to be obtained in this case study. Similar approach by Sahu *et al* [7] also needs to known service rate for individual queue. May *et al* in [6] only give out a solution to high priority queue. Both [6] and [7] are assuming the buffer is scheduled according to preemptive rule. Blondia in [13] provides a method to calculate queue length distribution and waiting time based on embedded Markov chain and recursive formula. But this method is difficult and complicated due to the recursive formulas for computing the Laplace Transform of busy period of preferred flow and blocking time of non-preferred flow. Apart from these, there is less discussion on this issue in literature. In this paper, a method to analytically approximate the packet loss probability and mean packet delay of finite non-preemptive priority queues for two traffic flows is proposed. Results of the approximation method are then verified with simulations.

The paper is structured as follows. Section 2 overviews the performance of PS, in particular, determines the impact of buffer partition on packet loss and mean packet delay. In section 3, an analytical approximation method has been proposed for computing the packet loss and mean packet delay of non-preemptive finite priority queues with two traffic classes. Results from both analytical approximation method and simulation are then compared. Section 4 concludes the paper.

## II PRIORITY SCHEDULING

A non-preemptive finite priority queuing model handling two traffic classes (preferred and non-preferred flows) is depicted in Figure 1. The packets of preferred class (preferred flow) will be served first. When the queue of preferred flow is empty, the packets from non-preferred flow will be served. We assume here that the packets arrive at the server according to Poisson process and the service time is exponentially distributed. Buffer sizes for the preferred and non-preferred flows are set to K and L, respectively. Service rate $\mu$ is normalized to 1.

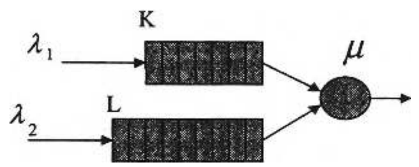



**Figure 1:** Priority finite Queues

Figure 2 shows simulation results for packet loss and mean packet delay experienced by the preferred and non-preferred flows in the PS mechanism as a function of the buffer size L allocated to the non-preferred flow. The total buffer size (K+L) was set to 15 and preferred and non-preferred flows were Poisson processes with mean arrival rates $\lambda_1$ and $\lambda_2$, respectively. No flow control and packet re-transmission were considered

The results show a clear trade-off between packet loss and mean packet delay when the buffer partition is changed. Mean packet delay curves for non-preferred flow show interesting behavior when buffer space allocated to non-preferred traffic is varied. The mean packet delay for non-preferred flow is small when the buffer space allocation is either small (less than 2) or large (more than 12). This is because when the allocated buffer size is small, the mean delay is bounded by the small buffer size. When more buffer space is allocated to non-preferred flow, however, the buffer space left for preferred flow will be reduced due to the constant total buffer size. Under this scenario, packets from non-preferred flow will spend less time waiting for the queue of preferred flow being empty. This behavior is due to the fact that we ignore packet re-transmission in our simulation and only consider the mean delay of those packets that were not dropped from the queue.

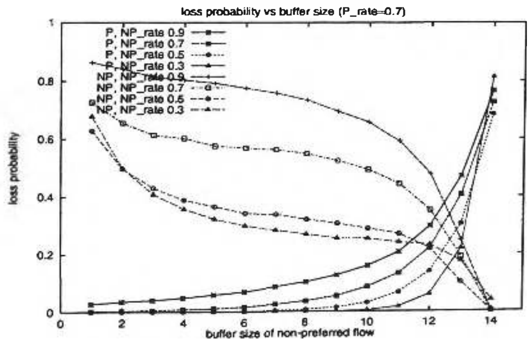
In Figure 2, we denote:

NP curves of non-preferred flow.

P curves of preferred flow.

NP_rate mean arrival rate of non-preferred flow.

P_rate mean arrival rate of preferred flow.



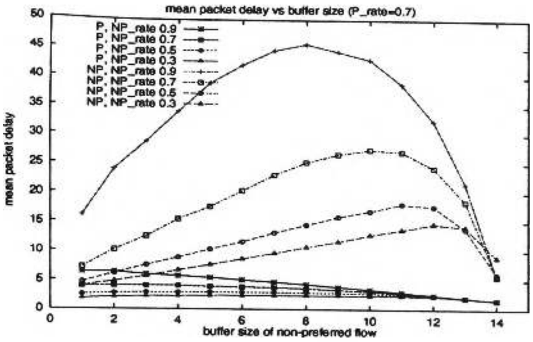



**Figure 2:** Packet loss and mean packet delay vs buffer partition for various values of NP_rate ($\lambda_2$). P_rate ($\lambda_1$) is set to 0.7 in this study.

## III APPROXIMATION METHOD FOR PRIORITY SCHEDULING

### AN APPROXIMATION METHOD FOR PRIORITY SCHEDULING WITH TWO SERVICE CLASSES

Assume that there is a router deployed with non-preemptive finite priority queues sharing a single processor. Also assume that there are 2 classes of traffic (packet flows) with different service preferences where class one has priority over class two. Both two flows arrive at the router according to Poisson process with exponential distributed service time. The mean arrival rate of class one and two is $\lambda_1$ and $\lambda_2$ respectively. Separate queue is maintained for each class. Since the buffers for both queues are finite (assuming size $K$ is assigned to queue with high priority and size $L$ is assigned to queue with low priority), all the packets that find queues full are dropped. Packet retransmission is not considered in this study. We intend to work out the packet loss probability and the mean packet delay experienced by both flows. The queueing model can be referred as figure 1 in section 2.

Denote:

$P_{i,j}$ probability of $j$ packets are found in queue $i$.

$\mu$ service rate of the processor which is normalized to $1$

$\mu_i$ service rate for queue $i$

$\lambda_i$ mean arrival rate of flow $i$.

$\varepsilon$ a factor that reducing the service rate of preferred flow due to the non-preemptive service rule.

$\rho_i = \dfrac{\lambda_i}{\mu_i}$ utilization factor for queue $i$

$\alpha_i$ the rate that packets are accepted by queue $i$.

$NQ_i$ the average number of packets in queue $i$.

$R$ the mean residual time of packet in server

Because after some $K_0$ ($K_0$=$K$ for the queue with high priority and $L$ for queue with low priority. For example, when $K_0$=$K$, all following packets from high priority flow will be dropped and hence $\lambda_i$=0) $\dfrac{\lambda_i}{\mu_i} < 1$, so all states in this process will be ergodic [4] and there exists the equilibrium probabilities $\{P_i\}$. Under steady state, we have

$$P_{1,0} = \frac{1}{1+\sum_{n=1}^{K}\left(\frac{\lambda_1}{\mu_1}\right)^n} = \frac{1-\frac{\lambda_1}{\mu_1}}{1-\left(\frac{\lambda_1}{\mu_1}\right)^{K+1}} \quad (1)$$

$$P_{2,0} = \frac{1}{1+\sum_{n=1}^{L}\left(\frac{\lambda_2}{\mu_2}\right)^n} = \frac{1-\frac{\lambda_2}{\mu_2}}{1-\left(\frac{\lambda_2}{\mu_2}\right)^{L+1}} \quad (2)$$

$$P_{1,K} = P_{1,0}\left(\frac{\lambda_1}{\mu_1}\right)^K = \frac{\left(1-\frac{\lambda_1}{\mu_1}\right)\left(\frac{\lambda_1}{\mu_1}\right)^K}{1-\left(\frac{\lambda_1}{\mu_1}\right)^{K+1}} \quad (3)$$

$$P_{2,L} = P_{2,0}\left(\frac{\lambda_2}{\mu_2}\right)^L = \frac{\left(1-\frac{\lambda_2}{\mu_2}\right)\left(\frac{\lambda_2}{\mu_2}\right)^L}{1-\left(\frac{\lambda_2}{\mu_{21}}\right)^{L+1}} \quad (4)$$

The basic idea of this approximation is to decompose the joint queues in figure 1 into equivalent two individual queues with derivable equivalent service rates. This will make the above probabilities obtainable. Since the priority is given to flow one (preferred flow), the flow two (non-preferred flow) can only get serviced during the processor is idle and the queue one is empty as well. If we approximate the service rate of class one $\mu_1$ with $\mu$ (the reason is $\mu_1 = \mu$ - $\varepsilon$ and $\varepsilon$ tends to be a very small value), then the service rate for class two can be derived as

$$\mu_2 = \left(1 - \frac{\alpha_1}{\mu}\right)\mu = \mu - \lambda_1 \left[ 1 - \frac{\left(1 - \frac{\lambda_1}{\mu}\right)\left(\frac{\lambda_1}{\mu}\right)^K}{1 - \left(\frac{\lambda_1}{\mu}\right)^{K+1}} \right] \quad (5)$$

From now on, the system can be equivalently decomposed as two individual queues with

$$\alpha_1 = \lambda_1 P_{1,0} + \lambda_1 P_{1,1} + \ldots + \lambda_1 P_{1,K-1} = \lambda_1 (1 - P_{1,K}) \quad (6)$$

$$\alpha_2 = \lambda_2 P_{2,0} + \lambda_2 P_{2,1} + \ldots + \lambda_2 P_{2,L-1} = \lambda_2 (1 - P_{2,L}) \quad (7)$$

In equilibrium, the average number of packets in both queues are derived as

$$NQ_1 = \sum_{n=0}^{K} nP_{1,n} = \frac{\rho_1 (1 - (K+1)\rho_1^K + K\rho_1^{K+1}}{(1 - \rho_1)(1 - \rho_1^{K+1})} \quad (8)$$

$$NQ_2 = \sum_{n=0}^{L} nP_{2,n} = \frac{\rho_2 (1 - (L+1)\rho_2^L + L\rho_2^{L+1}}{(1 - \rho_2)(1 - \rho_2^{L+1})} \quad (9)$$

According to Little's formula, the average waiting time for the packets in both queues are

$\frac{NQ_1}{\alpha_1}$ and $\frac{NQ_2}{\alpha_2}$

The mean residual time in server is [5]

$$R = \frac{1}{2} \sum_{i=1}^{2} \alpha_i \overline{\chi_i^2} \quad (10)$$

where $\overline{\chi_i^2}$ is the second moment of service time. When service times are exponentially distributed [5], $\overline{\chi_i^2} = \frac{2}{\mu_i^2}$. Since the packet in the processor is served at the same rate $\mu$ no matter which flow the packet belongs to, the mean residual time can be derived as

$$R = \frac{1}{2} \sum_{i=1}^{2} \alpha_i \overline{\chi_i^2} = \frac{1}{\mu^2} \sum_{i=1}^{2} \alpha_i \quad (11)$$

So the mean packet delays for flow one and flow two (preferred flow and non-preferred flow) are

$$Delay_1 = R + \frac{NQ_1}{\alpha_1} \quad (12)$$

$$Delay_2 = R + \frac{NQ_2}{\alpha_2} \quad (13)$$

## NUMERICAL RESULTS

The accuracy of the approximation will be affected if $\varepsilon$ increased. This can take place when the possibility that the server attends the packets of non-preferred flow increased. That is

1) The buffer size of non-preferred flow is very large (compared with preferred flow) or
2) The load of non-preferred flow is heavy (such as the load factor of non-preferred flow equal to 2.0).

In order to verify the efficiency and accuracy of the approximation method, simulation is carried out to implement the above situations.

First, we let the buffer size of non-preferred flow to be large (100 packets with comparison of 4 packets of preferred flow) and the load of preferred flow be moderate (load factor is 0.5). Then we observe the packets loss and mean packet delay for both flows while varying the load of non-preferred flow. The focus in this case has been the impact of non-preferred flow on packet loss when the load of non-preferred flow is more than 0.4. The comparison of results from simulation and approximation is shown in Figure 3. Mean packet delay is normalized with respect to service time.

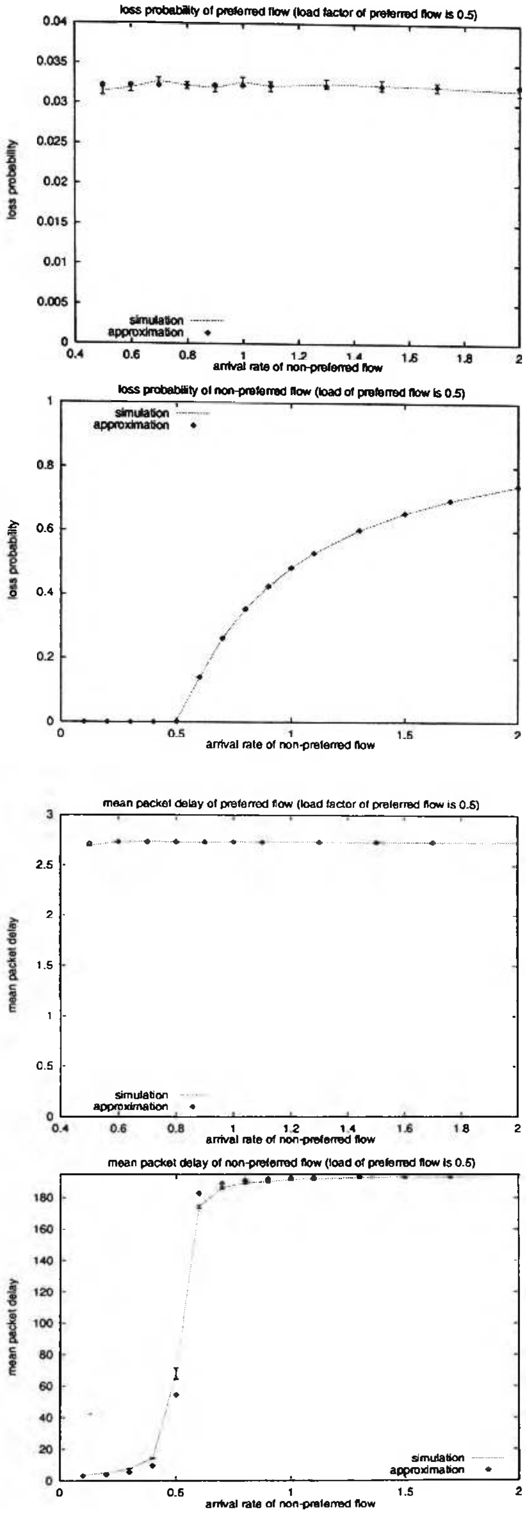



**Figure 3:** Comparison of results from simulation and approximation when buffer size of non-preferred flow is large.

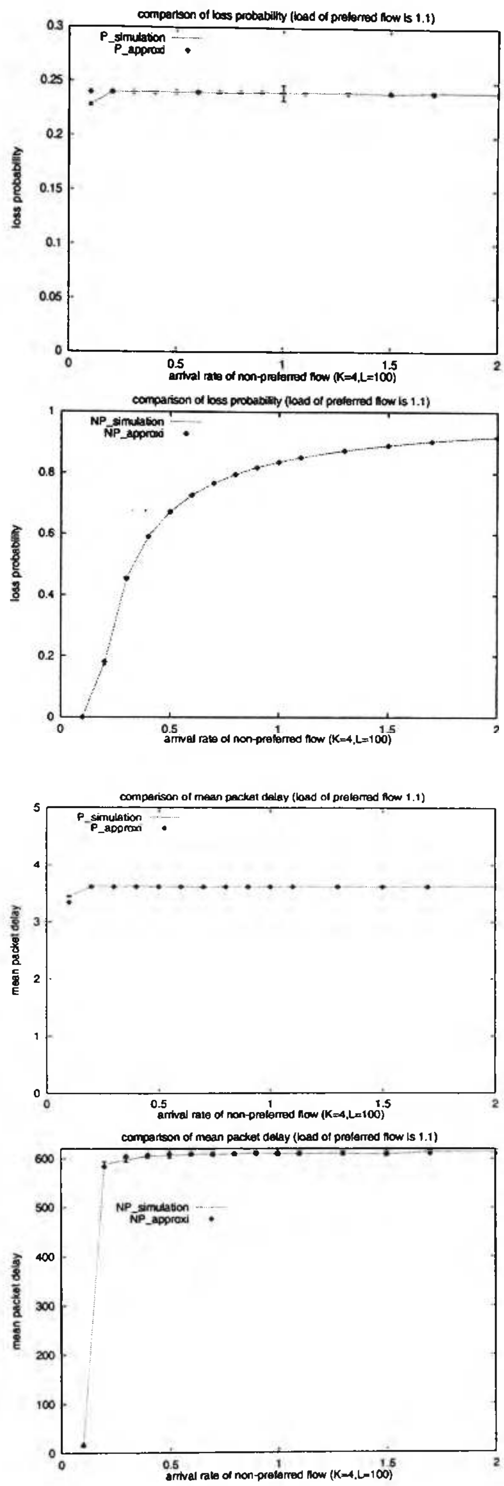



**Figure 4:** The comparison results when the load of preferred flow is heavy.

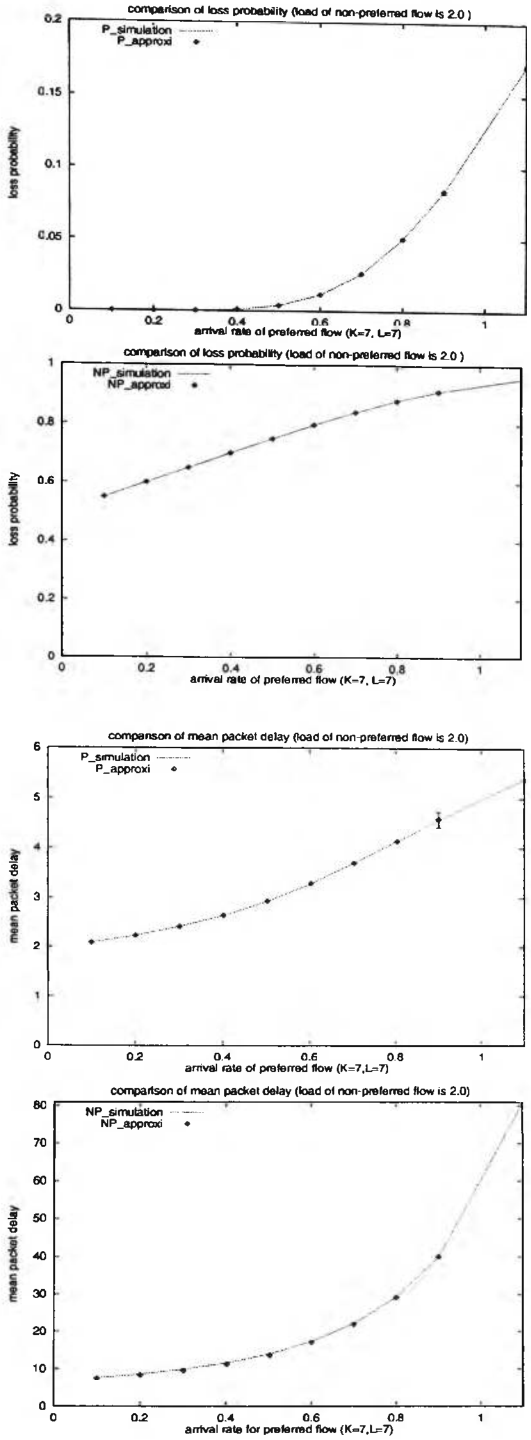



**Figure 5:** The comparison results when load of non-preferred flow is heavy

In both figures 4 and 5, we denote the results from simulation for preferred and non-preferred flow as P_simulation and NP_simulation respectively. Likewise, we denote the results from approximation method as P_approxi and NP_approxi. Same as Figure 3, the mean packet delay is normalized with respect to service time.

Figure 4 presents the comparison results when the load of preferred flow is heavy (load factor is 1.1). When the load of non-preferred flow is heavy (load factor 2.0) and buffer size of the two flows are kept the same (7 packets), we vary the load for preferred flow from light (load factor 0.1) to heavy (load factor 1.1). The results from simulation and approximation are depicted in Figure 5.

The observations of the good agreement between the curves of packet loss (for preferred flow and non-preferred flow) from both approximation and simulation are obtained in Figure 3, 4 and 5. Same observations are also obtained for mean packet delay. The agreements of the numerical results show that the approximation method is effective.

## IV CONCLUSION

A clear trade-off between packet loss and mean packet delay for preferred and non-preferred packets flow is observed in PS mechanism when the buffer partition is changed. To simplify the computation of packet loss and mean packet delay of finite non-preemptive priority queues, an approximation method is proposed for two traffic classes (preferred flow and non-preferred flow). The results from both analytical approximation method and simulation are compared. Numerical results indicate that the approximation method is effective in terms of accuracy. This approximation method provides a simple way in understanding traffic behavior of the future Internet where PS mechanisms are deployed.

## V REFERENCES

[1] J.C.R. Bennett and H. Zhang, "Why WFQ Is Not Good Enough for Integrated Services Networks?", *Proceedings of NOSSDAV'96*, Apr, 1996.

[2] J.C.R. Bennett and H. Zhang, "Hierarchical Packet Fair Queuing Algorithms," *Proceedings, SIGCOMM '96*, Palo Alto, California, August 1996.

[3] Clark, D., Fang, W., "Explicit Allocation of Best Effort Packet Delivery Service". *Internet draft*, Sept. 1997

[4] Leonard Kleinrock, Queueing Systems, Volume I, John Wiley & Sons, 1975, p93-94.

[5] Dimitri Bertsekas and Robert Gallager, Data networks, Prentice Hall, 1992, p186-189.

[6] Martin May, J.C. Bolot, Alain J. Marie and C. Diot, Simple Performance Models of Differentiated

Services Schemes for the Internet, *Proceedings of INFOCOM'99*, New York, March 1999
[7] Sambit Sahu, Don Towsley, Jim Kurose, A Quantitative Study of Differentiated Services for Internet, *Umass CMPSCI Technical Report* 99-09.
[8] S. Blake, D. Black, M. Davies, Z. Wang, W. Ewiss, An Architecture for Differenciated Services, *RFC 2475*, December 1998.
[9] D.D. Clark, J. Wroclaski, An Approach to Service Allocation in the Internet, *IETF Draft*, July 1997.
[10] K. Nicholos, V. Jacobson, L. Zhang, A Two-bit Differentiated Services Architecture for the Internet, *IETF Draft*, Nov. 1997.
[11] Hui Zhang, Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks, *Proceedings of The IEEE*, Vol. 83, NO. 10, October 1995.
[12] S. Floyd, V. Jacobson, Random Early Detection Gateways for Congestion Avoidance, *IEEEE/ACM Trans. Networking*, 1(4):397-413, August 1993.
[13] C. Blondia, An M/G/1 Finite Capacity Queue with Vacation and Priorities, *Proc. Performance '87*, Brussels, eds. P.-J. Courtois and G. Latouche (North-Holland, 1987) 305-325