Analysis of Priority-based Packet Schedulers for Proportional Delay Differentiation

Tan Chee-Wei
Bachelor in Electrical & Computer Engineering 2002

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Masters of Engineering

Department of Electrical and Computer Engineering

NATIONAL UNIVERSITY OF SINGAPORE

2002/2003

Abstract

In this thesis, priority-based packet schedulers are analyzed in order to provide relative and proportional delay differentiation. We investigate a Probabilistic Priority (PP) scheduler that provides relative delay differentiation to different classes. We present an integer PP algorithm and show that PP is a special scheme of applying lottery scheduling to bandwidth allocation in a strict priority sense. We then propose a Multi-winner PP (MPP) scheduler using multi-winner lottery scheduling to improve the throughput and response time accuracy and a flexible ticket transfer algorithm to improve the deadline violation probability in probabilistic scheduling. Finally, we investigate the issue of parameter assignment for an MPP scheduler and use our techniques to implement a prototype Assured Forwarding (AF) mechanism in a network processor. Proportional Delay Differentiation (PDD) has stricter requirement than relative delay differentiation. We study the schedulability conditions of the Waiting Time Priority (WTP) packet scheduler on achieving multi-class PDD under load variation. Based on a necessary condition for positive scheduler parameters in general $N-$class WTP, we derive a sufficient condition for WTP to achieve PDD. The sufficiency therefore implies that PDD delay dynamics can be readily employed. Hence, using these results, we can determine and re-adjust the load spacings that have passed the necessary condition for positive scheduler parameters. The results obtained also quantify the maximum operational target ratio achievable in WTP for a given load distribution and allow us to relate results for WTP to the PDD model for general

N-class in a precise manner. Next, based on an inequality relationship between scheduler parameters and target ratios, we propose a dynamic adjustment control technique to efficiently enhance the computation of scheduler parameters that uses iterative methods. We then evaluate the performance of this adjustment control mechanism. Lastly, we show that WTP can achieve both PDD and absolute QoS requirements under certain schedulability conditions by appropriate selection of scheduler parameters.

To My Mother

in gratitude and affection.

Contents

List of Figures

List of Tables

Acknowledgments

My sincere thanks to

Dr Tham Chen Khong for his support and feedback while I write this thesis. His financial support for me to attend MMNS 2003 is also much appreciated.

Prof. Loh Ai Poh for teaching me patiently an interesting mathematical course.

Dr Mohan Gurusamy for sharing much networking knowledge with me. His patient guidance in research and his rigorous attitude in pursuing research play a role of motivation.

Dr Jiang Yuming for providing invaluable comments to the work in Chap. 3.

Prof. John, Lui Chi Shing from the Chinese University Hong Kong for his useful comments and advice in my research. His ability to describe challenging problems astounds me greatly and his wisdom inspires me profoundly. His generous help is deeply appreciated.

Prof. Tay Yong Chiang who provided invaluable comments on my work and gave me useful advice which I appreciate a lot.

Anonymous reviewers from IEEE conferences, ICON 2003 and MMNS 2003, who helped to improve the quality of the work in Chap. 3. The IEEE ICNP 2003 anonymous reviewers and Technical Program Committee members helped to improve the quality of the work in Chap. 4. The coordinators of Intel Exchange Architecture (IXA) Network Processor Education Program were kind to showcase the work in Chap. 3 online.

Tony Low Aik-Seng, Liu Yong, Yao Qi, Phua Kok-Soon, Tan Chong-Jin and Santos Kumar Das for bringing fun and laughter to my graduate days.

Lin Ying who continuously encourages and supports me tremendously to finish all the work and get me going, even in unusual difficult times.

Symbols and Abbreviations

AF	Assured Forwarding
BE	Best Effort
CPU	Central Processing Unit
DAC	Dynamic Adjustment Control
DDP	Delay Differentiation Parameter
DiffServ	Differentiated Services
DSCP	Differentiated Services Code Point
EF	Expedited Forwarding
FIFO	First-in-first-out
HOL	Head-of-line
IP	Internet Protocol
LRD	Long Range Dependent
MDP	Mean Delay Proportional
MPP	Multi-winner Probabilistic Priority
MTU	Maximum Transmission Unit
PDD	Proportional Delay Differentiation
PHB	Per Hop Behavior
PP	Probabilistic Priority
QoS	Quality of Service
RISC	Reduced Instruction Set Computer
SP	Strict Priority

TCP	Transport Control Protocol
TDP	Time Dependent Priority
WTP	Waiting Time Priority
$A_i[t, t+\tau]$	Actual traffic arrival of class i from time t to $t+\tau$
$A_i^*[\tau]$	Upper bound of traffic constraint of class i in interval τ
δ_i	DDP for class i
b_i	Scheduler parameter for class i in WTP
p_i	Scheduler parameter for class i in PP/MPP
C_p	Set of connections with priority p
d_i	Maximum delay bound for class i
σ_i	Maximum burst size or token bucket depth in class i
s_i	Maximum transmission time of packet in class i
ρ_i	Offered load for class i
$\overline{W}_i^{SP}$	Average delay of class i in SP scheduler
$\overline{W}_i^{WTP}$	Average delay of class i in WTP scheduler
λ_i	Average arrival rate of class i
$M/G/1$	Kendall notation for Poisson input and generalized service distribution
x_i	Mean service time of class i
W_0	Mean residual service time in M/G/1
S_i	Target ratio of average delay between class i and class N
$r_{i,j}$	Target ratio between class i and class j
R_i	Regnier's ith inequality

Chapter 1

Introduction

1.1 Background

In order to support the Quality-of-Service (QoS) requirement of an aplication, a crucial network design issue is to decide what kind of network services should be provided. According to the QoS guarantees offered, we divide network resources into three categories: deterministic, statistical and best effort. Under deterministic services, deterministic QoS guarantees are provided and enforced based on the contract made between a user and the network. Under statistical services, statistical QoS guarantees are promised, but in many cases, they may not be strictly enforced or actually enforceable. Under best effort services, no QoS guarantee is supported. In recent years, several approaches have been proposed to allow resources in a network to be used efficiently. Among them, the Differentiated Services (DiffServ) approach is very promising because of its potential scalability to provide real-time applications with QoS guarantees and best effort services within the Internet. In the DiffServ architecture, individual flows with similar QoS requirements are aggregated, and given the same treatment as described by a Per-Hop-Behavior (PHB) in terms of QoS metrics such as average packet delay, packet loss and jitter. The routers do not keep per-flow states and there is no complex resource signaling mechanism involved [2]. The Expedited Forwarding (EF) PHB defines that premium traffic is guaranteed. In contrast, the Assured Forwarding (AF) PHB guarantees only that the assured traffic is delivered with a higher probability than the best-effort traffic; in the case of severe network congestion, the assured traffic can still experience severe losses and high delay. It remains a challenge in designing a framework to provide Assured Forwarding to data packets. The Proportional Differentiated Services framework is leading current intensive research in meeting this challenge in data networks [9].

1.2 Proportional Delay Differentiation

As most network providers are still unwilling to deploy large-scale QoS mechanisms due to the complexity involved, recent research in DiffServ has focused on a simplified approach, known as relative DiffServ [9]. The Class Selector PHB [24] which was recently standardized by the Internet Engineering Task Force (IETF) follows this service approach to provide a number of classes with increasing performance. The user has the flexibility to choose the level of service it wishes to have under cost constraints. Dovrolis *et al.* [9, 10] proposed a Proportional Delay Differentiation (PDD) model to provide ”tuning knobs” to control the performance spacing and to have predictable service guarantees in a DiffServ framework, independent of the class loads. In particular, PDD requires that the average class delay of packet $\overline{W_i}$, are spaced as

$$\frac{\overline{W_i}}{\overline{W_j}} = \frac{\delta_i}{\delta_j} \quad , 1 \leq i, j \leq N \tag{1.1}$$

where the parameters δ_i are the Delay Differentiation Parameters, and they are ordered so that classes with higher priorities provide lower delays, i.e. $\delta_1 > \delta_2 > \cdots > \delta_N > 0$.

A PDD model must be *predictable* such that differentiation is consistent (a higher class is better or at least no worse than a lower class) and the differentiation is independent of class loads. Second, the model must be *controllable* such that network operators can select the appropriate level of spacing between classes based on their delay spacings. It was shown in [9] for $N > 2$, the feasibility conditions for the PDD model are the following $N-1$ inequalities:

$$\sum_{i=k}^{N} \lambda_i \delta_i \leq \frac{\sum_{i=1}^{N} \lambda_i \delta_i}{\sum_{i=1}^{N} \lambda_i \overline{W_i}^{SP}} \sum_{i=k}^{N} \lambda_i \overline{W_i}^{SP} \quad , k = 2, \ldots, N \tag{1.2}$$

where $\overline{W_i}^{SP}$ denotes the average delay of class i in the Strict Priority (SP) scheduler and λ_i denotes the average arrival rate of class i.

Packet scheduling is an important mechanism that provides QoS guarantees. The scheduling discipline defines the order in which packets from different QoS categories are served. In this thesis, we concentrate only on work conserving inter-class packet schedulers, i.e., the server is never idle if there are arriving or buffered packets, and the packet that is being served cannot be preempted by other packets from another class. We assume the First-In-First-Out intra-class scheduling policy for each class. It is well known that the Strict Priority (SP) scheduler provides large differentiation among classes. Under the SP scheduler, packets in each priority class are served in a First-In-First-Out manner and a packet is serviced if and only if there is no buffered packet from a higher priority class.

As such, the SP scheduler is unfair to all classes except the highest priority class and may cause starvation in lower priority classes. In short, there is no *degree of freedom* in the SP scheduler. To achieve proportional delay differentiation, we analyze two different kinds of schedulers that also operate on the principle of priorities. The fundamental difference between these two schedulers and the SP scheduler is that they provide a *degree of freedom* to achieve delay differentiation. In other words, a high priority class will still always have better performance than a low priority class on the average but the shortcomings of the SP scheduler are overcome.

1.2.1 Proportional Probabilistic Priority-based Scheduling

In this paper, we analyze the Probabilistic Priority (PP) scheduling discipline within the framework of relative service differentiation. PP adopts a probabilistic relative service model. At every service round, each class takes a bid. Since higher priority classes have higher probabilities associated with them, in the long run, they will be served more often than lower priority classes. Compared to Strict Priority (SP), this increases fairness among classes and prevent the starvation of lower priority classes. We first show that PP is a cross application of lottery scheduling in a strict priority sense to provide proportional bandwidth sharing among classes. This in turn allows us to benefit from numerous techniques presented in [32, 33] to control PP. The lottery and stride scheduling algorithms are very well-known schedulers for statistical allocation of CPU resources [32, 33]. Lottery scheduling randomizes resource allocation among clients whose shares of resources are represented by tickets using policies such as ticket inflation and deflation. An allocation is performed by holding a lottery, and the resource is granted to the client with the winning ticket. Multi-winner lottery scheduling is a variant of lottery scheduling that produces better throughput accuracy for many workloads. Based on this multi-winner concept, we formulate a multi-winner PP algorithm to improve the response-time variability of PP. As lottery scheduling is effectively stateless, a great deal of complexity is removed in comparison to other proportional schedulers. The feasibility of using lottery scheduling in packet forwarding has been analyzed in [11, 16, 34] but no work has been done to address its weaknesses at the packet level due to its probabilistic nature. The probabilistic relative service model is only suitable for applications that are able to tolerate deadline violations of a few packets. We propose a technique that is analogous to the idea of dynamically-controlled ticket transfer which has been applied to graphics rendering and Monte-Carlo tasks [33] to address this problem.

1.2.2 Waiting Time Dependent Priority-based Scheduling

In [9], the Waiting Time Priority (WTP) scheduling discipline was found to be suitable to achieve PDD. WTP is based on Kleinrock's Time-dependent Priority (TDP) scheduling algorithm [18]. In the WTP algorithm, the service priority of a packet in class i at time t is given by $p_i(t) = w_i(t)\, b_i$, $i = 1, \ldots, N$ where $w_i(t)$ is the waiting time of the packet at time t and b_i is the weight of the delay class.

A packet's priority increases linearly from zero with time, in proportion to a rate assigned to the class [18]. Interestingly enough, a question was posed in [9]: Is there a work conserving scheduler that satisfies PDD ? We answer that question in this thesis. Specifically, Kleinrock's *Conservation Law* [18] which states that the weighted sum of average delay in a $M/G/1$ queueing model remains constant independent of the scheduling policy will be used in this paper to bridge the theoretical framework of PDD and WTP.

1.3 Thesis Scope and Overview

The rest of the thesis is organized as follows. In Chapter 3, we propose an efficient integer PP algorithm and show that PP is indeed a cross application of lottery scheduling. We use the multi-winner concept to generalize PP to improve its throughput accuracy and reduce its response-time variation. We present a technique based on flexible ticket transfer to reduce the deadline violation probability in times of congestion. Next, we investigate parameter assignment and propose a framework to implement Assured Forwarding. Finally, a performance study on a network processor-based router is presented.

In Chapter 4, we derive a sufficient condition for WTP to conform to the necessary and sufficient conditions of the PDD model for general N classes. We also derive the maximum target ratio achievable for a given system utilization achievable for $N > 2$. Next, we derive an inequality relationship between scheduler parameters and target ratios and then propose a Dynamic Adjustment Control (DAC) algorithm to identify infeasible load distributions. The performance of the DAC is also evaluated.

In Chapter 5, we obtain the maximum delay bound of WTP using general traffic specification and compare it with SP. We show a sufficient condition where all classes can perform better than SP by tuning the scheduler parameters. We conclude the thesis in Chapter 6.

Chapter 2

Related Work

In this chapter, we discuss related works on the PP and the WTP scheduler. We also elaborate the motivations of our work in this thesis.

2.1 PP Scheduler

Jiang *et al.* proposed the Probabilistic Priority scheduler to address the shortcomings of SP [16]. The authors showed in [16, 17, 30] that this algorithm exhibits the following properties that are very desirable to achieve service differentiation in a multi-class network by (a) providing diverse delay differentiation between classes, (b) supporting weighted max-min fairness among classes, (c) overcoming the starvation problem inherent in SP, (d) supporting relative differentiated services, and (e) providing explicit bandwidth reservation guarantees. However the problem of deadline violation probability associated with probabilistic scheduling due to randomness in a relative differentiated services framework was not addressed in these works. Reference [30] implemented PP on Linux machines but their design prohibits dynamic control of the PP scheduler parameters and thus is not scalable for large number of classes due to pre-calculation of all possible network states which increase exponentially with the number of classes. No previous work shows how the PP scheduler parameters are related to provide service differentiation which is essential because the scheduler parameters are the only tuning knobs available, hence, in this thesis, we derive necessary and sufficient conditions that relate scheduler parameters with the concept of relative service differentiation.

Earlier works in exploiting randomness to allocate bandwidth fairly include the statistical matching technique in [1] and partially connected operation in [14]. Eggleston *et al.* [11] investigated the benefits and drawbacks of using lottery queueing at the flow level and the trade-off between packet re-ordering and the number of flows whereas this work on PP assumes that class-aggregated flows are served in a FIFO order and lottery scheduling is performed at the class level thus avoiding the problem of packet re-ordering. Our service model also differs from theirs in that packets do not carry bid values. They used lottery scheduling to manage queue lengths whereas we focus on the scheduling of Head-of-line (HOL) packets. Another more recent related work to lottery scheduling is the Probabilistic Packet Scheduling (PPS) [34] which provides different level of proportional service to TCP flows. Their work applies the concept of ticket transaction and policies in lottery scheduling to adaptive marking in an end-to-end connection set-up by accommodating flows traversing multiple domains to exchange tickets between different currencies.

2.2 WTP Scheduler

Dovrolis *et al.* [9, 10] showed that WTP approximates PDD in heavy load conditions, even in short timescales. When the load tends to the system capacity, the delay ratios of two consecutive classes tend to converge to the reciprocals of the corresponding increasing rates of the priority functions. Based on Kleinrock's analysis in [18], Sethuraman *et al.* showed the solutions to minimizing response time variance for linear TDP and a recursive formula to compute the scheduler's parameters for the general N-class system under different loads. Similarly, Leung *et al.* [21] showed the exact solutions for two traffic classes. In particular, the scheduler parameters do not depend on the load distribution but only on the total utilization in the queuing system. They also proposed a numerical algorithm to calculate the scheduler parameters dynamically so that WTP can achieve a feasible set of DDPs based on feedback of current load conditions. The authors believed that certain distributions of load, ρ_i's will not lead to positive solutions of the scheduler parameters but did not show exactly how. Eaasfi *et al.* also showed similar results for two traffic classes in [12] and they also used iterative optimization technique to adapt the scheduler parameters to load variance. In [13], Essafi *et al.* used genetic optimization algorithms to dynamically adjust WTP for a finite number of classes with high accuracy. The authors also compared this offline optimization approach with the numerical iterative algorithm in [21]. Several issues related to feasibility conditions were raised in this paper. In particular, the authors could not conclude whether the infeasibilities of certain load distributions are due to the inaccuracy of the optimization algorithms or insufficient utilization. Our findings in this thesis show that the reason is due to the inappropriate load distribution and not due to inaccuracy of the optimization techniques. A novel architecture known as CoreLite described in [23] couples per hop proportional delay differentiation with end-to-end delay guarantees in core stateless networks. The authors propose a Mean-delay Proportional (MDP) scheduler and derive delay dynamics very similar to that of PDD. The main advantage of the CoreLite architecture is that packets do not carry state information. Likewise, we also define in this thesis the schedulability region of WTP where PDD dynamics is applicable.

Recently, Lee *et al.* [19] proposed a framework for admission control and dynamic adaptation for achieving proportional delay differentiation in a web server. The web server operator can specify ”fixed” performance spacings between each class and the proposed dynamic algorithms attempt to classify clients to its ”lowest” admissible class so as to achieve the lowest possible cost for each client. To provide differentiated services, the web server attempts to achieve consistency and controllability independent of variations in class load. Also, a central premise in the relative differentiated service model in [8] is that users can dynamically search for a class which provides the desired QoS level. Hence a natural question to ask is: How can the load on multi-class WTP scheduler be exactly characterized to achieve PDD with low complexity ? How does load distribution relate to the performance in computation of WTP scheduler parameters ? Earlier works in [20, 21] show that WTP is not predictable for more than two traffic classes as it is dependent on load distribution to certain extent. As such, making WTP controllable based on a given load distribution is the focus of this thesis. To this end, we propose a measurement-based load Dynamic Adjustment Control (DAC) algorithm to assure the feasibility of the PDD model using WTP.

Chapter 3

Proportional Probabilistic Priority Scheduling

In this chapter, we analyze the relationship between the PP scheduler and lottery scheduling. Next, we develop algorithms to improve the PP scheduler and implement our algorithms on network processor. We also derive relationships between scheduler parameters for parameter assignment to achieve relative delay differentiation.

3.1 Analysis of A PP Scheduler

3.1.1 Basic PP Integer Algorithm

The work conserving Probabilistic Priority Scheduler is based on the Strict Priority scheduler with each queue being assigned a probability p_i of getting served [16]. By appropriate setting of a parameter $p_i \in [0, 1]$, $i = 1, \ldots N-1$ and $p_N = 1$ in a multi-class system, a class is selected with a probability corresponding to equation (3.1) for service at every cycle. A class parameter of $p_i = 1$ means that the class i definitely gets served when polled if all higher priority classes are empty or not selected during the cycle. Hence PP reduces to SP when $p_i = 1.0$, $i = 1, \dots N$. In the following, we derive an integer algorithm and show that it is indeed a cross application of lottery scheduling in the strict priority sense. Lottery scheduling is a novel probabilistic CPU task scheduling mechanism that assigns each task some number of tickets [33]. When a task is to be selected for execution, a lottery is held, and the task holding the winning ticket is selected to run. On the average, a task is expected to run in proportion to the number of tickets it holds.

First, consider a multi-class system of N priority levels with the highest priority level denoted by 1. Let us define the weight of class i to share the server [16] as

$$r_i = p_i \prod_{j=1}^{i-1} (1 - p_j) \tag{3.1}$$

Without loss of generality, assume that all classes in the group are busy so that the normalized weight of class i among all classes is

$$\hat{r}_{i \in \Omega} = \frac{r_i}{\sum_{j \in \Omega} r_j} \tag{3.2}$$

where Ω consists of all queues in the group. After rearranging all r_i such that they share a common denominator, we have

$$\hat{r}_{i \in \Omega} = \frac{x_i}{\sum_{j \in \Omega} x_j} \tag{3.3}$$

where x_j is the numerator of the normalized relative weight r_i. It is easy to see that this will also be true for all network conditions:

$$\hat{r}_{i \in BQ} = \frac{x_i}{\sum_{j \in BQ} x_j} \quad , BQ \in \Omega \tag{3.4}$$

where BQ is the set of non-empty queues in Ω. The total number of possible network conditions is equal to $2^N - 1$ but the most interesting set would be the total number of possible network conditions with more than one non-empty queue which is equal to $M = \sum_{i=1}^{N-1} \sum_{j=1}^{N-i} \binom{N-i}{j} = 2^N - N - 1$. From equation (3.4), we now have numerator x_i to calculate $\hat{r_i}$ without having to store in advance $\hat{r_i}$ for all possible combinations of empty and non-empty queues with each combination corresponding to a particular instance of Ω. This effectively removes both the need for fractional arithmetic in recalculation of network states whenever p_i changes dynamically and the restriction for a small set of all possible network states. The integer algorithm of PP works *without* the need for *a priori* network state computation. One instantly recognizes that the numerator for each class corresponds to the number of tickets for each client in lottery scheduling. PP is analogous to having sets of different numbers of tickets that are present in a service round with each set corresponding to one of the network conditions in M. The winner is then selected from this set at each service round. In lottery scheduling, there is no preference for the priorities of the clients whereas PP defines that on every round, the winner of the lottery is searched for in a strict priority sense, i.e. the highest priority class is the first client on the search list. To set the p parameters such that the classes are served in a relative priority fashion, we have the following theorem.

Theorem 1 *A necessary and sufficient condition to assign average probability parameter for each class for relative service differentiation, i.e. Class* 1 *being the highest priority class has higher probability than class* 2*, is* $\frac{1}{N-i+1} < p_i \leq min\left(\frac{p_{i-1}}{1-p_{i-1}}, 1.0\right) \quad, i = 1, \ldots, N.$

Proof: We first give the proof for the sufficient condition. The inequality on the RHS can be proved easily using $r_i < r_{i-1}$ and equation (3.1), and using the fact that p_i is always less than 1. To prove the inequality on the LHS, we use the RHS inequality and the fact that $p_N = 1$ to get $\frac{p_{N-1}}{1-p_{N-1}} > 1$. Thus $p_{N-1} > \frac{1}{2}$. Again from the RHS inequality, $p_{N-1} \leq \frac{p_{N-2}}{1-p_{N-2}}$ hence $p_{N-2} > \frac{1}{3}$. Finally we obtain $p_1 > \frac{1}{N}$ for the highest priority class. Hence, in general, $p_i > \frac{1}{N-i+1}$ which completes the proof for the LHS inequality.

Next, to prove the necessary condition, we have to show that the above theorem holds for both inequalities. First, we look at the LHS inequality. Let us assume that for a particular class i where $1 \leq i \leq N-1$, $p_i = \frac{1}{N-i+1} - \triangle$ where $0 < \triangle < \frac{1}{N-i+1}$. Then we get

$$r_i = \left(\frac{1}{N-i+1} - \triangle\right)\prod_{j=1}^{i-1}(1-p_j) \tag{3.5}$$

Now, consider class i's immediate lower priority class, class $i+1$ with parameter p_{i+1}. Suppose that $p_{i+1} = \frac{1}{N-(i+1)+1} + \delta$ where δ is a positive real value. This would also imply that we assume the theorem holds, i.e., Class i will have a higher probability of getting served than Class $i+1$. Now, let $r_i - r_{i+1}$, and we have

$$\begin{aligned} r_i - r_{i+1} &= \left(\frac{1}{N-i+1} - \triangle\right)\prod_{j=1}^{i-1}(1-p_j) - \left(\frac{1}{N-i} + \delta\right)\prod_{j=1}^{i}(1-p_j) \\ &= \frac{1}{N-i+1}\prod_{j=1}^{i-1}(1-p_j) - \prod_{j=1}^{i-1}(1-p_j)\triangle - \frac{1}{N-i}\prod_{j=1}^{i-1}(1-p_j) \\ &\quad - \frac{1}{N-i}\left(1-p_i\right) - \prod_{j=1}^{i}(1-p_j)\delta \\ &= \frac{-1}{(N-i+1)(N-i)}\prod_{j=1}^{i-1}(1-p_j) - \prod_{j=1}^{i-1}(1-p_j)\triangle - \frac{1}{N-i}\left(1-p_i\right) \\ &\quad - \prod_{j=1}^{i}(1-p_j)\delta \\ &< 0 \end{aligned} \tag{3.6}$$

The inequality is true for all positive real $\triangle$ and δ. In other words, we can select any $\triangle$ and δ that would result in a violation of the service priority constraint. For the RHS inequality, it is sufficient to show that it is impossible to find a positive δ that satisfies the following inequality

$$\delta < \frac{1 - \triangle(N-i+1)}{N-i+\triangle(N-i+1)} - \frac{1}{N-i} \tag{3.7}$$

for all positive $\triangle$, N and i which results from the RHS inequality of the above theorem. Hence, we obtain a contradiction with our assumption that $r_i > r_{i+1}$.

3.1.2 Multi-winner PP (MPP) Integer Algorithm

Multi-winner lottery scheduling is a generalization of the basic lottery scheduling technique that produces better throughput accuracy and smaller response-time variation [33]. Instead of selecting a winner per round, N_w winners are selected with only the first winner being randomly selected and each winner is guaranteed the use of the resource for one quantum. The set of N_w consecutive quanta allocated by a single multi-winner lottery is referred to as a super-quantum. Due to the probabilistic nature of PP, the highest priority class can exhibit substantial variability over small time scales which can cause its HOL packet to miss its deadline if sufficient numbers of service round are given to its lower priority classes instead. At worst, this may cause buffer overflow and incoming high priority packets to be dropped. This necessitates incorporating a deterministic mechanism in PP to achieve predictable behavior at small time scales. We use the multi-winner concept to extend the original PP integer algorithm as shown in Table 3.1. In this paper, we use a fixed value of $N_w = 20$. The ordering of the winners in MPP is based on a fixed permutation that goes in a round robin fashion, starting from the first winner and followed by its immediate lower priority class. This integer algorithm requires a total of $2^N - 1$ uniform distributions of integer random numbers for N classes. This is analogous to the total number of tickets differing in every service round of lottery scheduling. Waldspurger *et al.* [32] provides a multiplicative linear congruential Park-Miller pseudo random number generator in MIPS assembly language code but we use a generic algorithm *U-map* described later to scale uniform distributions without using multiplication assembly language instructions. In our algorithm, each super-quantum is reset back to 0 when the network condition changes which would happen very often if the system is highly loaded. This implies that MPP is able to reduce the throughput error and response-time variability. Through extensive simulations under heavy load conditions, we observe that the super-quantum is reset on an average of about 85% of the total time. Hence N_w does not have a significant impact on the reduction rate of throughput error. The advantage of MPP over PP appears to be small for 8 classes but by keeping the number of classes small, we can increase the number of winners to provide stricter throughput guarantees within a class.

3.1.3 Flexible Ticket Transfer Algorithm

In the previous section, we described an extension of PP to achieve throughput guarantee. In this section, we aim to reduce the time given up to the lower priority classes by the higher priority classes ("slack" in probabilistic scheduling) by setting a rate of approaching strict prioritization using the relationship between delays of different classes. In particular, we use the following propositions of average delay of class i, $\overline{W_i}$ proved in [30] to affect p_i.

Table 3.1: Pseudo-code of Multi-winner Probabilistic Priority scheduling algorithm

```
  /* Start with segregation groups in a strict priority manner*/
1. if ( Segregation Group $> 1$ )
2.     get Group with highest priority
3.     get numerator_vector of selected Group=$(x_1, x_2, \ldots, x_N)$
4.     for all busy queues $j \in \Omega$ in Group
5.         n_winners=$\left\lceil \sum_{j\in\Omega} x_j / min\,(x_1, x_2, \ldots, x_j)_{j\in\Omega} \right\rceil$
6.         get denominator=$\sum_{j\in\Omega} x_j$
7.         if ( class parameter list $\neq P(1, 1, \ldots, 1)$)
8.             get random_number
9.             random_number =U-map(random_number, denominator)
10.        else dequeue packet using strict priority
11.         intra_space=denominator/n_winners
  /* Select next winner within super-quantum*/
12.        while(intra_cnt $\neq 0$)
13.              winner=random_number+intra_space*intra_sched[intra_cnt]
  /*handle wrap around of numerator space*/
14.            if(winner $\geq$ denominator)
15.                winner $- =$ denominator
16.            if(++intra_cnt==n_winners)
17.                intra_cnt=0
18.             for all busy queues $j \in \Omega$ in Group
19.                if($queue_j \rightarrow$ sum $>$ winner)
20.                    dequeue packet of $queue_j$
```

(1) As $p_j \uparrow [0 \rightarrow 1]$[1] for $j < i$, $\overline{W_i}$ is continuously and monotonically increasing.

(2) As $p_i \uparrow [0 \rightarrow 1]$, $\overline{W_i}$ is continuously and monotonically decreasing.

(3) As $p_j \uparrow [0 \rightarrow 1]$ for $j > i$, $\overline{W_i}$ is nearly constant under congested network conditions.

Let us define the initial parameter r_i for class i that satisfies the relationship $r_1 \geq r_2 \cdots \geq r_i \geq \cdots \geq r_N$ for the multi-class system where r_1 is the highest priority class. Such assignment means that the probability of higher priority class is larger. This algorithm consists of the following two steps. The first step is to reduce the probability of a lower priority class after it has been served by transferring some probability to its immediate higher priority class. Note that the transfer of tickets from the class served to its immediate higher priority class will create a snowball effect that will cause the highest priority class to be eventually served while still using probabilistic scheduling. The second step is to preserve as much as possible the priority allocation that is defined at the start of the algorithm by transferring probability starting from the lowest priority class even though it has not been served to the immediate higher priority class of the class being served if the first step persists. Eventually the class that continuously gets served will lose its bid after the probabilities of all lower priority classes have been depleted.

From the algorithm shown in Table 3.2 and equation (3.1), we can make the following propositions:

(a) If $p_{i+1} < \frac{p_i}{1-p_i} \leq 1$ and $\triangle_i$ of probability to be served is transferred from class i to class $i-1$, $\hat{p_i}$ decreases, $\hat{p}_{i-1}$ increases, and $\hat{p_j}$, $j \neq i, i-1$ remains constant.

(b) If $p_{i+1} = \frac{p_i}{1-p_i} \leq 1$, and $\triangle_k$ of probability to be served is transferred from class k, $i < k \leq L$ to class $i-1$, $\hat{p_j} \uparrow \left[p_j^{orig} \rightarrow 1\right]$, $j \leq i$ where p_j^{orig} is the original PP parameter of

[1]Following [30], the notation "$x \uparrow [0 \rightarrow 1]$" means "$x$ increases from 0 to 1".

Table 3.2: Outline of ticket transfer algorithm

At each service round, suppose classes 1 to L, corresponding to a particular network condition $BQ \in M = 2^N - N - 1$ where N is the total number of classes, are busy,

1. If class i, $1 < i \leq L$, gets served, then $r_i^{'} = max\,(r_i - \triangle_i, r_{i+1})$, and $r_{i-1}^{'} = min\,(r_{i-1} + \triangle_i, 1.0)$, such that $r_i^{'} \geq r_{i+1}$, i.e. transfer $\triangle_i$ of probability being served to the immediate next higher priority level with $r_i \neq 0$.

2. If $r_i = r_{i+1}$, then $r_k^{'} = (r_k - \triangle_k)^+$, $i < k \leq L$ where k is the lowest priority class in BQ that satisfies $r_k \neq 0$, and $r_{i-1}^{'} = min\,(r_{i-1} + \triangle_k, 1.0)$, i.e. transfer $\triangle_k$ of probability being served to the immediate next higher priority class $i - 1$.

3. If the highest priority class is served or the network condition BQ changes, $r_i^{'} = r_i$, i.e. reset all class parameters back to their original r_i.

class j.

Proposition (a) states that only the probabilities of the class served and its immediate higher priority class will change while the other classes will maintain the original PP configurations at the initial stages after the algorithm begins while proposition (b) states that higher class priority will approach the configuration of SP, i.e. $\hat{p}_j \rightarrow 1$, $\hat{p}_j \neq 0$, $1 < j \leq i$ if the situation where the highest priority class HOL packet is not served while class i is constantly being served persists. Therefore, from proposition (1) and (2), the average delays of classes with higher priorities than class i will decrease monotonically over time while those classes with lower priorities than class i will increase monotonically over time. We introduce an additional parameter $\triangle_i$ to provide a dynamic feed-forward mechanism based on the current workload or the slack of the corresponding high priority HOL packet. This user-tunable class parameter $\triangle_i$ can be a function of the class's burstiness or the higher priority classes' backlog. It provides a way for static PP to approach SP in a configurable length of time so that the HOL packet of higher priority classes will not exceed its deadline unnecessarily.

3.1.4 Simulation Studies

In this section, we consider scenarios with high traffic loads and tight deadlines for each class. For each class, we use long range dependent (LRD) traffic modeled as Pareto On-off processes with shape parameter 1.3 since aggregated traffic in real DiffServ networks is LRD in nature. The mean service time is taken to be the unit of time and the service times of packets in each class follow the same exponential distribution with mean 1.0 units. Results are averaged over 10^6 time unit simulation windows unless otherwise indicated. Throughout this paper, we use λ_i and ρ_i to denote the arrival rate and traffic

Table 3.3: Comparison of deadline violation probabilities under full utilization condition (%)

	PP/Lottery	MPP	MPP w/ ticket xfer	SP
Class 1	0.114	0.069	0.036	0.000
Class 2	0.172	0.082	0.068	0.010
Class 3	4.297	1.680	0.646	0.410
Class 4	23.513	17.883	11.451	9.647
Class 5	34.696	27.678	21.410	14.609
Class 6	57.679	45.020	35.453	27.650
Class 7	91.627	88.020	64.952	58.243
Class 8	94.831	90.671	67.316	100.000

Table 3.4: Comparison of average delay under full utilization condition (time units)

	PP/Lottery	MPP	MPP w/ ticket xfer
Class 1	1.350	1.170	1.201
Class 2	1.990	1.460	1.450
Class 3	4.920	3.550	3.471
Class 4	55.290	45.640	37.400
Class 5	198.490	214.620	120.810
Class 6	555.440	402.180	240.080
Class 7	6719.060	3726.320	1973.990
Class 8	22243.940	19847.430	7240.370

intensity of class i respectively. In Table 3.3, the arrival rates for all classes are the same, i.e. $\rho_i = 0.125$ so the system is not overloaded, i.e. $\rho = 1.0$. Each class has the same parameter i.e. $p_i = 0.6$, $i \neq N$. To compare the performance between the various schemes, we use deadline violation probability in Table 3.3 and Table 3.5 as a performance metric. The deadlines for class 1 to N are arbitrary selected as 11, 16.5, 22, 27.5, 33, 38.5, 44, and 49.5 time units respectively. The probability transfer quantum is the same for all classes, i.e. $\triangle_i = min\,(0.15, r_i)$.

The simulation experiments were run for a sufficiently long time and were repeated several times to get accurate values within 95% confidence interval. We use different random

Table 3.5: Comparison of deadline violation probabilities under overloaded condition (%)

	PP/Lottery	MPP	MPP w/ ticket xfer	SP
Class 1	0.104	0.082	0.062	0.000
Class 2	0.259	0.152	0.083	0.013
Class 3	6.960	2.302	1.552	0.988
Class 4	31.518	29.204	19.566	12.514
Class 5	50.316	43.976	34.253	20.460
Class 6	88.226	76.638	61.668	40.547
Class 7	99.492	99.835	90.286	83.241
Class 8	99.701	99.756	93.508	100.000

Table 3.6: Comparison of average delay under overloaded condition (time units)

	PP/Lottery	MPP	MPP w/ ticket xfer
Class 1	1.390	1.180	1.180
Class 2	2.220	1.490	1.490
Class 3	5.350	3.610	3.550
Class 4	64.720	48.400	24.400
Class 5	185.490	160.470	146.640
Class 6	539.960	521.950	486.590
Class 7	10083.030	10502.340	11348.750
Class 8	29159.260	24863.400	23929.430

seed values for packet generation in each run window and we record the average of ten windows in total. Results in Table 3.3, 3.4, 3.5 and 3.6 indicate that ticket transfer algorithm does not have an adverse effect on low priority class though it discriminates against them by allowing high priority classes to be selected as fast as possible. Table 3.4 and Table 3.6 show that the average delays in all classes except for Class 7 in Table 3.6 is smaller for the case of MPP with ticket transfer. Since we use different random seeds for generation of packets at each run, the results do not indicate that MPP with ticket transfer provides the smallest possible delays for all classes. Later simulated results in a smaller class system would show that the lowest priority class suffers longer delay using MPP with ticket transfer as compared to PP and MPP. Rather, our simulated results in the tables only suggest that this mechanism improves deadline violation probability and delays of lower priority classes and not necessarily all the lower priority class on the average as opposed to intuition which we shall investigate next.

We now consider the ticket transfer algorithm used in a 4-class system to evaluate its effectiveness. Each class has parameter $p_1 = 0.5, p_2 = 0.55, p_3 = 0.6$ and $p_4 = 1.0$. Note this parameter assignment provides lower priority classes with higher probabilities of being serviced than in previous simulations. Fig. 3.1 and Fig. 3.2 show the probabilities of all possible network conditions occurring in the system for SP, PP and MPP with ticket transfer schedulers at both short (10^3 time units) and long timescales (10^6 time units) with respect to packet service times. Each network condition is binary-coded as follows: bit 0 corresponds to the highest priority class, class 1 hence 0101B implies that only class 1 and 3 are present. Note that the network condition is a function of offered loads and

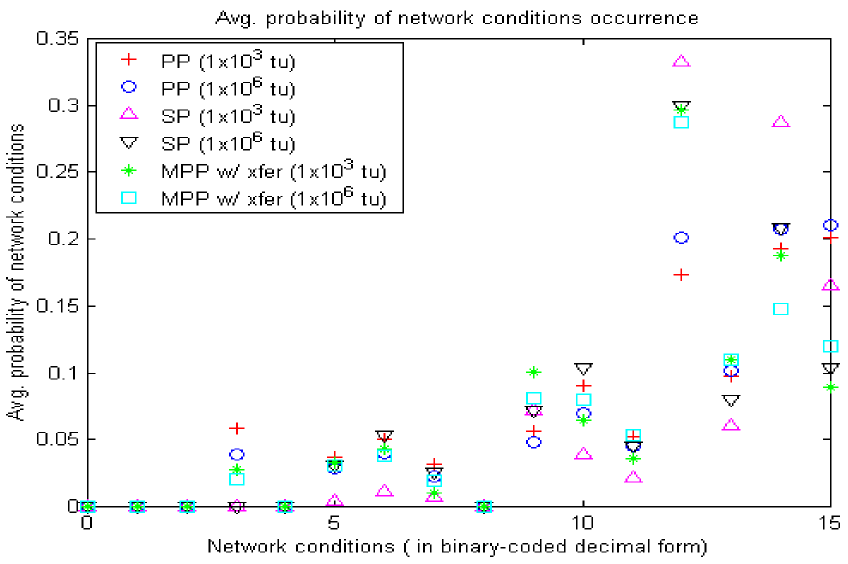


Figure 3.1: Comparison of network condition probabilities between PP, SP and MPP with ticket transfer scheme under light load

scheduling mechanism. We also compare the Pareto on-off traffic model with the token bucket-constrained traffic model with a bucket depth of 17 time units which exhibits short bursts.

Note that, in contrast to intuition, the deadline violation probability of the lowest priority class is improved significantly when the ticket transfer algorithm is used because higher priority classes are assured to get transmitted within short timescale and this implies that the probability of network conditions containing these high priority classes occurring within a longer time frame will be smaller than that in comparison to normal PP scheduling. From Fig. 3.1, Fig. 3.2 and Fig. 3.3, we can make the following observations:

- We found that MPP with ticket transfer can always achieve smaller average delay and deadline violation probability than PP and MPP scheme for most classes. Its deadline violation probability of the lowest priority class can be better than SP.

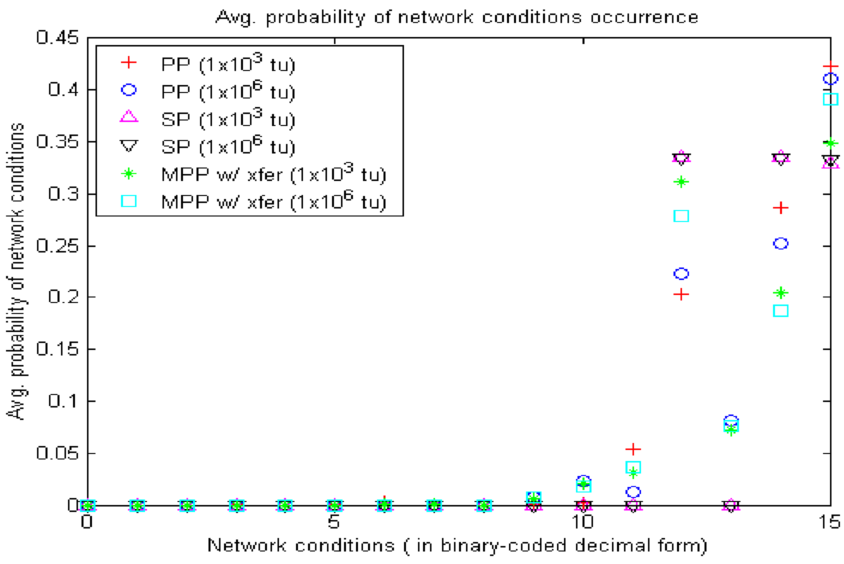
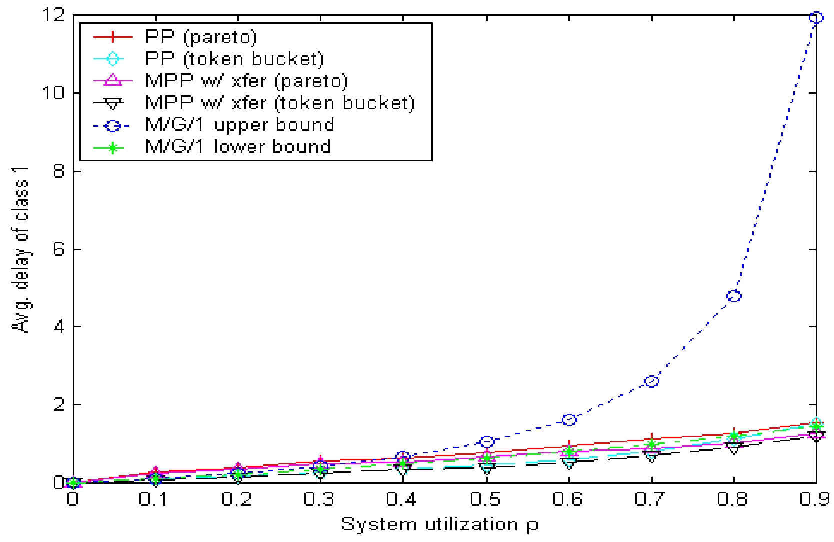
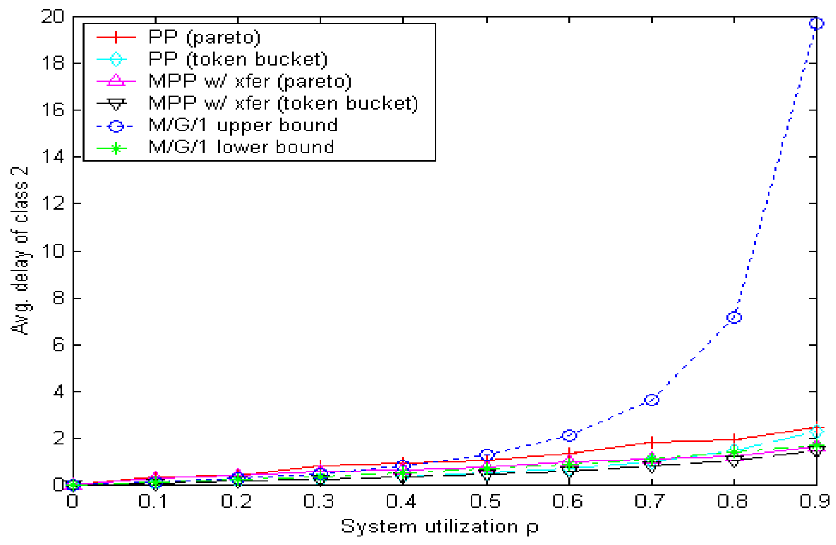
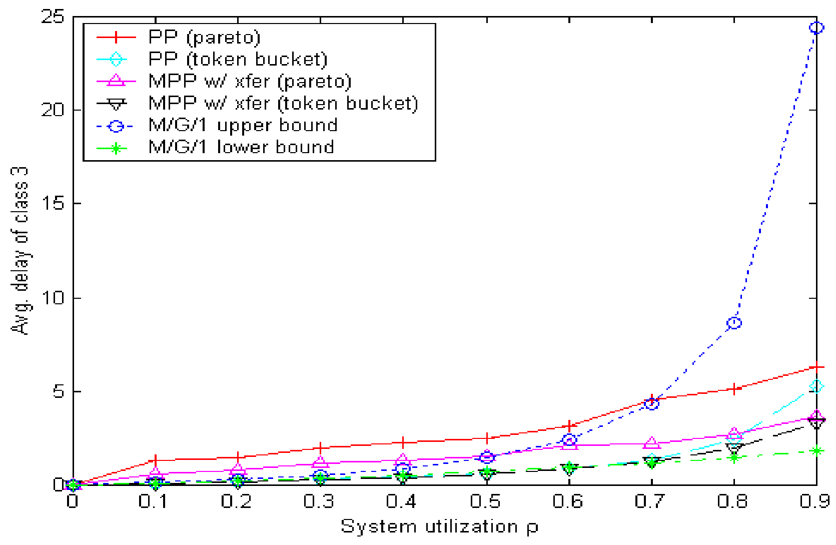


Figure 3.2: Comparison of network condition probabilities between PP, SP and MPP with ticket transfer scheme under heavy load




PP (pareto)
PP (token bucket)
MPP w/ xfer (pareto)
MPP w/ xfer (token bucket)
M/G/1 upper bound
M/G/1 lower bound
Avg. delay of class 2
System utilization ρ
0
2
4
6
8
10
12
14
16
18
20
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9


PP (pareto)
PP (token bucket)
MPP w/ xfer (pareto)
MPP w/ xfer (token bucket)
M/G/1 upper bound
M/G/1 lower bound
Avg. delay of class 3
System utilization ρ
0
5
10
15
20
25
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

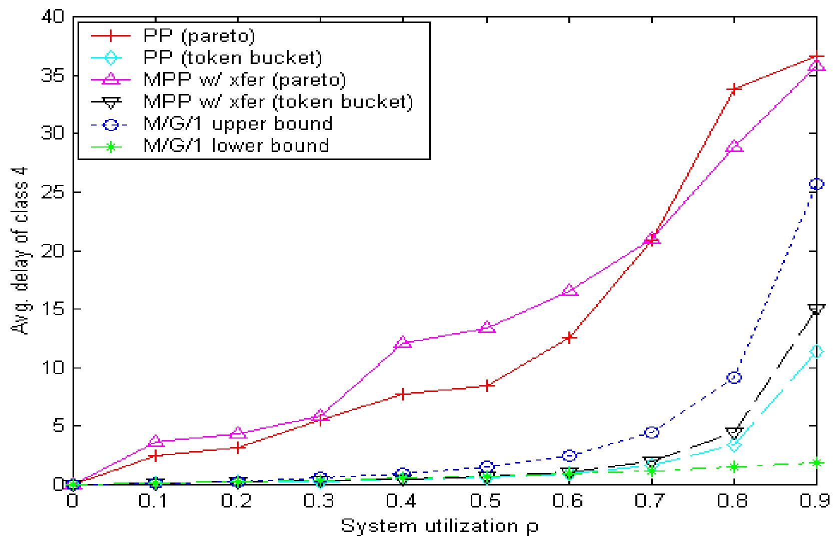


Figure 3.3: Average queueing delay under different traffic loads using Pareto on-off and token bucket filter constrained traffic

- Generally the delay of token bucket-constrained traffic lies in between the $M/G/1$ delay bounds derived in [30]. But the heavy-tailedness of Pareto on-off, for eg. with a shape parameter of 1.3, and burst rate 0.25 can cause the delay to exceed the $M/G/1$ delay bound.

- The ticket transfer algorithm has an evident impact on reducing the mean delay of all classes except the lowest priority class. This is due to: (a) the probability of the network condition 12 (1100B) that contains only the two lowest priority classes becomes higher, and (b) the probability of the network condition 15 (1111B) that contains all classes becomes smaller, and in both cases, they approach that of SP.

Both (a) and (b) increase the probability of the lower classes being serviced. Since the algorithm differentiates that higher priority classes are served as fast as possible when network conditions containing them appear, the mean delays of higher priority classes will therefore be much smaller than PP.

3.2 Achieving Assured Forwarding Using MPP

We consider 8 QoS classes and we configure a MPP scheduler to have 2 segregation groups AF_1 and AF_2. Each group has the last parameter $p_4^{AF_1} = p_4^{AF_2} = 1$. In each group, the AF classes are assigned parameters $p_1^{AF_i} < p_2^{AF_i} < \cdots < p_4^{AF_i}$, $i = 1, 2$. The following theorem ensures that this parameter assignment guarantees AF classes to obtain better statistical relative delay service differentiation than its immediate lower priority class. The group segregation property states that in PP, the service discipline among segregation groups is exactly the Strict Priority discipline hence the first AF group is guaranteed to have better service than the second group in terms of delay [16]. By means of segregation, this framework (a) provides more isolation among high priority classes that demand low delay and deadline violation probability, and low priority classes that require at least best effort service, and (b) reduces the number of classes within a group as this means a smaller number of network conditions within each group therefore we can configure more number of winners within each super-quantum, i.e. smaller spacing between consecutive winners to improve the response time variability in multi-winner scheduling. Since each group is based on MPP scheduling, there is fairness in the resource allocation within each group by means of fair distribution to excess capacity [16]. The ticket transfer algorithm is used in the first segregation group to provide improved deadline violation probability and average delay. Since we do not consider admission control, we expect some form of policing to limit the burst size and amount of bandwidth admitted to each class to prevent starvation if a non-conforming flow enters the node.

Theorem 2 *An assignment of average probability parameter for each class where* $1/2 < p_1 < p_2 \cdots < p_N = 1$ *satisfies the priority hierarchy for relative service differentiation.*

Proof: Define r_i as in equation (3.1) and q_i as the average queue length of class i. At steady state, we want higher priority classes to have shorter backlogs. Hence using the relationship that $\frac{r_i}{r_1} \propto \frac{q_i}{q_1}$, we can use Little's theorem [18] to show that $p_i = \frac{\lambda_i}{\prod_{k=1}^{i-1}(1-p_k)\sum_{j=1}^{N}\lambda_j}$, $i = 1, \ldots, N$. Since $p_{i-1} = \frac{\lambda_{i-1}}{\prod_{k=1}^{i-2}(1-p_k)\sum_{j=1}^{N}\lambda_j}$ therefore $\frac{p_{i-1}}{p_i} = \frac{\lambda_{i-1}}{\lambda_i}(1 - p_{i-1})$ which can be further simplified to $p_{i-1}\lambda_i\big(\frac{1}{p_i} + \frac{\lambda_{i-1}}{\lambda_i}\big) = \lambda_{i-1}$. Since the highest priority class gets served with the highest probability, its average departure rate must be the greatest among all classes, i.e. $\lambda_1 > \lambda_2 \cdots > \lambda_N > 0$. Thus we have $p_{i-1}\big(\frac{1}{p_i} + \frac{\lambda_{i-1}}{\lambda_i}\big) > 1$. Rearranging the term leads to $\frac{p_i p_{i-1}}{p_i - p_{i-1}} > \frac{\lambda_i}{\lambda_{i-1}}$ hence $p_{i-1} < p_i$. Thus the theorem is implied. Since this assignment is independent of the number of classes in the system, $p_1 > \frac{1}{2}$.

Theorem 3 *For the special case of* $\frac{p_{i-1}}{1-p_{i-1}} \le 1$ *and* $p_i \to \frac{p_{i-1}}{1-p_{i-1}}$, $i = 2, \ldots, N-1$, *then all classes in the system are served with equal probability, i.e.* $r_i \to \frac{1}{N}$, $i = 1, \ldots, N$.

Proof: Using the RHS inequality of Theorem 1, in the event that $p_i \to \frac{p_{i-1}}{1-p_{i-1}}$ subject to $\frac{p_{i-1}}{1-p_{i-1}} \le 1$, then we obtain $p_{i-1} < p_i < \frac{p_1}{1-(N-1)p_1}$. Since $p_{N-1} < p_N = 1$, thus $\frac{p_1}{1-(N-1)p_1} < 1$. Now, if $p_1 \to \frac{1}{N}$, then $p_i \to \frac{1}{N-i+1}$ hence we obtain $r_i \to \frac{1}{N}$, $i = 1, \ldots, N$

Theorem 4 *If proportional delay differentiation is used, the average departure rate of each class is proportional to its average probability of getting served.*

Proof: We use the proportional delay constraint $\frac{r_i}{r_j} = \frac{\delta_j}{\delta_i}\frac{q_i}{q_j}$ defined in [9] where r_i is defined in equation (3.1), δ_i is part of the proportional delay target ratio and q_i is the instantaneous backlog of class i. Together with the conservation of probability $\sum_{i=1}^{N} r_i = 1$, we can show that $r_1 = \left(\sum_{j=1}^{N} \frac{\delta_1}{\delta_j}\frac{q_j}{q_1}\right)^{-1}$. Since $p_1 = r_1$ then $p_i = \left(\frac{\delta_1}{\delta_i}\frac{q_i}{q_1}\right)\left(\prod_{k=1}^{i-1}(1-p_k)\sum_{j=1}^{N}(\frac{\delta_1}{\delta_j}\frac{q_j}{q_1})\right)^{-1}$, $i = 1, \ldots, N$. At steady state, we obtain $p_i = \frac{\lambda_i}{\prod_{k=1}^{i-1}(1-p_k)\sum_{j=1}^{N}\lambda_j}$, $i = 1, \ldots, N$. Using equation (3.1), we prove the theorem.

3.3 Router Architecture

Understanding the nature and constraint of each component in the network processor is the very first step taken in programming MPP packet schedulers because these schedulers operate at those fabrics that are at the center of fast packet routing. It is crucial that programmability does not outweigh the performance gain of parallel processing in the process of packet forwarding by designing suitable QoS algorithms at line speed [3]. The Intel IXP1200 network processor[2] consists of a StrongARM processor core and 6 parallel RISC processors (microengines). Each microengine supports hardware-based multi-threading [15] and all processors run at 200MHz. StrongARM and microengines have access to off-chip 8 Mbytes of 32-bit wide SRAM, 128 Mbytes of 64-bit wide SDRAM memory, and 32-bit wide on-chip Scratchpad memory. A set of media access controller chips implement 10 Ethernet ports ($8 \times 100Mbps + 2 \times 1Gbps$). MPP schedulers reside in the data plane. Initialization and computation intensive sub-tasks in the algorithms, and performance monitoring are offloaded on the StrongARM processor.

[2]We use Intel network processor IXM1200 c-PCI hardware based on IXP1240 chipset.

3.4 Efficient Implementation of MPP in IXP1200

We implemented our proportional bandwidth guaranteed probabilistic priority multi-class framework proposed in the previous section on Intel IXP1200 network processor. To convert the class parameter p_i to tickets in lottery scheduling, all the assigned parameters p_i within a group are normalized to their least common multiple. For a large number of classes, we use Euclid's Greatest Common Divisor algorithm to speed up computation in the StrongARM core before supplying the scheduler's parameters in a numerator vector string to the microengines. For an 8-class system at an egress port, all the class parameters are stored in only two SRAM memory words with each parameter p_i occupying 8 bits thus the smallest probability being addressable is $\frac{1}{256}$ which offers relatively high computational granularity. In comparison, earlier implementation [30] will require over 100 Bytes of parameters' storage and larger memory access overheads. Clearly, our approach reduces memory access overhead drastically and accommodates more classes in a multi-port setting.

3.4.1 Fast Algorithm for Scaling Uniform Distribution

The StrongARM is elected to run a periodic task of generating uniform pseudo-random numbers in the SRAM. When the microengines require a random number for computation, they simply do a table lookup. This table has to be updated often by StrongARM to prevent a microengine from reading the same entry twice. However, we note that too high a refreshing frequency will lead to a higher latency for a microengine's SRAM read operation to this shared table due to increased contention between StrongARM and microengines. Numerous techniques exist for scaling uniform random numbers. An exact scaling method would convert the random number from an integer to a floating-point number between 0 and 1, multiply it by X, and then convert the result back to the nearest integer [33]. Alternatively, 32-bit random numbers in a particular uniform distribution, $Uniform[0, X]$ can be obtained by dividing any random 32-bit wide number in the range 0 to $2^{32} - 1$ by X and keeping the remainder under the assumption that $X \ll 2^{32} - 1$ [33]. Due to the significant computation overhead of integer division (measured as 378 cycles and independent of the value size of X), this method is not scalable without a pseudo random number generation co-processor. From the observation that each bit in any 32-bit uniformly distributed random number has an equal chance of being a "1" or "0", we use a simple generic bitwise algorithm to map this uniform random number into another equally uniform random number, effectively scaling $Uniform\left[0, 2^{32} - 1\right]$ to $Uniform\left[0, X\right]$. This algorithm shown in Table 3.7 first performs the AND operation, and then re-claims those bits lost in the AND operation, ignoring bits which are outside the desired range. It is noteworthy that the instruction cycle count for this algorithm depends on the value size of X, i.e. we can trade-off computational granularity with speed. For X less than 255, this algorithm takes 41 instruction cycle counts. In the worst case, mapping a full 32-bit value of X requires a maximum of 173 instruction cycle counts but the gain is already an exponential increase in computational granularity to approximately $2^{32} - 1$.

Table 3.7: *U-Map* scaling algorithm

```
int result        =  0;
int comparator    =  denominator & random_number;
if(comparator == 0) comparator = denominator;
while(comparator)
{
result      | =    (comparator & random_number);
comparator  >>=   1;
}
if(result > denominator)    result = denominator ^ result;
return result;
```

3.5 Performance Study and Results

We implemented our proportional bandwidth guaranteed probabilistic priority multi-class framework proposed in the previous section on Intel IXP1200 network processor [15]. In this section, we evaluate its performance. In our experiments, we use token bucket metering to characterize the service and allocate a pre-calculated buffer space for each class. We present here the results in terms of mean delay and deadline violation probability. The topology of the experimental test-bed is shown in Fig. 3.4. All network links are full-duplex and have a capacity of 100 Mbps. We classify the traffic generated as Assured Forwarding (AF) and Best-Effort (BE). We implement 8 QoS classes with DiffServ Codepoints (DSCP) classification using our framework with two segregation groups. Each priority class in AF has marking 0x2e, 0x0a, 0x12, 0x1a, 0x22, 0x0c, 0x14, and 0 respectively. Class 1 and 2 traffic is sent from Sender 1 with the rest of the traffic in Class 3 to 8 from Sender 2. All flows are independent Poisson processes with exponentially distributed packet lengths and have the same mean sending rate and mean packet size. In order to simulate congestion, we use one IXP1200 (IXP Router 2) to generate high volume of traffic at the Gigabit out-

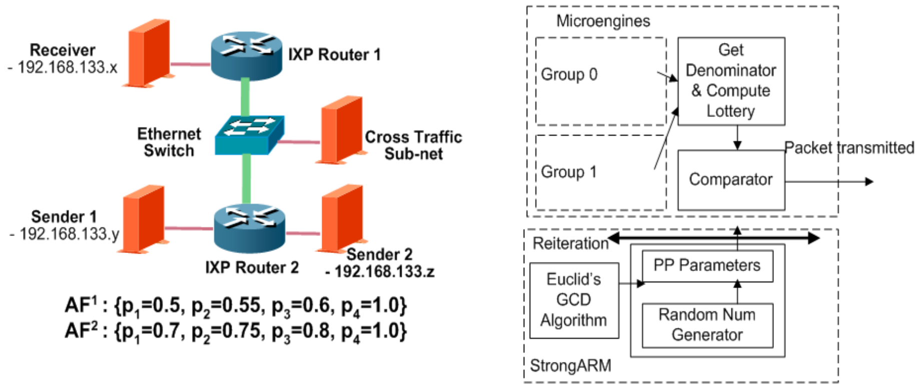


Figure 3.4: (a)Relative DiffServ Test-bed and Assured Forwarding framework configuration (b) Block diagram of implementation on IXP1200 network processor

put which is in turn forwarded to the Fast Ethernet output port on the other IXP1200 (IXP Router 1) which runs the MPP scheduler algorithm. Additional cross-traffic is also generated in the background to vary the congestion load pattern. All traffic terminates at Receiver.

The parameters for the framework are as shown in Fig. 3.4. The deadline violation probabilities of class 1 and class 2 are shown in Fig. 3.5. As expected, the deadline violation probabilities of class 1 and class 2 of MPP with ticket transfer scheme lie in between that of normal PP and SP. At low load, the deadline violation probability is very close to that of SP. Fig. 3.6 shows the average delay ratio between classes of the MPP with ticket transfer scheme measured within an interval of 1 hour. As the traffic load increases, the delay differences between classes become wider. Thus, with appropriating setting of the class parameter as described in section 4, higher priority classes get better delay differentiation at medium to high load. We also observe that the packet loss for each class in our experiments is strictly increasing as the priorities get lower. Note in Fig. 3.6 that the delay spacing between the

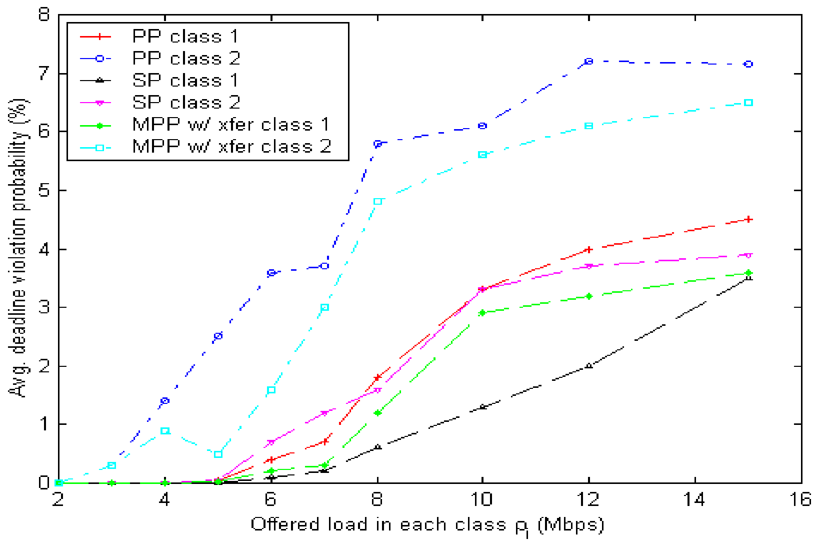


Figure 3.5: Deadline violation probabilities

last class in AF_1 and the first class in AF_2 is quite small. However, MPP scheduler observes the strict priority rule between segregation groups hence we can expect packet loss and deadline violation probability of the first class in AF_2 to be higher.

In order to compare the impact of packet sizes on the performance of the MPP with ticket transfer scheme with PP scheme under congested conditions, we repeated the same experiments with different packet size distributions. The observed experimental results were largely similar to those obtained above but we note that for large packet size close to MTU, the benefit of the ticket transfer algorithm is not so obvious because, at high load, the time for a single packet transmission becomes longer thereby increasing the probability of the network condition where all classes' HOL packets are present as is in the case of SP. Nevertheless, the queuing delays in this case are still not as high as those for SP or the PP scheme. In summary, the MPP with ticket transfer scheme is good when the primary goal

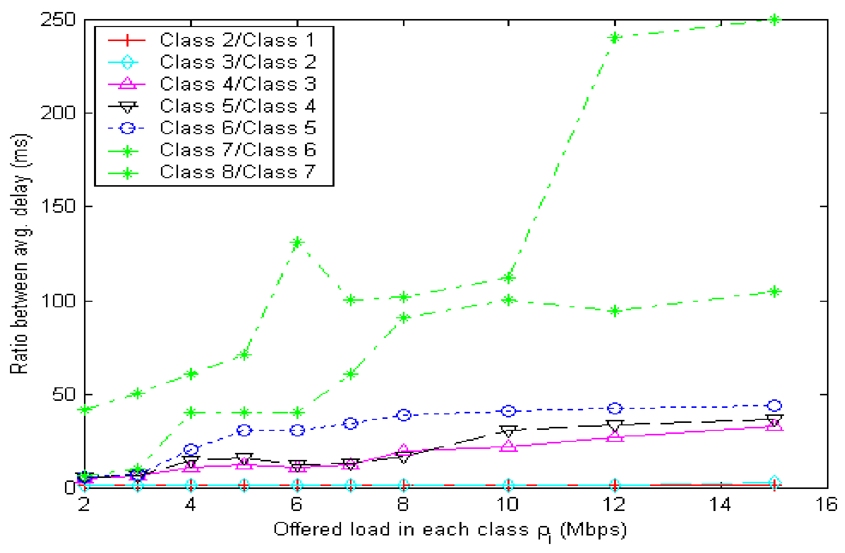


Figure 3.6: Delay ratios between classes

is to provide relative delay differentiation as in SP while ensuring that deadlines of higher priority classes are not unnecessarily violated, and also meeting specific timing requirements, for eg. small delay bounds for high priority classes as in absolute QoS.

Chapter 4

Waiting Time Priority Scheduling

In this chapter, we bridge the mathematical framework of WTP and PDD by deriving a sufficient condition for WTP to achieve PDD. We also develop technique to efficiently compute the WTP scheduler parameters.

4.1 A Sufficient Feasibility Condition For PDD

We first review some mathematical backgrounds of the WTP scheduler. The avid reader will recognize that all the assumptions made in the analysis here are based on related works [9, 21, 27]. Assuming Poisson arrival λ_i and general service time characterized by x_i and $\overline{x_i^2}$, Kleinrock [18] has shown that the average waiting time of WTP is

$$W_p = \frac{(W_o/1-\rho) - \sum_{i=1}^{p-1} \rho_i W_i[1-(b_i/b_p)]}{1-\sum_{i=p+1}^{N} \rho_i[1-(b_p/b_i)]} \quad , p = 1, \ldots, N \tag{4.1}$$

where $W_o = (1/2)\sum_{i=1}^{N} \lambda_i \overline{x_i^2}$ is the expected residual service time, b_i, $i = 1, \ldots, N$ is the scheduler parameter and the system utilization denoted by ρ is equal to $\sum_{i=1}^{N} \rho_i$ where $\rho = \lambda_i \overline{x_i}$.

For the general case of N classes, Leung *et al.* [21] has derived the following necessary condition for positive scheduler parameters to exist:

Theorem 5 *(Leung et al.) A necessary condition to have positive solutions of the b_i's is $R(1) > 0$ and $R(N) < 0$ where $R(i) = \frac{W_0}{1-\rho} - \sum_{k=1}^{i-1} \rho_k W_k - \left(1 - \sum_{k=i+1}^{N} \rho_k\right) W_i$, $i = 1, \ldots, N$.*

Using the above condition, we can easily determine if a given delay proportional differentiation can be achieved or not. However, if a given delay ratio cannot be achieved, the above theorem does not tell us exactly what the current load spacing can offer. Since Theorem 5 is a necessary condition for positive solutions of b_i's, there exist some load distributions that will pass the test and yet still be infeasible because the constraint that $b_{i+1} > b_i$ is not observed. For example, a load distribution of $\rho_1 = 0.43$, $\rho_2 = 0.29$ and $\rho_3 = 0.2$ in a 3-class WTP with a target ratio of 2 between each class will satisfy Theorem 5 but the scheduler parameters obtained are $b_1 = 1$, $b_2 = 1.27$ and $b_3 = 0.68$. By feasibility of a load distribution for general $N-$class, we mean that positive solutions of b_i's that satisfy the constraint $b_{i+1} > b_i$ can be found. If a given delay proportional differentiation cannot be achieved, it is also unknown how one can tune the existing load distribution and still achieve it. For example, a small reallocation of $\rho_1 = 0.42$, $\rho_2 = 0.29$ and $\rho_3 = 0.21$ will yield a feasible solution. Also, computing $R(1)$ and $R(N)$ involves iterative calculation of all N average delays. In the later part, we will show that Theorem 5 is equivalent to the average delay of the lowest priority class being smaller and the average delay of the highest priority class being higher than that of strict prioritization thus calculation is greatly simplified.

First, we extend the above theorem to provide more insights between load spacing and N-class system.

Theorem 6 *For N classes of traffic, let S_i be the target ratio of the average waiting time of class i traffic to that of class N traffic. Then for a maximum achievable target ratio S_1 in any system load distribution to exist, both S_1 and the total system utilization must satisfy*

$$S_1 < \left(1-\rho\right)^{-2}$$

Proof: From Theorem 5, we have $R(1) = \frac{W_0}{1-\rho} - \Big(1-(\rho-\rho_1)\Big)W_1$. Since $R(1) > 0$, therefore $W_0 > (1-\rho)(1-\rho+\rho_1)W_1$. Similarly, using the *conservation law principle* [18] where $\frac{W_0}{1-\rho} = \frac{1}{\rho}\sum_{i=1}^{N}\rho_i W_i$,

$$\begin{aligned} R(N) &= \frac{W_0}{1-\rho} - \sum_{k=1}^{N-1}\rho_k W_k - W_N \\ &= \frac{W_0}{1-\rho} - \left(\frac{\rho}{1-\rho}W_0 - \rho_N W_N\right) - W_N \\ &= W_0\left(\frac{1}{1-\rho} - \frac{\rho}{1-\rho}\right) + W_N\left(\rho_N - 1\right) \\ &= W_0 - \left(1-\rho_N\right)W_N \end{aligned} \tag{4.2}$$

Since $R(N) < 0$, therefore $W_0 - (1-\rho_N)\,W_N < 0$ and we get $W_0 < (1-\rho_N)W_N$. Hence

$$(1-\rho)(1-\rho+\rho_1)W_1 < W_0 < (1-\rho_N)W_N \tag{4.3}$$

Since $W_0 > 0$ hence we obtain $(1-\rho)(1-\rho+\rho_1)W_1 < (1-\rho_N)W_N$. Following [21], let us define $S_i = \frac{W_i}{W_N}$. Hence we have

$$\frac{1}{S_1} > \frac{(1-\rho)(1-\rho+\rho_1)}{1-\rho_N} \tag{4.4}$$

After rearranging the terms, we have

$$S_1\left(1-\rho\right)\rho_1+\rho_N<1-S_1\left(1-\rho\right)^2 \tag{4.5}$$

Since the terms on the LHS of the inequality is always positive for a stable system, i.e. $\rho<1$, the terms on the RHS of the inequality must be strictly positive. Hence

$$S_1<\left(1-\rho\right)^{-2} \tag{4.6}$$

Remarks: The implication of the above theorem is that to build any system using WTP scheduler, it is necessary that the total system utilization has to be sufficient for a desired maximum achievable target ratio between any two classes to exist regardless of the number of classes and load distribution in the system. For example, to achieve a target ratio of $S_1 = 10$, then the system regardless of the number of classes has to be at least 68% utilized so as to achieve the desired waiting-time spacing. Only after this fundamental requirement has been satisfied, then there exists the feasibility regions that are dependent on load distribution and the number of classes. In [21], the authors have shown that the system has to be at least 90% utilized in order to satisfy $S_1 = 10$ for two classes. Note that the fundamental requirement that the minimum system utilization of 68% has already been satisfied by the constraint of a minimum system utilization of 90%. Though for a total system utilization of 68%, a 2-class system can never achieve a ratio of 10 whereas a 3-class system can achieve close to $S_1 = 10$ with a load distribution of $\rho_1 = 0.001$, $\rho_2 = 0.678$ and $\rho_3 = 0.001$ (Service time is normalized to 1 so system load utilization is 0.68). On the other hand, if the system utilization is at most 67%, a target ratio of $S_1 = 10$ can never exist for any load distributions or number of classes. The physical interpretation of equation (4.6) is the maximum operational ratio that WTP can achieve for a given system utilization under light, moderate or heavy load condition.

Lemma 1 *For N classes of traffic, let S_i be the target ratio of the average waiting time of class i traffic to that of class N traffic. Then the target ratio for class i can be achieved only if the target ratios for all classes satisfy*

$$S_1 \sum_{i=1}^{N} \frac{\rho_i \overline{W_i}^{SP}}{\overline{W_1}^{SP}} < \sum_{i=1}^{N} \rho_i S_i < \sum_{i=1}^{N} \frac{\rho_i \overline{W_i}^{SP}}{\overline{W_N}^{SP}} \quad , i = 1, \ldots, N$$

where $S_i < S_1$, $i = 2, \ldots, N$ and the system load distribution satisfies

$$S_1 < \frac{\overline{W_1}^{SP}}{\overline{W_N}^{SP}}$$

Proof: For N classes of traffic, the constraint in equation (4.4) must be satisfied. Since $W_1 \geq W_2 \geq \cdots \geq W_N$ hence by definition, $S_1 \geq S_2 \geq \cdots \geq S_N$. Now, again using the *conservation law principle*, we have

$$\frac{W_0}{1-\rho} = \frac{1}{\rho} \sum_{i=1}^{N} \rho_i S_i W_N \tag{4.7}$$

Letting $W_0 = \frac{1-\rho}{\rho} \sum_{i=1}^{N} \rho_i S_i$ and substituting in equation (4.3), we get

$$S_1 \rho (1 - \rho + \rho_1) < \sum_{i=1}^{N} \rho_i S_i < \frac{\rho}{1-\rho} (1 - \rho_N) \tag{4.8}$$

Furthermore, using the *conservation law*, it can be shown that

$$\frac{\rho}{1-\rho} \left(1 - \rho_N\right) = \sum_{i=1}^{N} \frac{\rho_i \overline{W_i}^{SP}}{\overline{W_N}^{SP}}$$

$$\rho \left(1 - \rho + \rho_1\right) = \sum_{i=1}^{N} \frac{\rho_i \overline{W_i}^{SP}}{\overline{W_1}^{SP}}$$

$$\frac{1-\rho_N}{(1-\rho)(1-\rho+\rho_1)} = \frac{\overline{W_1}^{SP}}{\overline{W_N}^{SP}}$$

Substituting them into equations (4.4) and (4.8), we prove the lemma.

Remarks: The implication of the above lemma is that we just need to check the boundary condition for the maximum delay spacing between the highest priority and the lowest priority class to obtain a spectrum of achievable delay spacings between all classes. Moreover, if this maximum waiting-time ratio is achievable, the scheduler parameters b_i's, $i = 1, \ldots, N$ are guaranteed by Theorem 5 to be positive for feasible load distributions. To obtain a high delay ratio, it is necessary that the system utilization ρ is large *and* both ρ_1 and ρ_N is small. The theorem also implies that strict prioritization achieves the largest possible delay differentiation.

Another implication is, given that the system utilization remains unchanged, we are able to increase the achievable maximum waiting-time ratio by demoting some of highest priority class, class N traffic to lower priority classes or promoting some of the lowest priority class, class 1 traffic to higher priority classes. It is often that the delay ratio spacing increases faster for the latter case where the following corollary elaborates.

Corollary 1 *If the maximum waiting-time target ratio is large, it increases faster when some traffic from the lowest priority class switches to higher priority classes as compared to the case when some traffic from the highest priority class switches to the lower priority classes.*

Proof: Let us consider the first case where some lowest priority traffic moves to higher priority classes. Let $S_1^{max} = \frac{1-\rho_N}{(1-\rho)(1-\rho+\rho_1)}$. Taking partial derivatives, we obtain

$$\frac{\partial S_1^{max}}{\partial \rho_1} = -\frac{1-\rho_N}{(1-\rho)(1-\rho+\rho_1)^2}$$

Similarly for the second case where some highest priority traffic moves to lower priority classes, we get

$$\frac{\partial S_1^{max}}{\partial \rho_N} = -\frac{1}{(1-\rho)(1-\rho+\rho_1)}$$

The negative signs denote that S_1^{max} increases as ρ_1 or ρ_N decreases. Since $1-\rho+\rho_1$ is strictly less than 1, $\frac{1}{(1-\rho)(1-\rho+\rho_1)} \ll \frac{1}{(1-\rho)(1-\rho+\rho_1)^2}$. Hence if S_1^{max} is large, both ρ_1 and ρ_N are relatively small. Thus $\|\frac{\partial S_1^{max}}{\partial \rho_1}\| > \|\frac{\partial S_1^{max}}{\partial \rho_N}\|$ when ρ_N is very small.

Corollary 2 *Consider a system with N classes. For a given load distribution and given the same waiting-time target spacing $r_{i,i+1}$, $i = 1, \ldots, N-1$, between classes, the maximum waiting-time target ratio $r_{i,i+1}^{max}$ is*

$$r_{i,i+1}^{max} = \left(\frac{(1-\rho_N)}{(1-\rho)(1-\rho+\rho_1)}\right)^{\frac{1}{N-1}}$$

Proof: For the above system configuration, $S_1 = (r_{i,i+1})^{N-1}$. Putting this into equation (4.4), we obtain $(r_{i,i+1})^{N-1} < \frac{(1-\rho_N)}{(1-\rho)(1-\rho+\rho_1)}$. Thus

$$r_{i,i+1} < \left(\frac{(1-\rho_N)}{(1-\rho)(1-\rho+\rho_1)}\right)^{\frac{1}{N-1}} \tag{4.9}$$

Hence the corollary is proved. Alternatively, we can also find the maximum number of classes required to support a given waiting-time ratio which is
$N < 1 + \left(log_e r_{i,i+1}\right)^{-1} log_e\left(\frac{1-\rho_N}{(1-\rho)(1-\rho+\rho_1)}\right)$.

A consequence of Lemma 1 is that certain distribution of ρ_i's will not lead to a positive solution of b_i's due to the constraint that $b_{i+1} > b_i$. In such cases, the system cannot achieve the waiting time ratios. To determine only the feasible load distributions for PDD, we have the following theorem.

Theorem 7 *A sufficient condition for the feasibility of a set of N average class delays using WTP that conforms to the PDD model for $\rho < 1$ is*

$$\frac{\overline{W_1}^{WTP}}{\overline{W_1}^{SP}} < \frac{\sum_{i=k}^{N} \lambda_i W_i^{WTP}}{\sum_{i=k}^{N} \lambda_i W_i^{SP}} \leq \frac{\overline{W_N}^{WTP}}{\overline{W_N}^{SP}} \quad , k = 2, \ldots, N$$

where $\frac{\sum_{i=k}^{N} \lambda_i W_i^{WTP}}{\sum_{i=k}^{N} \lambda_i W_i^{SP}}$ *are the* $N-1$ *Regnier's inequalities [27].*

Proof: Using the substitution $\delta_i = S_i/S_1$, equation (1.2) can be expressed as

$$\sum_{i=1}^{N} \lambda_i S_i \leq \frac{\sum_{i=k}^{N} \lambda_i S_i}{\sum_{i=k}^{N} \lambda_i \overline{W_i}^{SP}} \sum_{i=1}^{N} \lambda_i \overline{W_i}^{SP} \quad , k = 2, \ldots, N \tag{4.10}$$

It can further be shown that equation (4.10) is equivalent to

$$\frac{\sum_{i=1}^{k-1} \lambda_i W_i}{\sum_{i=1}^{k-1} \lambda_i \overline{W_i}^{SP}} \leq \frac{\sum_{i=k}^{N} \lambda_i W_i}{\sum_{i=k}^{N} \lambda_i \overline{W_i}^{SP}} \quad , k = 2, \ldots, N \tag{4.11}$$

Assuming the same packet size distribution for all classes, i.e. $\overline{x} = 1$, and by definition of $S_i = \overline{W_i}^{WTP} / \overline{W_N}^{WTP}$, the LHS inequality of equation (4.8) can be expressed as

$$\frac{\overline{W_1}^{WTP}}{\overline{W_1}^{SP}} < \frac{\sum_{i=1}^{N} \lambda_i \overline{W_i}^{WTP}}{\sum_{i=1}^{N} \lambda_i \overline{W_i}^{SP}}$$

Rearranging, we have

$$\frac{\overline{W_1}^{WTP}}{W_1^{STP}} < \frac{\sum_{i=2}^{N} \lambda_i \overline{W_i}^{WTP}}{\sum_{i=2}^{N} \lambda_i \overline{W_i}^{SP}}$$

which is exactly the case of equation (4.11) for $k = 2$.

and the RHS inequality of equation (4.8) can be expressed as

$$\frac{\overline{W_N}^{WTP}}{\overline{W_N}^{SP}} > \frac{\sum_{i=1}^{N} \lambda_i \overline{W_i}^{WTP}}{\sum_{i=1}^{N} \lambda_i \overline{W_i}^{SP}}$$

or equivalently,

$$\frac{\sum_{i=1}^{N-1}\lambda_i\overline{W_i}^{WTP}}{\sum_{i=1}^{N-1}\lambda_i\overline{W_i}^{SP}} < \frac{\overline{W_N}^{WTP}}{\overline{W_N}^{SP}}$$

.

Similarly, this is exactly the case of equation (4.11) for $k = N$.

Moreover, by the *conservation law*, $\sum_{i=1}^{N}\lambda_i\overline{W_i}^{WTP} = \sum_{i=1}^{N}\lambda_i\overline{W_i}^{SP}$ hence it follows that Theorem 5 is equivalent to being that both the average delay of the highest priority class in WTP must necessarily be larger than that of the SP scheduler and the lowest priority class in WTP must necessarily be less than that of the SP scheduler.

Using equation (4.10), WTP conforms to the necessary and sufficient feasibility conditions of the PDD model if and only if

$$\frac{\overline{W_1}^{WTP}}{\overline{W_1}^{SP}} < \frac{\sum_{i=k}^{N}\lambda_i W_i^{WTP}}{\sum_{i=k}^{N}\lambda_i W_i^{SP}} \leq \frac{\overline{W_N}^{WTP}}{\overline{W_N}^{SP}} \quad , k = 2, \ldots, N \tag{4.12}$$

Notice that the Regnier's inequalities [27] are defined as

$$\frac{\sum_{i=k}^{N}\lambda_i\overline{W_i}}{\sum_{i=k}^{N}\lambda_i\overline{W_i}^{SP}} \geq 1 \quad , k = 2, \ldots, N$$

and therefore $\frac{\overline{W_N}^{WTP}}{\overline{W_N}^{SP}} \geq 1$ as expected.

Since the necessary condition of Theorem 5 is a subset of PDD model in equation (4.10), equation (4.12) characterizes the set where the feasible average delays of PDD are mapped to the average delays that satisfies theorem 5 in WTP. Note that in the case of $N = 2$, from equation (4.4) or (4.12), we obtain $\rho > 1 - \frac{1}{S_1}$ which is the main result of theorem 1 in [20, 21, 12] for the necessary and sufficient conditions to achieve the specified performance ratio S_1 in a 2-class system.

We conclude that the feasibility condition for WTP to achieve PDD is *dependent* on the class load distribution and the total system utilization for the general case of $N > 2$.

This allows us to relate results for WTP to the PDD model for general N-class in a precise manner. The result for the case of two traffic classes in the PDD model has been proved to be similar to that of WTP where feasibility is dependent only on the total system utilization [9, 20]. The derivation of Theorem 7 is based on the same assumption as the derivation of the PDD model, i.e., the N classes have the same packet size distribution hence if Theorem 7 is satisfied then WTP exhibits the delay dynamics in the PDD model [9] which we list below for completeness sake. Based on Theorem 7, we elaborate the following statement in [21]: Even though the system utilization ρ remains unchanged, it is still possible that certain distributions of ρ_i's will not lead to a positive solution of b_i's. Specifically, the system can search for positive solutions of b_i's to achieve the target waiting-time ratio if the load distribution ρ_i, $i = 1, \ldots, N$ and the average delay of each class satisfies equation (4.12) by using PDD delay dynamics.

Corollary 3 *If $\overline{W_i}, i = 1, \ldots, N$ falls in the schedulability region defined by equation (4.12),*

Property 1: *Increasing the input rate of a class, increases (in the wide sense [1]) the average delay of all classes.*

Property 2: *Increasing the rate of a higher class causes a large increase in the average class delays than increasing the rate of a lower class.*

Property 3: *Decreasing the delay differentiation parameter of a class increases (in the wide sense) the average delay of all other classes, and decreases (in the wide sense) the average delay of that class.*

Suppose that the class load distribution changes from λ_n to $\lambda_n^{'}$ with $\lambda_i^{'} = \lambda_i - \epsilon$,

[1]Increasing a function $f(x)$ in the wide sense means $\frac{df(x)}{dx} \geq 0$.

$\lambda_j^{'} = \lambda_j + \epsilon$, and $\lambda_k^{'} = \lambda_k$ for all $k \neq i, j (\epsilon > 0)$. Let $\overline{W_n'}$ be the average delay in class n when the class load distribution is $\lambda_n^{'}$.

Property 4: *If $i > j$ then $\overline{W_n'} \leq \overline{W_n}$ for all $n = 1, \ldots, N$. Similarly, if $i < j$ then $\overline{W_n'} \geq \overline{W_n}$*

Property 5: *If $i > j$ then $\overline{W_j'} \geq \overline{W_i}$. Similarly, if $i < j$ then $\overline{W_j'} \geq \overline{W_i}$.*

It is possible to obtain a special case of Theorem 7 that has physical implication.

Corollary 4 *For a feasible load spacing, if the target ratios of all consecutive classes in WTP are less than the consecutive ratios of average delays in SP, the average delays of all classes in WTP satisfy PDD.*

Proof: We prove the result in a converse manner. Assume that all the average delays in WTP satisfy PDD, i.e., they satisfy equation (4.12). We first prove that the result is true for the highest priority class and its immediate low priority class, i.e. Class $N-1$, since this result follows immediately in equation (4.12) for $k = N-1$. For the case of $2 \leq k \leq N-2$ in equation (4.12), we see that a constrained ordering of the Regnier's inequalities such that $R_i < R_{i+1}$, $i = 2, \ldots, N-2$ where R_i denotes the Regnier's inequality at $k = i$, that is repeatedly applied to equation (4.12) from $k = N-2$ to $k = 2$ would therefore yield the desired result.

It was proposed in [9] to deploy the SP scheduler in routers for short time intervals to measure the average delays of the SP scheduler $\overline{W}_i^{SP}$ that would result in a SP scheduler in order to access the feasibility condition of PDD. From Corollary 4, we see that this similar mechanism can be used to determine the achievable ratios for WTP to achieve PDD.

4.2 Improvement to Iterative Computation of Scheduler Parameters

In the previous section, we examined the relationship between WTP and PDD under all load conditions. In this section, we want to show that the ratio of consecutive scheduler parameters will always be more than the consecutive target ratio values. This helps to increase the efficiency of computation using iterative methods by detecting infeasible load distributions that satisfy Theorem 5 as early as possible. We use the Gauss-Seidel numerical algorithm in [20, 21] to solve the set of nonlinear equations and we set the maximum iteration count to be 200 with a predefined error threshold, $\epsilon = 10^{-3}$ for all computations, unless otherwise stated.

Theorem 8 *For $\rho < 1$, the feasibility condition of WTP satisfies the following $N-1$ inequalities*

$$\frac{b_p}{b_{p+1}} < \frac{W_{p+1}}{W_p} \quad , p = 1, \ldots, N-1$$

Proof: For $N = 2$, let $r_t = W_1/W_2$ then from [21], $b_2/b_1 = \rho/\left(\rho - 1 + \frac{1}{r_t}\right)$ and hence

$$\frac{b_2}{b_1} = r_t\left(\frac{\rho}{\rho r_t - r_t + 1}\right)$$

But $\rho r_t - r_t + 1 - \rho = (\rho - 1)(r_t - 1)$ and since $\rho < 1$ and $r_t > 1$ thus

$$\frac{b_2}{b_1} > r_t$$

For general N classes, Kleinrock has derived in [18] the relationship between the expected time, V_i for i^{th} customers (for $i > p$) that arrive and get served before the p^{th} packet does,

$W_p, p = 1, \ldots, N$.

$$V_i = W_p\left(1 - \frac{b_p}{b_i}\right) \quad i > p,\ p = 1, \ldots, N-1 \tag{4.13}$$

After rearranging, we have $b_p/b_i = (W_p - V_i)/W_p$ and since none of i^{th} customers that arrive and are buffered gets dequeued within the interval $W_p - V_i$, i.e. $W_p - V_i < W_i$ thus

$$\frac{b_p}{b_i} < \frac{W_i}{W_p} \quad i > p,\ p = 1, \ldots, N-1 \tag{4.14}$$

Since for all $i > p$, $p = 1, \ldots, N$, all the $(N-1)(N-2)/2$ inequalities with $i > p+1$ can be inferred from the $N-1$ inequalities with $i = p+1$ hence the $N(N-1)/2$ inequalities in equation (4.14) are equivalent to the following $N-1$ inequalities

$$\frac{b_p}{b_{p+1}} < \frac{W_{p+1}}{W_p} \quad , p = 1, \ldots, N-1 \tag{4.15}$$

A direct implication of Theorem 8 is that we are able to check if a given measured load distribution which has fulfilled Theorem 5 is feasible or not after the first few iterations of any numercial algorithm for example, the genetic optimization algorithm in [12] or Gauss-Seidel iteration method [20, 21] to solve the set of non-linear equations in equation (4.1) for the scheduler parameters. The rate of convergence to Theorem 8 can also be used as a means for comparison of the effectiveness between various numerical algorithms. In particular, the initial values of the scheduler parameters are the inverse of the target average delay values thus ratios of consecutive feasible scheduler parameters will increase to be larger than the initial values after a certain number of loops whereas ratio of infeasible scheduler parameters can be detected since it does not satisfy Theorem 8 at any iteration. As a comparison, for a load distribution of $\rho_1 = 0.43$, $\rho_2 = 0.29$ and $\rho_3 = 0.2$ in a 3-class WTP with a target ratio of 2 between each class ,the violation that $b_{i+1} < b_i$ is only detected after the Gauss-Seidel algorithm runs to completion after the maximum iteration count, i.e. 200 loops in our case, whereas this violation could be detected by setting a predefined iteration threshold less than the maximum iteration count and checking whether Theorem 8 is violated. Empirical studies using the Gauss-Seidel method show that the ratios of feasible scheduler parameters usually converge to values that satisfy Theorem 8 after 40 loops. Hence a predefined iteration threshold of 40 can be set to detect infeasible load spacings.

4.2.1 Load Dynamic Adjustment Control Technique

We now present an efficient Dynamic Adjustment Control (DAC) technique that enhances the computation of scheduler parameters. Solving the non-linear set of equations is a very computational intensive task in packet scheduling. As mentioned earlier, we need to identify those infeasible load distributions that satisfy Theorem 5 before using them as input for any iterative methods so that routers do not waste resources in computing infeasible scheduler parameters. To achieve this, the pseudo-code of the algorithm given in Table 4.1 is proposed.

Line 5 checks whether the total system utilization is adequate to provide for a desired target ratio. If the system is underutilized, the desired maximum target ratio S_1 has to be reduced. Line 7 computes the scheduler parameters using iterative methods if Thereom 5 in line 6 is satisfied and infeasible load distributions are simply identified after a predefined iteration threshold of 40. The complexity involved is only $N-1$ comparisons once the predefined iteration threshold is reached. If the load distribution is not feasible, the load distribution is slightly re-adjusted to satisfy Theorem 7 in line 18. The computational complexity to verify Theorem 7 is O(N) since $N - 2$ Regnier's inequalities have to be computed and compared. If computational budget is not tight, we can first compute the feasibility of the load distribution using Theorem 7. If it does not satifsy Theorem 7, we adjust the load spacing until Theorem 7 is satisfied and thereafter we compute the scheduler parameters using the iterative algorithm. The re-adjustment of load distribution can be made by relaxing certain desired target ratios between classes to obtain different average delays that satisfy Theorem 7. Alternatively, the router precomputes equation (4.12) and informs the sources that certain class selection parameters are not available.

4.3 Numerical Results

In this section, we present the results of our experiments.

4.3.1 Experiment 1: Comparison between maximum achievable target ratios and load distributions

In [13], the authors attempt to use offline genetic algorithm to compute scheduler parameters for a 4-class system with equal load distribution at $\rho = 0.7$ and $\rho = 0.75$ for a consecutive target ratio of 2 but could not conclude whether the infeasibility is due to insufficient system utilization or inaccuracy of the genetic algorithm. Using equation (4.9), we could compute the maximum achievable consecutive target ratio which turns out to be 1.796 and 1.951. We conclude that by Theorem 6, the system utilization is adequate but the load distribution is not feasible for a target ratio of 2. Next, we consider a 3-class system at a low load of $\rho = 0.6$. To achieve $r_{i,i+1} = 2$, we know by Theorem 6 that at $\rho = 0.6$,

Table 4.1: Outline of the Dynamic Adjustment Control algorithm

At each periodic computation, measure the load in each class 1 to N,

Input ρ_i, $\overline{x}_i$, $\overline{x^2}_i$, S_i for $i = 1, \ldots, N$

```
1.  begin
2.    init_IterativeAlgo();
3.    LoopThres= 40; Load=FEASIBLE; /* FEASIBLE=0, INFEASIBLE=1 */
4.    $\rho = \sum_{i=1}^{N} \rho_i$;
5.    if($S_1 < 1/(1-\rho)^2$)
6.        if(Theorem_5()==TRUE)
7.          while(IterativeAlgo())
8.              if(iteration_loop==LoopThres and Theorem_8()==TRUE )
9.                  continue;
10.                 else return ( Load=INFEASIBLE );
11.             end
12.         end/* End IterativeAlgo() */
13.   else
14.       reduce_TargetRatio();/* Inadequate total system utilization */
15.   end
16.   if(Load==INFEASIBLE)
17.       while(Theorem_7()==FALSE)
18.           re-adjust_Load();/* Infeasible load distribution */
19.       end
20.   end
21. end
```

some load distributions can satisfy this target ratio since the maximum target ratio is 4. However, it is hard to satisfy both Theorem 5 and 7 when the load in Class 1 and 3 is too high hence by Corollary 1 we see that by moving some of Class 1's load to Class 2 or Class 3's load to Class 2 as shown in Table 4.2, the target ratio between each class can be met. However, this may not be satisfactory in all cases, since we know from the delay dynamics of the PDD model that moving some of Class 1's load to Class 2 increases the average delays of all classes.

4.3.2 Experiment 2: Using predefined iteration threshold

To illustrate the effectiveness of a predefined iteration threshold using Gauss-Seidel algorithm, we select the load distributions that satisfy Theorem 5. We consider three classes of traffic. The arrival process of class i $(i = 1, 2, 3)$ is Poisson with a load of ρ_i. The number of packets is generated to be at least 50,000 for each class in each set of experiment. The mean service time is taken to be the unit of time and the service times of packets in each class follow the same exponential distrbution with unit mean. We first vary the load distribution, and show the number of loops that satisfy Theorem 8 and the scheduler parameters obtained in Table 4.3. As observed, feasible load distributions seldom take more than 40 loops to be larger than the target ratios.

Table 4.4 illustrates that the predefined error threshold parameter in the Gauss-Seidel algorithm has no effect on the number of loops executed to satisfy Theorem 8 although the total number of loops executed decreases significantly as the error threshold ϵ gets larger. In other words, the rate of convergence to ratios that satisfy Theorem 8 is independent of the parameters of the Gauss-Seidel algorithm. Table 4.5 shows that consecutive target ratios affect the rate of convergence. As consecutive target ratio increases, more number of loops are executed before Theorem 8 is satisfied. Table 4.6 shows some infeasible load distribution. We use R_2 to denote the Regnier's inequality at $k = 2$. As a ballpark estimate, if the load is measured as in Table 4.6, the Gauss-Seidel Algorithm with predefined iteration threshold executes $40 \times 7 = 280$ loops before declaring that the load distributions are infeasible while the normal Gauss-Seidel Algorithm executes $200 \times 7 = 1400$ loops hence the savings in computation is around 80%.

It is probable that some feasible load distributions will take more number of loops than the predefined iteration threshold to converge to satisfy Theorem 8 so rejecting a load distribution based on a hard threshold is not foolproof. However, we verify by simulation that, as time progresses, consecutive scheduler parameter ratios will tend to converge to values that satisfy Theorem 8 notwithstanding that it may take rather long for some feasible load distributions, for example as shown in Table 4.7. In the case of infeasible load distributions, this convergence is nonexistent. Hence, to identify infeasible load distribution, a better method would be to make use of this property of convergence and compare the $N-1$ inequalities at two predefined iteration thresholds.

We conclude from these experiments that the rate of convergence to Theorem 8 using the Gauss-Seidel algorithm is rather effective for most load distributions. For a relative accurate error threshold (10^{-3}), the rate of convergence takes approximately 10% of the total computation time.

Table 4.2: Comparison between maximum achievable target ratios S_1 and load distributions

Load distribution	S_1	Satisfied Thm. 5	Satisfied Thm. 6
0.2-0.2-0.2	3.333	No	No
0.15-0.35-0.1	4.000	No	No
0.1-0.4-0.1	4.5	Yes	Yes
0.05-0.5-0.05	5.278	Yes	Yes

Table 4.3: Estimation of predefined iteration threshold

Load distribution	Loops	Loops to satisfy Thm. 8	b_1	b_2	b_3
0.4-0.3-0.25	68	7	1	2.131	4.635
0.4-0.25-0.3	195	23	1	2.135	4.650
0.4-0.24-0.31	200	36	1	2.141	4.658
0.4-0.29-0.26	78	9	1	2.132	4.638
0.4-0.31-0.24	61	7	1	2.123	4.631
0.4-0.35-0.2	43	5	1	2.126	4.620
0.35-0.2-0.4	41	3	1	2.146	4.664
0.3-0.25-0.4	16	1	1	2.146	4.644
0.35-0.4-0.2	18	1	1	2.126	4.603
0.2-0.45-0.3	7	1	1	2.135	4.576

Table 4.4: Effect of predefined error in Gauss-Seidel algorithm on the predefined iteration threshold

Load distribution	Loops ($\epsilon = 10^{-2}$)	Loops to satisfy Thm. 8 ($\epsilon = 10^{-2}$)	Loops ($\epsilon = 10^{-1}$)	Loops to satisfy Thm. 8 ($\epsilon = 10^{-1}$)
0.4-0.3-0.25	45	7	22	7
0.4-0.25-0.3	128	23	61	23
0.4-0.24-0.31	200	36	100	36
0.4-0.29-0.26	52	9	25	9
0.4-0.31-0.24	40	7	20	7
0.4-0.35-0.2	29	5	14	5
0.35-0.2-0.4	27	3	13	3
0.3-0.25-0.4	11	1	5	1
0.35-0.4-0.2	12	1	6	1
0.2-0.45-0.3	5	1	3	1

Table 4.5: No. of loops to satisfied Theorem 8 under different ratio targets ($r_{i,i+1}$)

Ratio target (0.4-0.29-0.21)	Loops to satisfy thm. 8 ($\epsilon = 10^{-3}$)
1.1	3
1.7	9
2.0	13
2.7	17
3.0	17
3.5	17

Ratio target (0.35-0.3-0.3)	Loops to satisfy thm. 8 ($\epsilon = 10^{-3}$)
1.1	1
2.0	3
3.0	5
4.0	7
5.0	9
5.5	13

Table 4.6: Infeasible load distributions that satisfy Theorem 5 but not Theorems 7 & 8

Load distribution	Loops	$r_{1,2}, r_{2,3}$ at 40th Loop	R_2	$\overline{W_3}/\overline{W_3^{SP}}$
0.46-0.29-0.2	200	$-0.468, -2.067$	5.887	5.802
0.46-0.2-0.29	200	$-0.460, -2.764$	5.393	5.332
0.49-0.2-0.26	200	$-0.394, -3.075$	5.619	5.366
0.45-0.3-0.2	200	$-2.968, -0.700$	5.84..42	5.84..41
0.45-0.2-0.3	200	$-0.486, -2.660$	5.31..96	5.31..95
0.45-0.29-0.21	200	$-0.486, -2.660$	5.796	5.769
0.41-0.23-0.31	200	$3.085, 1.284$	5.799	5.439

Table 4.7: Comparison between infeasible and feasible load distributions that exceed pre-defined iteration threshold. The values at the 40th, 80th and 120th loops are shown.

Load distribution	$r_{1,2}, r_{2,3}$ (40th)	$r_{1,2}, r_{2,3}$ (80th)	$r_{1,2}, r_{2,3}$ (120th)
0.41-0.23-0.31 (infeasible)	$3.085, 1.284$	$18.453, 0.165$	$-1.901, -1.071$
0.42-0.29-0.21 (feasible)	$2.641, 1.789$	$2.563, 1.878$	$2.161, 2.534$

4.3.3 Experiment 3: Effectiveness of DAC to dynamic load variation

It is difficult to preallocate load distribution in advance so we want to evaluate the effectiveness of the DAC in detecting infeasible load spacings in dynamic load variation. We consider Long Range Dependent (LRD) traffic modeled as Pareto On-off processes with shape parameter 1.9 since aggregated traffic in real DiffServ networks is LRD in nature. In this experiment, we use fixed-size packets and packets with bimodal distribuition (64 bytes and 1500 bytes). The number of packets is generated to be at least 50,000 for each class. The mean service time for a packet is taken to be the unit of time and the monitoring timescale is 100 packet transmission times. The consecutive target ratio between each class is 2. We use $\rho_i\{i = 1, 2, 3\}$ to denote load distribution. We consider a 3-class and a 4-class system and the starting load vectors are $\rho_i\{0.45, 0.29, 0.21\}$ and $\rho_i\{0.35, 0.2, 0.3, 0.1\}$ respectively. We keep the total system utilization constant at $\rho = 0.95$ but each load variation in Class 2 and 3 is uniformly distributed with a mean of 0.01 and occurs at every 50 packet transmission times. At each monitoring timescale, we measure the load and compute the scheduler parameters when the load distribution is feasible.

In the first part, we use a single hard threshold of 40 loops . Table 4.8 and 4.9 show the number of infeasible load spacings detected and the number of feasible load spacings that are wrongly identified as infeasible for a 3-class and 4-class system respectively. Note that all the infeasible load distributions listed in the tables are observed to satisfy Theorem 5. In the second part, we repeat the setting but we use the method of two thresholds spaced at 40 loops apart and observe that no feasible load spacing is identified wrongly. Also, as observed from Table 4.8 and 4.9, the number of feasible load distributions increases as the variance of load variation in each class increases, i.e. the range of variation in each class is larger, for example ρ_1 varies from 0.01 to 0.5086 as compared to ρ_1 from 0.4516 to 0.6436 in Table 4.8. It is also observed from extensive empirical studies that as the number of classes increases, the probability of an infeasible load distribution occurring increases rapidly.

Table 4.8: Load variation for 3-class system. The entries denote the range of variation of arrival rates and the number of times feasible, infeasible and falsely detected loads are found

Class 1	Class 2	Class 3	Feasible	Infeasible	False
$0.01 - 0.5086$	0.1990-0.4899	0.1100-0.4100	4473	284	5
$0.4516 - 0.6436$	0.1990-0.2865	0.1100-0.2100	0	4757	0

Table 4.9: Load variation for 4-class system. The entries denote the range of variation of arrival rates and the number of times feasible, infeasible and falsely detected loads are found

Class 1	Class 2	Class 3	Class 4	Feasible	Infeasible	False
$0.2100 - 0.3500$	0.1511-0.2500	0.2600-0.3500	0.0693-0.1500	699	4274	2
$0.3620 - 0.4825$	0.1500-0.2087	0.2500-0.2968	0.0500-0.1050	0	4971	0

Since infeasible load distributions occur quite frequently in dynamic load, we conclude that there is a necessity to identify these infeasible load distributions as quickly as possible. The proposed load dynamic adjustment control technique can efficiently terminate Gauss-Seidel algorithm computation when these infeasible load distributions satisfy Theorem 5.

Chapter 5

Exact Schedulability Conditions of WTP

In the previous chapter, we have shown that the ratios of the WTP scheduler parameters play a significant role in obtaining average delays to achieve PDD. Likewise, an interesting and natural question to ask would be: How would the WTP scheduler parameters affect the maximum delay bound in each class ? This chapter analyzes the WTP scheduler in the context of maximum delay bound.

5.1 Maximum Delay Analysis Using General Traffic Specifications

We assume that each connection j is assigned a priority p with $1 \leq p \leq P$. We use C_p to denote the the set of connections with priority p. Note that we do not follow the priority convention in the previous chapter but our priority convention follows that in

[22] for easy comparison, i.e., a lower class index indicates a class of higher priority. All connections in C_p have the same delay bound d_p, with $d_p < d_q$ for $p < q$. Hence, the priority of a connection is high if the maximum delay bound is small.

We use the traffic specifications defined in [22] to describe the *actual traffic arrival* from connection j where $A_j[t, t+\tau]$ provides the actual arrivals from connection j in time interval $[t, t+\tau]$. The traffic specification uses three parameters to characterize the traffic from a connection j: the period T_j, the burst size B_j, and the maximum transmission time of a packet s_j. Following [22],

$$A_j[t, t+\tau] \leq A_j^*[0, \tau] \tag{5.1}$$

and the characterization for the traffic models as:

$$s_j^* = \begin{cases} \hat{A}_j^*(t) = B_j s_j + \lfloor \frac{t}{T_j} \rfloor s_j & \text{for discrete traffic specifications} \\ \\ \tilde{A}_j^*(t) = B_j s_j + t\frac{s_j}{T_j} & \text{for continuous traffic specifications} \end{cases}$$

Since with continuous traffic specifications the packets on a connection j are infinitesimally small, we define the maximum transmission time of packets by s_j^* for all $j \in N$ as follows:

$$s_j^* = \begin{cases} s_j & \text{for discrete traffic specifications} \\ \\ 0 & \text{for continuous traffic specifications} \end{cases}$$

For each priority level p, we define $s_p = max_{j \in C_p} s_j$ and $s_p^* = max_{j \in C_p} s_j^*$. We use the term priority-P busy period to denote a busy period that is generated by connections with all priorities $p \in P$, and we denote by B_1^P the first priority-P busy period where all connections $j \in \bigcup_{q \in P} C_q$ transmit according to their rate-controlling function A_j^*, that is,

$$B_1^P = min_{t>0} \sum_{q=1}^{P} \sum_{j \in C_q} A_j^*(t) - t = 0 \tag{5.2}$$

With these definitions, we give the necessary and sufficient schedulability conditions for WTP-schedulers.

Theorem 9 *A set N of connections, where each connection $j \in N$ is characterized by (A_j^*, d_p) is WTP-schedulable for all $A_j < A_j^*$ if and only if for all priorities p and for all $0 \leq t \leq B_1^P - d_p$, there exists a τ_p with $\tau_p \leq d_p - s_p^*$ such that:*

$$t_p + \tau_p \geq \sum_{q=1}^{P} \sum_{j \in C_q} A_j^* \Big(t_p + (1 - \frac{b_p}{b_q}) \tau_p \Big) + max_{q \in P} s_q^{max} - s_p^* \tag{5.3}$$

5.1.1 Proof of Sufficiency

For WTP, we obtain in [25] the workload for all $\tau_p, 0 \leq \tau_p \leq \delta_p$,

$$W^{p,t_p}(t_p + \tau_p) = \sum_{q=1}^{P} \sum_{i \in C_q} A_i \Big(t_p - \hat{\tau_p}, t_p + (1 - \frac{b_p}{b_q}) \delta_p \Big) + R(t_p - \hat{\tau_p}) - (\hat{\tau_p} + \tau_p) \tag{5.4}$$

The condition for stability of a work-conserving packet scheduler is given by:

$$\lim_{t \to \infty} \frac{\sum_{j=1}^{N} A_j^*(t)}{t} < 1 \tag{5.5}$$

From the definition of the traffic constraint A_j^* in [22],

$$\sum_{j \in C_p} A_j[t - \hat{\tau}, t] \leq \sum_{j \in C_p} A_j^*(\hat{\tau}) \tag{5.6}$$

Hence, we obtain

$$\sum_{q=1}^{P} \sum_{i \in C_q} A_j \Big(t_p - \hat{\tau_p}, t_p + (1 - \frac{b_p}{b_q}) \delta_p \Big) \leq \sum_{q=1}^{P} \sum_{i \in C_q} A_j^* \Big[\hat{\tau_p} + (1 - \frac{b_p}{b_q}) \delta_p \Big] \tag{5.7}$$

Since the remaining nonpreemptable transmission time of priority-r traffic ($r \in P$) at time $t - \hat{\tau_p}$ is maximal if traffic with maximum transmission time[1] of a lower service

[1]For discrete traffic specifications, the traffic will result from a single packet.

priority starts transmission at $t - \tau^{-}$, we obtain:

$$R(t - \hat{\tau_p}) \leq max_{q \in P} s_r \tag{5.8}$$

With equations (5.6), (5.7) and (5.8), we can give the following bound for $W^{p,t_p}(t+\tau_p)$ as

$$W^{p,t_p}(t + \tau_p) \leq \sum_{q=1}^{P} \sum_{i \in C_q} A_j^* \left(\hat{\tau_p} + (1 - \frac{b_p}{b_q})\delta_p \right) + max_{q \in P} s_q^{max} - (\hat{\tau_p} + \tau_p) \tag{5.9}$$

By choice of $\hat{\tau_p}$, we know that $A_j^*[\hat{\tau_p} + (1 - \frac{b_p}{b_q})\delta_p] \geq 0$ since we can select $\hat{\tau_p}$ such that $\hat{\tau_p} \geq |(1 - \frac{b_p}{b_q})\delta_p|$.

Now,

$$\delta_p = s + min\{z \ \ | \ \ W^{p,t_p}(t_p + z) = s, z \geq 0\} \tag{5.10}$$

where $s_{min} \leq s \leq s_{max}$ is the transmission time of the packet.

Thus, to avoid deadline miss, for all P and all $t_p \geq 0$, we obtain with condition Theorem 9 that there exists a $0 \leq \tau_p \leq d_p - s^{min}$ such that $W^{p,t_p}(t_p + \tau_p) \leq 0$. Hence there exists a $\tau'' \leq d_p - s_k^*$ such that $W^{p,t_p}(t_p + \tau'') = s_p^*$. Therefore, with equation (5.10), the tagged packet begins transmission at time $t + \tau''$. Since the packet has at most a transmission time of s_p^*, the tagged packet does not cause a deadline violation.

Hence

$$W^{p,t_p}(t + \tau_p) \leq \sum_{q=1}^{P} \sum_{i \in C_q} A_j^* \left(\hat{\tau_p} + (1 - \frac{b_p}{b_q})\delta_p \right) + max_{q \in P} s_q^{max} - (\hat{\tau_p} + \tau_p) \leq s_p^{min} \quad \text{for all } p \tag{5.11}$$

5.1.2 Examples

For discrete traffic models, it is typically not feasible to simplify the conditions in the Theorem 9. However, for the continuous traffic specification, the conditions can be simplified as follows. Let us assume that there is only one connection p in each priority set C_p. In this case, we can rewrite the condition in Theorem 9 as:

$$t_p\left(1-\sum_{j=1}^{P}\frac{s_j}{T_j}\right)+\tau_p\left(1-\sum_{j=1}^{P}(1-\frac{b_p}{b_q})\frac{s_j}{T_j}\right)\geq\sum_{j=1}^{P}B_js_j+max_{q\in P}s_q \tag{5.12}$$

Clearly, for fixed τ_p the condition is satisfied for all $t_p \geq 0$ if it is satisfied for $t = 0$. Thus, for $\sum_{j=1}^{P}\frac{s_j}{T_j} < 1$, the connections are schedulable if d_p is set to:

$$d_p \geq \frac{\sum_{j=1}^{P}B_js_j+max_{q\in P}s_q}{\left(1-\sum_{j=1}^{P}(1-\frac{b_p}{b_q})\frac{s_j}{T_j}\right)} \quad \text{for all } p=1,2,\ldots,P \tag{5.13}$$

Using the leaky bucket-contrained model (σ, ρ), assume that there is no restriction on the packet size s_p^{min},i.e. $s_p^{min} = 0$, and a single connection per class,

$$t+\tau_p \geq \sum_{q=1}^{P}\left[\sigma_q+\rho_q\left(t+\tau_p(1-\frac{b_p}{b_q})\right)\right]+max_{q\in P}s_q^{max} \tag{5.14}$$

$\Rightarrow$

$$t\left(1-\sum_{q=1}^{P}\rho_q\right)+\tau_p\left(1-\sum_{q=1}^{P}\rho_q(1-\frac{b_p}{b_q})\right)\geq\sum_{q=1}^{P}\sigma_q+max_{q\in P}s_q^{max} \tag{5.15}$$

For fixed τ_p, the condition is satisfied for all $t \geq 0$ if it is satisfied for $t = 0$, Thus for $\sum_{q=1}^{P}\rho_q(1-\frac{b_p}{b_q}) \leq 1$,

$$d_p \geq \frac{\sum_{q=1}^{P}\sigma_q+max_{q\in P}s_q^{max}}{1-\sum_{q=1}^{P}\rho_q(1-\frac{b_p}{b_q})} \tag{5.16}$$

where $\sum_{q=1}^{P}\rho_q \leq 1$.

5.1.3 Proof of Necessity

Let us assume that the condition in Theorem 9 does not hold, that is, there exists a priority p and a time interval $[t_p, t_p + d_p - s_p^*]$ within a priority-p busy period such that for all $0 \leq \tau_p \leq d_p - s_p^*$:

$$t_p + \tau_p < \sum_{q=1}^{P} \sum_{j \in C_q} A_j^* \left(t_p + (1 - \frac{b_p}{b_q}) \tau_p \right) + max_{q \in P} s * max_q - s_p^* \tag{5.17}$$

Now assume a scenario where the WTP-scheduler is empty before time 0^-, and at time 0^- traffic from connection $i \in C_r$ with $s_i = max_{r \in P} s_r$ arrives. Suppose that, starting at time 0, all connections j with priorities p or higher transmit the maximum traffic permitted by their rate-controlling functions A_j^* with one exception: the last packet arrival before t_p from a connection k with $k \in C_p$ and $s_k^* = s_p^*$ is delayed until time t_p. If the delayed packet from connection $k \in C_p$ with arrival time t_p has not started transmisson at time $t_p + \tau_p$, then the traffic that arrives in time interval $[0^-, t_p + \tau_p]$ and is transmitted before the delayed packet consists at least of:

- $s_i = max_{q \in P}$, the transmission time of traffic that arrived at time 0^-
- $A_j^*(t_p) - s_p^*$ with $j \in C_p$, the traffic from priority p that arrived in time interval $[0, t]$ excluding the packet with arrival time t
- $A_j^*(t_p + \tau_p)$ with $j \in C_q$ and $q < p$, the high priority traffic which arrives in time interval $[0, t_p + \tau_p)$

Here we do not assume that $t \leq B_1^p - d_p$, i.e., in time interval $[0, t]$, the WTP scheduler could be empty or transmit traffic from classes that have higher priorities. However, in the best case, the WTP scheduler is always transmitting traffic in the time interval

$[0^-, t+\tau]$. Hence, we obtain the following lower bound for $W^{p,t}(t+\tau)$, the workload that is transmitted before the delayed packet:

$$W^{p,t}(t+\tau) > \sum_{q=1}^{P} \sum_{j \in C_q} A_j^* \left(t_p + (1 - \frac{b_p}{b_q}) \tau_p \right) + max_{q \in P} s * max_q - (t+\tau) \tag{5.18}$$

With our assumption in equation 5.17, we obtain that $W^{p,t} > 0$ in the entire time interval $[t, t + d_p - s_p^*]$. Thus if the packet from connection k arrives at time t has a transmission time of s_p^*, a deadline violation occurs for this packet at $[t, t + d_p - s_p^*]$.

5.2 Comparison Of Delay Bound Between WTP and SP

In [22], the delay bound for SP using the continuous traffic model is:

$$d_p \geq \frac{\sum_{q=1}^{p} \sigma_q + max_{q>p} s_q}{1 - \sum_{j=1}^{p} \rho_j} \tag{5.19}$$

If WTP wishes to tolerate deadline miss much better than SP for all classes, we can define a constraint for a set of scheduler parameters as follows:

Theorem 10 *Any class p for $p = 1 \ldots, P$ in WTP can tolerate deadline miss better than SP if*

$$\begin{aligned}\sum_{q=1}^{P} \frac{b_p}{b_q} \rho_q > \left(1/(\sum_{q=1}^{p} \sigma_q + max_{q \in P} s_q^{max}) \right) \Bigg(\sum_{q=p+1}^{P} (\sigma_q + max_{q \in P} s_q^{max} \rho_q) + \\ \sum_{q=1}^{p} \sigma_q \sum_{j=P}^{p+1} \rho_j - \sum_{q=1}^{p} \rho_q \sum_{j=P}^{p+1} \sigma_j \Bigg)\end{aligned} \tag{5.20}$$

Proof: Assuming that the packet size distributions in each class are equal, then $max_{q \in P} s_q^{max}$ is equal to $max_{q>p} s_q^{max}$. Using the constraint $d_p^{WTP} < d_p^{SP}$ and after some algebraic manipulations, we get equation (5.20). Now, we have a sufficient condition that is feasible for the range of b_p/b_q defined in Theorem 10.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we showed that PP is a special scheme of lottery scheduling in the strict priority sense and this enabled us to generalize the basic PP algorithm into a Multi-winner PP algorithm which ensures that high priority classes get served within deterministic time quanta. MPP improved the throughput accuracy and response-time variability hence improving the convergence rate to steady state. We proposed a ticket transfer algorithm to overcome the problem of the highest priority class from missing its deadline at small time scales. Simulations showed that MPP with ticket transfer surpassed normal PP and SP using models of bursty traffic class aggregation. Our algorithm provided even lower deadline violation probability and mean delay to most classes than normal PP. We used the segregation group property and MPP scheduling with ticket transfer to build a framework for relative service differentiation in Assured Forwarding and derive appropriate priority assignment rules for relative delay differentiation. We then presented the performance of this framework on high-speed programmable routers.

Next, we showed the fundamental mathematical relationship between WTP and PDD. We also derived a relationship between the maximum waiting-time target ratio and total system utilization in WTP which is independent of the number of classes and load distribution. Based on the necessary condition for positive solutions of scheduler parameters, we derived a sufficient condition for the general case of N classes that enabled the average delays in WTP to conform to the PDD model. The sufficient condition thus obtained provided bounds for the Regnier's inequalities and therefore implies that PDD delay dynamics can be readily used to search for positive solutions of scheduler parameters in WTP. Furthermore, we obtained a relationship where consecutive scheduler parameter ratios for feasible load distributions are always larger than consecutive target ratios. Based on this, we proposed a measurement-based DAC algorithm using WTP for multi-class delay differentiation. The main advantage of this extension in comparison to earlier work is that we are able to determine infeasible load distributions quickly and efficiently. The performance of the DAC algorithm was analyzed and we verified by simulations that consecutive scheduler parameter ratios of most feasible load distributions will converge rather quickly to be larger than consecutive target ratios using Gauss-Seidel algorithm. The rate of convergence can be useful as a measure for the effectiveness of various numerical algorithms in computing WTP scheduler parameters. A summary overview of the mathematical relationship between WTP and PDD is illustrated in the appendix.

Last but not least, we obtained the worst case delay bound for WTP for the general case of P classes. We showed that, as in the analysis of average delay, we had a degree of freedom provided by the set of scheduler parameters to adjust the delay bound of each class. Recently, some researchers introduce a framework that combines the concept of PDD with absolute QoS requirement [5, 6] known as Quantitative Assured Forwarding (QAF). In QAF, service differentiation is enforced over the duration of a busy period [5, 6]. The QAF concept requires tight QoS mechanism to achieve stringent performance metrics, in other words, QoS mechanism must be able to provide proportional and absolute differentiation on losses, delays and throughputs to multiple classes of traffic. In this thesis, we showed that WTP can achieve both proportional delay differentiation and maximum delay bound requirement under certain schedulability conditions by appropriate selection of scheduler parameters. What we have analyzed in details are the delay differentiation mechanism and hence, this may be a first step in using WTP to provide only QAF delay differentiation in terms of proportional and absolute performance metrics.

6.2 Future Work

We conclude this thesis by outlining areas that may have potential interest for future research work.

It is possible to derive interesting results using the necessary and sufficient condition to adjust the PP scheduler parameters for relative service differentiation. Specifically, it is difficult to obtain a closed form solution to the average queueing delay for each class of the PP scheduler in terms of the PP scheduler parameters p_i, but the authors in [17] proposed two approaches to estimate the average queueing delays using a decomposition method. These two approaches relate the estimated queueing delay for each class in the PP scheduler with p_i, hence, a meaningful extension to the necessary and sufficient condition would be to obtain the lower and upper average delay bounds that a class suffers within its corresponding parameter range. The delay bounds so obtained would significantly contrast the use of probabilistic priority as opposed to strict priority. As the delay bound of each class in the SP scheduler fluctuates with load variation, it would be worth determining if the necessary and sufficient condition of the PP scheduler parameters p_i for relative delay differentiation can better that of the SP scheduler. We have not attempted in this thesis to derive how the PP scheduler parameters can be tuned such that PDD can be achieved. However, if the feasible region of p_i that satisfies PDD exists, we can conjecture that it will lie within the necessary and sufficient region for relative delay differentiation that we have sketched out in this thesis since PDD is a form of relative delay differentiation that has a stricter requirement for consistent fixed ratio spacing. Hence, a potential research area might be to define a relationship for p_i that satisfies both Theorem 1 and equation (1.2). Alternatively, we can also repeat the estimation approach as described earlier to obtain a relationship between p_i and the estimated queueing delays using the decomposition method. This approach may not satisfy equation (1.2), but is nonetheless straightforward, and if the set of p_i so obtained satisfies Theorem 1, then we can conclude that we have a set of p_i that approximates PDD reasonably.

Also, the algorithms proposed for MPP with ticket transfer are still probabilistic in nature. The algorithms proposed could be altered with other deterministic techniques for example in the way that lottery scheduling has been improved using stride scheduling and hierarchical stride scheduling [33] in order to strike a balance between the randomness of probabilistic scheduling and meeting hard deadlines.

A question that is worth pursuing is the impact of class selection from random number of users on our load DAC algorithm for WTP scheduler. The purpose of the load DAC algorithm is to facilitate the router in computing positive WTP scheduler parameters while class selection is an external factor that is highly dependent on the type of guarantees that the network can offer. Hence, it would be interesting to show how a user can be *dissatisfied* by the constraint that all or some of the nodes in the network use WTP scheduler and the DAC algorithm. There are already some research works that try to solve this problem. In [19], game theories are combined with the WTP scheduler to solve the problem of class selection. However, results are only available for the 3-class system. Repeating their experiments using our DAC algorithm and their dynamic adaptation algorithms for the general N-class system may be a possible research area to provide robust end-to-end class selection algorithms. Also, by combining the conditions for achieving PDD and maximum delay bounds obtained in this thesis, we could attempt to observe the performance of the WTP scheduler in comparison to the QAF framework in [5, 6]. For this purpose, the region of possible scheduler parameters have already been characterized in this thesis. The next possible extension to QAF is to couple a dropping mechanism which is the primary role of buffer management that works with the WTP scheduler.

Analyzing the rate of convergence of the WTP scheduler parameter ratios for other numerical algorithms and testing the DAC algorithm for other non-Poisson traffic models will have to be examined. Other optimization-based numerical algorithms that could be used to compute the WTP scheduler parameters include the steepest descent algorithm and the Netwon-Rapson method. It would then be necessary to determine what the conditions for the existence of a solution are. Since the DAC algorithm is based on load measurement, a natural question to ask is: How to choose a limited number of measurements for best estimation resolution and quality ? This question can be worth pursuing to help reduce the overhead of iterative computation in the router. Furthermore, it would be desirable to translate the number of loops in Gauss-Seidel algorithm into actual CPU cycles consumed by the DAC mechanism. These experiments can be measured accurately in high-speed programmable network processor such as Intel IXP1200 so as to determine the scheduling overhead of WTP. We note that devising good technique to reduce the cost of deploying intensive computational mechansim in the DiffServ networks remains an important open problem.

Chapter 7

List of Publications

The following papers have been accepted:

1. C.W. Tan and C.K. Tham, "Proportional Bandwidth Guaranteed Probabilistic Priority Scheduling for Assured Forwarding", *Proceedings of IEEE High Speed Networking Workshop 2003*, San Francisco, California, Mar 2003
2. C.W. Tan and C.K. Tham, "Efficient Implementation of Relative Differentiated Services using Proportional Probabilistic Priority", to appear in Proceedings of IEEE International Conference on Networks (ICON 2003), Sydney, Australia, 28 Sept - 1 Oct 2003.
3. C.W. Tan and C.K. Tham,"Achieving Relative Differentiated Services using Proportional Probabilistic Priority Scheduling on Network Processor", to appear in Proceedings of 6th IFIP/IEEE International Conference on Management of Multimedia Networks and Services (MMNS 2003), Belfast, Northern Ireland, 7-10 Sept 2003.

The following paper is in preparation:

C.W. Tan and C.K. Tham,“Achieving Relative Differentiated Services using Proportional Probabilistic Priority Scheduling on Network Processor”, Computer Communications Journal, Elsevier Science, 2004

C.W. Tan and C.K. Tham, “Load Characterization of Waiting Time Priority Scheduler for Multi-class Proportional Delay Differentiation”, Computer Networks Journal, Elsevier Science, 2004

Bibliography

[1] T. Anderson, S. Owicki, J. Saxe and C. Thacker, High-speed switch scheduling for local-area networks, *Proc. ACM Transaction on Computer Systems*, Nov 1993

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

[3] W. Bux, Wolfgang, Ton, Andreas and Ronald, Technologies and building blocks for fast packet forwarding, *IEEE Communications Magazine*, Jan 2001

[4] S. Chaudhry and A. Choudhary, Time dependent priority scheduling for guaranteed QoS systems, *Proc. IEEE ICCCN 97*, Sep 1997

[5] N. Christin and J. Liebeherr, A QoS architecture for quantitative service differentiation, *IEEE Communications*, vol. 41(6), pp. 38-45, Jun 2003

[6] N. Christin, J. Liebeherr and T.F. Abdelzaher, A Quantitative Assured Forwarding Service, *Proc. IEEE INFOCOM 2002*, Jun 2002

[7] D.D. Clark, S. Shenker and L. Zhang, Supporting real-time applications in an integrated services packet network: architecture and mechanism, *Proc. ACM SIGCOMM*, 1992

[8] C. Dovrolis and P. Ramanathan, Dynamic class selection: from relative differentiation to absolute QoS, *Proc. IEEE ICNP 2001*, Nov 2001

[9] C. Dovrolis, D. Stiliadis and P. Ramanathan, Proportional differentiated services : Delay differentiation and packet scheduling, *Proc. ACM SIGCOMM*, Sep 1999

[10] C. Dovrolis, Proportional differentiated services for the Internet, *Ph.D. Thesis University of Wisconsin-Madison*, 2000

[11] J. Eggleston and S. Jamin, Differentiated services with lottery scheduling, *Proc. Int'l Workshop on Quality of Service (IWQoS'01)*, Jun 2001

[12] L. Essafi, G. Bolch and A. Anders, An adaptive waiting time priority scheduler for proportional differentiation model, *Tech. Report TR-I4-00-06 University of Erlangen-Nuernburg*, Sep 2000

[13] L. Essafi, G. Bolch and H.D. Meer, Dynamic priority scheduling for proportional delay differentiated services, *Tech. Report TR-I4-01-03 University of Erlangen-Nuernburg* , Mar 2001

[14] L. Huston and P. Honeyman, Partially connected operation, *Proc. Second USENIX Symposium on Mobile and Location-independent Computing*, Apr 1995

[15] Intel IXP 1200 Network Processor: Microcode Programmer's Reference Manual Revision 11, Part No. 278304-011 March 2002

[16] Y. Jiang, C.K. Tham and C.C. Ko, A probabilistic priority scheduling discipline for multi-service networks, *Computer Communications 25 (13) 2002 pp. 1243-1254*, 2002

[17] Y. Jiang, C.K. Tham and C.C. Ko, Delay analysis of a probabilistic priority scheduling discipline, *European Trans. Telecommun.*, 2002

[18] L. Kleinrock, Queueing Systems: Vol.2, Computer applications, *John Wiley & Sons*, 1976

[19] S.C.M. Lee, J.C.S. Lui and D.K.Y. Yau, Admission control and dynamic adaptation for a proportional-delay DiffServ-enabled web server, *Proc. ACM SIGMETRICS*, Jun 2002

[20] M.K.H. Leung, J.C.S. Lui, D.K.Y. Yau, Characterization and performance evaluation for proportional delay differentiated services, *Proc. IEEE ICNP 2000*, Nov 2000

[21] M.K.H. Leung, J.C.S. Lui, D.K.Y. Yau, Adaptive Proportional Delay Differentiated Services: Characterization and Performance Evaluation, *Proc. IEEE/ACM Trans. Networking*, Vol. 9, No. 6, Dec 2001

[22] J. Liebeherr, D. Wrege and D. Ferrari, Exact Admission Control for Networks with a Bounded Delay Service, *Proc. IEEE/ACM Trans. Networking*, Vol. 4, No. 6, Dec 1996

[23] T. Nandagopal, N. Venkitaraman, R. Sivakumar and V. Barghavan, Delay differentiation and adaptation in core stateless networks, *Proc. IEEE INFOCOM 2000*, 2000

[24] K. Nicholas, S. Blake, F. Baker and D.L. Black, Definition of the differentiated services field (DS Field) in the IPv4 and IPv6 Headers, IETF RFC 2474, Dec 1998

[25] H.T. Ngin and C.K. Tham, Achieving Proportional Delay Differentiation Efficiently, *Proc. IEEE ICON*, Sep 2002

[26] R. Pan, B. Prabhaker and K. Psounis, CHoKe, a stateless active queue management scheme for approximating fair bandwidth allocation, *Proc. IEEE INFOCOM 2002*, Mar 2002

[27] J. Regnier, Priority assignment in integrated services networks, *Tech. rep. LIDS-TH-1565, Massachusetts Institute of Technology*, Dec 1986

[28] J. Sethuraman, M. Squillante, Optimal stochastic scheduling in multi-class parallel queues, *Proc. IEEE ACM SIGMETRICS* , 1999

[29] I. Stoica, Stateless Core: A scalable approach for Quality of Service in the Internet, *Ph.D. Dissertation, CMU-CS-00-176*, 2000

[30] C.-K. Tham, Q. Yao and Y. Jiang, Achieving differentiated services through multi-class probabilistic priority scheduling, *Computer Networks*,40, pp. 577-593, 2002

[31] L. Thiele, S. Chakraborty, M. Gries and S. Kuenzli, Design space exploration of network processor architectures, *Proc. First Workshop on Network Processors at the 8th International Symposium on High-Performance Computer Architecture (HPCA8)*, Feb 2002

[32] C. Waldspurger and W. Weihl, Lottery scheduling: Flexible proportional-share resource management, *Proc. the First USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Nov 1994

[33] C. Waldspurger, Lottery and stride scheduling: Flexible proportional-share

resource management, *Ph.D. Thesis, Massachusetts Institute of Technology*, Sep 1995

[34] M. Zhang, R. Wang, L. Peterson and A. Krishnamurthy, Probabilistic packet scheduling: Achieving proportional share bandwidth allocation for TCP flows, *Proc. IEEE INFOCOM 2002*, Jun 2002

Appendix A

Appendix

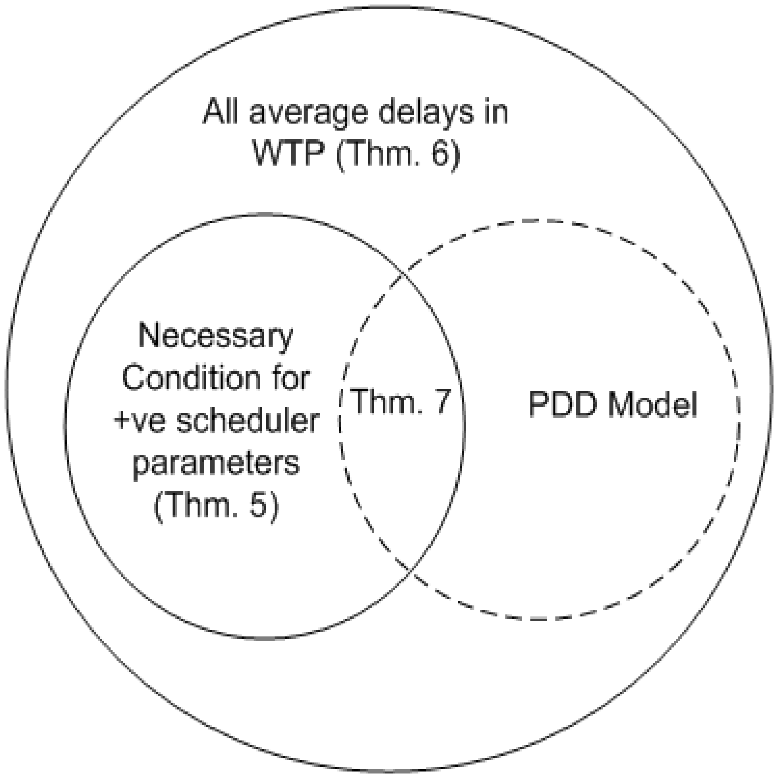

All average delays in WTP (Thm. 6)
Necessary Condition for +ve scheduler parameters (Thm. 5)
Thm. 7
PDD Model

Figure A.1: An interpretation of WTP and PDD using set diagrams. A point in a set denotes a vector of N average delays for N-class system