Modeling and Analysis of Buffer Management Schemes in a DiffServ Router

Guidance

Professor Masao Fukushima
Associate Professor Tetsuya Takine

Yujin Imagawa

2000 Graduate Course

in

Department of Applied Mathematics and Physics

Graduate School of Informatics

Kyoto University

February 2002

Abstract

The Internet was originally developed to provide best effort service, which supports no Quality of Service (QoS) guarantees. To provide QoS guarantees for applications which need real-time control, Integrated Service (IntServ) was proposed. This service is implemented by use of architectures such as RSVP which reserves per-flow based bandwidth. However, IntServ has a scalability problem. To solve this problem, Differentiated Service (DiffServ) was proposed. The basic idea of DiffServ is to focus on aggregate of packets by tagging them and to differentiate between service classes. DiffServ can provide simple and scalable service. Current proposals for DiffServ architectures, however, do not quantify the service they would provide for applications. Thus quantifying QoSs offered by DiffServ is the goal of this thesis.

In this thesis, we propose a new service scheme for a DiffServ router, which aims for differentiation of two QoSs, i.e., throughput and mean delay, at the same time. In this service scheme, arriving packets are divided into four service classes and differentiated by one of two buffer management schemes, i.e., separated buffer management scheme and shared buffer management scheme. To reflect properties of traffic, we model these two buffer management schemes with input streams changing their transfer rates reactively to network congestion. We describe the system dynamics by a Markov chain, and then propose an efficient way to compute the steady state probabilities. In the overloaded situation, we examine qualitative performance of the two QoSs through numerical experiments. As a result, we show that the proposed DiffServ router can work well regardless of experimental scenarios.

Contents

1 Introduction

In recent years, the Internet was developed and spread over the world. The Internet is simply designed for packet delivery. It is based on a single service class, i.e., best effort service. In this service, all packets are treated in the same way without any discrimination or explicit delivery guarantees. This service is implemented simply and inexpensively because of simple packet forwarding. With improvement of network technology, in addition to traditional applications such as HTML, e-mail and FTP, new applications which need real-time control, e.g., audio and movie, have appeared. In these situations, there is new service requirement that offers Quality of Service (QoS) guarantees for provisions of diverse service classes. To provide QoS guarantees, IETF investigation mainly focused on two different approaches: Integrated Service (IntServ) and Differentiated Service (DiffServ).

IntServ [2] was proposed to integrate non-real-time service and real-time service over the Internet. In this service, each flow reserves bandwidth by use of Resource ReSerVation Protocol (RSVP) [10]. However, IntServ has a scalability problem. Since each flow makes reservation at intermediate routers, it is difficult to handle a number of flows in large-scale networks. Also, the cost of large-scale routers becomes high. Thus, it is necessary to propose a simple architecture that is applicable to widespread high-speed networks. To solve the problems in IntServ, the IETF proposed a new architecture, called DiffServ [1].

The basic idea of DiffServ is to focus on aggregate of packets by tagging them and to differentiate between service classes rather than provide absolute per-flow QoS guarantees [9]. DiffServ can provide simple and scalable service. There are two service schemes, Assured Service and Premium Service, proposed mainly in DiffServ [7]. The Assured Service scheme [3] is argued that a guaranteed throughput is the one quality of service feature of interest to most applications, and thus that there is a need for a mechanism that directly reflects users' desires in terms of transfer time. One way to achieve this objective is to define a service profile at the source, which defines how packets should be sent so as to meet the rate objective, and to incorporate a profile meter. The profile meter monitors the transfer in progress and tags each packet of the date stream. The tag is set to 1 if the packet is sent according to the profile; the tagged packet is also referred to as an IN packet. The tag is set to 0 otherwise; the non-tagged packet is also referred to as an OUT packet. The tag information is used by intermediate routers in case of congestion. OUT packets are preferentially selected to receive a congestion push-back notification, which typically means that they are dropped. When buffering/scheduling mechanism is implemented, it is intuitively clear that IN packets will get higher throughput than OUT packets. On the other hand, the Premium Service scheme [8] is expected to provide significant delay reduction for the subscribers. To provide the Premium Service, routers implement two levels of priority queueing. Tagged (Premium) packets are sent first; non-tagged (best-effort) packets are queued and sent only when all tagged packets have been sent. There is consequently a very limited queue management at the network node level. Then, the Premium class behaves as a flow aggregation, and does not require the control of each flow within the network.

In any case, DiffServ architectures based on tagging packets are attractive because they appear to be simple to implement. However, current proposals for DiffServ architectures do not quantify the service they would provide for applications. Thus quantifying the QoSs offered

by DiffServ is the goal of this thesis.

In this thesis, we propose a new service scheme for a DiffServ router, which aims for differentiation of two QoSs, i.e., throughput and mean delay, at the same time. In this service scheme, arriving packets are divided into four service classes and differentiated by one of two buffer management schemes, i.e., separated buffer management scheme and shared buffer management scheme. To reflect properties of traffic, we model these two buffer management schemes with input streams changing their transfer rates reactively to network congestion. We describe the system dynamics by a Markov chain, and then propose an efficient way to compute the steady state probabilities. In the overloaded situation, we examine qualitative performance of the two QoSs through numerical experiments.

The rest of this thesis is organized as follows. In section 2, we describe a new service scheme for a DiffServ router, and two buffer management schemes. In section 3, we consider a mathematical model of the proposed new service scheme. In section 4, we analyze the model and derive expressions for performance measures. In section 5, we show numerical results and quantify the QoSs to examine performance of the proposed DiffServ router. Finally, section 6 concludes the thesis.

2 Proposals for a DiffServ Router

In this section, we propose a new service scheme for a DiffServ router. Throughput and mean delay are qualities of service feature of interest to most applications. In the new service scheme, a DiffServ router aims for differentiation of these two QoSs at the same time. The new service scheme needs a simple architecture because DiffServ should provide simple and scalable service. In what follows, we show the architecture for differentiation of QoSs in the new service scheme.

2.1 Packet classification

To implement DiffServ, a service provider makes a contract with each user. The contract specifies the mean transfer rate and the maximum burst length that the user should keep. According to the contract, the router tags arriving packets. DiffServ focuses on aggregate of packets by tagging them and differentiates between service classes. In the new service scheme, packets are divided into two classes relevant to delay priority. We denote two delay priority classes by $DP1$ and $DP2$, respectively. Packets of $DP1$ are served by priority, while packets of $DP2$ are served only when there are no packets of $DP1$ in the system. This priority rule is expected to provide significant delay reduction for $DP1$ packets. In each delay priority class, if packets are sent according to the service specification, they are set to IN, and otherwise, they are set to OUT. We denote IN packets of DPi $(i = 1, 2)$ by i-IN packets and OUT packets of DPi $(i = 1, 2)$ by i-OUT packets. The router differentiates between IN and OUT packets by drop mechanisms. In any drop mechanism, OUT packets are preferentially selected to drop by routers in case of congestion. Using such selective drop mechanisms, throughput of IN packets will be higher than that of OUT packets in each DP class.

2.2 Buffer management schemes

We argue a way to differentiate the four service classes in this subsection. As mentioned in section 2.1, we use a drop mechanism for throughput discrimination between IN and OUT packets. There are two popular ways to implement a selective drop scheme, a RED with IN and OUT packets (RIO) mechanism and a threshold drop mechanism. RIO [4] extends Random Early Detection (RED) to handle two classes of packets. RED [5] is designed for congestion avoidance in packet switched networks. RED detects incipient congestion by computing the average queue size. When the queue size exceeds a given threshold parameter, RED drops arriving packets with a certain probability depending on the average queue size. RIO has two sets of drop parameters, one is for IN packets and the other for OUT packets. Throughput differentiation between IN and OUT packets can be achieved by using two thresholds, where the threshold for OUT packets is set lower than that for IN packets, or by using drop probabilities that increase at different rates as the average queue length increases. On the other hand, in the threshold drop mechanism, when buffer occupancy reaches a preset threshold, incoming OUT packets are discarded, whereas IN packets are still admitted in the queue as long as the buffer is not full. We denote this mechanism as THRESH. If no specific buffer management is implemented, a router uses TAIL drop [6], which accepts both of IN and OUT packets unless the buffer is full. If a router uses either RIO or THRESH mechanism, it is intuitively clear that IN packets will get higher throughput than OUT packets. In numerical results in section 5, we make sure of this intuition.

In the new service scheme, we use the service priority discipline to differentiate mean delay between $DP1$ and $DP2$. A router serves $DP1$ packets preferentially, and $DP2$ packets get service only when there are no $DP1$ packets in the system. We handle two buffer management schemes for delay discrimination. We call one *separated* buffer management scheme, and the other *shared* buffer management scheme.

In the separated buffer management scheme (showed in Figure 1), we have two finite queues, each having capacity K_i $(i = 1, 2)$. The ith queue is accessible only by DP class i packets, e.g., the 1st queue is accessible only by 1-IN and 1-OUT packets. In each queue, packets are served in order of arrival. When the ith queue is full, arriving packets of DP class i are discarded, even if the other queue is not full. Packets of $DP1$ are served successively until the 1st queue becomes empty. When that queue becomes empty, waiting packets of $DP2$ get service.

In the shared buffer management scheme (showed in Figure 2), the router has only one buffer of size K, which is accessible by both of DP classes. However, in this buffer management scheme, packets of $DP1$ are lined up ahead of packets of $DP2$. In the same DP class, packets are served on a first-come first-served basis. If the queue is full, arriving packets are rejected regardless of their DP class. In a similar fashion of the separated buffer management scheme, packets of $DP1$ are served preferentially. Packets of $DP2$ are served only when there is no packet of $DP1$ in the queue.

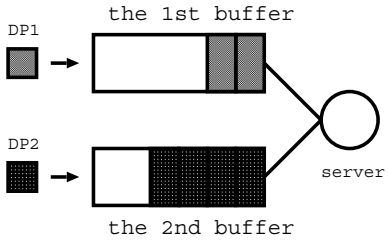

Figure 1: Separated buffer management scheme.

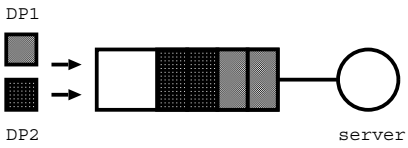

Figure 2: Shared buffer management scheme.

3 Mathematical Modeling

3.1 Phase

In the Internet, there are various sorts of traffic. Traffic is roughly classified into two sorts. Most of traffic is TCP-like traffic which controls the transfer rate of packets by reacting to network congestion. We denote this traffic by t-traffic. In recent years, with appearance of real-time applications, UDP-like traffic has increased, which sends packets independently of network congestion. We call this traffic u-traffic. To reflect the effects of traffic, we introduce phase structure.

The source state is expressed by a pair of two phase numbers (m_1, m_2). m_1 denotes a $DP1$ phase number and represents the source state regarding transmission of $DP1$ traffic, and we refer to m_2 as a $DP2$ phase number and show the state about $DP2$ traffic transfer. $DP1$ phases are numbered from 1 to M_1, and $DP2$ phases are numbered from 1 to M_2. In each phase, we define the arrival rates of four service classes and the transition rates between phases.

First, we argue about the arrival rates of four service classes in each phase. When the state of a source is in phase (m_1, m_2), sending rate of packets belonging to $DP1$ is given by a function of m_1, while that of $DP2$ packets is given by a function of m_2. Sending rates of each service class are showed in Table 1.

Table 1: Notation of sending rates when the source state is in (m_1, m_2).

service class	1-IN	1-OUT	2-IN	2-OUT
notation	$\lambda_{IN}^{(1)}(m_1)$	$\lambda_{OUT}^{(1)}(m_1)$	$\lambda_{IN}^{(2)}(m_2)$	$\lambda_{OUT}^{(2)}(m_2)$

We make following assumption on the arrival rate.

Assumption 1 *In each DP class i $(i = 1, 2)$, if $m < n$,*

$$\begin{aligned} \lambda_{IN}^{(i)}(m) &\leq \lambda_{IN}^{(i)}(n), \\ \lambda_{OUT}^{(i)}(m) &\leq \lambda_{OUT}^{(i)}(n). \end{aligned}$$

We discuss the phase transition rates. These transition rates are decided by what traffic sources send. We consider two sorts of traffic, i.e., t-traffic and u-traffic. For simplicity, we assume that phase transitions are divided into $DP1$ phase transitions and $DP2$ phase transitions, and these phase transitions are skip-free, that is, when the source state is in (m_1, m_2), allowable transition is to $\{(m_1+1, m_2), (m_1-1, m_2), (m_1, m_2+1), (m_1, m_2-1)\}$. We refer to transition from (m_1, m_2) to (m_1+1, m_2) as a $DP1$ upward transition and transition from (m_1, m_2) to (m_1-1, m_2) as a $DP1$ downward transition. In a similar way, we call transition from (m_1, m_2) to (m_1, m_2+1) a $DP2$ upward transition and transition from (m_1, m_2) to (m_1, m_2-1) a $DP2$ downward transition. The relation between the sorts of traffic and the transition rates are showed as follows.

- t-traffic (Figure 3)
  While there is a buffer space to enter packets of DPi $(i = 1, 2)$, the state of the source which sends t-traffic transits upward at rate $\tau_i(m_1, m_2)$ except that $m_i = M_i$. On the other hand, when buffer occupancy is full, the state of the source makes DPi $(i = 1, 2)$ downward transition at rate $\tilde{\tau}_i(m_1, m_2)$ if $m_i \neq 1$. By setting DPi $(i = 1, 2)$ downward transition rate $\tilde{\tau}_i(m_1, m_2)$ as DPi arrival rate at (m_1, m_2), $\lambda_{IN}^{(i)}(m_i) + \lambda_{OUT}^{(i)}(m_i)$, we can model that loss of DPi packets triggers downward transition of the source state.

- u-traffic (Figure 4)
  Regardless of buffer occupancy, the state of the source which sends u-traffic can transit to both upward and downward directions if $1 < m_i < M_i$. If $m_i = 1$, the source state can transit only upward, whereas if $m_i = M_i$, the source state can transit only downward. The rate of upward transition is denoted by $\gamma_i(m_1, m_2)$ and that of downward transition by $\tilde{\gamma}_i(m_1, m_2)$.

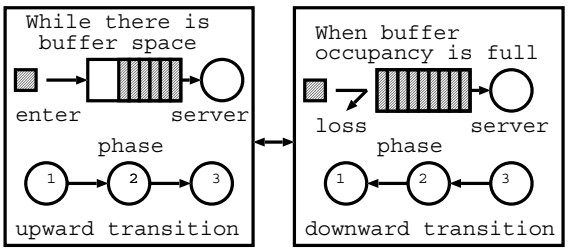

Figure 3: Transition of t-traffic.

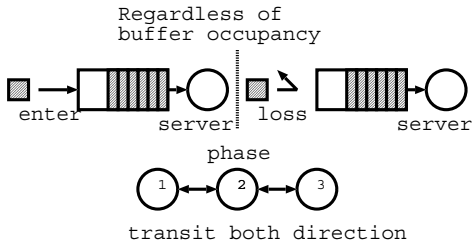

Figure 4: Transition of u-traffic.

Assumption 2 *When the source state is in (m_1, m_2), an arriving packet belongs to DPi t-traffic $(i = 1, 2)$ with probability $p_i(m_1, m_2)$ and DPi u-traffic with probability $1 - p_i(m_1, m_2)$.*

The transition rate of a source is expressed by the sum of transition rates multiplied by the corresponding probability. Let $g_{n_1,n_2}((m_1, m_2), (m_1', m_2'))$ denote the phase transition rate from (m_1, m_2) to (m_1', m_2'), given that n_1 packets of $DP1$ and n_2 packets of $DP2$ are in the system.

While there is a buffer space to enter DPi $(i = 1, 2)$ packets, that is, $n_i < K_i$ in the separated buffer management scheme or $n_1 + n_2 < K$ in the shared buffer management scheme,

$$g_{n_1,n_2}((m_1, m_2), (m_1', m_2')) =$$
$$\begin{cases} p_i(m_1, m_2)\tau_i(m_1, m_2) + (1 - p_i(m_1, m_2))\gamma_i(m_1, m_2), & m_i' = m_i + 1, 1 \le m_i < M_i, \\ (1 - p_i(m_1, m_2))\tilde{\gamma}_i(m_1, m_2), & m_i' = m_i - 1, 1 < m_i \le M_i, \\ 0, & \text{otherwise.} \end{cases}$$

When buffer occupancy is full, i.e., $n_i = K_i$ $(i = 1, 2)$ in the separated buffer management scheme or $n_1 + n_2 = K$ in the shared buffer management scheme,

$$g_{n_1,n_2}((m_1, m_2), (m_1', m_2')) =$$
$$\begin{cases} (1 - p_i(m_1, m_2))\gamma_i(m_1, m_2), & m_i' = m_i + 1, 1 \le m_i < M_i, \\ p_i(m_1, m_2)\tilde{\tau}_i(m_1, m_2) + (1 - p_i(m_1, m_2))\tilde{\gamma}_i(m_1, m_2), & m_i' = m_i - 1, 1 < m_i \le M_i, \\ 0, & \text{otherwise.} \end{cases}$$

We define the whole phase number m as $m = (m_1 - 1)M_2 + m_2$, when DP phase is given by (m_1, m_2).

3.2 Drop mechanisms

To model the drop mechanisms in section 2, we use functions of n_1 and n_2, where n_i $(i = 1, 2)$ denotes the number of DPi packets in the system. Specifically, let $\alpha_{IN}^{(i)}(n_1, n_2)$ $(i = 1, 2)$ denote the probability that an arriving IN packet of DPi is accepted, given that n_1 packets of $DP1$ and n_2 packets of $DP2$ are already present in the queue. Likewise, let $\alpha_{OUT}^{(i)}(n_1, n_2)$ be the acceptance probability for OUT packets of DPi when n_1 packets of $DP1$ and n_2 packets of $DP2$ are found in the queue. In the separate buffer management scheme, $\alpha_{IN}^{(1)}(K_1, n_2) = \alpha_{OUT}^{(1)}(K_1, n_2) = 0$ and $\alpha_{IN}^{(2)}(n_1, K_2) = \alpha_{OUT}^{(2)}(n_1, K_2) = 0$, where K_i $(i = 1, 2)$ denotes the buffer capacity of the ith queue. In the shared buffer management scheme, we denote the buffer size by K, then if $n_1 + n_2 = K$, $\alpha(n_1, n_2) = 0$. Modeling drop mechanisms can be done simply by choosing appropriate values for $\alpha(n_1, n_2)$.

We have examined three mechanisms, which we refer to as TAIL (tail drop), RIO (RED with IN and OUT packets) and THRESH (threshold drop). These are defined as follows.

- ***TAIL***: This mechanism has no specific buffer management. In the separated buffer management scheme, for $i = 1, 2$,

  $$\alpha_{IN}^{(i)}(n_1, n_2) = \alpha_{OUT}^{(i)}(n_1, n_2) = 1, \quad 0 \le n_1 < K_1, 0 \le n_2 < K_2.$$

  Similarly, in the shared buffer management scheme, for $i = 1, 2$,

  $$\alpha_{IN}^{(i)}(n_1, n_2) = \alpha_{OUT}^{(i)}(n_1, n_2) = 1, \quad 0 \le n_1 + n_2 < K.$$

- ***RIO***: In the separated buffer management scheme, IN (OUT) packets of DPi $(i = 1, 2)$ are accepted until the buffer occupancy of the ith queue becomes $TH_{IN}^{(i)}$ $(TH_{OUT}^{(i)})$. Then

i-IN packets are dropped with probability that increases linearly to $r_{IN}^{(i)}$, and *i-OUT* packets are dropped with $r_{OUT}^{(i)}$. Specifically,

$$\begin{cases} \alpha_{IN}^{(1)}(n_1,n_2)=1, & n_1 \le TH_{IN}^{(1)}, \\ \alpha_{OUT}^{(1)}(n_1,n_2)=1, & n_1 \le TH_{OUT}^{(1)}, \\ \alpha_{IN}^{(2)}(n_1,n_2)=1, & n_2 \le TH_{IN}^{(2)}, \\ \alpha_{OUT}^{(2)}(n_1,n_2)=1, & n_2 \le TH_{OUT}^{(2)}. \end{cases}$$

$$\begin{cases} \alpha_{IN}^{(1)}(n_1,n_2)=1-\dfrac{r_{IN}^{(1)}}{K_1-TH_{IN}^{(1)}}(n_1-TH_{IN}^{(1)}), & TH_{IN}^{(1)} \le n_1 < K_1, \\ \alpha_{OUT}^{(1)}(n_1,n_2)=1-\dfrac{r_{OUT}^{(1)}}{K_1-TH_{OUT}^{(1)}}(n_1-TH_{OUT}^{(1)}), & TH_{OUT}^{(1)} \le n_1 < K_1, \\ \alpha_{IN}^{(2)}(n_1,n_2)=1-\dfrac{r_{IN}^{(2)}}{K_2-TH_{IN}^{(2)}}(n_2-TH_{IN}^{(2)}), & TH_{IN}^{(2)} \le n_2 < K_2, \\ \alpha_{OUT}^{(2)}(n_1,n_2)=1-\dfrac{r_{OUT}^{(2)}}{K_2-TH_{OUT}^{(2)}}(n_2-TH_{OUT}^{(2)}), & TH_{OUT}^{(2)} \le n_2 < K_2. \end{cases}$$

Similarly, in the shared buffer management scheme, all packets of *i-IN* (*i-OUT*) are accepted for $i=1,2$ while the queue length is smaller than $TH_{IN}^{(i)}$ ($TH_{OUT}^{(i)}$). Then *i-IN* (*i-OUT*) packets are discarded with the rate that rises linearly to $r_{IN}^{(i)}$ ($r_{OUT}^{(i)}$).

$$\begin{cases} \alpha_{IN}^{(1)}(n_1,n_2)=1, & n_1+n_2 \le TH_{IN}^{(1)}, \\ \alpha_{OUT}^{(1)}(n_1,n_2)=1, & n_1+n_2 \le TH_{OUT}^{(1)}, \\ \alpha_{IN}^{(2)}(n_1,n_2)=1, & n_1+n_2 \le TH_{IN}^{(2)}, \\ \alpha_{OUT}^{(2)}(n_1,n_2)=1, & n_1+n_2 \le TH_{OUT}^{(2)}. \end{cases}$$

$$\begin{cases} \alpha_{IN}^{(1)}(n_1,n_2)=1-\dfrac{r_{IN}^{(1)}}{K-TH_{IN}^{(1)}}(n_1+n_2-TH_{IN}^{(1)}), & TH_{IN}^{(1)} \le n_1+n_2 < K, \\ \alpha_{OUT}^{(1)}(n_1,n_2)=1-\dfrac{r_{OUT}^{(1)}}{K-TH_{OUT}^{(1)}}(n_1+n_2-TH_{OUT}^{(1)}), & TH_{OUT}^{(1)} \le n_1+n_2 < K, \\ \alpha_{IN}^{(2)}(n_1,n_2)=1-\dfrac{r_{IN}^{(2)}}{K-TH_{IN}^{(2)}}(n_1+n_2-TH_{IN}^{(2)}), & TH_{IN}^{(2)} \le n_1+n_2 < K, \\ \alpha_{OUT}^{(2)}(n_1,n_2)=1-\dfrac{r_{OUT}^{(2)}}{K-TH_{OUT}^{(2)}}(n_1+n_2-TH_{OUT}^{(2)}), & TH_{OUT}^{(2)} \le n_1+n_2 < K. \end{cases}$$

- ***THRESH***: All *DPi* ($i=1,2$) packets are accepted until the queue size reaches $TH^{(i)}$, then all *OUT* packets of *DPi* are dropped, but all *IN* packets of both *DP* classes are accepted. Formally, in the separated buffer management scheme,

$$\begin{aligned} \alpha_{IN}^{(1)}(n_1,n_2) &= \alpha_{IN}^{(2)}(n_1,n_2)=1, \quad 0 \le n_1 < K_1, 0 \le n_2 < K_2. \\ \alpha_{OUT}^{(1)}(n_1,n_2) &= \begin{cases} 1, & n_1 \le TH^{(1)}, \\ 0, & n_1 > TH^{(1)}. \end{cases} \\ \alpha_{OUT}^{(2)}(n_1,n_2) &= \begin{cases} 1, & n_2 \le TH^{(2)}, \\ 0, & n_2 > TH^{(2)}. \end{cases} \end{aligned}$$

In the shared buffer management scheme,

$$\begin{aligned}
\alpha_{IN}^{(1)}(n_1,n_2) &= \alpha_{IN}^{(2)}(n_1,n_2) = 1, \quad 0 \le n_1+n_2 < K. \\
\alpha_{OUT}^{(1)}(n_1,n_2) &= \begin{cases} 1, & n_1+n_2 \le TH^{(1)}, \\ 0, & n_1+n_2 > TH^{(1)}. \end{cases} \\
\alpha_{OUT}^{(2)}(n_1,n_2) &= \begin{cases} 1, & n_1+n_2 \le TH^{(2)}, \\ 0, & n_1+n_2 > TH^{(2)}. \end{cases}
\end{aligned}$$

3.3 Effective arrival rate

As described in section 3.1, a source sends packets of each service class according to the source state which is showed by phase structure. When the source state is in phase (m_1, m_2), the arrival rate of 1-IN is $\lambda_{IN}^{(1)}(m_1)$, that of 1-OUT is $\lambda_{OUT}^{(1)}(m_1)$, that of 2-IN is $\lambda_{IN}^{(2)}(m_2)$ and that of 2-OUT is $\lambda_{OUT}^{(2)}(m_2)$. Incoming packets according to these arrival rates are dropped by the drop mechanisms in section 3.2. Thus, when n_1 $DP1$ packets and n_2 $DP2$ packets are found in the system and the state of the source is in (m_1, m_2), we denote the effective arrival rate of each DP class by $\Lambda_i(n_1, n_2, (m_1, m_2))$ $(i = 1, 2)$, and then

$$\Lambda_1(n_1,n_2,(m_1,m_2)) = \lambda_{IN}^{(1)}(m_1)\alpha_{IN}^{(1)}(n_1,n_2) + \lambda_{OUT}^{(1)}(m_1)\alpha_{OUT}^{(1)}(n_1,n_2),$$

and

$$\Lambda_2(n_1,n_2,(m_1,m_2)) = \lambda_{IN}^{(2)}(m_2)\alpha_{IN}^{(2)}(n_1,n_2) + \lambda_{OUT}^{(2)}(m_2)\alpha_{OUT}^{(2)}(n_1,n_2).$$

We assume that packets of DP class i $(i = 1, 2)$ require service whose length is exponentially distributed with parameter μ_i. We assume for simplicity that packets are served in a preemptive way. Within a DP class, packets are served on a first-come first-served basis.

4 Analysis

We examine the mathematical model mentioned in previous section. As mentioned in section 3.3, the number of packets in the system and the phase number form a Markov chain. We solve this model in use of properties of Markov chains. We argue the analysis of the separated buffer management scheme below. In the case of the shared buffer management scheme, we can deduce the argument in a similar way, and therefore we omit this case.

4.1 Problem formulation

Let X denote the set of the system states:

$$X = \Big\{\big(n_1,n_2,(m_1,m_2)\big); 0 \le n_1 \le K_1, 0 \le n_2 \le K_2, 1 \le m_1 \le M_1, 1 \le m_2 \le M_2\Big\},$$

where n_1 denotes the number of $DP1$ packets in the system, n_2 denotes that of $DP2$ packets in the system and (m_1, m_2) denotes a pair of DP phase numbers. We denote the steady

state probability vector whose element represents the probability that the system state is in $(n_1, n_2, (m_1, m_2)) \in X$ by

$$\boldsymbol{\pi} = \Big\{\boldsymbol{\pi}\big(n_1, n_2, (m_1, m_2)\big)\Big\},$$

where elements of this vector are laid out as

$$\begin{aligned}\boldsymbol{\pi} = \Big(&\big(0,0,(1,1)\big),\big(0,0,(1,2)\big),\ldots,\big(0,0,(1,M_2)\big),\big(0,0,(2,1)\big),\ldots,\\ &\big(0,0,(2,M_2)\big),\ldots,\big(0,0,(M_1,M_2)\big),\big(1,0,(1,1)\big),\ldots,\big(2,0(1,1)\big),\ldots,\\ &\big(K_1,0,(1,1)\big),\ldots,\big(K_1,0,(M_1,M_2)\big),\big(0,1,(1,1)\big),\ldots,\big(0,2,(1,1)\big),\ldots,\\ &\big(0,K_2,(1,1)\big),\ldots,\big(K_1,K_2,(M_1,M_2)\big)\Big).\end{aligned}$$

Our purpose in this section is to solve these steady state probabilities. We denote $\boldsymbol{Q}$ the transition rate matrix by

$$\boldsymbol{Q} = \{q_{x,x'}\},$$

where

$$q_{x,x'} = \begin{cases} (\text{transition rate from } x \text{ to } x'; x, x' \in X), & x \neq x', \\ (-1) \times (\text{the sum of all other elements in the same row}), & \text{otherwise.} \end{cases}$$

Then, we have stationary equations

$$\boldsymbol{\pi}\boldsymbol{Q} = \boldsymbol{0} \quad \text{and} \quad \boldsymbol{\pi}\boldsymbol{e} = 1, \tag{1}$$

where $\boldsymbol{e}$ denotes a column vector of ones with an appropriate dimension.

In our model, allowable transitions of this system are (a) increase of $DP1$ packets, (b) decrease of $DP1$ packets, (c) increase of $DP2$ packets, (d) decrease of $DP2$ packets and (e) phase transitions. Note that the number of packets can increase or decrease by one with a transition. Transition rates in each case are showed in Table 2.

Table 2: Transition rate.

transition	transition rate	condition
(a) increase of one $DP1$ packet	$\Lambda^{(1)}(n_1, n_2, (m_1, m_2))$	$n_1 < K_1$
(b) decrease of one $DP1$ packet	μ_1	$n_1 \neq 0$
(c) increase of one $DP2$ packet	$\Lambda^{(2)}(n_1, n_2, (m_1, m_2))$	$n_2 < K_2$
(d) decrease of one $DP2$ packet	μ_2	$n_1 = 0, n_2 \neq 0$
(e) phase transition from (m_1, m_2) to (m'_1, m'_2)	$g_{n_1,n_2}((m_1, m_2), (m'_1, m'_2))$	

Because of this limitation of allowable transitions, the matrix $\boldsymbol{Q}$ is written in a block form:

$$\boldsymbol{Q} = \begin{pmatrix} \mathcal{A}_0 & \mathcal{U}_0 & & & 0 \\ \mathcal{D}_1 & \mathcal{A}_1 & \mathcal{U}_1 & & \\ & \ddots & \ddots & \ddots & \\ & & \mathcal{D}_{K_2-1} & \mathcal{A}_{K_2-1} & \mathcal{U}_{K_2-1} \\ 0 & & & \mathcal{D}_{K_2} & \mathcal{A}_{K_2} \end{pmatrix},$$

where $\boldsymbol{Q}$ denotes the matrix with $(K_1+1)(K_2+1)M_1M_2$ dimension and then the $(K_1+1)M_1M_2 \times (K_1+1)M_1M_2$ matrices $\mathcal{A}_i$ $(i=0,\ldots,K_2)$, $\mathcal{U}_i$ $(i=0,\ldots,K_2-1)$ and $\mathcal{D}_i$ $(i=1,\ldots,K_2)$ are also denoted in block forms:

$$\mathcal{A}_i = \begin{pmatrix} \boldsymbol{A}_{i,0} & \boldsymbol{U}_{i,0} & & & 0 \\ \boldsymbol{D}_{i,1} & \boldsymbol{A}_{i,1} & \boldsymbol{U}_{i,1} & & \\ & \ddots & \ddots & \ddots & \\ & & \boldsymbol{D}_{i,K_1-1} & \boldsymbol{A}_{i,K_1-1} & \boldsymbol{U}_{i,K_1-1} \\ 0 & & & \boldsymbol{D}_{i,K_1} & \boldsymbol{A}_{i,K_1} \end{pmatrix},$$

$$\mathcal{U}_i = \begin{pmatrix} \boldsymbol{U}^{(2)}_{i,0} & & & 0 \\ & \boldsymbol{U}^{(2)}_{i,1} & & \\ & & \ddots & \\ 0 & & & \boldsymbol{U}^{(2)}_{i,K_1} \end{pmatrix}, \mathcal{D}_i = \begin{pmatrix} \boldsymbol{D}^{(2)}_{i,0} & & & 0 \\ & 0 & & \\ & & \ddots & \\ 0 & & & 0 \end{pmatrix}.$$

Then, $M_1M_2 \times M_1M_2$ matrices $\boldsymbol{A}_{i,j}$, $\boldsymbol{U}_{i,j}$, $\boldsymbol{D}_{i,j}$, $\boldsymbol{U}^{(2)}_{i,j}$ and $\boldsymbol{D}^{(2)}_{i,0}$ are represented as

$$[\boldsymbol{A}_{i,j}]_{k,l} = \begin{cases} g_{j,i}(k,l), & k \neq l, \\ -\sum(*), & \text{otherwise}, \end{cases} \quad (i=0,\ldots,K_2, j=0,\ldots,K_1)$$

$$[\boldsymbol{U}_{i,j}]_{k,l} = \begin{cases} \Lambda^{(1)}(j,i,(k,l)), & k=l, \\ 0, & \text{otherwise}, \end{cases} \quad (i=0,\ldots,K_2, j=0,\ldots,K_1-1)$$

$$[\boldsymbol{D}_{i,j}]_{k,l} = \begin{cases} \mu_1, & k=l, \\ 0, & \text{otherwise}, \end{cases} \quad (i=0,\ldots,K_2, j=1,\ldots,K_1)$$

$$\left[\boldsymbol{U}^{(2)}_{i,j}\right]_{k,l} = \begin{cases} \Lambda^{(2)}(j,i,(k,l)), & k=l, \\ 0, & \text{otherwise}, \end{cases} \quad (i=0,\ldots,K_2-1, j=0,\ldots,K_1)$$

$$\left[\boldsymbol{D}^{(2)}_{i,0}\right]_{k,l} = \begin{cases} \mu_2, & k=l, \\ 0, & \text{otherwise}, \end{cases} \quad (i=1,\ldots,K_2)$$

where k and l denote the whole phase number and $\sum(*)$ denotes the sum of all other elements in the same row of $\boldsymbol{Q}$.

By solving the stationary equations (1), we obtain the steady state probabilities. However, in the case of large buffer size, it is hard to solve them directly because of a large number of states. Then, we focus on the structure of a block tridiagonal matrix $\boldsymbol{Q}$. In use of this structure, we can decrease the size of matrices which we handle.

If we partition the vector $\boldsymbol{\pi}$ into a sequence of contiguously laid out vectors $\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_{K_2}$, where $\boldsymbol{\pi}_j$ is of $(K_1+1)M_1M_2$ dimension, then the stationary equations (1) become

$$\begin{cases} \boldsymbol{\pi}_0\mathcal{A}_0 + \boldsymbol{\pi}_1\mathcal{D}_1 = \boldsymbol{0}, \\ \boldsymbol{\pi}_{j-1}\mathcal{U}_{n_2-1} + \boldsymbol{\pi}_j\mathcal{A}_{n_2} + \boldsymbol{\pi}_{j+1}\mathcal{D}_{j+1} = \boldsymbol{0}, \qquad j=1,2,\ldots,K_2-1, \\ \boldsymbol{\pi}_{K_2-1}\mathcal{U}_{K_2-1} + \boldsymbol{\pi}_{K_2}\mathcal{A}_{K_2} = \boldsymbol{0}. \end{cases}$$

From $\boldsymbol{\pi}_0\mathcal{A}_0 + \boldsymbol{\pi}_1\mathcal{D}_1 = \boldsymbol{0}$, we have

$$\boldsymbol{\pi}_0 = \boldsymbol{\pi}_1\mathcal{R}_1,$$

where $\mathcal{R}_1 = \mathcal{D}_1(-\mathcal{A}_0)^{-1}$. Substitute $\boldsymbol{\pi}_0 = \boldsymbol{\pi}_1\mathcal{R}_1$ for $\boldsymbol{\pi}_0\mathcal{U}_0 + \boldsymbol{\pi}_1\mathcal{A}_1 + \boldsymbol{\pi}_2\mathcal{D}_2 = \mathbf{0}$, we obtain

$$\boldsymbol{\pi}_1 = \boldsymbol{\pi}_2\mathcal{R}_2,$$

where $\mathcal{R}_2 = \mathcal{D}_2(-\mathcal{A}_1 - \mathcal{R}_1\mathcal{U}_0)^{-1}$. In a similar way, we then have

$$\boldsymbol{\pi}_i = \boldsymbol{\pi}_{i+1}\mathcal{R}_{i+1}, \quad i = 1, \ldots, K_2 - 1, \tag{2}$$

where $\mathcal{R}_i$ $(i = 2, 3, \ldots, K_2)$ is given by

$$\mathcal{R}_i = \mathcal{D}_i(-\mathcal{A}_{i-1} - \mathcal{R}_{i-1}\mathcal{U}_{i-1})^{-1}.$$

Thus, by induction, it follows that

$$\boldsymbol{\pi}_i = \boldsymbol{\pi}_{K_2}\mathcal{R}_{K_2}\mathcal{R}_{K_2-1}\cdots\mathcal{R}_{i+2}\mathcal{R}_{i+1}, \quad i = 0, \ldots, K_2 - 1. \tag{3}$$

Using $\boldsymbol{\pi}_{K_2-1} = \boldsymbol{\pi}_{K_2}\mathcal{R}_{K_2}$ in $\boldsymbol{\pi}_{K_2-1}\mathcal{U}_{K_2-1} + \boldsymbol{\pi}_{K_2}\mathcal{A}_{K_2} = \mathbf{0}$, we have

$$\boldsymbol{\pi}_{K_2}(\mathcal{A}_{K_2} + \mathcal{R}_{K_2}\mathcal{U}_{K_2-1}) = \mathbf{0}. \tag{4}$$

By solving equation (4), we obtain $\boldsymbol{\pi}_{K_2}$ up to a multiplicative constant. From equation (3), for $i = 0, 1, \ldots, K_2 - 1$, $\boldsymbol{\pi}_i$ is given in terms of $\boldsymbol{\pi}_{K_2}$. Using the normalization condition $\boldsymbol{\pi}\boldsymbol{e} = 1$, we obtain all the steady state probabilities.

The system of equation (4) has $(K_1 + 1)M_1M_2$ equations. The number of equations which we handle decreases as compared to solving $\boldsymbol{\pi}\boldsymbol{Q} = \mathbf{0}$ directly. However, it is still hard to solve system equation when we deal with a large system. To solve stationary equations efficiently, we utilize the structure of matrices $\mathcal{A}$, $\mathcal{D}$, $\mathcal{U}$ and $\mathcal{R}$.

4.2 Algorithmic procedure to solve the steady state probabilities

In equations (3) and (4), to obtain the steady state probabilities, we need to calculate matrices $\mathcal{R}_j$, $j = 1, \ldots, K_2$. In what follows, we propose the efficient calculation of these matrices.

Recall that matrices $\mathcal{R}_j$ are denoted by

$$\mathcal{R}_j = \begin{cases} \mathcal{D}_1(-\mathcal{A}_0)^{-1} & j = 1, \\ \mathcal{D}_j(-\mathcal{A}_{j-1} - \mathcal{R}_{j-1}\mathcal{U}_{j-2})^{-1}, & j = 2, \ldots, K_2. \end{cases} \tag{5}$$

As showed in previous section, $\mathcal{A}$, $\mathcal{U}$ and $\mathcal{D}$ are sparse matrices. Specifically, matrix $\mathcal{D}_j$ has only one matrix element, $\boldsymbol{D}_{j,0}^{(2)}$, thus from equation (5), we see that matrices $\mathcal{R}$ have sparse structure. Here, define matrices $\mathcal{T}_j$ as

$$\mathcal{T}_j = \begin{cases} (-\mathcal{A}_0)^{-1}, & j = 1, \\ (-\mathcal{A}_{j-1} - \mathcal{R}_{j-1}\mathcal{U}_{j-2})^{-1}, & j = 2, \ldots, K_2. \end{cases}$$

Furthermore, let $\boldsymbol{T}_{ji}$ denote the first row elements of matrices $\mathcal{T}_j$ $(j = 1, \ldots, K_2)$. It then follows from equation (5) that

$$\mathcal{R}_j = \begin{pmatrix} \boldsymbol{D}_{j,0}^{(2)}\boldsymbol{T}_{j,0} & \boldsymbol{D}_{j,0}^{(2)}\boldsymbol{T}_{j,1} & \cdots & \boldsymbol{D}_{j,0}^{(2)}\boldsymbol{T}_{j,K_1} \\ \boldsymbol{O} & \cdots & \cdots & \boldsymbol{O} \\ \vdots & & & \vdots \\ \boldsymbol{O} & \cdots & \cdots & \boldsymbol{O} \end{pmatrix}, \quad j = 1, \ldots, K_2.$$

Thus, to obtain matrices $\mathcal{R}$, we need to know matrices $\boldsymbol{T}_{ji}$. Denoting the inverse matrix of $\mathcal{T}_j$ by $\mathcal{V}_j$ $(j = 1, \ldots, K_2)$, then $\mathcal{T}_j\mathcal{V}_j = \mathcal{I}$ holds for all j. Then, from $\mathcal{T}_1\mathcal{V}_1 = \mathcal{I}$, we have

$$
\begin{aligned}
-\boldsymbol{T}_{1,0}\boldsymbol{A}_{0,0} - \boldsymbol{T}_{1,1}\boldsymbol{D}_{0,1} &= \boldsymbol{I}, && (6)\\
-\boldsymbol{T}_{1,j-1}\boldsymbol{U}_{0,j-1} - \boldsymbol{T}_{1,j}\boldsymbol{A}_{0,j} - \boldsymbol{T}_{1,j+1}\boldsymbol{D}_{0,j+1} &= \boldsymbol{O}, \quad j = 1, \ldots, K_1 - 1, && (7)\\
-\boldsymbol{T}_{1,K_1-1}\boldsymbol{U}_{0,K_1-1} - \boldsymbol{T}_{1,K_1}\boldsymbol{A}_{0,K_1} &= \boldsymbol{O}. && (8)
\end{aligned}
$$

Similarly, from $\mathcal{T}_i\mathcal{V}_i = \mathcal{I}$ $(i = 2, \ldots, K_2)$, we obtain

$$
\begin{aligned}
-\boldsymbol{T}_{i,0}(\boldsymbol{A}_{i-1,0} + \boldsymbol{D}^{(2)}_{i-1,0}\boldsymbol{T}_{i-1,0}\boldsymbol{U}^{(2)}_{i-2,0}) - \boldsymbol{T}_{i,1}\boldsymbol{D}_{i-1,1} &= \boldsymbol{I}, && (9)\\
-\boldsymbol{T}_{i,0}\boldsymbol{D}^{(2)}_{i-1,0}\boldsymbol{T}_{i-1,j}\boldsymbol{U}^{(2)}_{i-2,j} - \boldsymbol{T}_{i,j-1}\boldsymbol{U}_{i-1,j-1} - \boldsymbol{T}_{i,j}\boldsymbol{A}_{i-1,j} - \boldsymbol{T}_{i,j+1}\boldsymbol{D}_{i-1,j+1} &= \boldsymbol{O}, && (10)\\
j = 1, \ldots, K_1 - 1,\\
-\boldsymbol{T}_{i,0}\boldsymbol{D}^{(2)}_{i-1,0}\boldsymbol{T}_{i-1,K_1}\boldsymbol{U}^{(2)}_{i-2,K_1} - \boldsymbol{T}_{i,K_1-1}\boldsymbol{U}_{i-1,K_1-1} - \boldsymbol{T}_{i,K_1}\boldsymbol{A}_{i-1,K_1} &= \boldsymbol{O}. && (11)
\end{aligned}
$$

From equations (6), (7) and (8), we have for $j = 1, \ldots, K_1 - 1$,

$$\boldsymbol{T}_{1,j+1} = \boldsymbol{T}_{1,j}\boldsymbol{R}_{0,j}, \tag{12}$$

where

$$
\boldsymbol{R}_{0,j} = \begin{cases} \boldsymbol{U}_{0,K_1-1}(-\boldsymbol{A}_{0,K_1})^{-1}, & j = K_1 - 1, \\ \boldsymbol{U}_{0,j}(-\boldsymbol{A}_{0,j+1} - \boldsymbol{R}_{0,j+1}\boldsymbol{D}_{0,j+2})^{-1}, & j = 1, \ldots, K_1 - 2. \end{cases}
$$

By induction, we deduce

$$\boldsymbol{T}_{1,j} = \boldsymbol{T}_{1,0}\boldsymbol{R}_{0,0}\boldsymbol{R}_{0,1}\cdots\boldsymbol{R}_{0,j-1}, \quad j = 1, 2, \ldots, K_1.$$

From equation (6), we have

$$\boldsymbol{T}_{1,0} = (-\boldsymbol{A}_{0,0} - \boldsymbol{R}_{0,0}\boldsymbol{D}_{0,1})^{-1}.$$

Furthermore, it follows from equation (9), (10) and (11) that for $j = 1, 2, \ldots, K_1 - 1$,

$$\boldsymbol{T}_{i,j+1} = \boldsymbol{T}_{i,j}\boldsymbol{R}_{i-1,j} + \boldsymbol{T}_{i,0}\boldsymbol{S}_{i-1,j}, \tag{13}$$

where

$$
\boldsymbol{R}_{i-1,j} = \begin{cases} \boldsymbol{U}_{i-1,K_1-1}(-\boldsymbol{A}_{i-1,K_1})^{-1}, & j = K_1 - 1, \\ \boldsymbol{U}_{i-1,j}(-\boldsymbol{A}_{i-1,j+1} - \boldsymbol{R}_{i-1,j+1}\boldsymbol{D}_{i-1,j+2})^{-1}, & j = 1, \ldots, K_1 - 2, \end{cases}
$$

$$
\boldsymbol{S}_{i-1,j} =
\begin{cases} \boldsymbol{D}^{(2)}_{i-1,0}\boldsymbol{T}_{i-1,K_1}\boldsymbol{U}^{(2)}_{i-2,K_1}(-\boldsymbol{A}_{i-1,K_1})^{-1}, & j = K_1 - 1, \\ (\boldsymbol{S}_{i-1,j+1}\boldsymbol{D}_{i-1,j+2} + \boldsymbol{D}^{(2)}_{i-1,0}\boldsymbol{T}_{i-1,j+1}\boldsymbol{U}^{(2)}_{0,j+1})(-\boldsymbol{A}_{i-1,j+1} - \boldsymbol{R}_{i-1,j+1}\boldsymbol{D}_{i-1,j+2})^{-1}, & j = 1, \ldots, K_1 - 2. \end{cases}
$$

From equation (9), we obtain

$$\boldsymbol{T}_{i,0} = (-\boldsymbol{A}_{i-1,0} - \boldsymbol{R}_{i-1,0}\boldsymbol{D}_{i-1,1} - \boldsymbol{S}_{i-1,0}\boldsymbol{D}_{i-1,1} - \boldsymbol{D}^{(2)}_{i-1,0}\boldsymbol{T}_{i-1,0}\boldsymbol{U}^{(2)}_{i-2,0})^{-1}.$$

Thus, we can solve equation (4) by using matrices $\boldsymbol{T}_{ji}$ deduced here. If we divide the vector $\boldsymbol{\pi}_{K_2}$ into a sequence of vectors $\boldsymbol{\pi}_{K_2}(0), \boldsymbol{\pi}_{K_2}(1), \ldots, \boldsymbol{\pi}_{K_2}(K_1)$, where $\boldsymbol{\pi}_{K_2}(i)$ $(i = 0, \ldots, K_1)$ is a column vector with $M_1 M_2$ dimension, then the equation (4) is rewritten to be

$$\boldsymbol{\pi}_{K_2}(0)(\boldsymbol{A}_{K_2,0} + \boldsymbol{D}^{(2)}_{K_2,0}\boldsymbol{T}_{K_2,0}\boldsymbol{U}^{(2)}_{K_2-1,0}) + \boldsymbol{\pi}_{K_2}(1)\boldsymbol{D}_{K_2,1} = \boldsymbol{0}, \tag{14}$$

$$\boldsymbol{\pi}_{K_2}(0)\boldsymbol{D}^{(2)}_{K_2,0}\boldsymbol{T}_{K_2,j}\boldsymbol{U}^{(2)}_{K_2-1,j} + \boldsymbol{\pi}_{K_2}(j-1)\boldsymbol{U}_{K_2,j-1} + \boldsymbol{\pi}_{K_2}(j)\boldsymbol{A}_{K_2,j} + \boldsymbol{\pi}_{K_2}(j+1)\boldsymbol{D}_{K_2,j+1} = \boldsymbol{0}, \quad j = 2, 3, \ldots, K_1 - 1, \tag{15}$$

$$\boldsymbol{\pi}_{K_2}(0)\boldsymbol{D}^{(2)}_{K_2,0}\boldsymbol{T}_{K_2,K_1}\boldsymbol{U}^{(2)}_{K_2-1,K_1} + \boldsymbol{\pi}_{K_2}(K_1-1)\boldsymbol{U}_{K_2,K_1-1} + \boldsymbol{\pi}_{K_2}(K_1)\boldsymbol{A}_{K_2,K_1} = \boldsymbol{0}. \tag{16}$$

From equations (14), (15) and (16), we have for $j = 1, 2, \ldots, K_1$,

$$\boldsymbol{\pi}_{K_2}(j) = \boldsymbol{\pi}_{K_2}(j-1)\boldsymbol{R}_{K_2,j-1} + \boldsymbol{\pi}_{K_2}(0)\boldsymbol{S}_{K_2,j-1}, \tag{17}$$

where

$$\boldsymbol{R}_{K_2,j-1} = \begin{cases} \boldsymbol{U}_{K_2,K_1-1}(-\boldsymbol{A}_{K_2,K_1})^{-1}, & j = K_1, \\ \boldsymbol{U}_{K_2,j-1}(-\boldsymbol{A}_{K_2,j} - \boldsymbol{R}_{K_2,j}\boldsymbol{D}_{K_2,j+1})^{-1}, & j = 1, \ldots, K_1 - 1, \end{cases}$$

$$\boldsymbol{S}_{K_2,j-1} = \begin{cases} \boldsymbol{D}^{(2)}_{K_2,0}\boldsymbol{T}_{K_2,K_1}\boldsymbol{U}^{(2)}_{K_2-1,K_1}(-\boldsymbol{A}_{K_2,K_1})^{-1}, & j = K_1, \\ (\boldsymbol{S}_{K_2,j}\boldsymbol{D}_{K_2,j+1} + \boldsymbol{D}^{(2)}_{K_2,0}\boldsymbol{T}_{K_2,j}\boldsymbol{U}^{(2)}_{K_2-1,j})(-\boldsymbol{A}_{K_2,j} - \boldsymbol{R}_{K_2,j}\boldsymbol{D}_{K_2,j+1})^{-1}, & j = 1, \ldots, K_1. \end{cases}$$

Then, from equation (14), we have

$$\boldsymbol{\pi}_{K_2}(0)(\boldsymbol{A}_{K_2,0} + \boldsymbol{R}_{K_2,0}\boldsymbol{D}_{K_2,1} + \boldsymbol{S}_{K_2,0}\boldsymbol{D}_{K_2,1} + \boldsymbol{D}^{(2)}_{K_2,0}\boldsymbol{T}_{K_2,0}\boldsymbol{U}^{(2)}_{K_2-1,0}) = \boldsymbol{0}. \tag{18}$$

By solving equation (18), we obtain $\boldsymbol{\pi}_{K_2}(0)$ up to a multiplicative constant. From equation (17), we can describe $\boldsymbol{\pi}_{K_2}(j)$ $(j = 1, \ldots, K_1)$ in terms of $\boldsymbol{\pi}_{K_2}(0)$, that is, we obtain vector $\boldsymbol{\pi}_{K_2}$. Using this $\boldsymbol{\pi}_{K_2}$ in equation (3), we can write $\boldsymbol{\pi}_i$ $(i = 0, \ldots, K_2 - 1)$ using $\boldsymbol{\pi}_{K_2}$. Finally, from the normalization condition $\boldsymbol{\pi}\boldsymbol{e} = 1$, we have all the steady state probabilities.

We proposed an efficient way to solve the steady state probabilities here, and then we can quantify the two QoSs, i.e., throughput and mean delay as follows.

- Throughput
  We define $\tilde{\lambda}^{(i)}_{IN}$ (i=1,2) as effective throughput of i-IN packets and $\tilde{\lambda}^{(i)}_{OUT}$ as that of i-OUT packets, and then for $i = 1, 2$, these are given by

$$\tilde{\lambda}^{(i)}_{IN} = \sum_{(n_1,n_2,(m_1,m_2))\in X} \boldsymbol{\pi}(n_1, n_2, (m_1, m_2))\Lambda^{(i)}_{IN}(n_1, n_2, (m_1, m_2)).$$

$$\tilde{\lambda}^{(i)}_{OUT} = \sum_{(n_1,n_2,(m_1,m_2))\in X} \boldsymbol{\pi}(n_1, n_2, (m_1, m_2))\Lambda^{(i)}_{OUT}(n_1, n_2, (m_1, m_2)).$$

- Mean delay
  Let L_i $(i = 1, 2)$ denote the expected number of DPi packets in the system. We then have

$$L_i = \sum_{(n_1,n_2,(m_1,m_2))\in X} n_i \boldsymbol{\pi}(n_1, n_2, (m_1, m_2)).$$

  We denote mean delay of each DP class by W_i $(i = 1, 2)$, and then they are given by

$$W_i = L_i/(\tilde{\lambda}^{(i)}_{IN} + \tilde{\lambda}^{(i)}_{OUT}).$$

5 Numerical Results

In this section, we show numerical results to examine qualitative performance by using the QoSs given in the previous section. To determine parameters of the mathematical model, we first provide an experimental scenario.

5.1 Scenario

When the traffic load in the network is low users send packets according to the contract, a DiffServ router is expected to differentiate performance of each service class easily. If network congestion happens, however, it is hard to discriminate QoSs. Even so, the provider must discriminate the QoSs in such a situation. Thus, we examine performance of the proposed DiffServ router in an overloaded situation.

In the overloaded network, users will send too many packets against the contract. It means that users send more OUT packets in a heavily loaded situation than in a lightly loaded situation. In the experimental scenario, we suppose that the arrival rate of i-IN $(i = 1, 2)$ packets is fixed to $\bar{\lambda}_{IN}^{(i)}$, and the arrival rate of i-OUT $(i = 1, 2)$ packets is changed from 0 to $\bar{\lambda}_{OUT}^{(i)}$. We realize this scenario by using the phase structure in section 3.1. Then, arrival rates in each service class are given by for $i = 1, 2$,

$$\begin{aligned} \lambda_{IN}^{(i)}(m_i) &= \bar{\lambda}_{IN}^{(i)}, \quad \text{for all } m_i, \\ \lambda_{OUT}^{(i)}(m_i) &= \frac{m_i - 1}{M_i - 1}\bar{\lambda}_{OUT}^{(i)}, \quad m_i = 1, \ldots, M_i. \end{aligned}$$

The phase structure determines the source state by two factors, i.e., arrival rates in respective phases and transition rates between phases. Note that the transition rates depend on the sort of traffic. In our model, we consider two types of traffic, t-traffic and u-traffic. When the source state is in (m_1, m_2), the transition rates of t-traffic are given by $\tau_i(m_1, m_2)$ and $\tilde{\tau}_i(m_1, m_2)$, and those of u-traffic are given by $\gamma_i(m_1, m_2)$ and $\tilde{\gamma}_i(m_1, m_2)$. For simplicity, we assume that for $i = 1, 2$,

$$\begin{aligned} \tau_i(m_1, m_2) &= \tilde{\tau}_i(m_1, m_2) = \bar{\tau}_i, \\ \gamma_i(m_1, m_2) &= \tilde{\gamma}_i(m_1, m_2) = \bar{\gamma}_i. \end{aligned}$$

The ratio of the two types of traffic is supposed in assumption 2. These ratios are decided according to experimental situations.

5.2 Basic characteristics of the proposed DiffServ router

In this section, we examine qualitative performance of the proposed DiffServ router. In particular, we consider effects of the followings.

- Buffer management schemes
  We consider two buffer management schemes, i.e., the separated buffer management scheme and the shared buffer management scheme. It is expected that performance of two buffer management schemes are different. Here after the model in the separated

buffer management scheme is called the separated buffer model and that in the shared buffer management scheme is called the shared buffer model.

- System load
  We change the arrival rates of OUT packets by the phase structure. We set the number of DP phases as $M_1 = 5$ and $M_2 = 5$. Recall that for $i = 1, 2$,

$$\begin{aligned} \lambda_{IN}^{(i)}(m_i) &= \bar{\lambda}_{IN}^{(i)}, \quad \text{for all } m_i, \\ \lambda_{OUT}^{(i)}(m_i) &= \frac{m_i - 1}{M_i - 1}\bar{\lambda}_{OUT}^{(i)}, \quad m_i = 1, \ldots, M_i. \end{aligned}$$

  We denote the total fixed arrival rate of IN packets by $\bar{\lambda}_{IN}$ and the total maximum arrival rate of OUT packets by $\bar{\lambda}_{OUT}$. We consider three cases of $\bar{\lambda}_{IN}$ with 0.6, 0.7 and 0.8. For each case of $\bar{\lambda}_{IN}$, we consider three values of $\bar{\lambda}_{OUT}$ with 0.2, 0.3 and 0.4. Then, we fix the ratio of arrival rates between $DP1$ and $DP2$ to 1/3 here. We examine the case of other ratios in section 5.5. We set service rates of each DP class as $\mu_1 = \mu_2 = 1$. Using these parameters, we can realize our scenario where the system is overloaded. In this overloaded situation, sources send IN packets of DPi $(i = 1, 2)$ at a fixed rate $\bar{\lambda}_{IN}^{(i)}$. Thus, these arrival rates of IN packets themselves become the target throughput of the proposed service scheme.

- Traffic sorts
  As for traffic sorts, we consider three situations in Table 3. In Situation 1, sources send t-traffic in both DP classes, whereas in Situation 2, sources send only u-traffic. In Situation 3, sources send u-traffic of $DP1$ and t-traffic of $DP2$. We set the transition rates between phases to $\bar{\tau}_1 = \bar{\tau}_2 = 0.01$ and $\bar{\gamma}_1 = \bar{\gamma}_2 = 0.01$.

Table 3: Experimental situations.

Situation	DP1	DP2
1	t-traffic	t-traffic
2	u-traffic	u-traffic
3	u-traffic	t-traffic

- Drop mechanisms
  To model the drop mechanisms, we use functions of n_1 and n_2, where n_i $(i = 1, 2)$ denotes the number of DPi packets in the system. We choose the parameters of the functions in each mechanism.

  - RIO: In both buffer models, we choose the same dropping rates with $r_{IN}^{(1)} = r_{IN}^{(2)} = 0.1$, $r_{OUT}^{(1)} = r_{OUT}^{(2)} = 0.5$, and different thresholds as follows: In the separated buffer model,

$$\begin{cases} TH_{IN}^{(1)} = TH_{OUT}^{(1)} = K_1/2, \\ TH_{IN}^{(2)} = TH_{OUT}^{(2)} = K_2/2, \end{cases}$$

while in the shared buffer model,

$$\begin{cases} TH_{IN}^{(1)} = TH_{OUT}^{(1)} = K/2, \\ TH_{IN}^{(2)} = TH_{OUT}^{(2)} = K/2. \end{cases}$$

- THRESH: We choose thresholds of OUT packets as $TH^{(1)} = K_1/2$ and $TH^{(2)} = K_2/2$ in the separated buffer model. In the shared buffer model, we use the same thresholds in both DP classes as $TH^{(1)} = TH^{(2)} = K/2$.

- Buffer size
  The buffer size will have an impact on QoSs, particularly on mean delay. However, in this subsection, we fix buffer size, $K_1 = 100$, $K_2 = 100$ and $K = 200$. In section 5.3, we argue how buffer size works on QoSs.

5.2.1 Situation 1

Here, we examine the QoSs in Situation 1. In this situation, we suppose that sources send only t-traffic, that is, the sources change the sending rates reactively to network congestion.

-Throughput in TAIL drop mechanism

When we examine throughput, we are interested in whether or not IN packets and OUT packets of each DP class are discriminated. Note that the proposed DiffServ routers need to differentiate throughput independently of mean delay discrimination. To argue qualitative performance of throughput, we first suppose that a router uses no specific drop mechanism, that is, uses TAIL mechanism. Table 4 shows throughput in the separated buffer model, and Table 5 shows that in the shared buffer model.

Table 4: Throughput in the separated buffer model.

$\bar{\lambda}_{IN}$	0.6			0.7			0.8		
Target	DP1=0.2, DP2=0.4			DP1=0.233, DP2=0.467			DP1=0.267, DP2=0.533		
$\bar{\lambda}_{OUT}$	0.2	0.3	0.4	0.2	0.3	0.4	0.2	0.3	0.4
1-IN	0.200	0.200	0.200	0.233	0.233	0.233	0.267	0.267	0.267
1-OUT	0.0667	0.100	0.133	0.0667	0.100	0.133	0.0667	0.100	0.133
2-IN	0.400	0.400	0.395	0.467	0.460	0.414	0.526	0.470	0.423
2-OUT	0.133	0.200	0.261	0.133	0.196	0.219	0.131	0.163	0.177

In both buffer models, if the maximum system load which equals to $\bar{\lambda}_{IN} + \bar{\lambda}_{OUT}$ is low, the target throughput of IN packets can be guaranteed. These results show that the proposed DiffServ router can differentiate throughput easily in the low system load. However, if the system is overloaded, the effective throughput is less than the target throughput and the results of the two buffer models are different.

In the separated buffer model, the target throughput of 1-IN is guaranteed regardless of the system load, whereas that of 2-IN is not guaranteed. In this model, the ith ($i = 1, 2$) queue

Table 5: Throughput in the shared buffer model.

$\bar{\lambda}_{IN}$	0.6			0.7			0.8		
Target	DP1=0.2, DP2=0.4			DP1=0.233, DP2=0.467			DP1=0.267, DP2=0.533		
$\bar{\lambda}_{OUT}$	0.2	0.3	0.4	0.2	0.3	0.4	0.2	0.3	0.4
1-IN	0.200	0.200	0.199	0.233	0.232	0.216	0.265	0.246	0.230
1-OUT	0.0667	0.100	0.132	0.0667	0.0993	0.118	0.0662	0.0876	0.103
2-IN	0.400	0.400	0.398	0.467	0.465	0.431	0.531	0.491	0.459
2-OUT	0.133	0.200	0.265	0.133	0.199	0.236	0.132	0.176	0.208

is accessible only by DP class i packets. Owing to the service priority rule, $DP1$ packets are served preferentially, and then throughput of these packets is assured. However, packets of $DP2$ are served only when there are no packets of $DP1$ in the 1st queue, and therefore packets of $DP2$ cannot be served sufficiently and the 2nd buffer is filled with $DP2$ packets. This causes loss of $DP2$ packets, so that throughput of $DP2$ decreases.

In the shared buffer model, throughput of both 1-IN and 2-IN packets are less than the target throughput. The reason is that all packets share one buffer in this model, and thus, if the buffer is full, arriving packets are dropped regardless of service classes.

-Throughput in RIO and THRESH drop mechanisms

To examine the impact of drop mechanisms, we show results in the case that a router uses RIO or THRESH mechanisms. The purpose of drop mechanisms is the improvement of IN packets throughput by dropping OUT packets. We know from the consideration in TAIL mechanism that it is easy to differentiate throughput in the low system load. However, throughput of IN packets are less than the targets in the overloaded situation. Thus, we pay our attention to the overloaded situation, i.e., we set $\bar{\lambda}_{IN} = 0.8$, $\bar{\lambda}_{OUT} = 0.4$. Results in both buffer models are showed in Table 6.

Table 6: Throughput in drop mechanisms ($\bar{\lambda}_{IN} = 0.8$, $\bar{\lambda}_{OUT} = 0.4$, target of DP1= 0.267, DP2= 0.533).

model	separate			share		
mechanism	TAIL	RIO	THRESH	TAIL	RIO	THRESH
1-IN	0.267	0.267	0.267	0.230	0.246	0.267
1-OUT	0.133	0.133	0.133	0.103	0.0867	0.0667
2-IN	0.423	0.453	0.532	0.459	0.493	0.533
2-OUT	0.177	0.147	0.0682	0.208	0.173	0.133

In the separated buffer model, RIO decreases 2-OUT throughput and increases 2-IN throughput. However, RIO cannot secure the target throughput of 2-IN packets. Moreover, RIO has no influence on 1-OUT throughput in this buffer model. The reason is that owing to service

priority rule, the queue length of the 1st buffer does not become large and therefore RIO does not drop 1-OUT packets sufficiently. This is a drawback of the separated buffer model. In the shared buffer model, by RIO mechanism, throughput of 1-OUT and 2-OUT are decreased and 1-IN and 2-IN are increased, compared with TAIL. However, in the overloaded situation, throughput of 1-IN and 2-IN is less than the target.

When the router uses THRESH mechanism, it can guarantee the target throughput of IN packets in each DP class. In the separated buffer model, THRESH makes the performance of 2-IN packets better than TAIL and RIO. However, these results are caused not by decreasing 1-OUT packets, but by decreasing 2-OUT packets. The reason that THRESH cannot decrease throughput of 1-OUT as in the RIO case. In the shared buffer model, throughput of IN packets in each DP class equals to the target throughput, even in the highly loaded situation. Because dropping OUT packets leads to improving that of IN packets, THRESH is a recommended mechanism. Because the shared buffer model can differentiate throughput between IN and OUT packets independently of DP classes, we see that this model is better than the separated buffer model.

-Mean delay

When we look at differentiation of mean delay, we do not make a distinction between IN and OUT packets. We first argue mean delay in the TAIL mechanism, and then examine how drop mechanisms make effects on mean delay. To focus on a heavily loaded situation, we fix $\bar{\lambda}_{IN} = 0.8$ and change only $\bar{\lambda}_{OUT}$. Recall that we set $\mu_1 = \mu_2 = 1.0$.

From Figure 5 and Figure 6, we note that in both buffer models, mean delay of $DP1$ is very small, whereas that of $DP2$ increases as $\bar{\lambda}_{OUT}$ increases. Owing to service priority discipline, a router provides low mean delay for $DP1$ packets. Furthermore, drop mechanisms have no effects on mean delay of $DP1$. Thus, the proposed DiffServ router can guarantee mean delay of $DP1$. On the other hand, delay of a $DP2$ packet consists of two factors. One is the sum of service times of packets found on arrival. The other is the sum of service times for $DP1$ packets which arrive during this $DP2$ packet's waiting time. If the system load increases, the influences of these two factors becomes large, and then $DP2$ packets have large delay. In the overloaded situation, $DP2$ packets cannot be served easily due to service priority rule, and then the buffer is filled with $DP2$ packets. Thus, in such a situation, $DP2$ packets have delay in proportion to the buffer size. $DP2$ packets enter the 2nd buffer in the separated buffer model, and enter the shared buffer in the shared buffer model. So, mean delay of $DP2$ is in proportion to the buffer size of the 2nd buffer in the separated buffer model, and that of the shared buffer in the shared buffer model. Now, the size of the 2nd buffer in the separated buffer model is 100, and the buffer size in the shared buffer model is 200. Thus, mean delay of $DP2$ in the shared buffer model is two times as large as that in the separated buffer model in Figure 6.

In the case that a router uses RIO or THRESH, mean delay of $DP2$ decreases compared with TAIL. These mechanisms drop packets before the buffer becomes full, and then the mean waiting time becomes smaller than TAIL. In the proposed new service scheme, originally, a DiffServ router uses drop mechanisms to differentiate throughput between IN and OUT packets. Moreover, these mechanisms are effective to decrease mean delay of $DP2$.

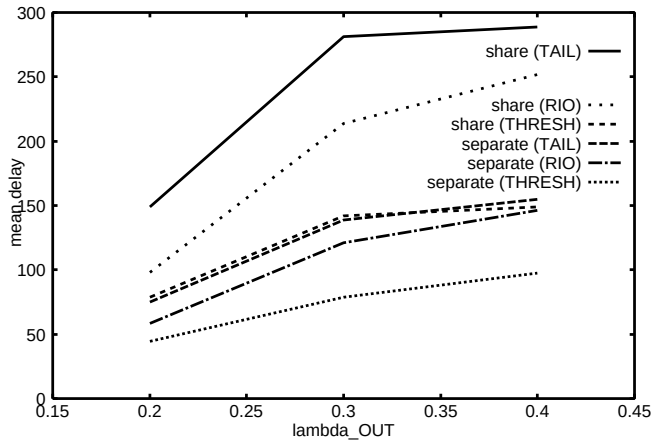

Figure 5: Mean delay of $DP1$ ($\bar{\lambda}_{IN} = 0.8$).

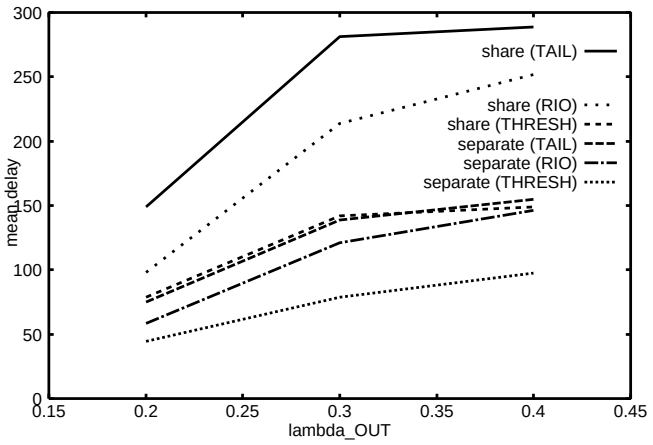

Figure 6: Mean delay of $DP2$ ($\bar{\lambda}_{IN} = 0.8$).

5.2.2 Other situations

In previous section, we argued the performance in Situation 1. In this situation, sources send only t-traffic in each DP class. If sources change the sorts of traffic they send, how does the proposed DiffServ router work? To answer this question, we show numerical results in Situations 2 and 3 to examine effects of traffic sorts. Recall that sources send only u-traffic in Situation 2, and sources send u-traffic of $DP1$ packets and t-traffic of $DP2$ packets in Situation 3. To focus on the overloaded system, we set $\bar{\lambda}_{IN} = 0.8$ below.

-Throughput and impact of drop mechanisms

The results in Situation 1 showed that in TAIL mechanism, a router can differentiate throughput in light traffic. However, heavy traffic reduces throughput of IN packets. In Table 7, we show throughput in TAIL in Situation 2, and in Table 8, show that in Situation 3.

Table 7: Throughput in TAIL in Situation 2 ($\bar{\lambda}_{IN} = 0.8$, target of $DP1 = 0.267$, $DP2 = 0.533$).

model	separate			share		
$\bar{\lambda}_{OUT}$	0.2	0.3	0.4	0.2	0.3	0.4
1-IN	0.267	0.267	0.267	0.267	0.266	0.262
1-OUT	0.0333	0.0500	0.0667	0.0333	0.0499	0.0651
2-IN	0.533	0.530	0.513	0.533	0.533	0.524
2-OUT	0.0667	0.0989	0.125	0.0667	0.0998	0.130

From these tables, we observe that there are common phenomena in Situations 2 and 3. In both buffer models, if the system load is low, a router can differentiate throughput without drop mechanisms, while if the system load becomes high, it is hard to differentiate throughput. These results in Situations 2 and 3 agree with the results in Situation 1.

Table 8: Throughput in TAIL in Situation 3 ($\bar{\lambda}_{IN} = 0.8$, target of $DP1 = 0.267$, $DP2 = 0.533$).

model	separate			share		
$\bar{\lambda}_{OUT}$	0.2	0.3	0.4	0.2	0.3	0.4
1-IN	0.267	0.267	0.267	0.267	0.255	0.239
1-OUT	0.0333	0.0500	0.0667	0.0333	0.0474	0.0587
2-IN	0.532	0.501	0.459	0.533	0.511	0.478
2-OUT	0.133	0.181	0.208	0.133	0.187	0.224

In Situation 1, when a router uses RIO drop mechanism, it improves throughput of IN packets, but does not achieve the targets. On the other hand, if a router uses THRESH drop mechanism, throughput of IN packets reaches the target throughput completely. Now, we show throughput in drop mechanisms in Situation 2 in Table 9, and that in Situation 3 in Table 10. Using drop mechanisms such as RIO or THRESH, a router can differentiate throughput as in Situation 1. RIO can improve throughput by dropping OUT packets, and throughput on RIO does not reach the target throughput. Throughput in THRESH mechanism equals to the target one in both situations.

Table 9: Throughput in drop mechanisms in Situation 2 ($\bar{\lambda}_{IN} = 0.8$, target of DP1= 0.267, DP2= 0.533, $\bar{\lambda}_{OUT} = 0.4$).

model	separate			share		
mechanism	TAIL	RIO	THRESH	TAIL	RIO	THRESH
1-IN	0.267	0.267	0.267	0.262	0.264	0.267
1-OUT	0.0667	0.0667	0.0667	0.0651	0.0621	0.0585
2-IN	0.513	0.520	0.533	0.524	0.527	0.533
2-OUT	0.125	0.115	0.0979	0.130	0.123	0.114

Table 10: Throughput in drop mechanisms in Situation 3 ($\bar{\lambda}_{IN} = 0.8$, target of DP1= 0.267, DP2= 0.533, $\bar{\lambda}_{OUT} = 0.4$).

model	separate			share		
mechanism	TAIL	RIO	THRESH	TAIL	RIO	THRESH
1-IN	0.267	0.267	0.267	0.239	0.252	0.267
1-OUT	0.0667	0.0667	0.0667	0.0587	0.0476	0.0379
2-IN	0.459	0.486	0.533	0.478	0.504	0.533
2-OUT	0.208	0.181	0.133	0.224	0.196	0.162

As for throughput differentiation, the proposed DiffServ router can work well by using

appropriate drop mechanism in any situations. Using THRESH drop mechanism, a router can provide the target throughput for IN packets in both buffer models. As mentioned in Situation 1, to differentiate throughput independently of DP classes, a router should use the shared buffer management scheme.

-Mean delay

In Table 11, we show mean delay of $DP1$ in Situations 2 and 3. In both situations, a router can keep mean delay of $DP1$ low as in Situation 1. We also note that drop mechanisms have no effects on mean delay of $DP1$ in both situations. Moreover, there is little difference of mean delay in both buffer models.

Table 11: Mean delay of $DP1$ in Situations 2 and 3 ($\bar{\lambda}_{IN} = 0.8$).

situation	model	separate			share		
	$\bar{\lambda}_{OUT}$	0.2	0.3	0.4	0.2	0.3	0.4
2	TAIL	1.43	1.47	1.52	1.43	1.47	1.49
	RIO	1.43	1.47	1.52	1.43	1.47	1.50
	THRESH	1.43	1.47	1.52	1.43	1.47	1.49
3	TAIL	1.43	1.47	1.52	1.43	1.42	1.38
	RIO	1.43	1.47	1.52	1.43	1.44	1.43
	THRESH	1.43	1.47	1.52	1.43	1.44	1.43

In Figure 7, we show mean delay of $DP2$ in Situation 2, and in Figure 8, that in Situation 3. As $\bar{\lambda}_{OUT}$ increases from 0.2 to 0.4, mean delay of $DP2$ increases in both situations. As observed in Situation 1, mean delay in the shared buffer model is two times as large as that in the separated buffer model.

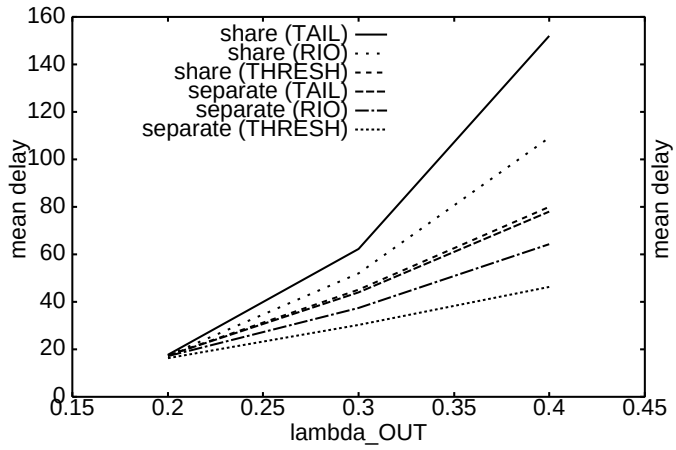

Figure 7: Mean delay of $DP2$ in Situation 2 ($\bar{\lambda}_{IN} = 0.8$).

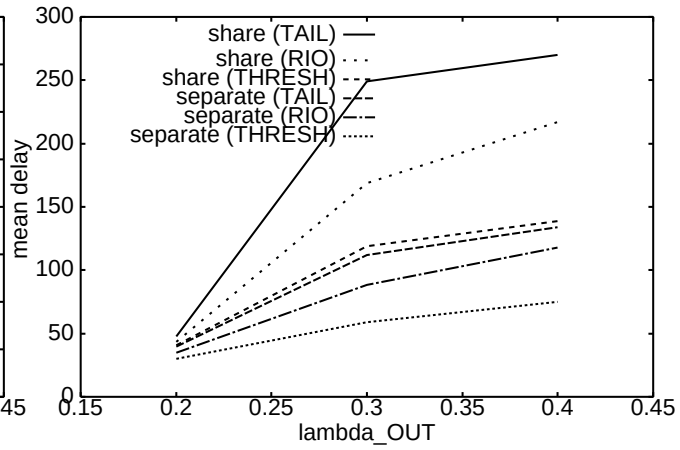

Figure 8: Mean delay of $DP2$ in Situation 3 ($\bar{\lambda}_{IN} = 0.8$).

In any situations where sources send the different types of traffic, a router can provide low

mean delay for $DP1$ packets by using one of the two buffer management scheme. Mean delay of $DP2$ is in proportion to the 2nd buffer size in the separated buffer model and the shared buffer size in the shared buffer model. We show that besides throughput differentiation, the proposed DiffServ router can provide mean delay differentiation.

5.3 Impact of buffer size

We examined numerical results with the fixed buffer size in both buffer models, i.e., $K_1 = K_2 = 100$ and $K = 200$. In this subsection, we argue the impact of the buffer size on performance. First, we change the total buffer size, which is given by $K_1 + K_2$ in the separated buffer model or by K in the shared buffer model. Then, we change the ratio of the 1st buffer size to the 2nd buffer size in the separated buffer model. From the consideration in section 5.2, we know that we can deduce the same conclusion in any experimental situations. Therefore, we assume that sources send u-traffic of $DP1$ and t-traffic of $DP2$, that is, Situation 3 in section 5.2. We use the same parameters in section 5.2 except for buffer sizes.

5.3.1 Total buffer size

Table 12 shows throughput with the total buffer size 100, 200 and 400. In the separated buffer model, the ratio of the buffer sizes between $DP1$ and $DP2$ is fixed to $K_1 = K_2$, here. In each buffer model, there are no differences when the buffer size is changed. If a router uses drop mechanisms such as RIO or THRESH and changes the total buffer size, it makes no influence on throughput performance. Thus, we note that the total buffer size does not affect on throughput performance.

Table 12: Throughput in TAIL with the total buffer size 100, 200 and 400 ($\bar{\lambda}_{IN} = 0.8$, $\bar{\lambda}_{OUT} = 0.4$).

model	separate			share		
K	100	200	400	100	200	400
1-IN	0.267	0.267	0.267	0.239	0.239	0.239
1-OUT	0.0667	0.0667	0.0667	0.0587	0.0587	0.0587
2-IN	0.458	0.458	0.458	0.478	0.478	0.478
2-OUT	0.207	0.207	0.207	0.224	0.224	0.224

Figure 9 shows mean delay of $DP1$ and Figure 10 shows that of $DP2$ when the total buffer size is changed from 50 to 400. Likewise to throughput results, mean delay of $DP1$ is not changed when we change the buffer size. On the other hand, mean delay of $DP2$ increases in proportion to the total buffer size. Therefore, it is verified that mean delay of $DP2$ is in proportion to the buffer size. When a router uses drop mechanisms, we can get the same results as in TAIL.

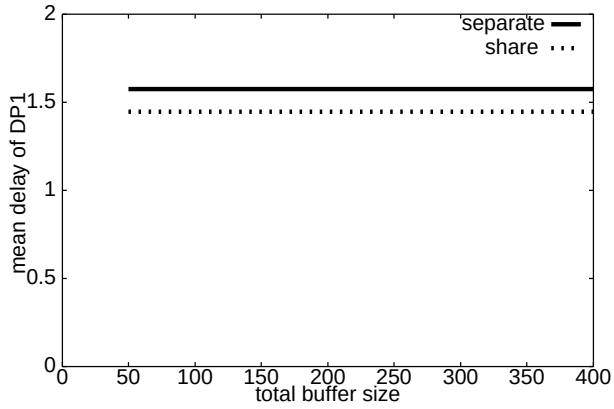

Figure 9: Mean delay of $DP1$ ($\bar{\lambda}_{IN} = 0.8$, $\bar{\lambda}_{OUT} = 0.4$).

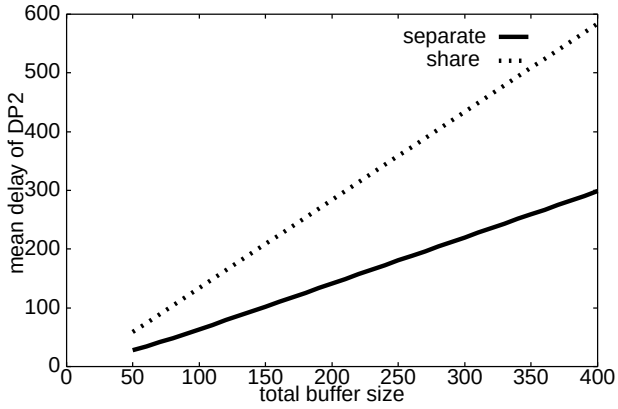

Figure 10: Mean delay of $DP2$ ($\bar{\lambda}_{IN} = 0.8$, $\bar{\lambda}_{OUT} = 0.4$).

We showed that the total buffer size affects only on mean delay of $DP2$. Mean delay of $DP2$ increases in proportion to the total buffer size. The proposed DiffServ router can differentiate throughput and mean delay independently of the total buffer size.

5.3.2 Ratio of buffer sizes in the separated buffer model

We examined performance with the fixed buffer ratio, i.e., K_1/K_2. Thus in this subsection, we change this ratio. We can argue the impact of this ratio only in the separated buffer model. We fix the total buffer size to $K_1 + K_2 = 200$ and change the 1st buffer size from 1 to 199, that is, change the 2nd buffer size from 199 to 1. We set arrival rates as $\bar{\lambda}_{IN} = 0.8$, $\bar{\lambda}_{OUT} = 0.4$ and use only TAIL mechanism to examine qualitative performance in the overloaded case. We suppose that sources send traffic according to Situation 3 in section 5.2.

Figure 11 shows throughput when the 1st buffer size is changed from 1 to 199. If we set a very small buffer size for the 1st buffer, throughput of IN packets decreases and that of OUT packets increases. When the 1st buffer size is very small, for example, $K_1 = 1$, all arriving $DP1$ packets are discarded if one $DP1$ packet is found in the 1st queue. As a result, throughput of $DP1$ decreases and $DP2$ packets get high throughput. If we set a very large buffer size for the 1st buffer, that is, set a very small one for the 2nd buffer, $DP2$ packets cannot get throughput because the drop probability of $DP2$ packets becomes high, i.e., the probability that the 2nd buffer is full is large in addition to disadvantage due to service priority discipline. However, if we provide an appropriate size for each buffer, packets of both DP classes get the same throughput independently of the ratio of the buffer sizes.

In Figure 12, we show mean delay of $DP1$ as a function of the 1st buffer size, and in Figure 13 we show that of $DP2$. When the 1st buffer size is not small, mean delay of $DP1$ is not changed regardless of the buffer size. If we set a very small buffer size for the 1st buffer, it takes a little time for $DP1$ packets to get service. Therefore, mean delay of $DP1$ decreases, and as a result, mean delay of $DP2$ also decreases. As the 1st buffer size increases, that is, the 2nd buffer size decreases, mean delay of $DP2$ decreases. This shows again that the mean waiting time of $DP2$ is in proportion to the 2nd buffer size.

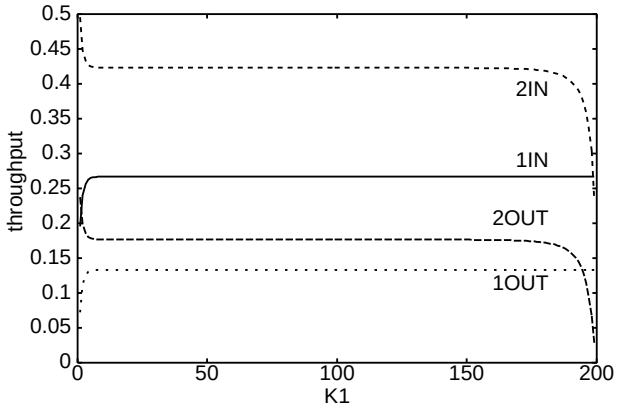

Figure 11: Throughput in TAIL in the separated buffer model as a function of the 1st buffer size ($\bar{\lambda}_{IN} = 0.8$, $\bar{\lambda}_{OUT} = 0.4$).

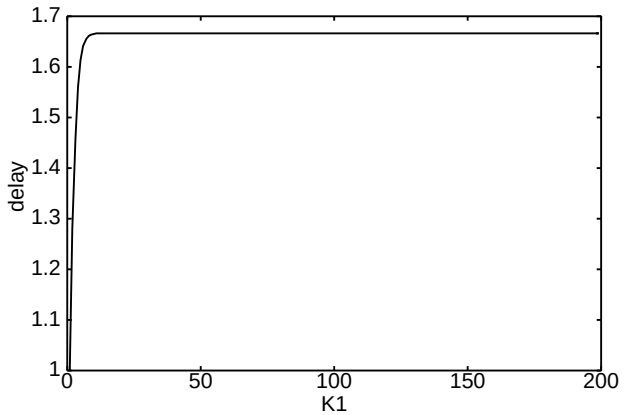

Figure 12: Mean delay of $DP1$ as a function of the 1st buffer size ($\bar{\lambda}_{IN} = 0.8$, $\bar{\lambda}_{OUT} = 0.4$).

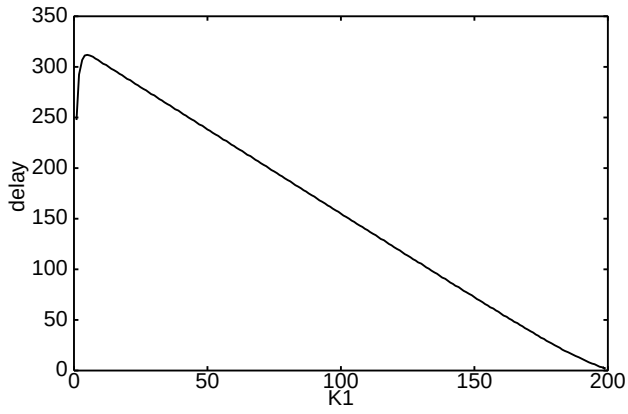

Figure 13: Mean delay of $DP2$ as a function of the 1st buffer size ($\bar{\lambda}_{IN} = 0.8$, $\bar{\lambda}_{OUT} = 0.4$).

From the results in this section, we show that if we provide a sufficient buffer size for both DP classes, the proposed DiffServ router can differentiate the two QoSs independently of the ratio of the buffer sizes.

5.4 Ratio of arrival rates between DP classes

We fixed the ratio of arrival rates between DP classes, i.e., $\bar{\lambda}_{IN}^{(1)}/\bar{\lambda}_{IN}^{(2)} = \bar{\lambda}_{OUT}^{(1)}/\bar{\lambda}_{OUT}^{(2)} = 1/3$. We examine performance in the case of various ratios of arrival rates. To quantify the impact of this ratio, we show performance measures in TAIL drop mechanism. We use the same parameters as in section 5.2 except for this ratio, and we suppose that sources send u-traffic of $DP1$ and t-traffic of $DP2$. To consider the overloaded situation, we fix arrival rates of IN and OUT packets to $\bar{\lambda}_{IN} = 0.8$ and $\bar{\lambda}_{OUT} = 0.4$, respectively.

We show throughput of the separated buffer model in Figure 14, and that of the shared buffer model in Figure 15. If the ratio of arrival rates between DP classes approaches to 1, arrival rate of $DP1$ becomes high. On the contrary, when this ratio approaches to 0, arrival rates of $DP2$ becomes high. In both buffer models, as the arrival rate of $DP1$ packets increases, throughput of $DP1$ packets increases. In a similar way, as the arrival rate of $DP2$ increases, throughput of $DP2$ packets increases. In the separated buffer model, when the arrival rate of $DP1$ becomes over a certain value, a router cannot assure throughput of $DP2$, because $DP2$ packets have little opportunity to get service. In this viewpoint, the shared buffer model differs from the separated buffer model because the shared buffer model can differentiate throughput between IN and OUT packets independently of DP classes.

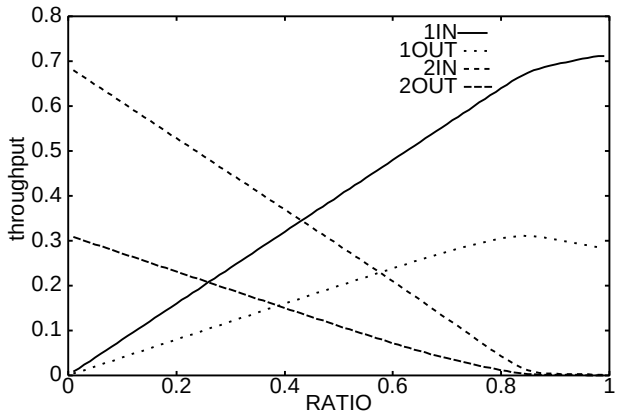

Figure 14: Throughput as a function of the ratio of arrival rates between DP classes ($\bar{\lambda}_{IN} = 0.8$, $\bar{\lambda}_{OUT} = 0.4$).

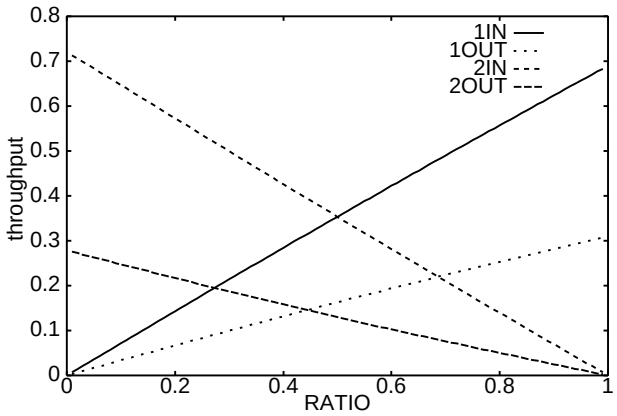

Figure 15: Throughput as a function of the ratio of arrival rates between DP classes ($\bar{\lambda}_{IN} = 0.8$, $\bar{\lambda}_{OUT} = 0.4$).

In Figure 16 and 17, we show mean delay of $DP1$ class and $DP2$ class, respectively. When the ratio approaches to 1, mean delay of each DP class becomes large, in particular, mean delay of $DP2$ packets becomes very large. In the overloaded situation, if most traffic belongs to $DP1$, the system is congested by this $DP1$ traffic, and then $DP2$ packets loose the opportunity to get service, so that they have large waiting time. However, if this ratio is under a certain value, mean delay of each DP class is not changed regardless of the ratio.

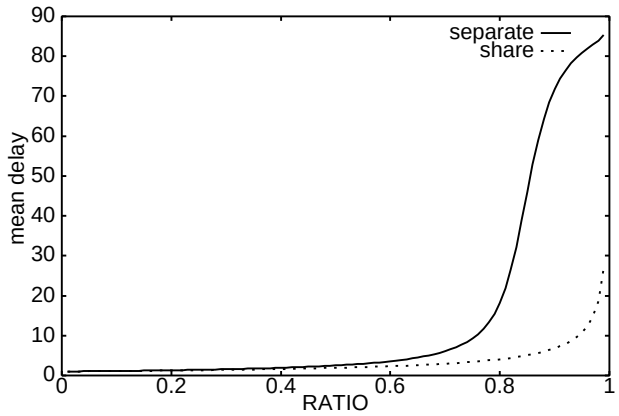

Figure 16: Mean delay of $DP1$ with the ratio of arrival rates between $DP1$ and $DP2$ packets ($\bar{\lambda}_{IN} = 0.8$, $\bar{\lambda}_{OUT} = 0.4$).

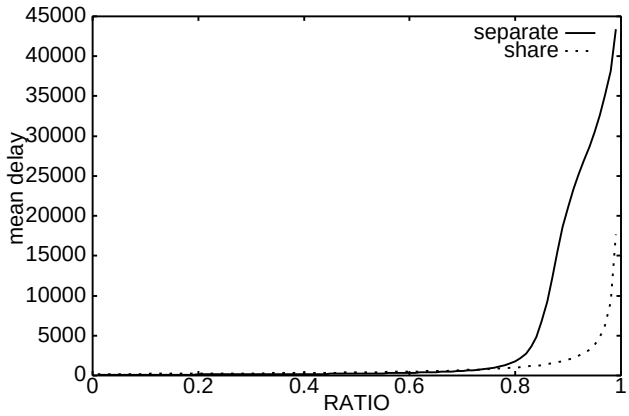

Figure 17: Mean delay of $DP2$ with the ratio of arrival rates between $DP1$ and $DP2$ packets ($\bar{\lambda}_{IN} = 0.8$, $\bar{\lambda}_{OUT} = 0.4$).

If we set the arrival rate of $DP1$ packets very large, it deteriorates performance of the proposed DiffServ router. However, if the arrival rate of $DP1$ is small, the DiffServ router can work well.

6 Conclusion

In this thesis, we proposed a new service scheme for a DiffServ router, which aims for differentiation of the two QoSs, i.e., throughput and mean delay, at the same time. To implement the proposed new service scheme, we proposed two buffer management schemes, the separated buffer management scheme and the shared buffer management scheme. In both buffer management schemes, we obtained expressions for performance measures, which can be used in numerical experiments.

As for throughput differentiation, our purpose is achieved by the shared buffer model with the threshold drop mechanism because this model can discriminate throughput between IN and OUT packets independently of DP classes. The proposed DiffServ router can differentiate mean delay between DP classes. Due to service priority discipline, packets of $DP1$ get low mean delay. Mean delay of $DP2$ is closely related to the buffer size in both buffer management schemes. In the separated buffer model, mean delay of $DP2$ is in proportion to the 2nd buffer size and that in the shared buffer model is in proportion to the shared buffer size.

Finally, we suggest a design of a DiffServ router. A provider should make the arrival rate of $DP1$ low because the high arrival rate of $DP1$ brings deterioration of the QoSs. To differentiate throughput, we recommend the threshold drop mechanism, even through in this mechanism, OUT packets are discarded even if there are buffer spaces. For the provision of the service in this mechanism, we should use an architecture such as MPLS not to increase too OUT packets. If the router uses RIO drop mechanism, we should evaluate the target throughput as IN throughput and part of OUT throughput. In mean delay discrimination, we should pay our attention to set the parameters about the buffer sizes because these parameters affect on delay

of $DP2$. In any parameter settings, however, the router can guarantee mean delay of $DP1$ in each buffer management scheme.

Acknowledgments

I gratefully acknowledge valuable guidance and helpful suggestions with Associate Professor Tetsuya Takine. I would like to express my thanks to Professor Masao Fukushima for thoughtful comments. I wish to thank the members in Professor Fukushima's laboratory for supporting my research. My thanks are also due to my friends and my family for encouraging words.

References

[1] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z. and Qeiss, W., "An architecture for Differentiated Services," RFC 2475, 1998.

[2] Braden, R., Clark, D. and Shenker, S., "Integrated Services in the Internet Architecture: an Overview," RFC 1633, 1994.

[3] Clark, D., "Adding Service Discrimination to the Internet," LCS MIT Technical report, 1995.

[4] Clark, D. and Fang, W., "Explicit Allocation of Best-Effort Packet Delivery Service," *IEEE/ACM Transactions on Networking*, vol. 6, pp. 362–373, 1997.

[5] Floyd, S. and Jacobson, V., "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 397–413, 1993.

[6] Labrador, M. and Banerjee, S., "Enhancing Application Throughput by Selective Packet Dropping," *Proc. of IEEE ICC*, pp. 1217–1222, 1999.

[7] May, M., Bolot, J. and Diot, C., "Simple Performance Models of Differentiated Service Schemes for the Internet," *IEEE INFOCOM '99*, pp. 1385–1394, 1999.

[8] Nichols, K., Jacobson, V. and Zhang, L., "A Two-bit Differentiated Services Architecture for the Internet," RFC 2638, 1999.

[9] Sahu, S. Towsley, D. and Kurose, J., "A Quantitative Study of Differentiated Services for the Internet," *Proc. of IEEE Global Internet, GLOBECOM '99*, pp. 1808–1817, 1999.

[10] Zhang, L., Estrin, D., Deering, S., Shenker, S. and Zappala, D., "RSVP: A New Resource ReSerVation Protocol," *IEEE Network*, vol. 5, pp. 8–18, 1993.