PROPORTIONAL INTERACTIVE BUFFER MANAGEMENT AND SCHEDULING

by

Yifan Peng
Bachelor of Science Degree in Electrical Engineering
South China University, China, 1996

A thesis
presented to Ryerson University
in partial fulfillment of the
requirements for the degree of
Master of Applied Science
in
Electrical and Computer Engineering Program
with Specialization in Computer Networks

Toronto, Ontario, Canada, 2006

I hereby declare that I am the sole author of this thesis.
I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Yifan Peng

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Yifan Peng

Ryerson University requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Proportional Interactive Buffer Management and Scheduling, Master of Applied Science, 2006, Yifan Peng, Electrical and Computer Engineering Program with Specialization in Computer Networks, Ryerson University

Abstract

The proportional differentiated services model has been gaining attention in recent years. Many effective algorithms have been proposed to provide delay and packet loss differentiated services to different traffic classes according to their pre-defined constraints.

In this article, we propose three novel QoS mechanisms to enhance the control of class-based QoS. First, we introduce a Scheduling to Dropping feedback mechanism, which is called SDF model. The main usage of the SDF is to provide dynamic trade-off between different basic QoS metrics. SDF provide differentiated services based on both proportionality constraints and absolute constraints. We also introduces an additional feature called Adaptive Safety Margin (ASM). Usually, The Earlier Due Date (EDD) [7] employs a fixed safety margin to pre-assign more bandwidth for time-critical traffics. ASM can adaptively changes this margin according to the queuing length or packet head-drop rate of the time-critical traffics when the congestion happened. By employing ASM with SDF model, it is found that the total packet drop can greatly be reduced.

Finally, we introduce two sorting methods, called one-sort and complete-sort scheme, for the queuing systems in the end-to-end QoS environment. By sorting the order of the packets in the queue based on their deadlines, we can reduce the probability of end-to-end deadline violation.

Acknowledgment

I wish to acknowledge those who I feel have greatly aided me in completing this thesis.

I would like to thank Professor Bobby Ma for providing an open study and discussion environment, having faith in me through out the project, constant encouragement and willingness to provide advice, and very clear guidance towards the success of this thesis project. I would also like to thank Professor Lian Zhao for his enlightening guidance, constructive suggestions, high-standard requirement, and unconditional support.

Special thanks to my classmates, Xiaoli Li, Zhenning Xu, Xiaoyan Chen, and Jinyuan Sun, without whose help I would not have been able to get through this difficult and emotional time. Thanks to all friends who ever helped me over the past two years.

Special thanks go to my family especially my mom and dad, and sister's for their constant supports.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

In the last decade, Internet has become more and more important in our society. Today, Internet is not only used in research, work, or study, but also widely used in our daily life, such as reading news and weather forecast, checking the stock market, and even entertainment and shopping. With the rapid development of Internet, many new Internet applications have been developed. Different from the earlier applications, which mainly focus on file transmission, printer sharing, email exchange, or WEB browsing, the new applications are more related to multimedia services, such as Voice over IP (VoIP), Video on Demand (VoD), and multimedia conferencing. The multimedia applications have more restricted demands on the network services. Since Internet is a world wide sharing network with limited resources, how to provide better service for selective traffic classes becomes the topic of all Quality of Service (QoS) study.

In a network with relatively low traffic load, QoS is not a concern because most of the requirements of different traffic classes can be satisfied. However, as the load becomes higher, and the congestion occurs, some QoS tools must be applied to distribute network resources among different applications appropriately in order to satisfy the requirements of the more urgent traffic classes. Different traffic classes, based on their different applications, may ask for different requirements, such as high transmission bandwidth, low end-to-end delay, small

packet drop rate. Roughly speaking, there are two basic QoS Models. One of them focus on the absolute QoS guarantee of the transmission, called absolute QoS model. The other concerns about the relative QoS performance between different traffic classes, called relative QoS model.

1.2 Motivation and Methods Used

Recently, a refinement of relative QoS model, proportional differentiated service model [1] has been receiving a lot of attention. The proportional differentiated service model provides a predictable and controllable differentiated services between different traffic classes. Unfortunately, proportional differentiated services model is a relative model, it can not provide absolute QoS guarantee. In practice, the absolute guarantee requirements are more important than the relative ones in many cases. Because of that, new studies, such as [4] and [5], which will be introduced in the next chapter, work on the integration of both absolute QoS and proportional QoS models.

In the proportional services models, the proportional differentiated parameters for QoS metrics, such as delay and packet loss are independent from each other. In our model, we facilitate a tradeoff between delay and packet loss among different traffic classes. The tradeoff is achieved by applying a feedback from the scheduler to the packet dropper. The trade off mechanism is called Scheduling to Dropping Feedback (SDF). By using SDF the change of the delay ratio can affect the packet drop rate among different traffic classes. This may prevent one of more real-time traffic classes from consuming too much network resources during the congestion period.

In this thesis, we also propose a modification to Earlier Due Date (EDD) scheduling [7] called Adaptive Safety Margin (ASM) to enhance the QoS for the real-time traffic class. When the congestion occurs, ASM can adaptively change the ratio between safety margin

and the delay deadline of a real-time traffic class, according to the queuing length or packet head drop-rate of that class. This greatly cut down the packet drop-rate of the real-time traffic class.

Finally, in this thesis, we presents two algorithms to enhance end-to-end delay performance. These algorithms use the delay parameter carried by the packet for the transmission scheduling. The results show that these algorithms reduce the probability of delay deadline violation.

1.3 Thesis Organization

The rest of this thesis is structured as follows. Chapter 2 introduces some important QoS concepts and the existing QoS frameworks. It also introduces the proportional services model. In Chapter 3, we introduce the concept of SDF and ASM. We present the model of SDF with ASM and study the performance of the model through simulation. In Chapter 4, The end-to-end delay algorithms are presented and their performances are also studied. Chapter 5 gives the conclusion.

Chapter 2

Quality of Service

2.1 QoS Tools

The purpose of QoS is to provide appropriate treatments to different traffic flows or classes. The quality of QoS can be qualified by QoS metrics, such as delay, packet-drop rate, jitter, and etc.

There are many QoS tools that can be used to provide QoS. These tools include scheduling algorithms, packet dropping algorithms, traffic shaping, and traffic policing. We will give a brief description of these tools in this section.

2.1.1 Scheduling algorithms

An important QoS metric is delay. Delay is the accumulated latency of traffic in a network. You can measure delay by adding up all of the individual contributing components. An overall delay consists of many components, including packetization delay, queuing delay, serialization delay, network delay, and de-jitter buffer delay. In this thesis, we mainly concentrate on queuing delay.

Usually, scheduling algorithm is used for the delay tradeoff among different traffic classes. A network node uses the scheduling algorithm to control the order in which packets are sent

out. Scheduling entails the creation of queues, assignment of packets to those queues based on the classification of the packet, and scheduling of the packets in a queue for transmission. There are many scheduling algorithms, each of which allows the network node to provides greater or lesser degrees of differentiation of traffic.

The most common scheduling algorithms are First In First Out (FIFO), Weighted fair queueing (WFQ), Custom Queueing (CQ), and Priority Queueing (PQ). FIFO has no concept of priority or classes of traffic. In FIFO, the packet arrives to the node first will be transmitted first. Both WFQ and CQ allow the bandwidth to be shared fairly among traffic flows or classes. In particular, the minimum bandwidth allocated to each flow/class can be specified. With PQ, packets belonging to a higher priority class of traffic are sent before packets belonging to lower priority classes to ensure timely delivery of the higher priority packets.

2.1.2 Packet Dropping Algorithms

Packets may be dropped by a node due to many reasons, such as not enough buffer space or congestion avoidance. The three most common used dropping algorithms are briefly described below.

Tail drop

This is the default congestion avoidance behavior. Tail drop treats all traffic equally and does not differentiate among classes. Queuing buffer will fill during periods of congestion. Tail drop is ineffective when the queuing buffer is full, packets are dropped until the congestion is cleared and the queue is no longer full.

Head drop

Generally a packet that reaches the head of a queue should enter the service process and be sent to the next hop and destination. Head drop takes effect when a packet at the head is dropped instead of being sent. Unlike tail drop, packet head drop will not happen automatically. It is activated by some predefined trigger. For instance the queuing system will use head drop to drop the packet of a real-time traffic if the delay of that packet exceeds the pre-defined delay deadline.

Random Early Detection

Random Early Detection (RED) [6] is aimed to control the average queue size by indicating to the end hosts when they should temporarily slow down the transmission rate of packets. RED takes advantage of the congestion control mechanism of TCP. By randomly dropping packets prior to periods of high congestion, RED tells the packet source to slow-down. Assuming the packet source is using TCP, it will reduce its congestion window size, thus decreasing the transmit rate. RED maintains low queue depth while absorbing traffic spikes occasionally.

The packet drop probability of RED is based on the minimum threshold, maximum threshold, and mark probability denominator. When the average queue depth is above the minimum threshold, RED starts dropping packets with certain probability. The probability of packet drop increases linearly as the average queue size increases until the average queue size reaches the maximum threshold, in which case, the probability of packet drop will be 100%.

The RED algorithm is used as the drop controller in our thesis.

2.1.3 Traffic Policing

Traffic policing is used to control the maximum transmitting or receiving data rate of a traffic. If the traffic rate is higher than the maximum limit, the system can choose to drop or mark the excessive data.

2.1.4 Traffic Shaping

Traffic shaping is used to control the out-going traffic in order to match the speed of the remote target and to ensure that the traffic conforms to policies contracted for it. By using traffic shaping, the network can reduce the probability of congestion and eliminate bottlenecks in topologies with data-rate mismatches.

2.2 Internet QoS Frameworks

In current Internet, two common QoS frameworks has been widely adopted. They are Integrated Services model and Differentiated Services model.

2.2.1 Integrated Services Model

Integrated services model (IntServ) provides absolute QoS guarantee. In Integrated Services, traffic is classified into a number of flows with different QoS assigned to each flow. Essentially, IntServ Provides absolute QoS to individual traffic flow. The absolute QoS (also called hard QoS) is expressed as an absolute bound.

The IntServ is flow-based. Because flow-based QoS works on each traffic flow that pass through a network node, service behavior is more predicable and more precise. However, one of the main problem in flow-based approach is that it is not very scalable as the size of the network increases. To provide flow-based QoS guarantee with many traffic flows, a great

deal of resources (CPU and Buffers) are required for classification and scheduling.

To provide the support of the absolute QoS, networks resources for a flow must be reserved before the transmission takes place. The reservation is made by using Resource Reservation Protocol (RSVP). Through RSVP, the application provides the network its traffic profile and requests a particular kind of service that encompasses its bandwidth and delay requirements. The application is expected to send data only after the reservation is successful along the path of the flow.

During the reservation, each network node along the path of the traffic flow performs admission control. Once the resources are successfully reserved in each node of the path, the network essentially commits to meet the QoS requirements of the application as long as the traffic remains within the profile specifications. The network fulfills its commitment by maintaining per-flow state and then performing packet classification, policing, and intelligent queueing based on that state.

2.2.2 Differentiated Services Model

Differentiated Services Model (DiffServ) provides class-based per-hop relative service. One of the main difference between DiffServ and IntServ is that in DiffServ, a number of similar traffic flows could be grouped into one traffic class. All the traffic flows belonging to the same class receive the same QoS treatment. Since core routers only need to deal with small number of traffic classes, this service is more scalable than the flow-based approach in IntServ, and thus, best suited for large ISP networks.

DiffServ also provides relative QoS instead of absolute QoS. In relative QoS, some traffic flows or traffic classes are treated better than others, for example, more bandwidth, lower average delay and less average packet drop-rate. However the better treatment is relative,

not a absolute one.

2.3 Proportional Differentiated Services

Even Differentiated Services model is widely applied in practice for its flexibility and scalability, it has some intrinsic problems. In particular, DiffServ can not provide consistent class differentiation. The differentiation varied with the traffic load. Generally, the traffic class with better QoS should receive better service than the one with lower QoS. However, in some situations, the class with lower assigned QoS may actually receive better service than the one with better QoS.

For example, consider two traffic classes, class i and class j. Suppose $10Mbps$ and $5Mbps$ of bandwidthes are allocated to class i and class j, respectively. Assume both class i and class j have the same traffic load, then class i in general will receive a better service than class j. However if class j has less than half of a traffic load of class i, it actually will receive a better service than class i.

By using the priority scheduling scheme, the service to the higher priority class can be made always better than that of the lower priority class. However, it is hard to derive a consistent relative QoS between them. In generally, the differentiation of delay and packet loss rate between different classes is hard to predict.

Recent study, like Proportional Differentiated Services model ([1]), further refines and quantifies the differentiation model. This model is aimed to provide the network operator with the "tuning knobs" for adjusting the quality spacing between classes, independent of the class loads. The primary goal of proportional differentiated services is to make the model more predictable and more controllable. Predictable means that the differentiation should be consistently independent of the class load variations, i.e., higher priority classes always

receive better service; and Controllable means that the network operators should be able to adjust the QoS differentiation between classes based on their criteria.

In the proportional differentiation services model, certain QoS metrics among traffic classes (e.g. delay or packet loss rate) are set to be proportional to the predefined differentiation parameters. Suppose we measure a QoS metric of a certain traffic class (class i) in a queuing system, during the time interval from time t to $t+\tau$. Let $\bar{q}_i(t+\tau)$ is the average metric value of class i measured during that time interval. The proportional differentiation model imposes a proportional constraint that for all classes (such as class i and class j) belong to a queuing system, in all time intervals, we have

$$\frac{\bar{q}_i(t,t+\tau)}{\bar{q}_j(t,t+\tau)} = \frac{c_i}{c_j} \tag{2.1}$$

where c_i and c_j are the predefined Quality Differentiation Parameters(QDPs). The proportional differentiated services model is predictable that, even though the actual quality level of each class will vary with the class loads, the quality ratio between classes will remain fixed. This model is also controllable because the predefined parameters can easily be adjusted by the network operator.

The proportional differentiation model can be used to control the proportionality of a QoS metric, such as delay or the packet loss rate, among all traffic classes in one queuing system.

One of the main design issues of the proportional differentiated service model is the choice of τ. A small τ or large τ lead to short-term service proportionality or long-term service proportionality, respectively. Unfortunately, short-term proportionality does not guarantee long-term proportionality or vice versa.

2.3.1 Proportional Delay Models

When the controlling object is delay, Eqn. 2.1 can be rewritten as

$$\frac{\bar{d}_i(t, t+\tau)}{\bar{d}_j(t, t+\tau)} = \frac{\delta_i}{\delta_j} \tag{2.2}$$

where $\bar{d}_i(t+\tau)$ and $\bar{d}_j(t+\tau)$ are the average delay values of class i and class j measured in the time interval, t to $t+\tau$, while δ_i and δ_j denote the predefined Delay Differentiation Parameters (DDPs).

Considering different measured time-frames, the authors in [2] proposed a long-term measurement proportional delay model PAD (Per-class Average Delay), a short-term measurement model WTP (Waiting Time Priorities), and a mixed model HPD (Hybrid Proportional Delay).

PAD Model

In PAD model, the average delay is measured from the beginning up to the most current time. Thus, the delay proportionality between class i and class j is governed by the following equation,

$$\frac{\bar{d}_i(0, \tau)}{\bar{d}_j(0, \tau)} = \frac{\delta_i}{\delta_j} \tag{2.3}$$

where τ represents the period from the start to the most current time, inclusively. When a PAD scheduler decides to choose a packet to sent among all classes, it will first check the average queuing delay of each class. The class with the maximum normalized average delay will be chosen for transmission. The normalize average delay of a class is the average delay of the class divided by its DDP,

$$\tilde{d}_i(0,\tau) = \frac{\bar{d}_i(0,\tau)}{\delta_i} \tag{2.4}$$

where $\tilde{d}_i$ denotes the normalized average delay of class i. Assume there are totally N traffic classes, the mechanism of the PAD scheduler can be illustrated by the following pseudo code,

```
d̃_max = 0;

For traffic class i <= N
  {
  Calculate d̃_i = d̄_i/δ_i
  If d̃_i > d̃_max
    {
    d̃_max = d̃_i;
    winner = i;
    }
  }

Send the head packet from class winner
Update the delay of class winner
```

WTP Model

Unlike the PAD model that uses the long-term average delay measurement for scheduling, WTP uses the queuing delay of the head packet of each class. The delay proportionality between class i and class j in the WTP model is governed by the following equation,

$$\frac{d_i(\tau_0)}{d_j(\tau_0)} = \frac{\delta_i}{\delta_j} \tag{2.5}$$

Where τ_0 represents the current time and $d_i(\tau_0)$ is the delay of the head packet in class i at time τ_0. When a WTP selects a packet to sent, it merely checks the queuing delay of the

head packet of each class. Then calculate the normalized delay. The one with maximum value will win the competition. Similar to Eqn. 2.4, we define the normalized delay as,

$$\tilde{d}_i(\tau_0) = \frac{d_i(\tau_0)}{\delta_i} \tag{2.6}$$

Assume there are totally N traffic classes, the operation mechanism of the WTP scheduler can be described by the following code,

$\tilde{d}_{max} = 0;$

For traffic class $i <= N$
{
Calculate $\tilde{d}_i = d_i/\delta_i$
If $\tilde{d}_i > \tilde{d}_{max}$
{
$\tilde{d}_{max} = \tilde{d}_i;$
$winner = i;$
}
}

Send the head packet from class *winner*

HPD Model

As described in [2], PAD attempts to minimize the differences between normalized average queuing delays. Thus, it can not react quickly to the proportional change among classes. On the other hand, WTP attempts to minimize the differences between normalized head packet queuing delays. It react to load change quickly. However, because of that, it is not stable since it ignores the effects of the past packets. So the authors combine the operation of both models. The new delay model is called Hybrid Proportional Delay (HPD). In HPD the normalized combined delay is the function of the normalized average delay and the normalized most current delay,

$$\tilde{d}_i^{HPD} = g\tilde{d}_i^{PAD} + (1-g)\tilde{d}_i^{WTP} \tag{2.7}$$

Where $\tilde{d}_i^{PAD}$ and $\tilde{d}_i^{WTP}$ are the normalized delays calculated using Eqn. 2.4 and Eqn. 2.6, respectively, while g is the HPD parameter ($0 \leq g \leq 1$). The paper also suggested to use the value $g = 0.875$ to provide a balanced tradeoff between the behaviors of PAD and WTP.

2.3.2 Proportional Packet Loss-rate Models

If the controlled object is packet loss rate, Eqn. 2.1 can be rewritten to

$$\frac{\bar{l}_i(t, t+\tau)}{\bar{l}_j(t, t+\tau)} = \frac{\sigma_i}{\sigma_j} \tag{2.8}$$

where $\bar{l}_i(t, t+\tau)$ or $\bar{l}_j(t, t+\tau)$ is the average loss-rate value of class i or class j in the time interval $(t, t+\tau)$, respectively. σ_i and σ_j denote the predefined Loss-rate Differentiation Parameters (LDPs).

Similar to the proportional delay model, [3] also proposed a long-term measurement proportional loss-rate model PLR(∞) (Proportional Loss-Rate Infinity) and a short-term measurement model PLR(M) (Proportional Loss-Rate M).

In the PLR(∞) model, the average packet loss-rate value of each class is calculated by dividing total dropped packet number by total arrival packet number. Since it considers all past packets, PLR (∞) has the same characteristic issue as the PAD has. The PLR (∞) equation can be written as,

$$\frac{\bar{l}_i(0, \tau)}{\bar{l}_j(0, \tau)} = \frac{\sigma_i}{\sigma_j} \tag{2.9}$$

Where $(0, \tau)$ represents the period that from the start to the most current time, exclusively. The normalized average packet loss rate is,

$$\tilde{l}_i = \frac{\bar{l}_i}{\sigma_i} \tag{2.10}$$

The class with the lowest normalized average packet loss rate will lose the competition and a packet of that class will be dropped. Assume there are totally N traffic classes, the operation mechanism of the PAD scheduler can be represented by the following code.

```
l̃_min = 1.0;

For traffic class i <= N
  {
  Calculate l̃_i = l̄_i/σ_i
  If l̃_i < l̃_min
    {
    l̃_min = l̃_i;
    failure = i;
    }
  }

Drop a packet from class failure
Update the packet-loss rate of class failure
```

PLR(M) Model

Just like the PAD model, the PLR(∞) model does not respond quickly to the current proportional change among classes. To overcome this issue, PLR(M) [3] calculates the packet loss rate based on the most recent M arrival packets, instead of using all packets in the past. M here is the history packet number, which determines the size of the Loss History Table (LHT). LHT is used to calculate the average packet drop-rate among M packets. In LHT,

the final status (buffered of dropped) of the most recent M packets in each class are recorded. The LHT can be adjusted by the network operator on demand. By choosing an appropriate M value, the PLR(M) algorithm can achieve fast response with stability.

2.4 Integrated Approach

Proportional differentiation models provides effective tuning knobs on adjusting differentiated services. Based on the proportional differentiation model, many researches proposed different approaches to try to improve and supplement the original model. For proportional delay, Nandagopal et al. [12] presented a core-stateless quality of service architecture between flows. Leung et al. [13] provided a waiting time "spacing" which can be controlled by the ISP among different traffic classes. For proportional packet loss, Bodin et al. [14] modified the measurement method of proportional packet drop by using average drop distances (ADDs). Zeng et al. [17] investigated the "packet shortage" phenomenon and try to enhance the dropping scheme. To provide an integrated proportional services for both delay and packet loss, Vuong et al. [15] defined a new relative differentiated services, and determined a break-point rate to improve the drop-rate differentiation algorithm, and tried to avoid traffic starvation. Li et al. [16] employed probabilistic longest queue first (PLQ) mechanism, which reacts effectively to queue-length fluctuation of each class. Li et al. [18] investigated a flow-based service in class-based routers.

However, all these models are based on proportional differentiated guarantee services, and do not incorporate with absolute guarantee services. In practice, it is desirable to integrate these two services and offer an unified approach. Two such approaches found in the literature are briefly described below.

2.4.1 Joint Buffer Management and Scheduling

JoBS (Joint Buffer Management and Scheduling) [4] provides relative and absolute per-class QoS guarantees on two QoS metrics (delay and packet loss rate). If both absolute QoS guarantees and the proportional QoS guarantees can not be satisfied simultaneously, JoBS applied a method called constraint relaxation. The constraint relaxation selectively ignores some of the proportional QoS guarantees for they are usually not as cirtical as absolute constraints.

The JoBS algorithm also integrates scheduling and buffer management into a single scheme. It predicts the delay value of buffered packets. The predictions are used to allocate the appropriate differentiated services among classes in the scheduler, and decide the amount of packets to be dropped in the drop controller.

2.4.2 Uniform and Practical Solution

Another unified solution that combines both the class-based proportional QoS guarantees and absolute QoS guarantees is presented in [5]. Comparing with the JoBS model, this solution is much simpler and requires less computation. Consequently, it is more practical.

In this approach, the PLR (∞) model is employed as the dropper at the entrance of the queuing system. The WTP model is employed as the scheduler at the exit of the queuing system. In order to deal with real-time traffic, Early Due Date (EDD) is employed.

Early Due Date Algorithm

If the delay of a packet belong to a time-critical traffic class has reached the delay deadline, the expiration of the packet is considered as a packet loss, the traffic is considered in "congestion mode". In practice, it is better to pre-assign more bandwidth to the class to

prevent the class entering the "congestion mode". EDD [7] algorithm uses the concept of safety margin such that the system can assigns more bandwidth to the congested class.

Assume the delay deadline of class i is D_i, EDD define a safety margin Δ_i, where $\Delta_i/D_i = 1/10$. If the queuing delay of a class i packet is already higher than $D_i - \Delta_i$ but less than D_i, this packet is scheduled to transmit directly, thus class i essentially gains more bandwidth in the short-term to avoid congestion.

If the delay of a packet exceeds its deadline, thus larger than D_i, the packet is dropped to save the system bandwidth. This mechanism is called Shortest Time to Extinction (STE).

Chapter 3

Scheduler to Dropper Feedback and Adaptive Safety Margin

3.1 Scheduling to Dropping Feedback

In the general proportional differentiated services model, the pre-defined Loss-rate Differentiated Parameters (LDPs) and the Delay Differentiated Parameters (DDPs) of the traffic classes are fixed. The parameters are defined based on the characteristics of each traffic class. For instance, some traffic classes such as real-time traffic are delay sensitive, so a lower DDP should be used. Some other traffic classes like UDP traffic of some special applications are drop-rate sensitive. Consequently, a lower LDP should be used.

As described in section 2.3, Proportional differentiation services model provides relative QoS. In practice, the absolute QoS is more critical for some applications. For example, absolute delay bound is very important for a time-critical application. If both proportional and absolute QoS are employed in one queuing system at the same time, A conflict of satisfying both absolute and proportional QoS requirements may arise.

3.1.1 The Conflict between Absolute and Relative QoS

Since the absolute constraint usually has a higher precedence than the proportional constrains, the proportional constrains are often relaxed when both constrains can not be satis-

fied at the same time. During the time when the proportional constrains are relaxed, classes with absolute constraints can actually take an advantage over other classes. The following example illustrates the point.

Assume we have a general proportional differentiated services queuing system. There are two traffic classes class i and class j in this queuing system. Class i is a delay-sensitive traffic. It has an absolute delay deadline of $20msec$, while class j doesn't. Initially, the DDPs distributed to class i and class j are 1 and 2, respectively. Thus, the delay ratio between the two classes is $1 : 2$.

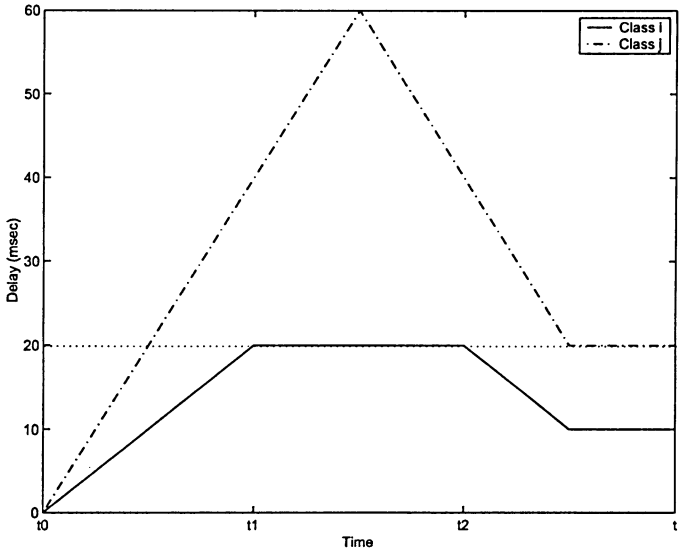

Figure 3.1: Delay Example (deadline of class $i = 20msec$)

Suppose in the interval (t_1, t_2), the system is congested. Without the absolute constraint imposed on traffic i, the delays for both classes increase proportionally with the same ratio of $1 : 2$. With the absolute constraint, class i's delay will be bounded by the absolute constraint. It means that more resource is allocated to class i. Consequently, class j will suffer higher delay (Figure 3.1) and the delay ratio is no longer $1 : 2$ (Figure 3.2).

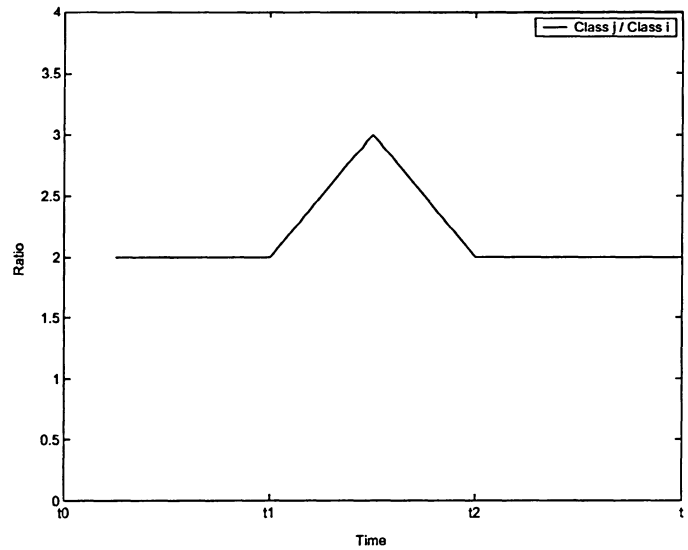

Figure 3.2: Delay Ratio Example

In this case, class i is always taking advantage from the system when there is a congestion. Both the JoBS model and the Uniform and Practical Solution model provide some solutions to integrate the absolute QoS guarantee and the proportional differentiated QoS guarantee. In this thesis, we propose a method that the queuing system will measure the change of delays of class i and class j. The result is then feedbacked to the dropper. The dropper penalizes class i by slightly increasing the LDP of class i. The result is that more packets from class i will be dropped, thus preventing class i to consume too much bandwidth.

3.1.2 Scheduling to Dropping Feedback Algorithm (SDF)

In a queuing system that supports different traffic classes, the total resource allocation can be reflected by several type of metrics. They are delay metric, packet loss metric, jitter metric, or other QoS metrics (we only consider the delay and the packet-loss rate in this thesis). Each traffic class belonging to a queuing system will be allocated a portion of queuing resources. How much resources each class consumes can be reflected by delay and packet loss.

The proportional differentiated services model is the mechanism which measures the QoS metrics of each traffic classes, and compares them with the predefined QDPs (Quality Differentiation Parameters) to determine which class is selected for transmission or packet drop. In our study, it is more convenient to use weight to replace QDP. Weight is inversely proportional to QDP. For delay, we define the normalized packet delay weight as,

$$w_{di} = \frac{1}{\delta_i} / \sum_{i=0}^{n-1} \frac{1}{\delta_i} \tag{3.1}$$

Similarly, the normalized packet-loss weight w_{li} is defined as,

$$w_{li} = \frac{1}{\sigma_i} / \sum_{i=0}^{n-1} \frac{1}{\sigma_i} \tag{3.2}$$

In the previous example where $\delta_i = 1$ and $\delta_j = 2$, the designed normalized delay weights of class i and class j are $\frac{2}{3}$ and $\frac{1}{3}$, respectively.

The two metrics, delay and packet loss are often treated independently in most proportional models. However, they are inter-related. When a packet arrives at a queuing system, the system has two choices: dropping this packet or keeping it to be transmitted later. If the packet is dropped, the packet-loss rate will increase but the bandwidth is saved. The extra bandwidth may reduce the delay of other queued packet. Conversely, keeping the arrived packet cuts down the total loss rate but will probably cause a larger overall delay.

As discussed in section 3.1.1, with the use of the absolute delay constraint, the time-critical traffic classes may take the advantage on delay. In order to provide some fairness among different traffic classes, here we proposed a feedback mechanism called Scheduling to Dropping Feedback (SDF). In this mechanism, the queuing system first checks the current delay values of all traffic classes, and then calculates the deviation from the designed delay weights. The

deviation of delay weight of class i, Δw_{di} can be determined by the following equation,

$$\Delta w_{di} = \hat{w}_{di} - w_{di} \tag{3.3}$$

where $\hat{w}_{di}$ is the measured normalized delay weight and is given by

$$w_{di} = \frac{1}{d_i} / \sum_{i=0}^{n-1} \frac{1}{d_i} \tag{3.4}$$

If class i takes the advantage of the absolute delay constraint, and thus resulting a proportionally smaller delay, Δw_{di} will be negative. The SDF mechanism uses Δw_{di} to change the packet-loss weight of class i. SDF calculates the packet-loss weight change, Δw_{li} using the following relation,

$$\frac{\Delta w_{li}}{w_{li}} = -\frac{\Delta w_{di}}{w_{di}} \tag{3.5}$$

The new loss weight $\hat{w}_{li}$ for class i can be calculated by,

$$\begin{aligned} \hat{w}_{li} &= w_{li} + \Delta w_{li} \\ &= w_{li} - \frac{w_{li}}{w_{di}} \cdot (\hat{w}_{di} - w_{di}) \end{aligned} \tag{3.6}$$

The detail of packet-loss weight calculation will be presented in section 3.1.4.

3.1.3 Feasibility of SDF

There are three reasons that make the SDF mechanism desirable.

First, it makes the system fair. By applying SDF algorithm, the queuing system will not overly favors a particular traffic class. Any traffic class taking an advantage of delay is penalized on packet loss.

Second, by using the feedback, the system can reduce the amount of traffic from the class with absolute constraint entering the system. Since scheduling these packets usually forces the queuing system to relax the proportional constraint, the feedback may alleviate the degree of deviation from the proportional constraints.

Third, real-time traffic, such as voice or video, can tolerate certain degree of packet loss. Thus, slightly increasing the packet-loss rate of the real-time traffic may not significantly affect the overall quality of such traffic. At the same time, the feedback will reduce the packet-loss rate of other traffic, especially data traffic, which is usually loss-sensitive.

In our model, the feedback happens from the scheduler to the dropper. Thereby, the deviation of delay weight affects the weight distribution of packet loss. Conceptually, this can be used in other direction: from packet loss to delay. However, we found that it is very difficult to employ both direction simultaneously, since the weight distribution of delay and packet loss will affect each other and causes serious diversion from their original settings.

For example, consider two traffic classes, class i and class j and suppose the feedback is used in both directions. Class i is a real-time traffic class and has a delay deadline. Initially, the delay weight distribution for class i and class j are $2:1$, and the weight distribution of packet loss are $1:2$. In the case when congestion happens, since class i has an absolute delay deadline, it takes the advantage on delay. Assume that in a time the measurement delay weight ratio between class i and class j has become $3:1$, by the effect of the feedback from delay to packet loss, the packet-loss weight distribution becomes $1:3$. However, the new packet loss ratio will in turn affect the delay weight by the feedback of another direc-

tion. This interaction will continue until a new balance is achieved. The new proportional values will be totally different from the original one. The system will lose its proportional guarantees.

If we still want to use the bi-directional feedback, some new components must be added to force the delay and packet-loss weights back to their original settings after the diversion. However, this will make the model become too complicated and much tougher to control, and it is not desirable in current situation. Consequently, we will only study unidirectional feedback. Because delay constraint is much more critical than packet loss constraint for the real-time traffic, only delay to packet loss feedback is studied in this thesis.

In additional, the scheduling to dropping feedback can cause the packet loss of time-critical traffic class become very high in some cases such as the delay deadline is set too low or there is too much traffic load. If the time-critical traffic class has a packet loss limit, when this limit is reached, other mechanisms can be apply to control also the packet loss. These mechanisms will not be discussed in this thesis.

3.1.4 The SDF Model

The Figure 3.3 shows the structure of SDF. Based on the functionality, the SDF model can be decomposed into two components: the arrival component and the departure component. The arrival component processes on the arrival packets. It can affect the packet loss. The departure component controls the transmission of the packets in the queue. It mainly affects the traffic delay. It can also affect the packet loss when head-drop algorithm is used.

Mechanism of the Arrival Component

When a new packet arrives at the queuing system, the classifier first classifies which traffic class that this packet belongs to. The system then puts this packet to its corresponding

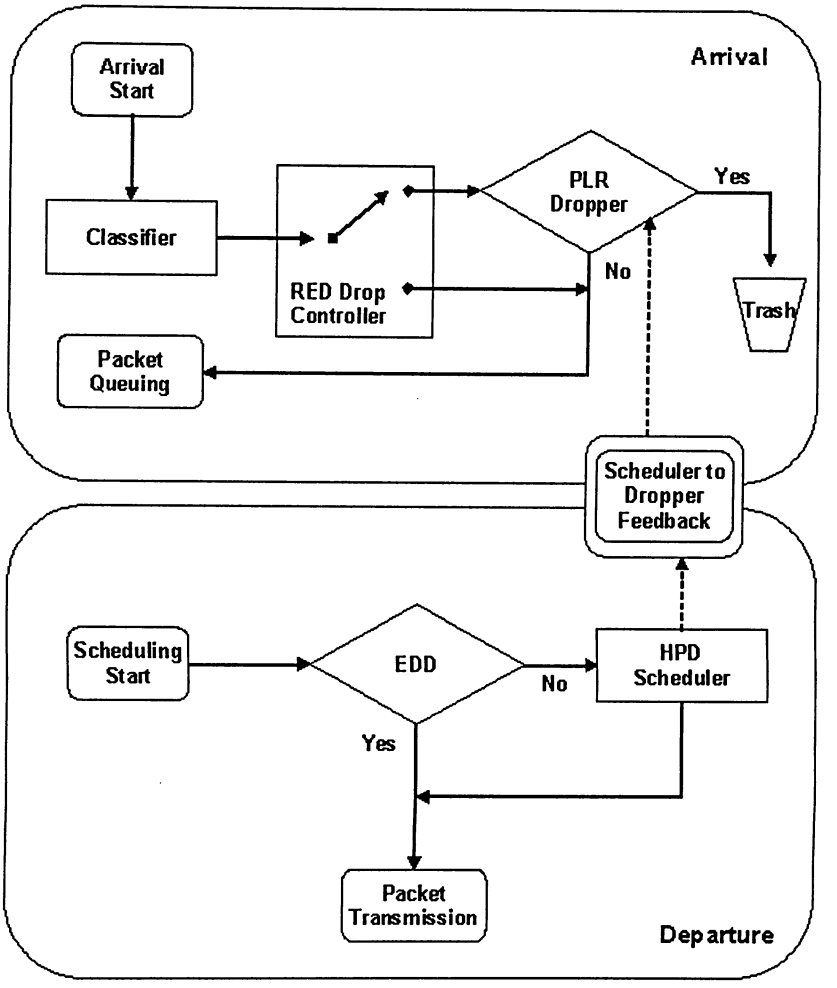

Figure 3.3: SDF Model

sub-queue, which is a virtual queue. After that, the system will check with the dropping process to decide if it has to drop any packet.

The dropping process contains two components, a drop controller and a dropper. The drop controller decides whether or not the system should drop a packet. If a drop is required, the dropper will select a traffic class, from which a packet is dropped.

The drop controller in our model uses the RED algorithm. As introduced in section 2.1.2, RED randomly drops packets prior to periods of high congestion. If the RED drop controller decides to drop a packet, the dropper will select a buffered packet among the traffic classes. The SDF model employs the PLR(M) (section 2.3.2) algorithm for the dropper. PLR(M) compares the normalized packet-loss rate of all classes. The class with the lowest value is selected, and a packet from that class will be dropped. If this class currently has no buffered packet, the class with next lowest packet-loss rate will be selected.

Mechanism of the Departure Component

The departure component consists EDD, and PAD scheduler. The EDD algorithm is used here to accelerate the transmission of urgent packets and cut down the head-drop rate for the real-time traffic class. If no classes are in congestion mode, the scheduler calculates the normalized delay values of all classes by using the HPD algorithm. These values are compared among different classes. The class with the highest normalized HPD delay value will win the competition and its head packet is sent out by the queuing system.

Scheduling to Dropping Feedback

The SDF model consists of two steps. The first step is to adjust the loss weight of the real-time traffic class by using the delay-feedback information. The second step is to assign the loss weights of the remaining classes.

The following example illustrates the SDF algorithm, Let's consider three traffic classes, classes 0, 1, and 2. Among the three classes, class 0 is the real-time traffic class, which has a delay deadline. The other two classes, class 1 and class 2 are traffic classes that do not have a delay deadline. The deadline and weights are listed in Table 3.1.

Table 3.1: Delay and Packet Loss Weight (Example)

Class	Designed Delay Weight	Measured Delay Weight	Designed Packet-loss Weight	New Packet-loss Weight
0	2/5	3/6	1/4	3/24
1	2/5	2/6	1/4	7/24
2	1/5	1/6	2/4	14/24

Introduces in section 3.1.4, when the dropper PLR(M) is going to drop a packet, the SDF process first checks the current delay value of all the three classes, which are the HPD delay value in this model. The designed normalized delay weight among class 0, class 1, and class 2 are $\frac{2}{5} : \frac{2}{5} : \frac{1}{5}$ while the designed normalized packet-loss weight among them are $\frac{1}{4} : \frac{1}{4} : \frac{2}{4}$. Assume that in congestion period, the measured normalized delay weight of class 0 becomes $\frac{3}{6}$. Thus, the delay weight deviation of class 0, $\Delta w_{d0} = \frac{1}{10}$. By using Eqn. 3.6, we can calculate the new packet-loss weight of class 0, w_{l0}. The new packet-loss weight is $w_{l0} = \frac{3}{24}$. Likewise, the new packet-loss weights of class 1 and class 2 are $\frac{7}{24}$ and $\frac{14}{24}$ respectively. The packet-loss weight ratio between class 0 and class 1 (and class 2) is still 1 : 2. Therefore in the SDF model, a traffic class will relax its proportional constraint only if the proportional constraint conflicting with the absolute constraint. The QoS among other traffic classes will remain in proportion.

3.2 Adaptive Safety Margin (ASM)

In 2.4.2, we introduce EDD to enhance the delay performance of time-critical traffic class. Recall in EDD, if the delay of a packet exceeds deadline Δ, where Δ is the safety margin, the packet will be scheduled to transmit immediately. By using this approach, we can reduce the delay of the time-critical traffic. Note that in EDD, Δ is fixed and is equal to 1/10 of the deadline.

3.2.1 ASM Algorithm

In this thesis, we propose an approach that dynamically adjusts Δ based on the traffic condition. The reason of this approach is that by adjusting the value of Δ, we can control the amount of additional bandwidth assigned to a traffic class. For example if Δ of a traffic class is set to 2/10 of the deadline, more packet will be scheduled to transmit directly without affected by the underlying scheduler. This is equivalent to allocate more bandwidth to the traffic class. By using the concept of adaptive safety margin, we could improve the delay performance of the delay sensitive traffic class.

First, we define $\tilde{\Delta}$, where $\tilde{\Delta} = \Delta/deadline$, as the normalized Δ with respect to the deadline. In this thesis, we propose a simple piecewise linear function for $\tilde{\Delta}$, as shown in Figure 3.4. When the congestion level is below the minimum threshold, $\tilde{\Delta} = \tilde{\Delta}_{min}$. As the congestion level between minimum and maximum thresholds, $\tilde{\Delta}$ increases linearly. Finally when the level exceeds the maximum threshold, $\tilde{\Delta} = \tilde{\Delta}_{max}$. We investigate two methods to measure the congestion level. In the first method, the average queue length is used. In the second method, the measurement is based on the head-drop rate.

If the queue-length is used to measure the congestion level, different time-critical traffic classes may have a different $\tilde{\Delta}$ function. For example, consider two traffic classes 1 and 2. If the delay of traffic class 1 has a smaller deadline, its minimum and maximum thresholds

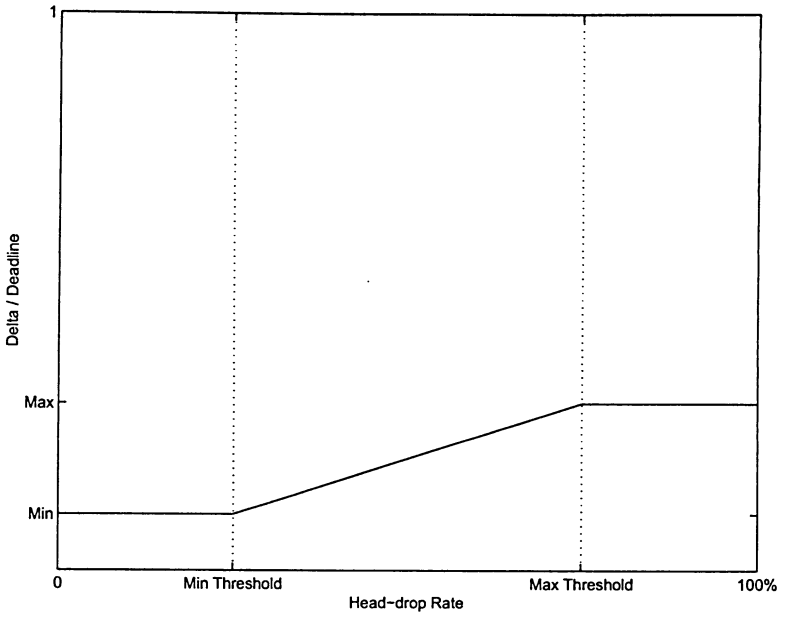

Figure 3.4: ASM

should be smaller than those of class 2. In addition, $\tilde{\Delta}_{max}$ and $\tilde{\Delta}_{min}$ could also be different. The system can use different $\tilde{\Delta}_{min}$ and $\tilde{\Delta}_{max}$ values for class 1 and class 2. The second method is more desirable, because the use of head-drop rate gives a more consistent view of congestion and the measurement is not affected by the deadline parameter. Consequently, we can use the same $\tilde{\Delta}$ function for all traffic class based on their packet head-drop rate. The ASM with head-drop rate method is much simpler for it does not require the administrator to assign different minimum and maximum threshold pairs to each traffic class. However, it can only be used while STE also exists. Without STE, there is no head drop at all. In this case, the ASM based on queue length measurement should be used. The algorithm ASM with head-drop rate is described by the following pseudo code,

Assume the parameters of ASM algorithm are list as below,
- minimum threshold = min_t
- maximum threshold = max_t
- maximum $\tilde{\Delta} = max_d$

The algorithm is performed when the scheduler is scheduling a packet transmission

if head-drop rate $\leq min_t$
  $\tilde{\Delta} = 0.1$
else if $min_t \leq$ head-drop rate $\leq max_t$
  $\tilde{\Delta}$ = head-drop rate$\cdot(max_d - 0.1)/(max_t - min_t)$
else
  $\tilde{\Delta} = max_d$

if $\tilde{\Delta} \leq$ (queuing delay/deadline) ≤ 1
  EDD schedules to transmit this packet directly

3.2.2 Integration of SDF and ASM

SDF is a feedback mechanism. ASM is a feedforward mechanism. It seems that it is beneficial to integrate both mechanisms. On one hand, ASM can cut down the queuing delay of time-critical traffic classes by adaptively assigning more bandwidth. On the other hand, because these classes take advantage on queueing delay, SDF will increase their packet-loss weights in the dropper, which causes more tail drop in the time-critical classes. This may cut down the total queue length. Because RED drops less packets with short queue-length, the total packet loss of the whole queuing system may decrease.

The integrated SDF&ASM model also includes the STE mechanism, the STE mechanism is applied to drop real-time traffic packets whose delays exceed the deadline. This will prevent the unnecessary transmission of expired packets, thus saving bandwidth.

The Figure 3.5 shows the integrated model, where an ASM feedforword block is added

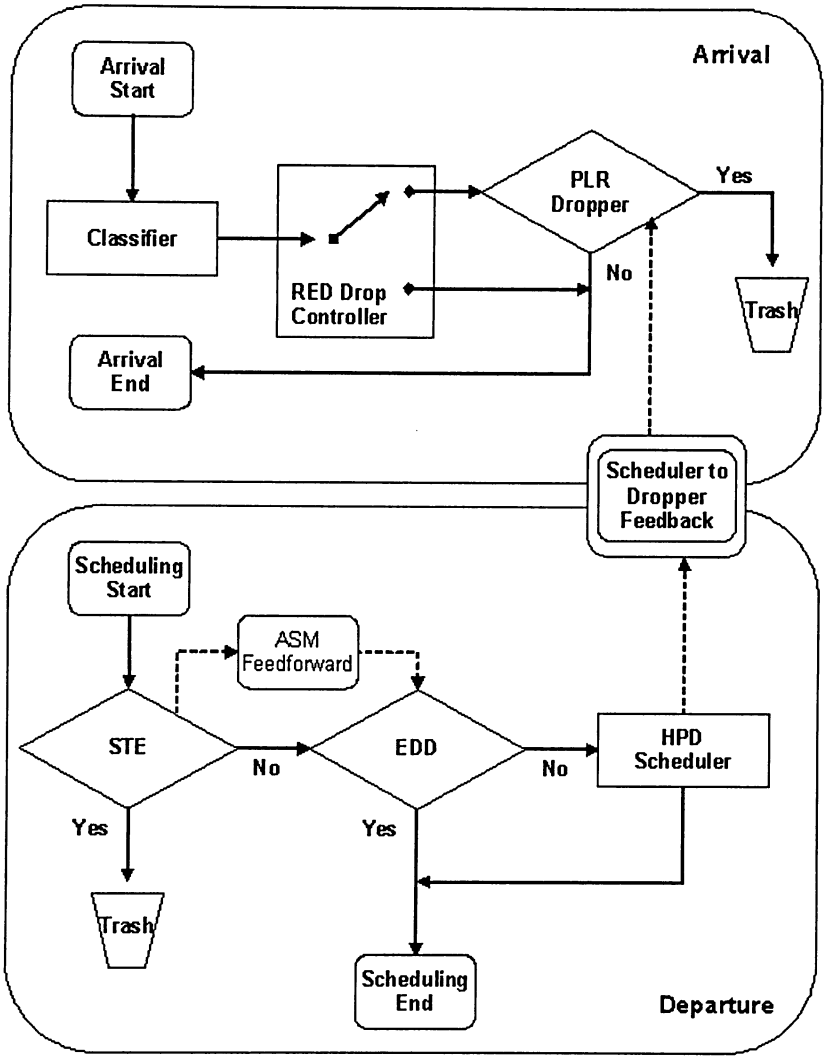

Figure 3.5: SDF&ASM Model

to the original SDF model. The safety margin used in EDD of the SDF&ASM model is not fixed anymore. As the introduction in section 3.2, the new safety margin value is calculated based on the queue length or the head-drop rate.

3.3 Simulation

In this section, we will study the performance of our proposed algorithms: SDF and SDF&ASM, through simulation. There are two simulation scenarios. The first scenario illustrates the effectiveness of SDF. In this scenario, delay deadline is set for the real-time traffic class. The packet loss of each traffic class are compared under two situations, with and without SDF. In the second scenario, the performance of packet-head-drop rate of the real-time traffic and the total packet-loss rate of all traffic classes are studied.

Both scenarios use the same topology as shown in Figure 3.6. In our simulation setup, the simulation duration is 600 seconds. There are 3 traffic classes, classes 0, 1, and 2. For class i ($i = 0, 1, 2$), packets are generated by source i and source i'. During the simulation period, source i continues sending packets from the beginning to the end, but source i' only transmits packets in the period between 180 sec and 420 sec. This creates a congestion during that period. In a non-busy period, where only one source from each class is transmitting, the queuing system utilization is 80%. This simulates the normal-mode operation, which is not in congestion. If both sources of each class are transmitting, the system utility becomes 95%. This simulates the congestion-mode operation. The packet inter-arrival rates for each traffic class are given at the normal and congestion periods in Table 3.2. Other parameters, which support the operation of the queuing system are listed below.

Low-pass-filter factor in RED calculation = $1/2^4 = 0.0625$
Minimum threshold of RED = 8
Maximum threshold of RED = 16

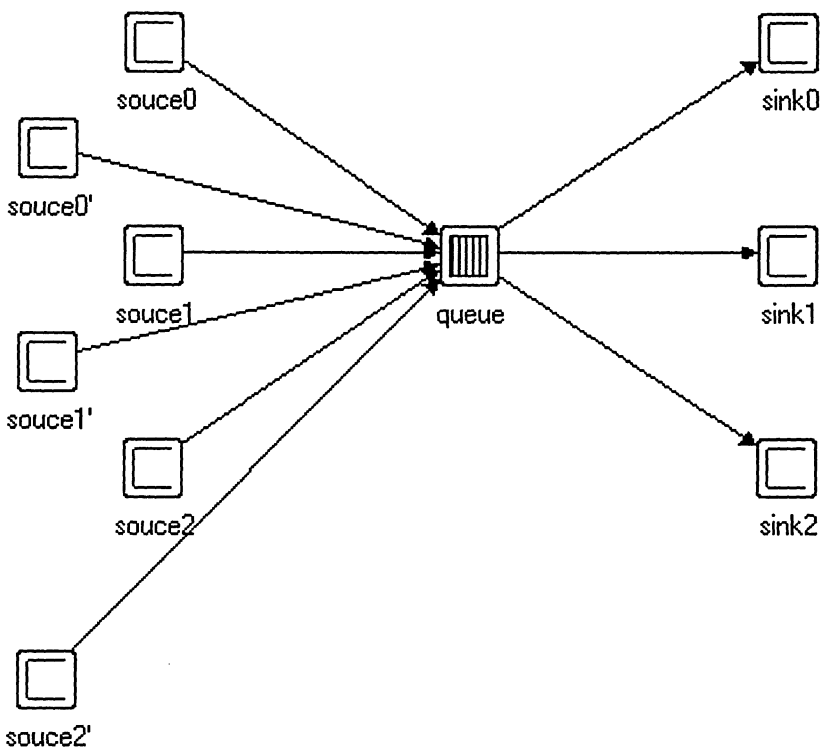

Figure 3.6: Topology

Maximum drop probability denominator of RED = 0.1

Window size in counting loss = 1,000.

Low-pass-filter factor in counting HPD queuing delay = 0.875 [2]

Service-rate = 1,000,000 bps

Packet size for all traffics = 150 byte

Table 3.2: Transmission Rate and Utility

	Packet Inter-arrival Time Period 0 ~ 180 sec	Packet Inter-arrival Time Period 180 ~ 420 sec	Packet Inter-arrival Time Period 420 ~ 600 sec
Class 0	Pareto (0.00225, 2)	Pareto (0.00225, 2) + Pareto (0.012, 2)	Pareto (0.00225, 2)
Class 1	Pareto (0.00225, 2)	Pareto (0.00225, 2) + Pareto (0.012, 2)	Pareto (0.00225, 2)
Class 2	Pareto (0.00225, 2)	Pareto (0.00225, 2) + Pareto (0.012, 2)	Pareto (0.00225, 2)
Utility	80% (normal)	95% (congestion)	80% (normal)

3.3.1 Scenario I

In the first scenario, we validate the effectiveness of SDF by comparing the packet loss of every traffic class under two situations, with and without SDF. Among the three classes, class 0 represents a real-time traffic class, which has a delay deadline (the absolute delay constraint). The other two classes, class 1 and class 2 represents normal traffic classes, which do not have a delay deadline.

Because class 0 has a absolute delay constraint, when congestion happens (during the congestion period), the queuing system tries to limit the queuing delay of all transmitted packets from class 0 to within the delay deadline, which is 5 *msec.* If this constraint conflicts with the proportional delay constraint, it will cause the delay ratio between class 0 and class 1 (and class 2) deviated from the designed value. The effect of SDF will be illustrated by the change of packet-loss proportionality among classes.

Delay Deadline = 5 *msec*

As listed in Table 3.3 the delay weight distributed for classes 0, 1, and 2 are 2 : 2 : 1. Since the weight is defined as inversely proportional to QDP. By using Eqn. 3.1, the delay proportion for classes 0, 1, and 2 should be 1 : 1 : 2 in all time. The packet-loss weight distributed for classes 0, 1, and 2 are 1 : 1 : 2, so the their packet loss ratio for classes 0, 1, and 2 should be 2 : 2 : 1.

Table 3.3: Delay and Packet Loss Weight of SDF Model

Class	Normalized Delay Weight	Delay Bound	Normalized Packet-loss Weight	Packet-loss Bound
0	2/5	5 msec	1/4	-
1	2/5	-	1/4	-
2	1/5	-	2/4	-

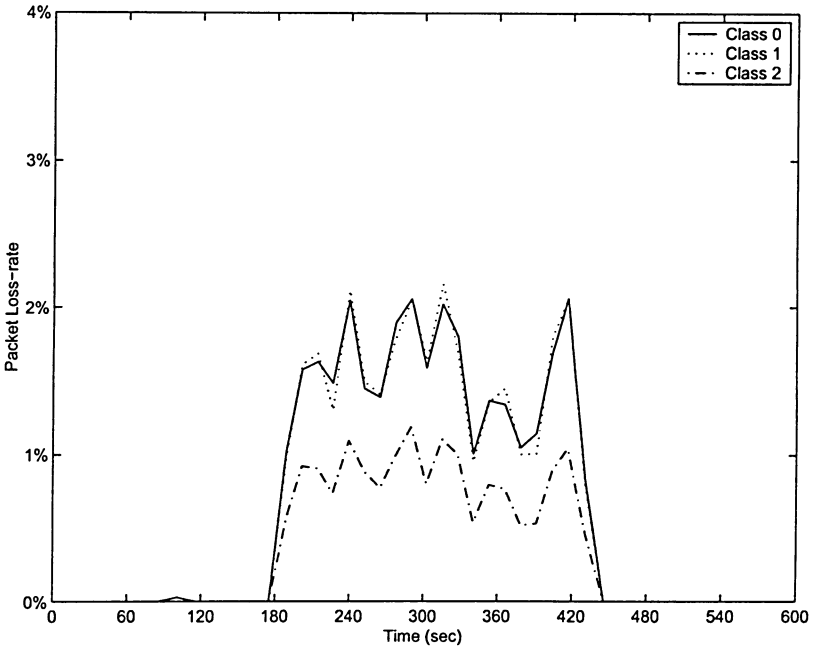

Figure 3.7: Packet-loss Rate (delay deadline = 5 *msec*, without SDF)

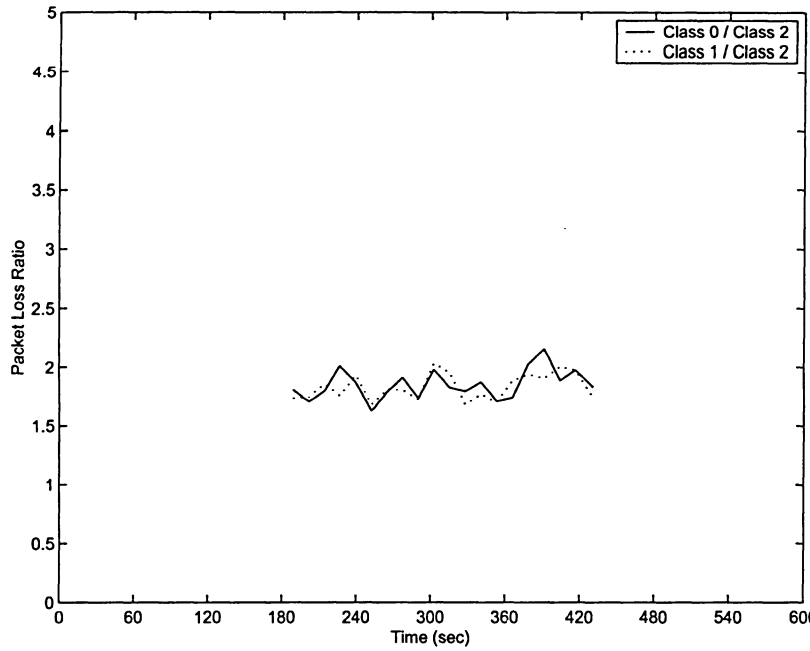

Figure 3.8: Packet-loss Ratio (delay deadline = 5 *msec*, without SDF)

The Figure 3.8 shows the packet-loss ratio of classes 0, 1, and 2 of the system without SDF. The ratio is around 1.8 : 1.8 : 1. The deviation caused by the parameter of the PLR(M) algorithm. Assume that at a given time, PLR determines to drop a packet from class 0, and it happens that no class 0 packet is in the buffer at that time, PLR will drop packet from the other classes that causes the deviation from the designed ratio. In our simulation M is set to 1,000. It seems that this value is not large enough to allow the proportional algorithm reach the desirable steady-state value.

We can eliminate this error by increasing the M size. In Figure 3.10, when M = 10,000, both average packet loss ratios are very close to the pre-defined 2. However, increasing the window size significantly increases the system overhead. This is because when PLR is trying to select a packet to drop, it has to examine the history of the last M packet arrivals and compute the packet-loss rate of each class among them. If M = 1,000, PLR only needs to go over 1,000 packets. In the case when M = 10,000, this computation process will use

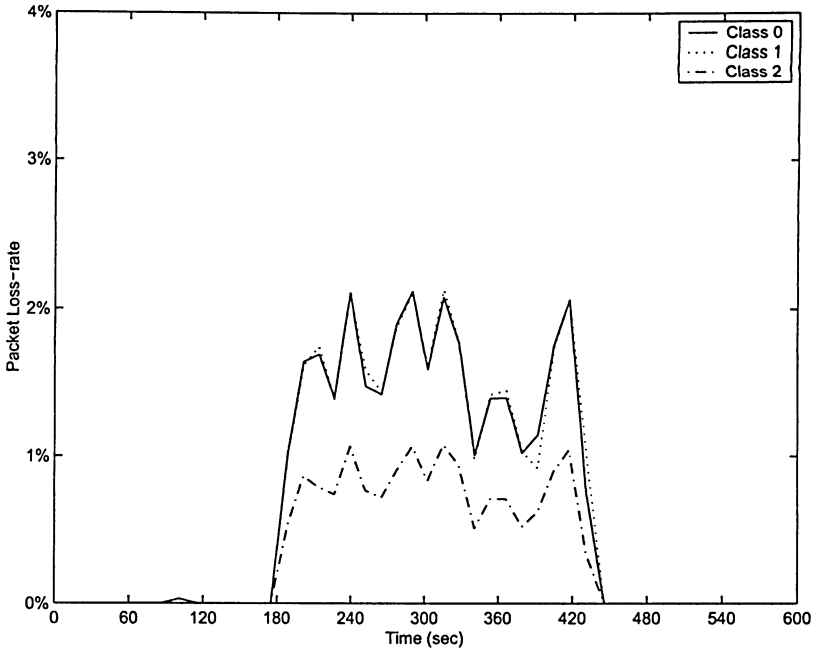

Figure 3.9: Packet-loss Rate (M = 10,000)

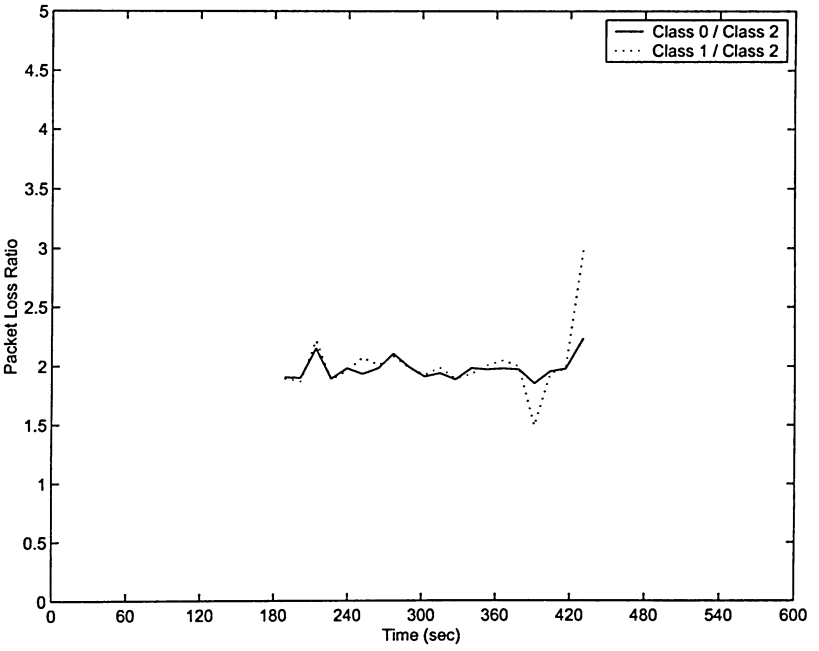

Figure 3.10: Packet-loss Ratio (M = 10,000)

more CPU resources. The appropriate M value is a trade-off between the accuracy and the computation overhead. In this thesis, we use M = 1,000.

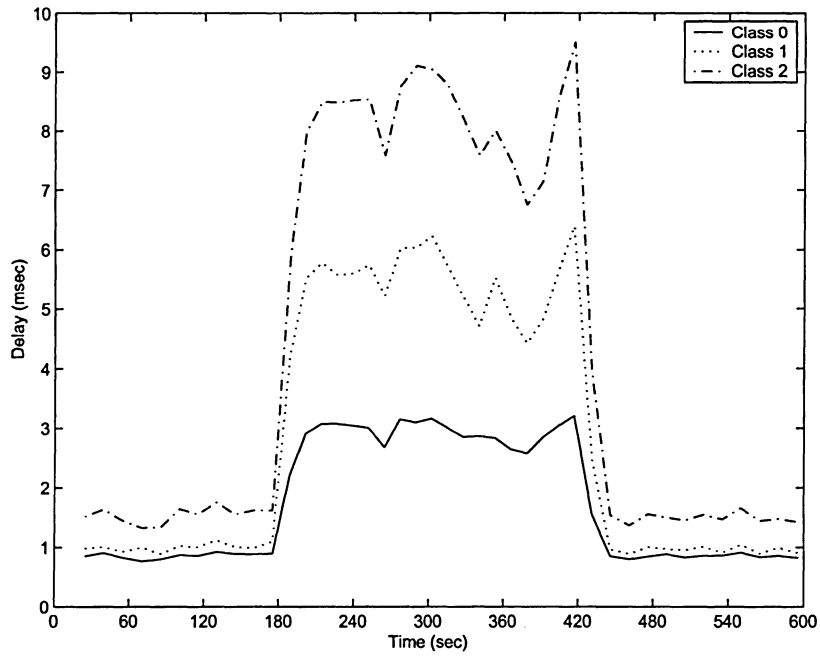

Figure 3.11: Delay (delay deadline = 5 *msec*)

The upper delay limit for class 0 is 5 *msec*. If the queuing delay of a packet belonging to class 0 is higher than 90% of the deadline, 4.5 *msec*, the EDD will schedule to transmit this packet immediately. Consider in the congestion period. Since the queuing delay increases, many class 0 packets are taking the EDD shortcut. The actual delay ratio is not equal to the pre-defined value 2 any more, instead it is larger than 2 (Figure 3.11 and Figure 3.12).

If there is no SDF deployed, the packet loss ratios among classes (Figure 3.7 and Figure 3.8) are not affected by the violation of the delay proportional constraint. However, after we employ the delay to packet loss feedback in the queuing system, the packet loss of class 0 is increased due to the reduction of the weight for the packet loss (Figure 3.13). By comparing the results in Figure 3.7 with Figure 3.13, we can see that the packet loss of class 1 and class

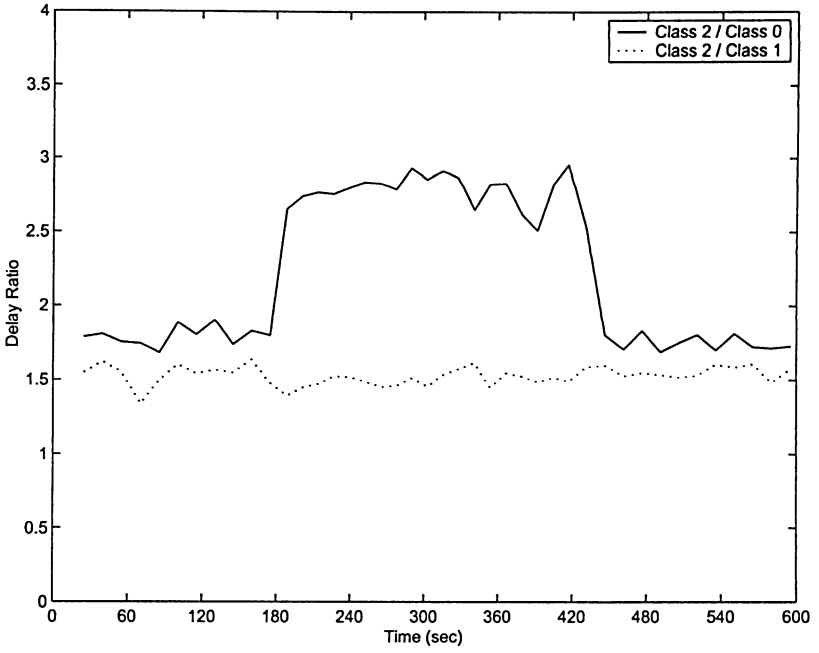

Figure 3.12: Delay Ratio (delay deadline = 5 *msec*)

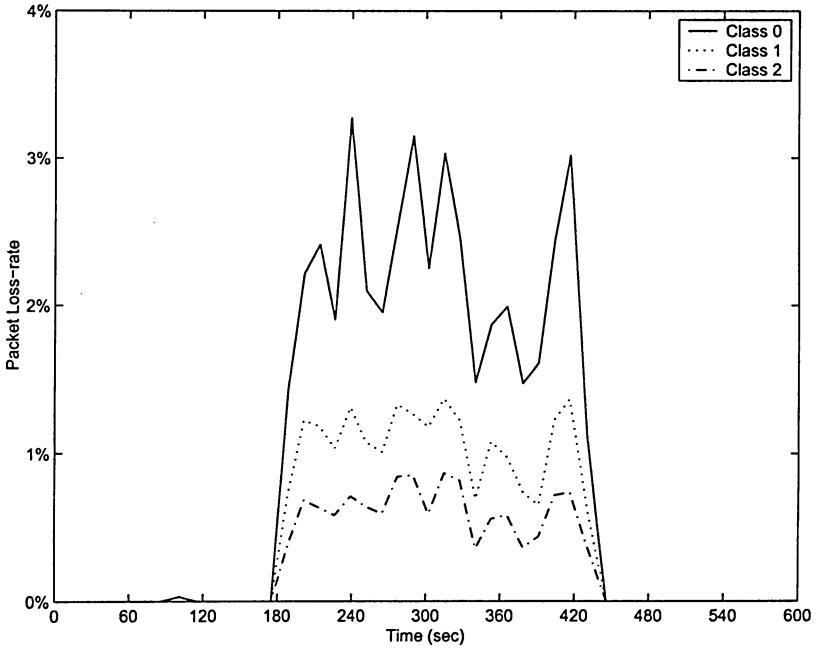

Figure 3.13: Packet-loss Rate (delay deadline = 5 *msec*, with SDF)

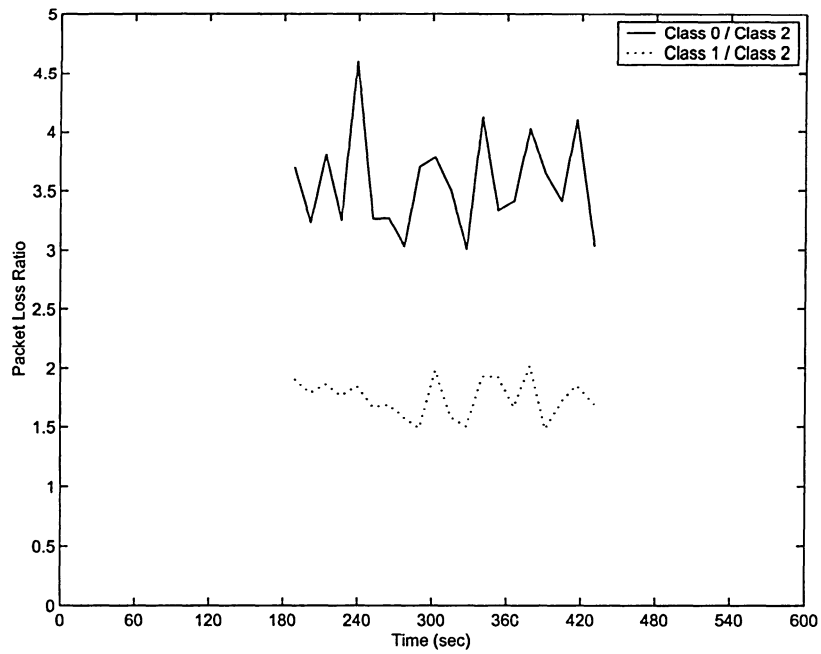

Figure 3.14: Packet-loss Ratio (delay deadline = 5 *msec*, with SDF)

2 are decrease as the packet loss of class 0 increases.

Delay Deadline = 5 *msec* with Different RED Threshold

In this case, we change the RED minimum threshold and maximum threshold from 8 and 16 to 16 and 32. The results of delay and delay ratio of all traffic classes are shown in figures below,

Compare Figure 3.15 with Figure 3.11, we can see that the delays of class 1 and class 2 is much higher. This is because since both the minimum and maximum threshold of RED increase, the queuing system then drops less packet, which causes the increase of average queue length increase, so the average delay increases. Since class 0 has a delay deadline, its delay will remain in a low level, while class 1 and class 2 do not has a limit, so their delays are higher. Although the delays of class 1 and class 2 has changed, their delay ratio

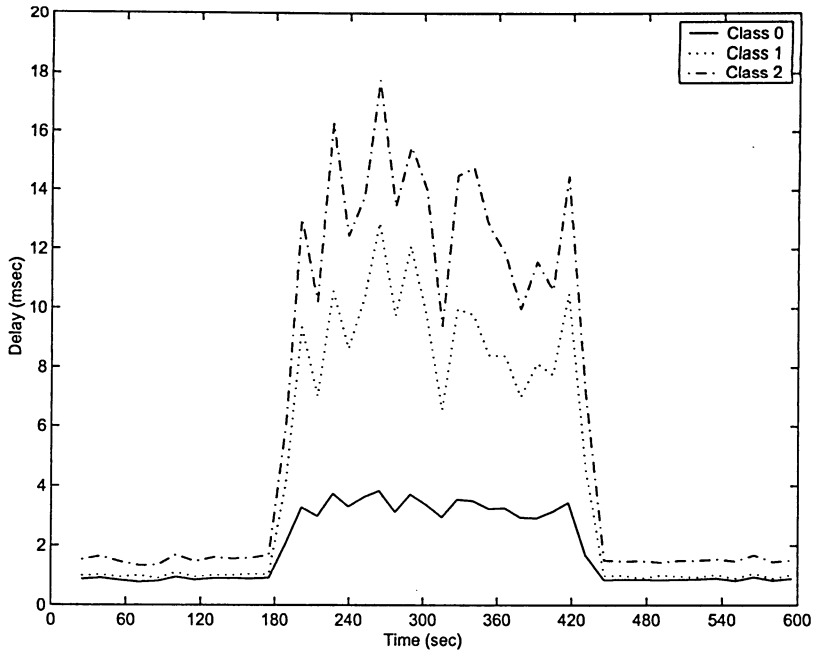

Figure 3.15: Delay (delay deadline = 5 *msec*, with SDF, RED Threshold is 16, 32)

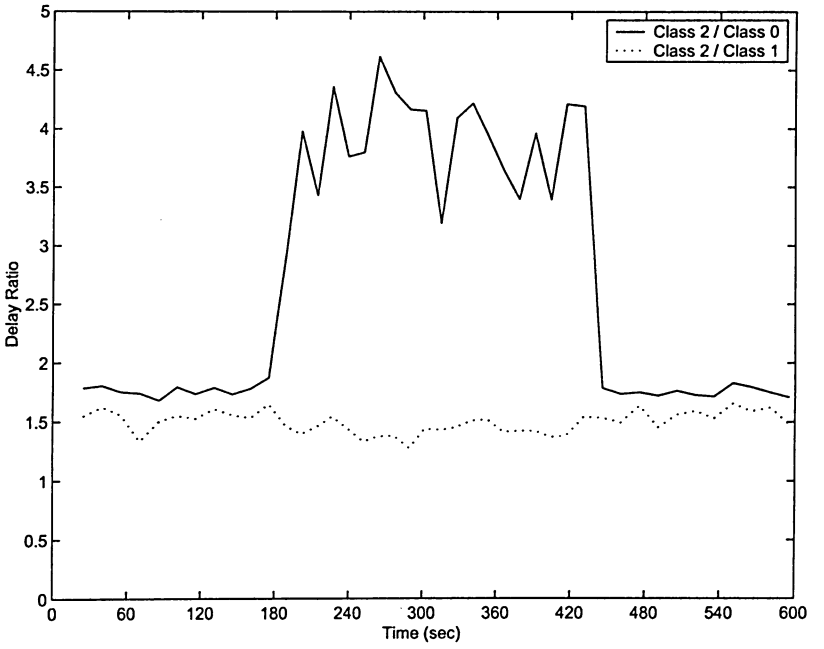

Figure 3.16: Delay Ratio (delay deadline = 5 *msec*, with SDF, RED Threshold is 16, 32)

(Figure 3.16) does not change too much (3.16). This proved that when applying SDF, the pre-defined proportional differentiation of delay will not be affect by the parameters of RED.

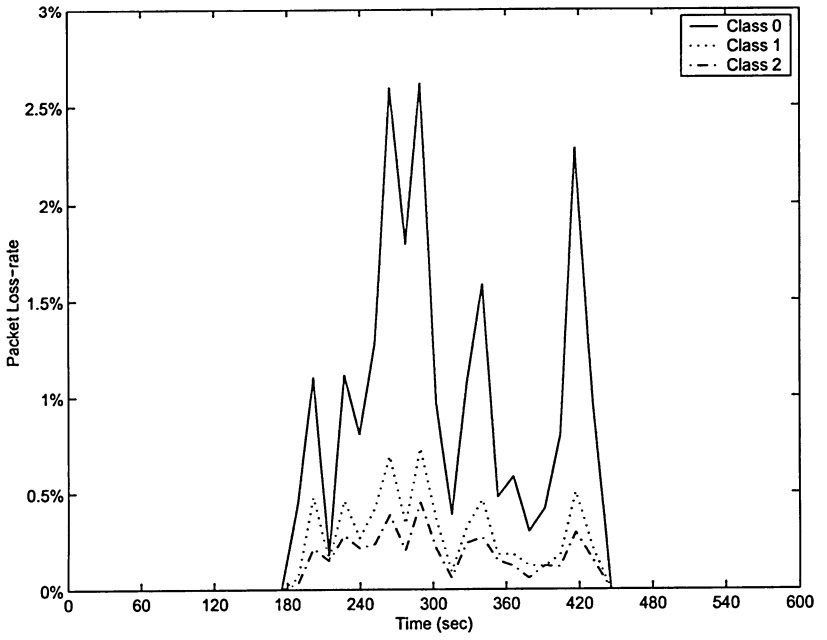

Figure 3.17: Packet-loss Rate (delay deadline = 5 *msec*, with SDF, RED Threshold is 16, 32)

Since the thresholds of RED increase, the total packet loss decreases. However, the measured packet-loss ratio between class 0 and class 2 in Figure 3.18 is much higher than the pre-defined ratio, which is 2. This is the result of the SDF, which proved that the SDF can work well under various thresholds of RED.

Various Delay Deadline

Table 3.4 and Table 3.5 compare the differences of packet-loss rates with vary delay deadlines. The RED minimum and maximum threshold are restored to 8 and 16, respectively. In this simulation, we consider two modes, normal mode and congestion mode. Packet-loss rate during the two different modes are completely different. From Figure 3.7, we can see

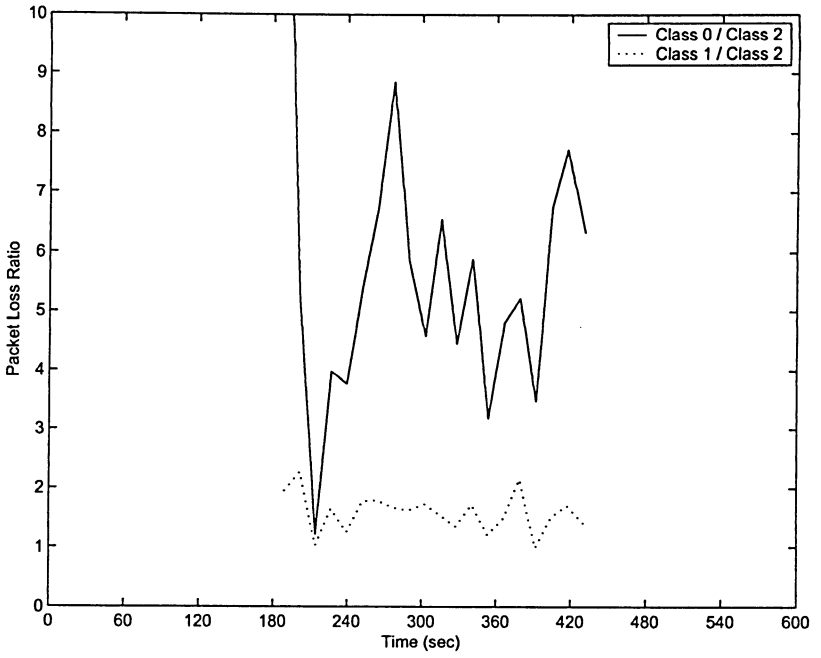

Figure 3.18: Packet-loss Ratio (delay deadline = 5 $msec$, with SDF, RED Threshold is 16, 32)

that the packet-loss rate during the normal mode is nearly 0. To emphasize the effectiveness of SDF, the packet-loss rates that shown in the tables are only the average value during the congestion mode. Thus, the average value during the period of 180 $\sim$ 420 sec.

From the tables, we can observe the characteristics of SDF. With the decreasing of delay deadline, the effectiveness of SDF becomes more obvious. This is because if the deadline of class 0 is lower, class 0 benefit more in delay, and $|\Delta w_{d0}|$ is larger. Accordingly the absolute value of Δw_{l0} is also larger. Thus the tradeoff in SDF is more distinct.

Moreover, In Figure 3.20, the packet loss ratio between class 2 and class 0 keeps increasing with the decrease of the delay deadline. However, the ratio between class 2 and class 1 is almost a constant value, in spite of the change between class 2 and class 0. This shows that

Table 3.4: Packet Loss Rate without Feedback

Delay Deadline (msec)	Packet Loss Rate			Packet Loss Ratio	
	Class 0	Class 1	Class 2	Class 0 / Class 2	Class 1 / Class 2
20	0.016083705	0.016134379	0.008711191	1.846326868	1.852144025
18	0.016147214	0.016055444	0.008727145	1.850228701	1.839713217
16	0.016194846	0.016055444	0.008679281	1.865920169	1.849858694
14	0.016099582	0.016134379	0.008695236	1.851540601	1.855542454
12	0.015988441	0.016181741	0.008759054	1.825361613	1.847430132
10	0.016226601	0.016008083	0.008695236	1.866148416	1.84101766
9	0.016099582	0.016134379	0.008695236	1.851540601	1.855542454
8	0.016067828	0.016165954	0.008695236	1.847888647	1.859173653
7	0.016210724	0.016039657	0.008679281	1.867749502	1.848039757
6	0.016147214	0.016134379	0.008647372	1.867297232	1.865812984
5	0.016147214	0.016039657	0.0087431	1.846852372	1.834550416

Table 3.5: Packet Loss Rate with Feedback

Delay Deadline (msec)	Packet Loss Rate			Packet Loss Ratio	
	Class 0	Class 1	Class 2	Class 0 / Class 2	Class 1 / Class 2
20	0.019100392	0.014082061	0.007753917	2.46332177	1.816122283
18	0.019021006	0.014050487	0.007865599	2.418252725	1.786321351
16	0.019195656	0.01376632	0.007977281	2.40629065	1.725690794
14	0.019148024	0.013924191	0.007865599	2.434401324	1.77026453
12	0.019306797	0.013861042	0.007769871	2.484828392	1.783947447
10	0.019529079	0.013734746	0.007674144	2.544789253	1.789743023
9	0.019767239	0.013513727	0.007658189	2.581189567	1.764611191
8	0.020164171	0.013355856	0.007418871	2.717956834	1.800254528
7	0.020973914	0.012803309	0.007163598	2.927846228	1.787273496
6	0.022005938	0.012187613	0.006748779	3.260728579	1.805898906
5	0.023164981	0.011256177	0.006525416	3.549962545	1.724974676

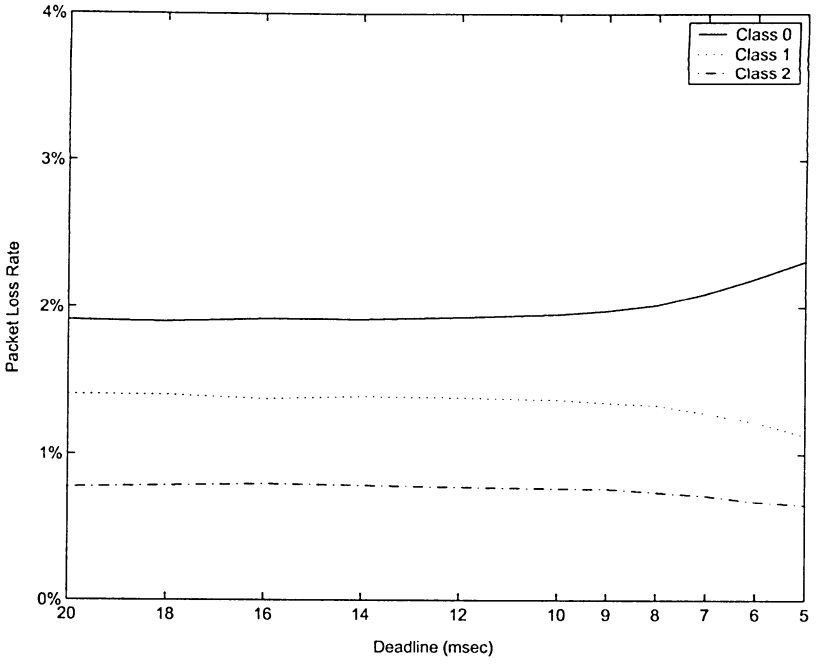

Figure 3.19: Packet-loss Rate (With SDF)

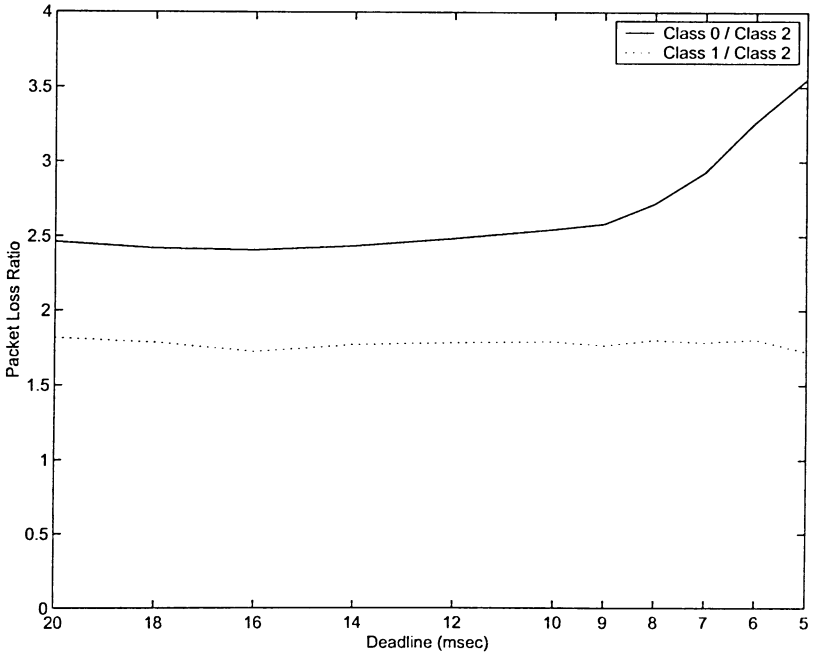

Figure 3.20: Packet-loss Ratio (With SDF)

in SDF model, the feedback used by one traffic class only affects the proportional ratios between itself and other traffic classes. It does not affect the ratio between other traffic classes.

3.3.2 Scenario II

In this scenario, the effectiveness of the SDF&ASM model is investigated under three different systems. The first system is a general proportional differentiated services system with STE for the real-time traffic (class 0 traffic). The structure of the system is similar to SDF model in Figure 3.3, except that EDD and SDF are not deployed. The second system deploys EDD but not SDF. The third system is based on the SDF&ASM model (Figure 3.5). We will see that the SDF&ASM system have the best head-drop rate performance among the three systems.

Delay Deadline = 5 *msec*

Figure 3.21 and Figure 3.22 illustrate the packet head-drop rate of class 0 and the total packet-loss rate of all traffic classes in the case that the delay deadline of class 0 equals to 5 *msec*. To emphasize the effectiveness of ASM, we choose the same delay and packet-loss weights for all three traffic classes (Table 3.6).

Table 3.6: Delay and Packet Loss Weight of SDF&ASM Model

Class	Delay Weight	Delay Bound	Packet-loss Weight	Packet-loss Bound
0	1/3	5 msec	1/3	-
1	1/3	-	1/3	-
2	1/3	-	1/3	-

Figure 3.21 shows the head-drop rates of the three systems. The head drop of the general

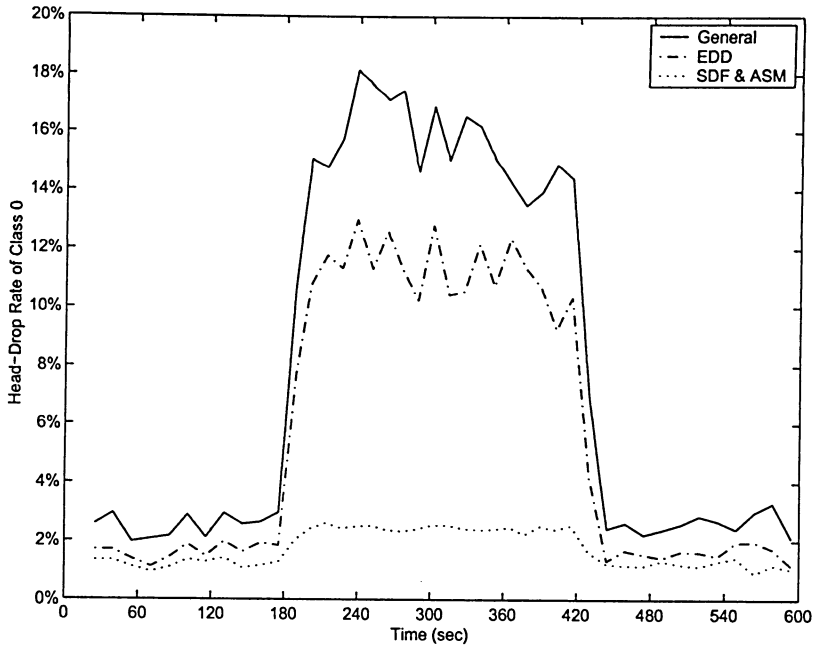

Figure 3.21: Packet-head-drop Rate of Class 0 (delay deadline = 5 *msec*)

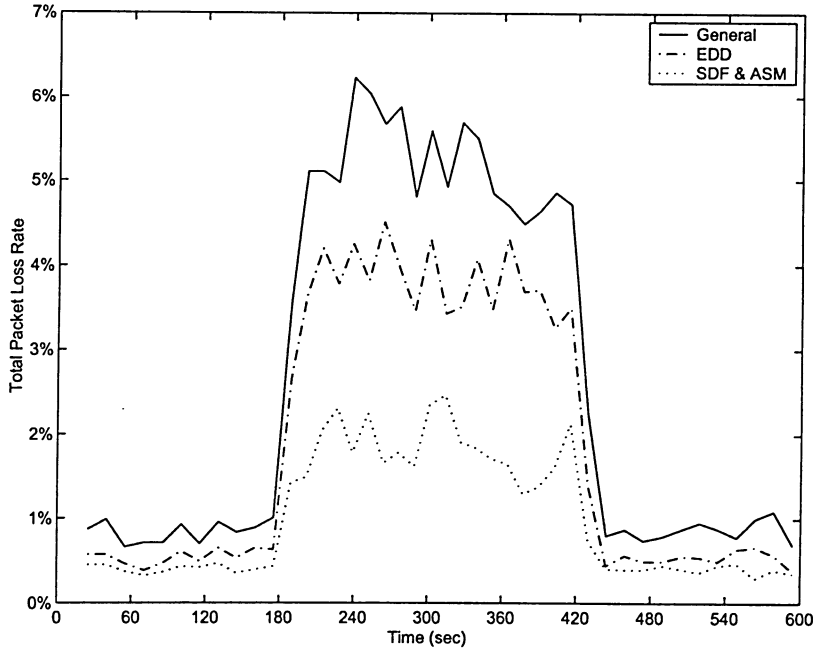

Figure 3.22: Total Packet-loss Rate (delay deadline = 5 *msec*)

system is quite high. Nearly 16% of packets from class 0 have violated their delay deadline. In practice, this is usually not acceptable. When the EDD mechanism applied in system 2, the head-drop rate reduces to around 11% which is an improvement, but still not enough. The inadequate performance of the second system is due to the fact that the delay deadline is set too low. with the ASM mechanism introduced in the third system, the head drop is reduced to around 2%. This is because ASM adaptively increases the safety margin, which effectively increases the bandwidth for class 0 during the congestion period.

As a result of ASM and STE, class 0 receives a very low packet head-drop rate and also a very short queuing delay. SDF may penalize class 0 with more tail drop. With the introduction of SDF with ASM, the total queuing length decreases. This may reduce the drop probability in RED, so the total packet loss decreases (Figure 3.22).

Various Delay Deadline

Table 3.7 and Table 3.8 illustrate the tendency of the packet head drop of class 0 and the total packet loss under various deadlines. The results are also displayed in Figure 3.23 and Figure 3.25 as well. And with the same reason, the packet head-drop rate and the total packet loss are also the average values measured during the congestion period.

Consider the head-drop rate of class 0 in Table 3.7 and Figure 3.23. For the first system, the head-drop rate of class 0 increases very rapidly as delay deadline decreases. For the EDD case, the drop rate is kept to a rather low value until the delay deadline is smaller than 12 *msec.* However, it quickly becomes out of control after the deadline is less than 12 *msec.* This demonstrates that with the fixed safety margin, 1/10 of the deadline, EDD does not adapt to the change of the environments. For the SDF&ASM case, the head-drop rate is alwys kept in a relatively low value.

Table 3.7: Head Drop-rate of Class 0

Delay Deadline (msec)	Packet Head-drop Rate of Class 0			Head Drop Improvement	
	General	EDD	SDF&ASM	EDD over General	SDF&ASM over General
15	0.058041613	0.001111019	0.000993399	98.09%	98.29%
14	0.065344885	0.001162922	0.001423893	98.22%	97.82%
13	0.072452938	0.00221547	0.002588438	96.94%	96.43%
12	0.078910042	0.003403945	0.003013189	95.69%	96.18%
11	0.082452465	0.010819605	0.007438793	86.88%	90.98%
10	0.091600152	0.020315242	0.010634931	77.82%	88.39%
9	0.102829385	0.034621158	0.012715211	66.33%	87.63%
8	0.111265121	0.051270281	0.014962357	53.92%	86.55%
7	0.123472133	0.070089985	0.017412387	43.23%	85.9%
6	0.135671143	0.088714907	0.020603745	34.61%	84.81%
5	0.155895334	0.11227595	0.024407482	27.98%	84.34%

Table 3.8: Total Packet Loss Rate

Delay Deadline (msec)	Total Packet Loss Rate			Packet Loss Improvement	
	General	EDD	SDF&ASM	EDD over General	SDF&ASM over General
15	0.023100703	0.014709464	0.014143112	36.32%	38.78%
14	0.025795438	0.014182672	0.014787537	45.02%	42.67%
13	0.026603418	0.015042032	0.015992647	43.46%	39.88%
12	0.028790827	0.013688261	0.015178947	52.46%	47.28%
11	0.029310591	0.01639612	0.015449668	44.06%	47.29%
10	0.031909109	0.018264985	0.015722524	42.76%	50.73%
9	0.035235208	0.019873717	0.016925625	43.6%	51.96%
8	0.037758997	0.02331963	0.016780079	38.24%	55.56%
7	0.04167938	0.027229154	0.018255887	34.67%	56.2%
6	0.045192535	0.03194826	0.018860081	29.31%	58.27%
5	0.051948052	0.038081857	0.018507747	26.69%	64.37%

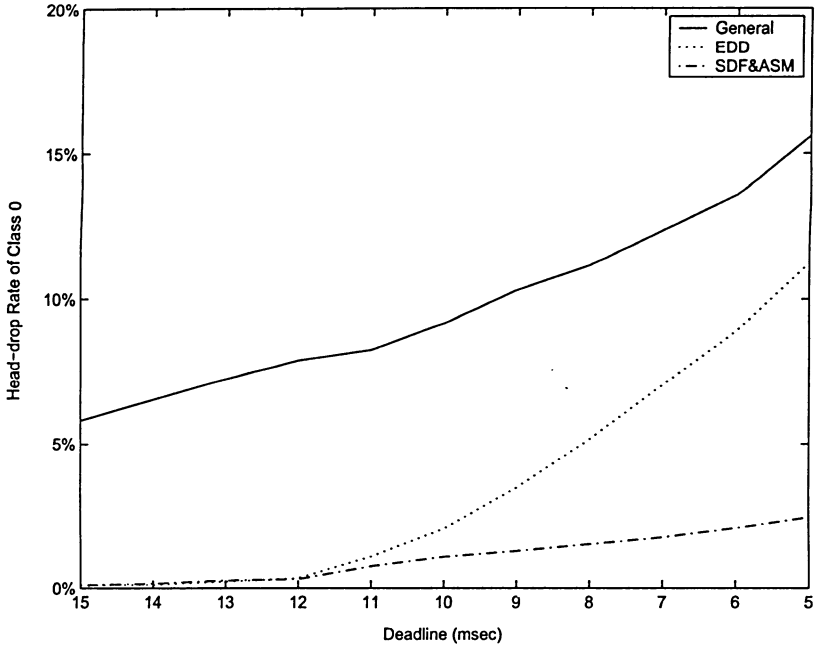

Figure 3.23: Packet-head-drop Rate of Class 0

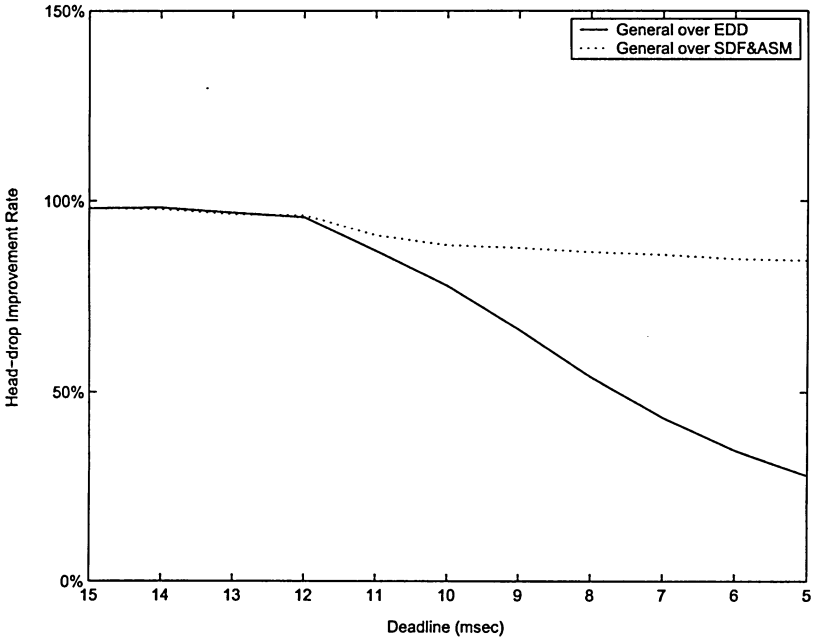

Figure 3.24: Packet-head-drop Ratio

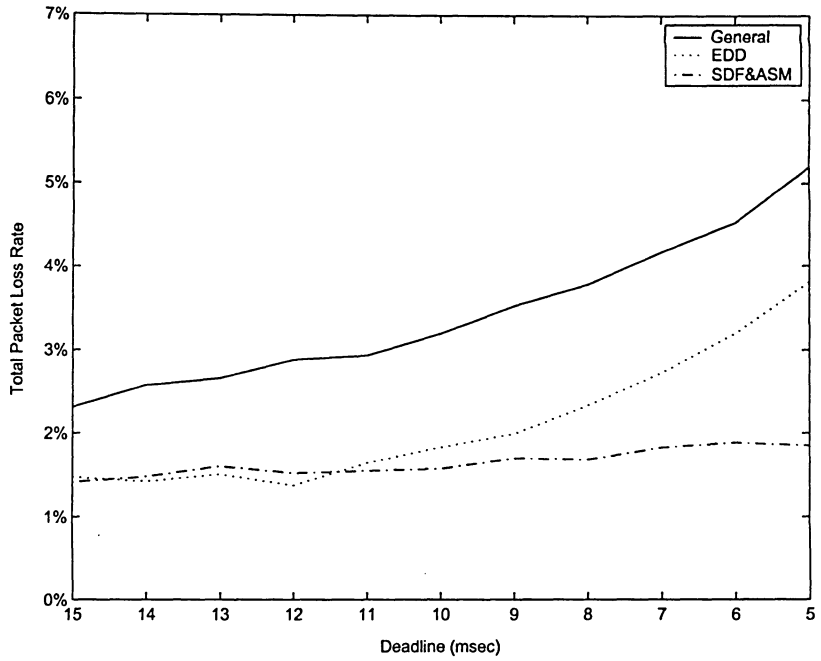

Figure 3.25: Total Packet-loss Rate

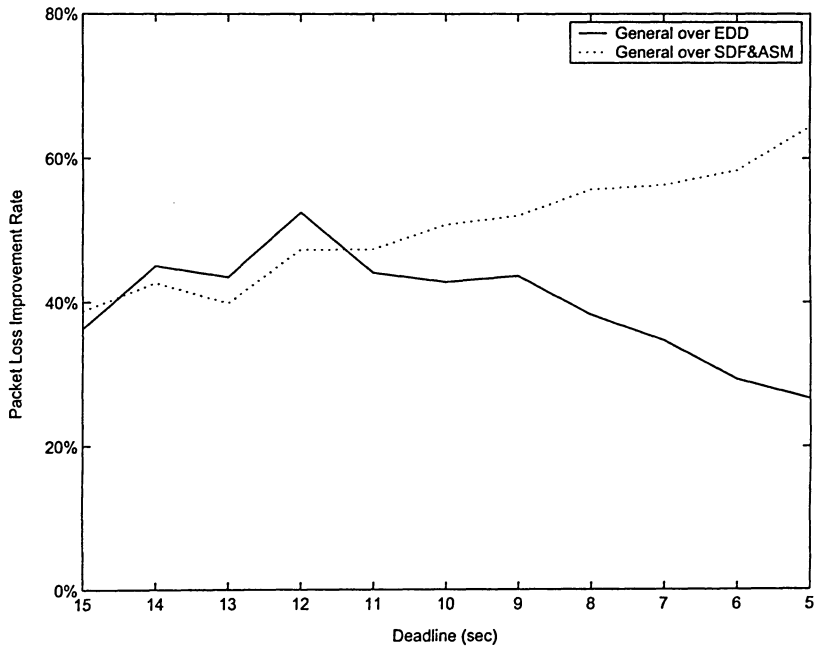

Figure 3.26: Total Packet-loss Ratio

Now consider the total packet-loss rate in Table 3.8 and Figure 3.25. For the first system without EDD, again the total packet loss increases rapidly with the decrease of the delay deadline. For the system with EDD, the 12 *msec* is also the performance threshold. When large than 12 *msec*, it works well. However, when the deadline is getting smaller, the drop rate increases, significantly. For the SDF&ASM case, the total packet loss changes slowly.

Table 3.9: Delay under different deadlines

Class	Delay Deadline (msec)	Average Delay (msec)			Percentage of Increase in Delay	
		General	EDD	SDF&ASM	General over EDD	General over SDF&ASM
0	15	3.3548	4.3461	4.3018	29.55%	28.23%
	10	2.4279	3.5426	3.5424	45.91%	45.9%
	5	1.3581	1.7825	2.3868	31.25%	75.75%
1	15	4.3579	6.4169	6.3658	47.25%	46.07%
	10	3.5855	6.1304	6.2317	70.98%	73.8%
	5	2.4043	3.6245	6.4366	50.75%	167.71%
2	15	4.3774	6.3456	6.2951	44.96%	43.81%
	10	3.5772	6.1663	6.3227	72.38%	76.75%
	5	2.4026	3.6116	6.4154	50.32%	167.02%

The average delays and delay changes of the real-time traffic class, class 0 and other non-real-time traffic classes, class 1 and class 2 are listed in Table 3.9. In this table, we can see that the average delay increases when EDD and SDF&ASM are used. The increase of average delay of the traffic is mainly caused by the decrease of the total packet loss. When the total packet loss is reduced, the average queue length becomes longer, so the average delay increases as well. For class 0, since deadline violation is the main concern, so the increase of average delay should not affect the overall performance.

Nevertheless, the delay changes of class 1 and class 2 are very high, especially in the case when delay deadline of class 0 is very low. There are two reasons that cause the higher delay

increase rate. The first reason is the same as the reason of class 0, which is the decrease of total packet loss. The second reason is when class 0 gains more bandwidth from EDD or ASM in congestion, class 1 and class 2 have to receive less bandwidth. However, class 1 and class 2 represent non-real-time traffic, which is not delay sensitive. The reduction of packet loss out weights the increase of delay.

Chapter 4

End-to-end Delay Model

4.1 The Limitation of Single-node Scheduler

In a single node model, traffic flows that belong to the same QoS class will receive the same services in the node. However, packets belonging to the same QoS class do not necessary belong to the same flow. Because of that, they may experience different end-to-end delays.

For example, consider two packets, packet 1 and packet 2. They are from two different flows. Both of them belong to class 0 and both have their deadlines equal to 100 *msec*. Now assume that packet 1 arrives to a node before packet 2, further assume that packet 1 has been delayed in the network by 30 *msec*, while packet 2, which has a longer path and has to go through more nodes, has accumulated a delay of 70 *msec*. If a single node model is employed, since there is no differentiated treatment for packets belonging to the same class, and the FCFS (First Come First Serve) scheduling is used within the class, packet 1 will be transmitted first, even though packet 2 should be sent for it is more urgent.

4.2 End-to-end Delay Scheme

Obviously, to provide a better end-to-end delay performance, a scheduler for a class should take into account the delays experienced by all packets currently in the queue. In this sec-

tion, we propose schedulers that sort the packets in the queue based on the delay of each packet in descending order.

4.2.1 Implementation Requirement

In order to sort packets based on delay, Each packet must carry a delay value to indicate the amount of time that the packet has been in the network. We assume that each router in the network will measure the queuing delay of each packet and update the delay when the packet is sent out according to the following equation,

$$new\ delay = old\ delay + queuing\ delay + transmission\ delay + propogation\ delay \quad (4.1)$$

The transmission delay can be calculated based on the size of the packet and the channel transmission rate. The propagation delay can be pre-configured at the outgoing interface. In addition, each arrived packet is time-stamped upon arrival and the queuing delay can be computed using the time stamp.

4.2.2 The benefit of Sorting Packets

There are at least two benefits to sort the packets in the queue. The first benefit is it may cut down the total packet drop rate. Since a real-time traffic usually has a predefined deadline. If the packets in the buffer are sorted based on their delays instead of the arrival order, more urgent packets may meet the deadline during congestion and less packets are eventually dropped.

To use the example in section 4.1. Assume the queueing system is the last hop of both packet 1 and packet 2. In this moment, only these two packets are in the queue. The packet size of them are equal. The service time (propagation delay + transmission delay) for both of them are 20 *msec.* If we send them by their arrival order, packet 1 is sent first and it

beats the deadline by 50 *msec* when it reach its destination. On the other hand, packet 2 will be delayed further by 40 *msec* before it reaches the destination. Since packet 2 has already used up 70 *msec* before. The end-to-end delay of packet 2 will be 110 *msec*. When it reaches the destination, it will be dropped. If we sort these two packets based on their delays and send packet 2 first, both packet 1 and 2 will meet the deadline.

The second advantage of using the sorting is that the scheme also decreases the traffic jitter. It is easy to see that from the example above, sorting-by-delay approach will smooth out the packet delay.

4.2.3 Sorting Schemes

In this article, we will introduce two sorting schemes: The first one is called one-sort scheme and the second one is called complete-sort scheme. The one-sort scheme compares the delay of a newly arrived packet with the backlog packets only once to find a reasonable position for this packet. The complete-sort scheme continues the sorting in the queue, until the most appropriate position for the packet is determined.

One-sort Scheme

In the one-sort scheme, when a new packet arrives, the system merely compares the delay of this packet with the delay of the middle packet in the queue. The middle packet is defined as following: if the number of packets in the queue is an odd number, the system just selects the packet in the middle of them; If the number is a even number, since there are two middle packets, the system may selects the packet which is closer to the head of the queue. For example, assume the queue has four packets, then the second packet, measured from the head of the queue, will be selected for comparison.

After the comparison, if the newly arrival packet has a longer delay than the middle packet,

it will be insert into the position just before the middle packet. Otherwise it will be put to the tail of the queuing buffer. Note that before the comparison, the delay associated with the middle packet is updated by adding the old delay with the queuing delay experienced by the packet so far. The queuing delay, again, can be derived from the time stamp.

Complete-sort Scheme

In the complete-sort scheme, the newly arrived packet is first compared with the middle packet of the queue similar to the One-sort Scheme. If the new one is more urgent, it will be further compared with the new middle packet between the head packet and the old middle packet. Otherwise it will be compared with the new middle packet, which is between the old middle packet and the tail packet. This process will continue until the most appropriate location of the newly arrived packet are located such that all the packets in the queue are sorted into a descending order again.

Needless to say, the second scheme are more precise than the first one. However, it may need more system resources because a higher number of comparison steps is required. The maximum number of comparison steps for the second scheme is $\lceil \log_2 n \rceil$, where n is the number of packets currently in the queue.

Sorting Process Thresholds

Both sorting schemes may consume a lot of system resources for the sorting process is involved in each packet arrival. We observe that during a non-congestion period, the queue will be small and it is not much benefit to do the sorting. Only in the long queue, which occurs during congestion, will the system benefit most from the sorting process. Consequently, our schemes will only perform sorting when the queue is large.

We define two queue-length thresholds: high threshold and low threshold. When a packet

arrives, the system checks the current queue length. If the queue length is higher than the high threshold, the sorting process is trigged. The sorting process will continue until the queue length retreats below the low threshold, then the process stops.

Note that before the system starts the sorting process, there are already some backlogs in the queue. Neither the one-sort scheme nor the complete-sort scheme can be applied directly in this situation because the packets already in the queue have not been sorted and they are in a random order. To smooth out and simplify the transition from non-sorting state to sorting state (Figure 4.1), the system should ignore the old packets and just put the newly arrived packet in the tail position. After that, all the subsequently arrived packets will be sorted among themselves. These packets will be transmitted after all the old packets are transmitted.

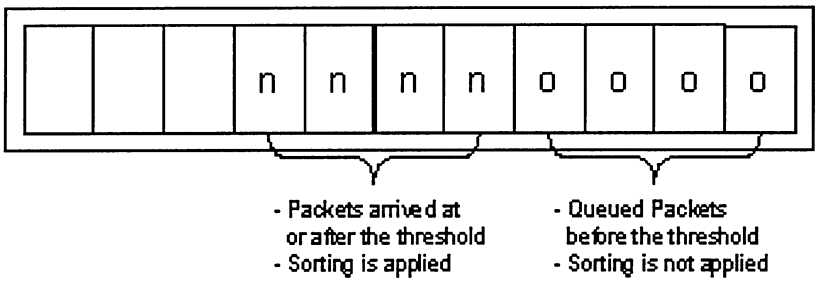

Figure 4.1: From Non-sorting State to Sorting State

4.3 Simulation

Three simulation scenarios are studied here. In three scenarios, the packet drop rate and end-to-end delay of real-time traffic flows are compared among FCFS, one-sort, and complete-sort schemes. The first scenario works in a pure real-time environment, where only real-time traffic flows exist, and all flows belong to the same traffic class. The second scenario works in a mixed environment, where the real-time traffic class has to share the queuing resources

with other traffic classes. In the third scenario, we investigate the relative performance of three schemes incorporated with the SDF&ASM model.

4.3.1 Scenario I Real-time Traffic Only

In this scenario, we trace five real-time traffic flows, flow 0 (flowing from source 0 to sink 0), flow 1 (flowing from source 1 to sink 1), flow 2 (flowing from source 2 to sink 2), flow 3 (flowing from source 3 to sink 3), and flow 4 (flowing from source 4 to sink 4). All five traffic flows belong to the same traffic class. There are also four nodes in the topology as shown in Figure 4.2. Traffic flow 0 passes through all four nodes, traffic flow 2 passes through three nodes, and traffic flows 1, 3, and 4 pass through two nodes. Parameters and nodes are listed in table 4.1.

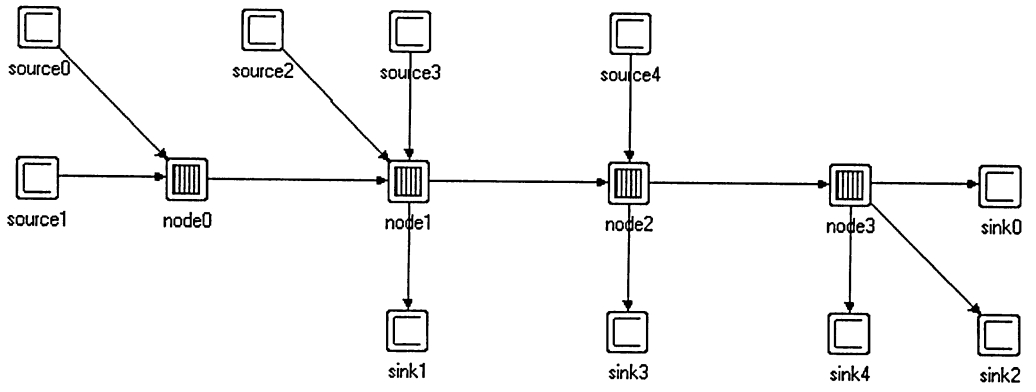

Figure 4.2: Topology I

Under this scenario, three schemes, FCFS, one-sort, and complete-sort, are tested. In the FCFS case, packets are queued in FCFS . From table 4.2, we can see that even the one-sort scheme can greatly improve the packet drop-rate for all traffic flows. As discussed in section 4.2.2, by sorting the packets in the queue based on their experienced delays, more urgent packets can be scheduled for earlier transmission. Therefore, more packets will meet their delay bound, and less packet will be dropped. Comparing with the FCFS scheme, the one-

Table 4.1: Transmission Rate and Utility in Real-time Traffic Environment

Real-time Traffic Flow	Packet Inter-arrival Time	Node	Service Rate (bps)	Utility
0	Pareto (0.002, 2)	0	666,667	90%
1	Pareto (0.002, 2)	1	1,000,000	90%
2	Pareto (0.004, 2)	2	833,333	90%
3	Pareto (0.004, 2)	3	666,667	90%
4	Pareto (0.004, 2)			

sort scheme can decrease the packet drop-rate of all traffic flows from 35% to 44%, and the complete-sort scheme can further decrease the packet drop-rate up to 53% $\sim$ 67%.

Table 4.2: Packet Drop-rate in Real-time Traffic Environment

Flow	Packet Drop-rate			Packet Drop Improvement	
	FCFS	One Sort	Complete Sort	FCFS to One Sort	FCFS to Complete Sort
0	0.179174871	0.108296628	0.069188481	39.56%	61.38%
1	0.098450846	0.055420252	0.032329571	43.71%	67.16%
2	0.106950063	0.071404372	0.050234327	33.24%	53.03%
3	0.070297665	0.040469224	0.02772508	42.43%	60.56%
4	0.082624266	0.053465962	0.03765048	35.29%	54.43%

More over, sorting the backlog packets can also smooth out their arrival times. That means the delay variation, or jitter is smaller. table 4.3 shows the average end-to-end delay and the delay variation of flow 0, which passing through all nodes. From table 4.3, we can see that the average delay of each scheme has almost no change, because the total system resources remain the same. However, the average delay variation of both sorting schemes are reduced in comparison with the FCFS scheme.

Table 4.3: Delay and Delay Variation in Real-time Traffic Environment for Flow 0

	FCFS	One Sort	Complete Sort	FCFS to One Sort	FCFS to Complete Sort
Average Delay	67.8023	67.111	67.0978	1.02%	1.04%
Variation	4.15E+07	2.73E+07	2.19E+07	34.26%	47.12%

4.3.2 Scenario II Mixed Traffics without Proportional Differentiation

In this simulation scenario, beside the five traffic flows in scenario I, a new traffic flow, flow 5 (flowing from source 5 to sink 5) is added. The traffic flow 5 is not a real-time traffic. It simulates the background traffic, which shares system resources with the real-time traffic. The delay weight of both the real-time traffic flow and the non-real-time traffic flow are the same, that is no proportional differentiation is applied. This scenario proves that the end-to-end schemes also works well in the mixed environment.

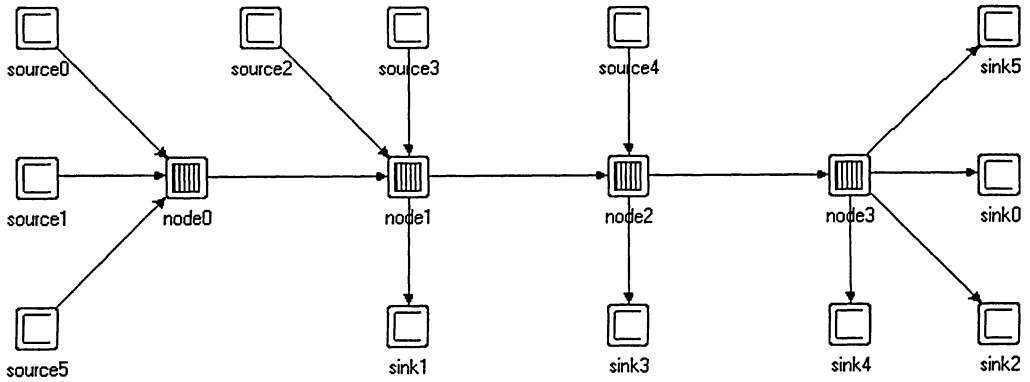

Figure 4.3: Topology II

There are a few changes in the traffic parameters. The changes are presented in table 4.4. Figure 4.3 shows that the traffic flow 5 passes through all four nodes, and consumes about 1/3 of the system resource in each node.

Table 4.4: Transmission Rate and Utility in Mixed Traffic Environment

Traffic Flow	Packet Inter-arrival Time	Node	Service Rate (bps)	Utility
0	Pareto (0.002, 2)	0	1,000,000	90%
1	Pareto (0.002, 2)	1	1,333,333	90%
2	Pareto (0.004, 2)	2	1,166,667	90%
3	Pareto (0.004, 2)	3	1,000,000	90%
4	Pareto (0.004, 2)			
5	Pareto (0.002, 2)			

Table 4.5: Packet Drop-rate in Mixed Traffic Environment

Flow	Packet-drop Rate			Packet Drop Decreasing Rate	
	FCFS	One Sort	Complete Sort	FCFS to One Sort	FCFS to Complete Sort
0	0.104739191	0.067149423	0.052222139	35.89%	50.14%
1	0.058748473	0.034098935	0.024812795	41.96%	57.76%
2	0.064907788	0.04545949	0.03812368	29.96%	41.26%
3	0.043673486	0.029216252	0.022582083	33.10%	48.29%
4	0.048930915	0.034697449	0.028657637	29.09%	41.43%

table 4.5 and table 4.6 show that the improvement in drop-rate and jitter is not as much as in the first scenario. This is because the traffic from flow 5 sharing the same resources with the traffic from class 0, thus it has major impact on the delay of the class 0 traffic. This dilutes the effect of the sorting methods.

Table 4.6: Delay and Delay Variation in Mixed Traffic Environment for Flow 0

	FCFS	One Sort	Complete Sort	FCFS to One Sort	FCFS to Complete Sort
Average Delay	60.7893	60.4857	60.4575	0.50%	0.55%
Variation	3.90E+07	3.16E+07	2.97E+07	18.97%	23.83%

4.3.3 Scenario III Mixed Traffics with Proportional Differentiation

In this scenario, Both the topology and the parameters of senders, nodes, and receivers are the same as in the last scenario except that the normalized delay weight assigned for the real-time traffic is 2/3 while the non-real-time traffic is 1/3. The nodes are using SDF&ASM mechanism. This scenario proves that both sorting schemes can also works well with the proportional differentiated services algorithm.

Table 4.7: Packet Drop-rate in Mixed Traffic Environment with Proportional Differentiation

Flow	Packet-drop Rate			Packet Drop Decreasing Rate	
	FCFS	One Sort	Complete Sort	FCFS to One Sort	FCFS to Complete Sort
0	0.085358585	0.058677916	0.051161612	31.26%	40.06%
1	0.048147544	0.030203703	0.024238985	37.27%	49.66%
2	0.055193921	0.039127255	0.036327906	29.11%	34.18%
3	0.037090336	0.024847979	0.02163873	33.01%	41.66%
4	0.041760845	0.031756561	0.027763184	23.96%	33.52%

Table 4.8: Delay and Delay Variation in Mixed Traffic Environment with Proportional Differentiation for Flow 0

	FCFS	One Sort	Complete Sort	FCFS to One Sort	FCFS to Complete Sort
Average Delay	59.2621	58.8351	58.5904	0.72%	1.13%
Variation	3.81E+07	3.33E+07	3.22E+07	12.74%	15.64%

Since class 0, which includes flows 0 - 4, has a higher normalized delay weight than class 1, class 0 is assigned more bandwidth than class 1 when the node is in congestion. Because of that, the sorting scheme in this scenario is not as critical as in scenario II, which does not has delay differentiation. So the improvement of the packet-drop rate between the cases with and without applying the sorting scheme (table 4.7) is not as great as in scenario II. Nevertheless, the improvement is still considerable.

Chapter 5

Conclusions

In this thesis, we have proposed three novel QoS mechanisms to enhance the control of class-based service. Among these mechanisms, the Scheduler to Dropper Feedback (SDF) and the Adaptive Safety Margin (ASM) are designed for single-node QoS control, while sort-packet schemes are designed for end-to-end delay control.

The main advantage of SDF is that it provides a dynamic trade-off between QoS metrics. With this tradeoff, traffic classes that suffer un-proportionally higher delays because of the relaxation of the proportional constraint due to the absolute constraint, can receive compensation in terms of lower packet loss. Thus, SDF provides a fairer QoS treatments among traffic classes.

ASM is an adaptive feedforward mechanism based on Early Due Date (EDD) algorithm. By applying ASM, the time-critical traffic class can be dynamically pre-assigned more bandwidth. Therefore ASM reduces the packet head-drop rate. Comparing with the original EDD, ASM is more effective. By integrated with SDF, the queuing system can also cut down the total packet drop rate.

We have proposed two types of sorting schemes for the end-to-end delay scheduling. Both of them can effectively reduce the probability of delay deadline violation for time-critical

traffic classes. However, there is a trade-off between them. The one-sorting scheme has less computational overload while the complete-sorting scheme has a better deadline-violation performance.

There are two topics we would take to investigate in the further.

Bi-directional feedback

SDF provides a single direction feedback mechanism from scheduling to dropping. The bi-direction feedback is difficult because of the diversion problem. If this problem could be resolved by a simple mechanism, the bi-direction feedback may provide more flexible control mechanism between delay and packet loss metrics.

End-to-end Packet Loss

Beside the end-to-end delay study, the end-to-end packet loss is also a very important topic. Further work should be done to derive an effective end-to-end packet loss schemes in the future.

Bibliography

[1] Dovrolis C.; Ramanathan P., "A case for relative differentiated services and the proportional differentiation model," *Network, IEEE*, Volume: 13, Issue: 5, Sep/Oct 1999, Page(s): 26-34.

[2] Dovrolis C.; Stiliadis D.; Ramanathan, P., "Proportional Differentiated Services: Delay Differentiation and Packet Scheduling," *Networking, IEEE/ACM Transactions*, Volume: 10, Issue: 1, Feb 2002, Page(s): 12-26.

[3] Dovrolis, C.; Ramanathann, P., "Proportional Differentiated Services, Part II: Loss Rate Differentiation and Packet Dropping," *Quality of Service, 2000. IWQOS. 2000 Eighth International Workshop on 2000*, Page(s): 53-61.

[4] Liebeherr J.; Christin N., "Rate allocation and buffer management for differentiated services," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, Volume 40 , Issue 1 (September 2002), Special issue: Towards a new internet architecture, Pages: 89 - 110.

[5] Yang Chen; Hamdi, M.; Tsang, D.H.K.; Qiao, C., "Proportional QoS provision: a uniform and practical solution," *Communications, 2002. ICC 2002. IEEE International Conference*, Volume: 4, 2002, Page(s): 2363- 2367.

[6] Floyd, S.; Jacobson, V., "Random early detection gateways for congestion avoidance," *Networking, IEEE/ACM Transactions*, Volume: 1, Issue: 4, Aug 1993, Page(s): 397-413.

[7] Liu C.; Layland J., "Scheduling algorithms for multiprogramming in a hard real-time environment," *Journal of the ACM*, Vol. 20, No. 1, Jan. 1973, pp. 46-61.

[8] Shivendra S. Panwar; Don Towsley; Amherst Jack K. Wolf, "Optimal scheduling policies for a class of queues with customer deadlines to the beginning of service," *Journal of the ACM (JACM)*, Volume 35, Issue 4 (October 1988), Pages: 832 - 844.

[9] S. Bodamer, "A New Scheduling Mechansim to Provide Relative Differentiation for Real-time IP Traffic," *Proceedings of IEEE GLOBECOM 2000*, Vol. 1, pp. 646-650.

[10] T.Quynh et al., "Relative Jitter Packet Scheduling for Differentiated Services," *Proc. 9th IFIP Conf. Perf. Modeling and Eval. of ATM & IP Networks, 2001.*

[11] Kumar A.; Kaur J.; M.Vin H., "End-to-End Proportional Loss Differentiation," *Univ. of TX Austin, Tech. rep. TR-01-33*, Sept. 2001.

[12] Nandagopal, T.; Venkitaraman, V.; Sivakumar, R.; Bharghavan V., " Delay differentiation and adaptation in core stateless networks," *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, Volume: 2, 2000, Page(s): 421-430.

[13] Leung, M.K.H.; Lui, J.C.S.; Yau, D.K.Y., "Adaptive proportional delay differentiated services: characterization and performance evaluation," *Networking, IEEE/ACM Transactions*, Volume: 9 Issue: 6 Dec 2001, Page(s): 801-817.

[14] Ulf Bodin; Andreas Jonsson; Olov Schelen, "On Creating Proportional Loss-Rate Differentiation: Predictability and Performance," *Lecture Notes In Computer Science, Proceedings of the 9th International Workshop on Quality of Service*, Vol. 2092, Pages: 372 - 388.

[15] Son Vuong; Xizheng Shi, "A proportional differentiation service model for the future Internet differentiated services," *Communication Technology Proceedings, 2000. WCC - ICCT 2000. International Conference*, Volume: 1, 2000, Page(s): 416-423.

[16] Jung-Shian Li; Hsing-Chien Lai, "Providing proportional differentiated services using PLQ," *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, Volume: 4, 2001, Page(s): 2280-2284.

[17] Jingdi Zeng; Ansari, N., "An enhanced dropping scheme for proportional differentiated services," *Communications, 2003. ICC '03. IEEE International Conference*, Volume: 3, 11-15, May 2003, Page(s): 1897- 1901.

[18] Li J.-S.; Mao C.-S., "Providing flow-based proportional differentiated services in class-based DiffServ routers," *Communications, IEE Proceedings*, Volume: 151, Issue: 1, 28 Feb. 2004, Page(s): 82- 88

Appendix A

Abbreviation List

ASM	Adaptive Safety Margin
DDP	Delay Differentiation Parameter
DiffServ	Differentiated Services
EDD	Earlier Due Date
FCFS	First Come First Serve
HPD	Hybrid Proportional Delay
IntServ	Integrated services
ISP	Internet Service Provider
JoBS	Joint Buffr Management and Scheduling
LDP	Loss-rate Differentiation Parameter
LHT	Loss History Table
PAD	Per-class Average Delay
PLR	Proportional Loss Rate
QDP	Quality Differentiation Parameter
QoS	Quality of Service
RED	Random Early Detection
RSVP	Resource Reservation Protocol
SDF	Scheduling to Dropping Feedback
STE	Shortest Time to Extinction
WTP	Waiting Time Priorities