12-1-2003

Preserving conformance for GCRA regulated flows

Deryck Hong

Preserving Conformance for GCRA Regulated Flows

by

Deryck Hong

Thesis submitted to the Department of Computer Engineering
in
Partial fulfillment of the requirements for the degree of
Master of Science
At the
Rochester Institute of technology
December, 2003

Dr. Shanchieh Jay Yang
Primary Advisor – Dept. of Computer Engineering, RIT

Dr. Muhammad Shaaban
Secondary Advisor – Dept. of Computer Engineering, RIT

Dr. Juan Cockburn
Secondary Advisor – Dept. of Computer Engineering, RIT

Release Permission

Rochester Institute of Technology

I, Deryck Hong, hereby grant permission to RIT to reproduce and to distribute copies of this thesis document in whole or in part.

_____ Date: 1/12/04

Deryck Hong

Abstract:

Traffic policing/shaping has been employed at the edge of networks to ensure proper provisioning of network resources and Quality of Service (QoS) guarantees. As shown in this thesis, however, network flows that have been regulated at the network edge based on traffic descriptors, e.g., GCRA, may still become non-conforming in the network core, depending on the packet scheduling algorithm used. Two supplemental schemes to scheduling algorithms are proposed and analyzed in this thesis to ensure conformance for GCRA regulated flows. The first scheme is to add an additional traffic regulator to shape the traffic more aggressively than required constraints before entering the scheduler. The second scheme explicitly computes the eligible departure time for the next packet of each flow in the scheduler. Performance achievable by both schemes, in terms of the percent non-conforming packets and the average delay, are investigated via simulation, when implemented for the First Come First Serve (FCFS) and the Weighted Fair Queuing (WFQ) schedulers.

Acknowledgement:

This thesis could not be completed without Dr. Yang's consistence and willingness of help. His remarkable insight contributed substantially to the results of this thesis. I would like to express my gratitude and thanks to Dr. Yang for his guidance, advice and help from choosing the topic until the finish of this thesis. I will always be grateful to him for helping me on my first significant research. Also I would like to thanks Dr. Shaaban, and Dr. Cockburn to be the members of my thesis committee.

Glossary

ATM: Asynchronous Transfer Mode.

BT: Burst Tolerance.

CDVT: Cell Delay Variation Tolerance.

DRR: Deficit Round Robin.

ET: Eligible Timer.

ET+FER: Eligible Timer with Front-End Regulator.

FCFS: First Come First Serve.

FER: Front-End Regulator.

GCRA: Generic Cell Rate Algorithm.

GPS: Generalized Processor Sharing.

MBS: Maximum Burst Size.

PCR: Peak Cell Rate.

QoS: Quality of Service.

RED: Random Early Detection.

SCR: Sustained Cell Rate.

TAT: Theoretical Arrival Time.

TBF: Token-leaky Buffer.

TCP: Transmission Control Protocol.

WFQ: Weighted Fair Queuing.

WF2Q: Worst-Faired Weighted Fair Queuing.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

High-speed networking has introduced opportunities for new multimedia applications such as video conferencing, scientific visualization, and medical imaging. These applications have stringent network performance requirements in terms of throughput, delay, delay jitter, packet loss, or any combination of above. To serve these applications, researchers have proposed suites of admission control mechanisms [6, 7, 12, 13], traffic shaping/policing policies [3, 15, 19] and packet scheduling algorithms [2, 8]. These algorithms together ensure proper allocation of network resources, such that multi-level fine-grained Quality of Service (QoS) requirements maybe satisfied. To properly use these algorithms together, the network and the users shall agree upon their usage of network resources. That is, how much and how fast a traffic stream may send in their packets. The idea then is that as long as the user sends in streams of packets that conform to the traffic descriptor, the network should guarantee the QoS agreements. Unfortunately, network flows that are conforming to the traffic descriptors at the network edge, however, may not conform in the network core due to multiplexing of packets from various flows. This in turn may lead to packet loss or conservative provisioning of network resources. This thesis investigates how to ensure that the conforming flows will not be penalized due to the scheduling algorithms that are implemented in the network core.

Unlike the voice traffic on traditional telephone networks, multimedia real-time traffic is bursty and harder to predict. Two extreme approaches can be taken to ensure QoS for such bursty traffic. First, the network can allocate the absolute maximum resources, e.g., bandwidth that each flow requires. Apparently, this may lead to inefficient resource allocation hence not cost effective for network providers. Alternatively, the network can allocate the average amount resource each flow may require throughout its lifetime. Although this approach has been shown to be 'stable', i.e., no flow will stay in the network forever, temporary congestion may happen and lead to packet losses if the buffer size were not big enough. Typical solution may lie in between these two solutions, by the use of admission control, traffic shaping / policing, and packet scheduling.

The role of admission control is to determine whether to accept the connection requests based on information such as the performance requirements, the existing volume of network traffic, and the estimated or explicitly specified traffic descriptors for the new request. The admission control decision may be as straightforward as a simple inequality calculation: accept the connection request if the sum of the bandwidth usage of total traffic flows, including the new request, is smaller than the network's total bandwidth, and vice versa. The traffic descriptor serves the purpose not only for the admission control, but also for the network to monitor whether the admitted connections send in traffic in the manner as specified. Traffic descriptor, e.g., Token bucket, or Generic Cell Rate Algorithm (GCRA) [18], is a set of parameters that characterizes how fast and how much each traffic source may send to the network. One may categorize the traffic descriptors: as deterministic and stochastic descriptors. Deterministic traffic descriptors contain "hard" restrictions, hence no tolerance for violations. In this case, the network has to allocate precisely the amount of bandwidth that is specified in the traffic descriptor, and the packet stream needs to arrive absolutely no faster and no more than specified. By contrast, stochastic traffic descriptors characterize the arrival streams as random process. For example, a traffic stream may be described as Markov On-Off process with average rate of 100 Kbps and peak rate of 2 Mbps. While the stochastic traffic descriptors provide flexibility for the admission control algorithms and lead to efficient network resource utilization, it is easier to implement traffic regulator (policer or shaper) for deterministically described traffic streams in either hardware or software.

Traffic policing or shaping is typically done at the network edge so as to avoid high complexity in the network core, and therefore enables high-speed processing and easy upgrades of the network. Once a traffic flow enters the network, it will likely traverse through multiple network nodes. The scheduling algorithms implemented in these network nodes determine the service schedule of packets from different traffic flows to ensure performance guarantees and efficient resource utilization. Packets that are multiplexed through these schedulers, however, may not conform to their traffic descriptors, even though they were conforming when they entered the network. This is due to that scheduling algorithms can spit out the backlogged burst of packets for a connection more than that specified in the traffic descriptor. When such non-conforming

bursts reach other nodes in the network, either a conservative resource provision is needed or some of the packets may be dropped due to buffer overflow. Even in the case where no packet is dropped within the network core, additional packet delay or dropping may be encountered once the non-conforming bursts enter the next network. This thesis identifies the scenarios where the network core routers cannot keep 'their promise' made during admission control. Two schemes are proposed in this thesis as supplement of common scheduling algorithms, such that conforming packet streams may experience zero or minimal packet drops.

The organization of the thesis is as follows. An overview of traffic shaping and policing for regulating traffic flows is provided in Chapter 2. A discussion of why the scheduling algorithms that are implemented in network elements may produce non-conforming packets is given in Chapter 3. Chapter 4 introduces the two proposed conformance-preserving schemes, and illustrates how to derive corresponding parameters for these schemes when they are used to supplement the First Come First Serve (FCFS) and the Weighted Fair Queuing (WFQ) schedulers. Simulator design and simulation results to exhibit the performance achieved by the proposed schemes are discussed in Chapter 5. Potential challenges to implement the proposed schemes and the scheduling algorithms in network processors are discussed in Chapter 6, followed by the conclusion and a highlight of future work in Chapter 7.

Chapter 2: Traffic Shaping / Policing

Gigabit speed transmission capacities have paved the way for many exciting multimedia applications, such as video conferencing and real-time distributed computing, to be supported on computer networks. Most of these applications have stringent QoS constraints in terms of throughput, delay, delay jitter, packet loss, or any combination of above. In order to satisfy QoS requirements of each application, it requires an integration of schemes including admission control, traffic enforcement and shaping at the edges of the network, and multiclass packet scheduling inside network nodes. Traffic enforcement schemes have a major role in a resource sharing environment because end users may, inadvertently or otherwise, attempt to exceed the traffic agreements specified at the time of connection setup. Were a few flows sending more than specified traffic, an unfair share of network resources may happen and lead to penalizing conforming flows. Therefore, it is desired to have only "well-formed" traffic arriving to the network core, which may be enforced either by the users themselves or at the network edge. One way to enforce the end system to send "well-formed" traffic is to employ traffic shaping/policing schemes. Many traffic shaping/policing schemes, such as Leaky Bucket and Jumping Window, have been proposed and analyzed [21].

Most of the traffic shaping/policing schemes consider deterministic traffic descriptors since they are easy to implement and the parameters are determined upon connection establishment. Table 2.1 compares the differences between a traffic shaper and a traffic policer. A traffic policer sits at the network side and acts as a "policeman" to prevent the end users from breaking the agreed traffic specification. If the measured traffic exceeds the negotiated parameters, the traffic policer may drop non-conforming packets – see Figure 2.1. With traffic shaping, excessive packets are buffered, hence packets are delayed rather than dropped in order to meet the specifications.

	Shaping	Policing
Objective	Packets are delayed when they exceed the required specifications.	Packets are dropped when they excess the required specifications.
Bursts	Packet bursts are smoothed out.	Packet bursts are chopped.
Advantages	Packets are buffered instead of dropped (unless buffer overflow), so no retransmissions are required.	Excessive packets are dropped; no extra queuing delay is introduced.
Disadvantages	Queuing delay is introduced.	Reduce the overall output rate when too many packets are dropped.

Table 2.1: Comparison between Traffic Shaping and Policing

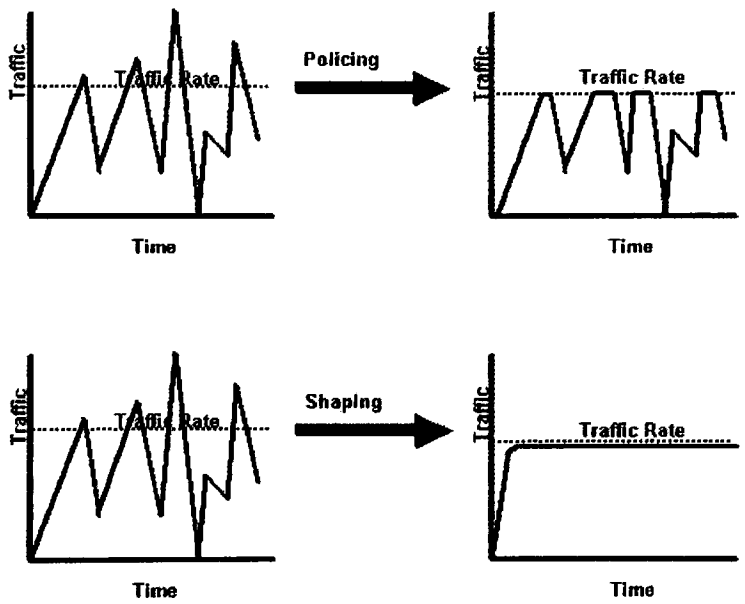

Figure 2.1: Comparison between Traffic Shaping and Policing [3]

In essence, the different effects induced by the traffic shaping and policing lie in the amount of buffer space used. In the case of the shaper, a large amount of buffer space is used to delay the packets and not causing buffer overflow. By contrast, the policer utilizes relatively small buffer space, so that packets are dropped when the buffer is full. In the sequel, the commonly used Generic Cell Rate Algorithm (GCRA) regulator and how it will be used for this thesis will be introduced.

2.1 Generic Cell Rate Algorithm (GCRA)

GCRA is used to model the packet stream arrivals and to check the arrival times of packets so as to determine whether each of them is conforming. It was initially proposed for Asynchronous Transfer Mode (ATM) network where small fixed size cells are used instead of general variable size packets. A packet stream arrival may be modeled using a generic $GCRA(I,L)$ or $GCRA(R,N)$ model. In the $GCRA(I,L)$ model, packets should arrive with at least I time unit spacing, but no more than L time units before the theoretical arrival time (TAT). The TAT for the next packet is updated upon each packet arrival, and equals to $I+t$, where t is the latest packet arrival time. A $GCRA(I,L)$ model can service two different traffic categories: one that specifying peak cell rate (PCR) or one that specifying sustained cell rate (SCR) for ATM networks. A $GCRA(I,L)$ descriptor that specifies traffic flows with PCR and SCR is equivalent to using $GCRA(1/PCR, CDVT)$ and $GCRA(1/SCR, BT)$, respectively, where PCR, SCR, CDVT and BT are defined in Table 2.2. A simple example of a bursty traffic with Maximum Burst Size (MBS) of 4 packets is shown in Figure 2.2.

Peak Cell Rate (PCR)	The maximum rate that packets can be transmitted.
Sustained Cell Rate (SCR)	The average packet rate as measured over an infinite long time interval.
Cell Delay Variation Tolerance (CDVT) and Burst Tolerance (BT)	These are the variations of the inter-packet-arrival time that the source can transmit at any give rate. They are typically used in accompany to the PCR and SCR parameters, respectively.

Table 2.2: Definitions of PCR, SCR, CDVT and BT

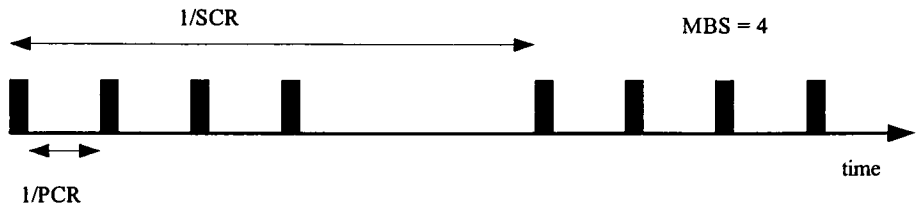

Figure 2.2: An example of PCR, SCR, and MBS

Notice that in Figure 2.2, the MBS is in fact the parameter N in the $GCRA(R,N)$ model [1]. The $GCRA(R,N)$ model is equivalent to the $GCRA(I,L)$ model, where R (packets/sec) is the average rate of the traffic flow, and N (packets) is the maximum burst size (MBS). The uses of the parameters of I, L, R, and N are interchangeable, and can be stated as follows.

$$R = \frac{1}{I}. \tag{2.1}$$

$$N = \left\lfloor \frac{L}{I - T_{trans}} \right\rfloor + 1. \tag{2.2}$$

where T_{trans} is the time it takes to transmit one packet.

The two GCRA traffic descriptors have been adopted for the Internet and other networks due to their ease of use. In the sequel, we shall illustrate how one may determine the conformance based on the $GCRA(I,L)$ model.

Upon each packet arrival, the packet arrival time and its relation to the previous packet arrival time, etc, is checked against the GCRA constraints to determine whether the packet conforms. Three possible scenarios could happen upon a packet arrival. Figure 2.3 provides the flow chart that determines the conformance of the arriving packet under the three scenarios and is illustrated below. Let $T\_arrival(K)$ be the arrival time of the K^{th} packet, and $TAT(K)$ be the theoretical arrival time of the K^{th} packet.

1. $T\_arrival(K) \geq TAT(K)$: The arriving packet is conforming., and the TAT for the $(K+1)^{th}$ packet is updated to be $I + T\_arrival(K)$.

2. $T\_arrival(K) \geq TAT(K) - L$: The arriving packet is conforming, and the TAT for the $(K+1)^{th}$ packet is updated to be $I + TAT(K)$.
3. $T\_arrival(K) < TAT(K) - L$: The arriving packet is non-conforming, and the TAT for the next packet is unchanged.

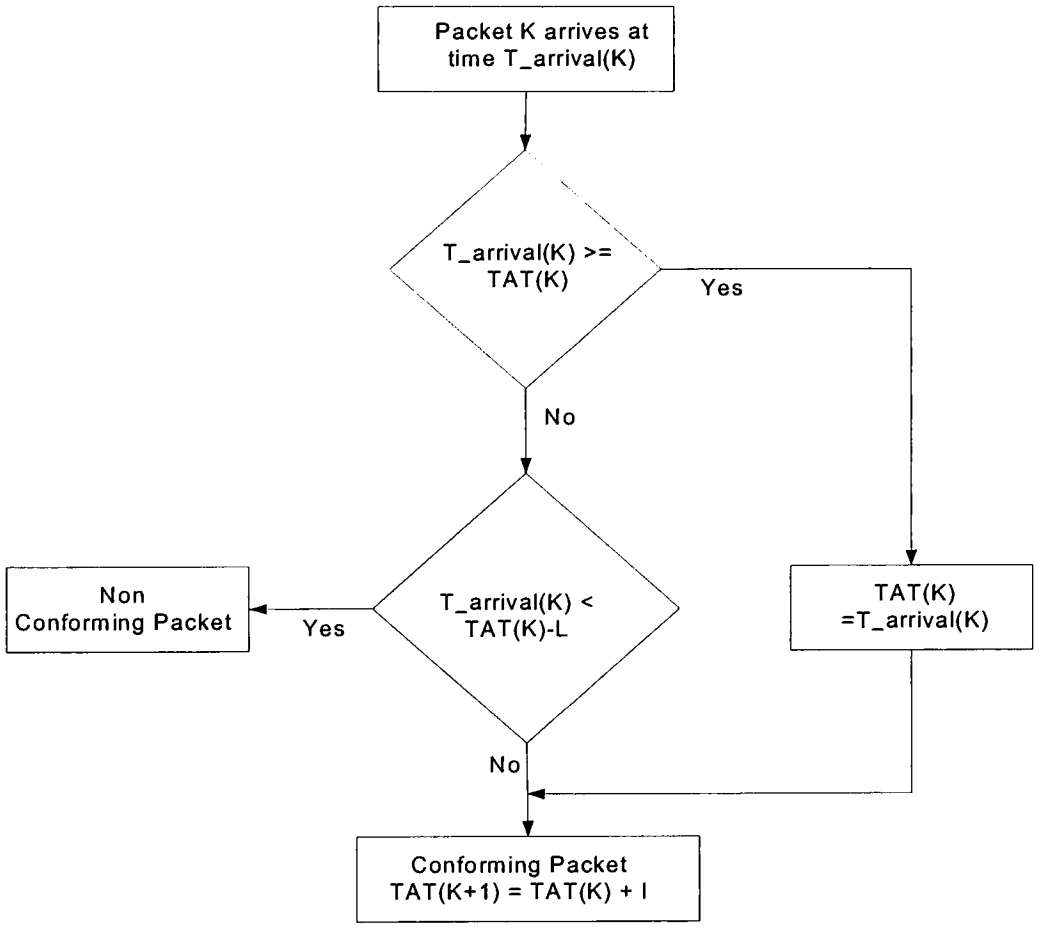

Figure 2.3: Algorithm that determines the conformance of the arrival packet based on the GCRA(I, L) constraint.

2.3 Leaky and Token Bucket

Two common implementations of a GCRA regulator are leaky bucket and token bucket. For the leaky bucket, the size of the bucket is $L + I$. When a packet arrives, if the packet causes the bucket to overflow, then the packet is said to be non-conforming and is discarded. Otherwise, the packet is said to be conforming, and the occupancy of the bucket is incremented by I unit. The bucket is leaking I unit per time unit. A token bucket is very similar to a leaky bucket. A token bucket can contain maximum of $L/I + 1$ tokens. Tokens are generated with a constant rate of $1/I$. If the bucket is full, the incoming tokens will be thrown away. For each conforming packet, a token is removed from the token bucket. An arriving packet is non-conforming if there are no tokens in the token bucket. Leaky bucket and Token bucket are illustrated in Figures 2.4 and 2.5. A detailed implementation of the GCRA regulator in our simulator is discussed in Chapter 5.

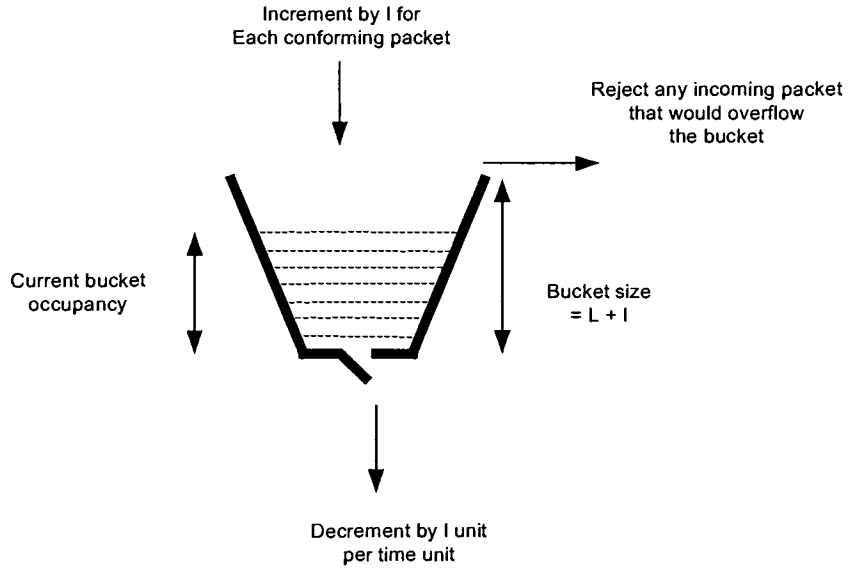

Figure 2.4: Leaky Bucket

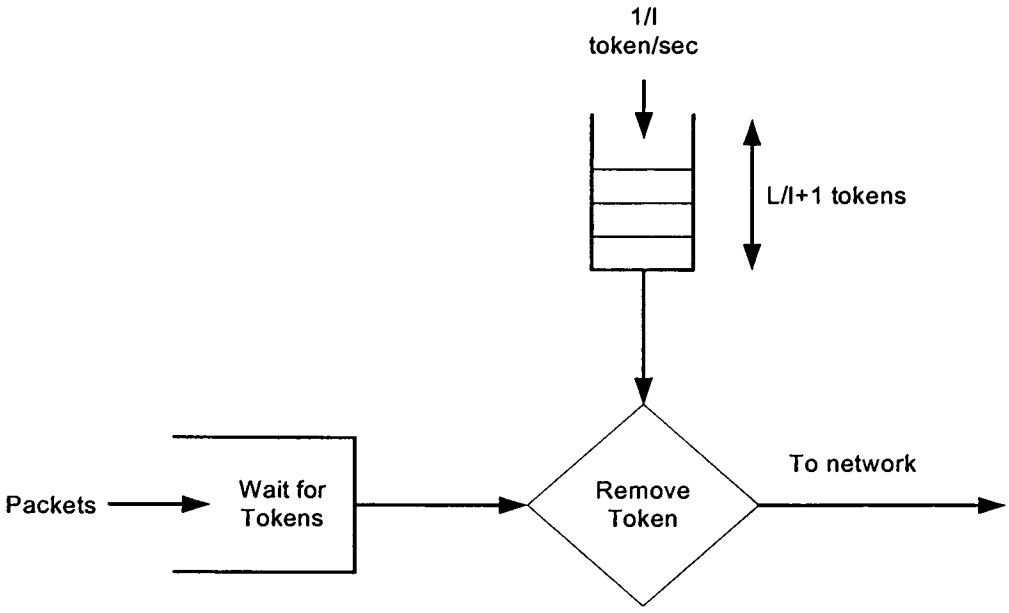

Figure 2.5: Token Bucket

Chapter 3: The Conformance Preservation Problem

As described in Chapter 1, GCRA conforming packet streams entering a core router may become non-conforming when they exit. Whether or not and how severe a conforming packet stream will be 'made' non-conforming depends on the packet scheduling policy employed in the router as well as the traffic mix in the router. This chapter will briefly describe commonly used packet scheduling algorithms and discuss how a conforming packet stream can be 'made' non-conforming.

3.1 Packet Scheduling Algorithms

A packet scheduling discipline is used to resolve contentions among network traffic flows by deciding the service order and managing queues of service requests. Depending on the scheduling disciplines used and the type of traffic it is mixed with, a traffic flow may perceive different levels of performance in terms of the bandwidth, delay, and the packet loss, which together determine whether a fair share of the link bandwidth and buffer is given to each flow. An ideal scheduling discipline would be easy to implement, provide fairness and performance bounds, and allow easy admission control decisions. It is however a challenging task to have a scheduling discipline achieving all these characteristics. More than often, a scheduling discipline aiming at achieving one characteristic would violate some other characteristics. For example, as will be discussed below, the Generalized Processor Sharing (GPS) scheduling discipline provides the ideal fairness among different traffic flows, but is impossible to implement in practice for packet scheduling.

The FCFS scheduling algorithm is one of the simplest packet scheduling algorithms that can be implemented. It is non-preemptive, and serves packets in the order of their arrival times. FCFS is easy to implement, but it does not provide any bandwidth guarantee or fairness among different input traffic flows. The GPS [13] scheme is an ideal scheduler that allocates weighted fair share of bandwidth to each flow depending on the weight agreed between each user and the network. GPS scheduler however is

infeasible in common practice due to the granularity of packet sizes. Among many others, Deficit Round Robin (DRR) [21], Weighted Fair Queuing (WFQ) [13], and Worst-case Fair Weighted Fair Queuing (WF2Q) [8] packet scheduling algorithms attempt to approximate the ideal fairness achieved by the GPS. Before discussing the 'conformance preservation' problem, these common packet scheduling algorithms are briefly reviewed in the sequel.

GPS (Generalized Processor Sharing):

The Generalized Processor Sharing (GPS) is a flow-based multiplexing discipline. GPS scheduler serves each active flow at every instant with bandwidth respect to its reserved weight; in addition, the excess bandwidth from flows that are not using their reservations is distributed among active flows in proportion to their individual reservations. Let $J(t)$ to be a set of active flows at time t, then the bandwidth that is allocated to flow j, where $j \in J(t)$, can be calculated using the following equation.

$$R_j = \frac{w_j}{\sum_{j \in J(t)} w_j} * C . \qquad (3.1)$$

R_j : bandwidth allocated to flow j
w_j : weight of flow j
$J(t)$: set of active flows at time t
C : total capacity

In a GPS system, all flows are served simultaneously, which makes it infeasible when dealing with packets from different flows.

DRR (Deficit Round Robin):

The DRR algorithm serves packets from different flows in a round-robin fashion while accounting for the flow weights and the differences in packet size. Each flow has a quantum number (in bytes), which is associated to flow's weight to determine the access duration of each flow in each round. For example, consider that a DRR scheduler is serving three flows (flow 1 to 3) with weights of 0.1, 0.1, and 0.8, respectively and the packet sizes from all three flows are the same. Then one set of quantum numbers for

flow 1 to 3 can be 100, 100, and 800 in bytes, which is the allocated amount of data to be served in each round. The following is the pseudo code for the DRR algorithm.

$DC_j(k)$: Deficit counter of flow j at round k
$L_j(k)$: Head of line packet size of flow j at round k
Q_j : Quantum associated with flow j

if $DC_j(k) \geq L_j(k)$

serve head of line packet

$DC_j(k+1) = DC_j(k) - L_j(k)$

load $L_j(k+1)$ if there is next packet

$DC_j(k+1) = DC_j(k) + Q_j$

if flow j is empty

$DC_j(k) = 0$

WFQ (Weight Fair Queuing):

WFQ is a packetized version of GPS. In WFQ, the packets are served in approximately the same order as they would have been served in GPS by referring to a virtual time ($V(t)$). WFQ calculates the expected finish times of packets based on the virtual time, and serves the packet with the smallest finish time among those from all flows. The following equation is used to calculate the finish time of a packet (the total link capacity is assumed to be 1 for simplicity).

$$F_k^j = Max(F_{k-1}^j, V(a_k^j) + \frac{L_k^j}{w_j}. \quad (3.2)$$

w_j : Weight of flow j
a_k^j : Arrival time of packet k from flow j
L_k^j : Length (size) of packet k from flow j
F_k^j : Finish time of packet k from flow j
$V(t)$: virtual time at actual time t under GPS

The function $V(t)$ represents the progression of the virtual time in the GPS model and is defined as follows.

$$\frac{d}{dt}V(t) = \frac{1}{\sum_{j \in J(t)} w_j}. \tag{3.3}$$

This means that the more inactive flows, the faster that the virtual time progresses, since the remaining active sessions can get more service per round.

WF2Q (Worst-case Fair Weighted Fair Queuing):

While it has been shown that the actual packet departures in a WFQ system may fall behind GPS by only one packet size in the worst case [13]. WFQ can actually be far ahead of GPS in terms of the number of bits served for a session [8]. These facts can indeed lead to unfairness in the worst case scenario. The goal of WF2Q, therefore, is to eliminate such unfairness [8]. In a WF2Q system, when choosing the next packet for service at a time instant t, it only considers the packets that have started receiving service in the corresponding GPS system rather than choosing among all the packets in the system as in WFQ. Instead of using the finish time of a packet in GPS for WFQ, WF2Q actually uses the start time of a packet that would be served in GPS to determine the service order of a packet. The start time of packet k from flow j is determined based on the following equation.

$$S_k^j = Max(F_{k-1}^j, V(a_k^j)). \tag{3.4}$$

where F_{k-1}^j and $V(a_k^j)$ is the finish time of $(k-1)^{th}$ packet from flow j and the virtual time upon the arrival of the k^{th} packet from flow j, respectively, calculated in the same way as WFQ does. When there is more than one packet are eligible for service, the packet with smallest finish time will be chosen to be serviced.

3.2 Performance Preservation Problem

A sequence of packets arriving to a network node may experience distortions in their service pattern as compared to their arrival pattern. This happens when a larger number of backlogged packets exist in the scheduler, and are waiting to be served. In

particular, when bursts of packets of different flows arrive to a scheduler at the same time, a maximum number of backlogged packets may be created. While processing of those backlogged packets, packets arrive from another flow may become a big burst of back-to-back packets that are waiting to be served by the scheduler. This big burst can in fact be larger than the maximum burst specified in the traffic descriptor agreed upon connection admission. Figure 3.1 exhibits an example to illustrate how a bigger-than-specified burst may occur when the FCFS scheduler is used.

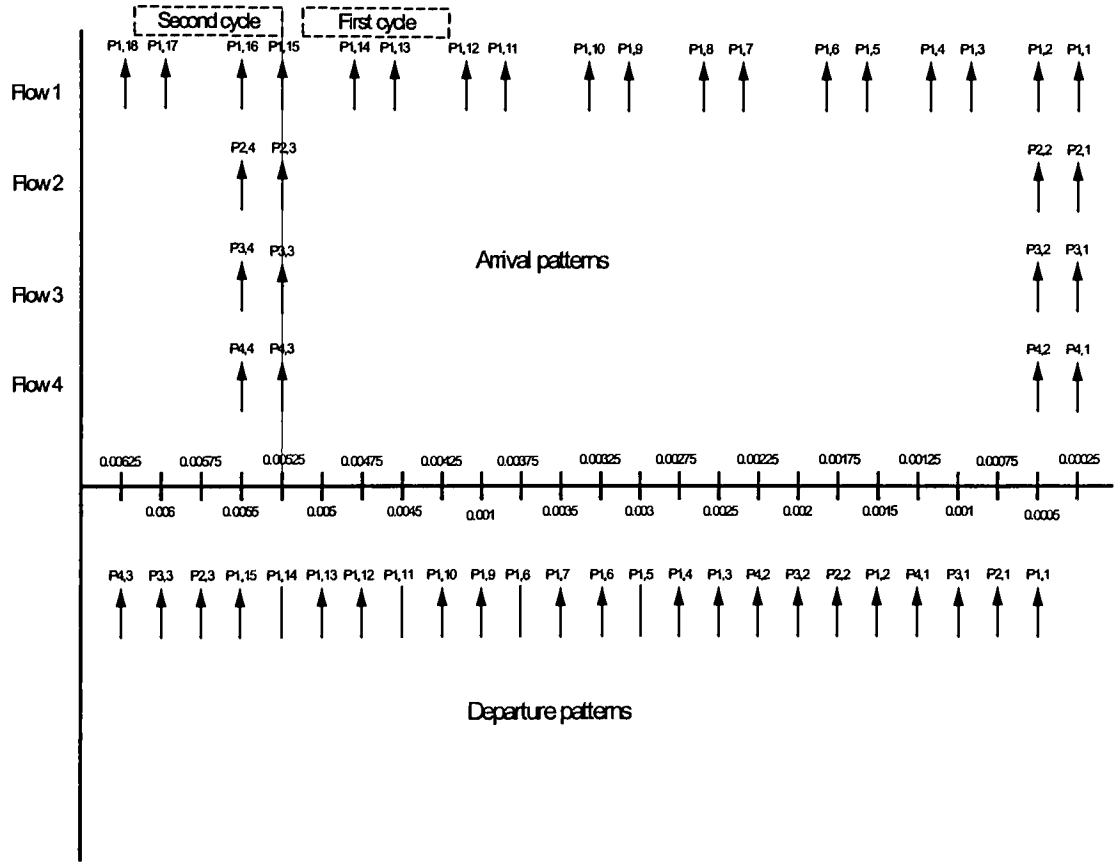

Figure 3.1: An example of Arrival and Departure Patterns

In this example, there are total of four flows with the average packet arrival rates of 0.7C, 0.1C, 0.1C, and 0.1 C for flow 1 to 4, respectively, where C is the total service capacity. The burst sizes of all four flows are assumed to be the same and equal to 2 packets. Also assume that flow 1 would have the highest priority if more than one flow

have a packet ready to be departed. The example shown in Figure 3.1 illustrates that there are total of 20 packets departed from the first cycle, and 4 of them are non-conforming packets. Let $P(i, j)$ denote the j^{th} packet from the i^{th} flow. The non-conforming packets are $P(1,5)$, $P(1,8)$, $P(1,11)$, and $P(1,14)$ out of the overall 13 back-to-back packets from flow 1. If this arrival patterns continue in the second cycle, 4 more packets are expected to be non-conforming. This means that 20% of total packets would be non-conforming packets. Since all non-conforming packets are from flow 1, flow 1 would have about 4/13 = 29% of total packets that are non-conforming.

The example exhibited above is an extreme case where a significant number of packets become non-conforming while passing through the FCFS scheduler. In more common cases and with more 'advance' schedulers, fewer, yet still non-zero number of packets may suffer. Figure 3.2 exhibits the % non-conforming packets that can be created out of four flows, when various common scheduling algorithms are used and as the traffic load increases. The % non-conforming packets (y-axis) is defined as the ratio of the number of non-conforming packets with respect to the total number of packets that enter the scheduler. As shown in Figure 3.2, FCFS scheduler does worse than other schedulers in terms of the % non-conforming packets. Recall that packets in the FCFS scheduler do not have any bandwidth guarantee, neither any delay bounds. In fact, without service isolation among flows, a flow in the FCFS scheduler may experience a large delay due to other flows sending in bursts before it, as the case shown in Figure 3.1. This in turn will cause a large portion of packets to be non-conforming. Since DRR, WFQ, and WF2Q all provide certain level of fairness among flows, it is less likely to have flows create additional delay to each other and cause non-conforming packets, nevertheless, a non-negligible number of over 5% of packets become non-conforming even with DRR, WFQ and WF2Q in the heavy load regime, as shown in Figure 3.2. The increasing of the traffic load, as expected, worsens the problem of creating non-conforming packets. Notice that, however, when it is lightly loaded, e.g., at 30%, all schedulers would achieve zero non-conforming packets.

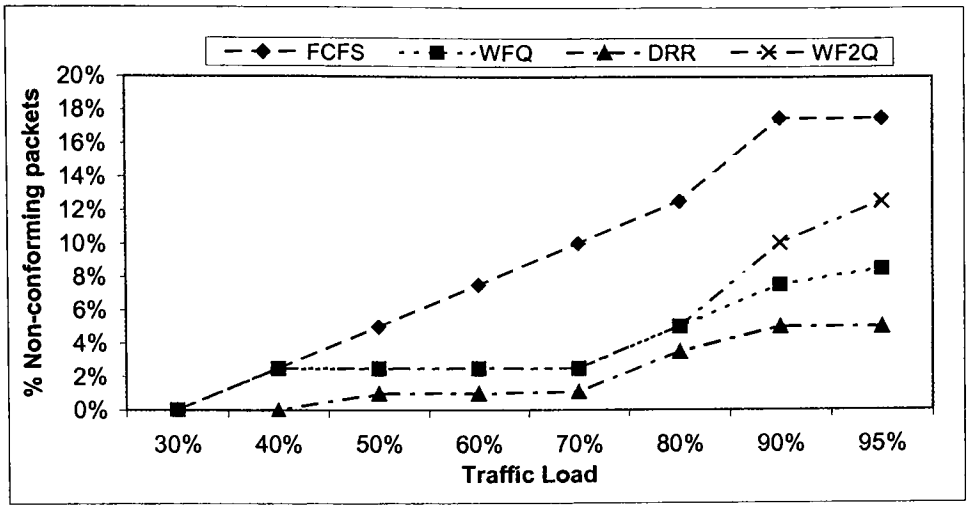

Figure 3.2: % non-conforming packets for common scheduling algorithms as traffic load increases.

Chapter 4: Conformance Preservation Schemes

This chapter proposes two schemes that preserve the conformance for packet streams when passing through the FCFS and WFQ schedulers. The first scheme consists of a 'front-end regulator' that restricts the number of back-to-back packets that may enter a scheduler and therefore controls the undesirable backlog. The second scheme involves explicit computation of the eligible service time for the next-in-line packet of each flow in the scheduler. Notice that the author of [1] also proposes the use of a front-end regulator for priority queues and WFQ schedulers. His work, however, focuses on buffer requirement of the scheduler, and does not provide sufficient information on the resulting performance of the proposed scheme. In fact, this chapter will show that the front-end regulator approach only reduce the number of non-conforming packets, but cannot guarantee 'no non-conforming packets' for conforming arriving packet streams. Detailed discussion on how to obtain system parameters for the two proposed schemes will also be included in this chapter.

4.1 Front-end Regulator

The Front-end Regulator (FER) is a traffic shaper that regulates each incoming packet stream, which already conforms to its GCRA specifications, based on even more stringent GCRA constraints. The goal of FER is to limit the number of back-to-back packets, i.e., burst size that may be accumulated in the queue when an arriving flow of packets are waiting to be served by a scheduler. This upper bound on burst size of a flow shall equal to the flow's GCRA descriptor and thus prevent the flow from violating its contract. Therefore, the question is how stringent one should 'shape' the incoming traffic streams such that each flow can have its maximum service burst equal to or less than its specified burst size N. Recall the $GCRA(R,N)$ model, which restrict packet streams in terms of their average arrival rates and burst sizes. In order to prevent unnecessary reduction in their service rates, FER scheme only shapes the incoming traffic streams in

terms of their burst sizes, and have their average rates remain constant[1]. That is, the FER scheme will employ a front-end GCRA regulator with parameter R_i and $\overline{N_i}$ for each flow i such that the flow i will have its maximum service burst size N_i^* equal to N_i, the actual GCRA burst size constraint of flow i. The sequel details the derivation process for finding the value of $\overline{N_i}$, in the case of FCFS and WFQ schedulers. For simplicity, all packets are assumed to have the same size. Let C denote the total capacity of the system and assume the scheduler is stable, i.e., $\sum_{j \in J} R_j \leq C$, where J is the set of traffic flows entering the scheduler. The goal is to compute the maximum burst size, N_i^*, that may be experienced by flow i, assuming that each of the flows entering the scheduler conforms to $GCRA(R_i, \overline{N_i})$. By setting $N_i^* = N_i$, the value for $\overline{N_i}$ can be found to set up the front-end regulator for flow i.

4.1.1 Front-end Regulator for FCFS scheduler

The architecture of the FER for FCFS scheduler is shown in Figure 4.1. Each incoming traffic flow has a dedicated buffer at the regulator, in which the packets are stored. The regulator holds each packet until it is eligible for transmission, where a packet of flow i is said to be eligible if it does not violate $GCRA(R_i, \overline{N_i})$. When a packet becomes eligible, it is put into the scheduler buffer and ready to be served. There is only one buffer for the FCFS scheduler.

[1] Experiments have also been conducted and exhibit that it does not help to shape the incoming packet streams in terms of their average arrival rates.

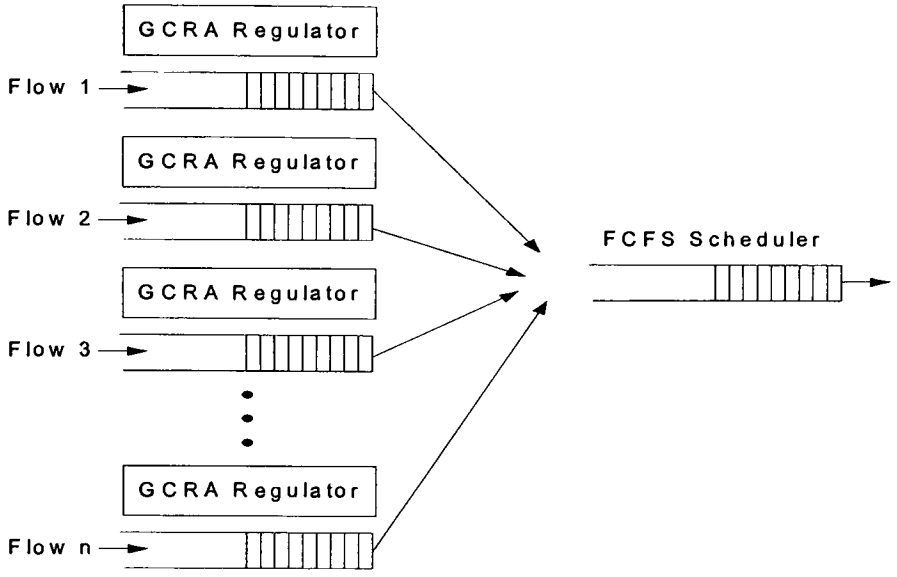

Figure 4.1: Architecture of the FER for FCFS scheduler. Incoming packets are placed in a per connection buffer until the GCRA traffic regulator determines that they are eligible for transmission (FCFS scheduler).

Since the FCFS scheduler services packets in the order of their arrival times, the maximum burst size that can be created for a given flow i is the number of packets that can arrive from flow i during the longest continuous time window, within which no packets from other flows would arrive and the FCFS queue never empties. To find this longest time window, one needs to find (1) the maximum number of packets from flows other than flow i that can exist in the queue at any point of time, and (2) the number of packet that can arrive from flow i to the queue before the queue empties. Considering a time window T_w, the maximum number of packets that can arrive to the scheduler from flow j within T_w is

$$A_j(T_w) = K_j(T_w) * \overline{N_j} + (T_w - K_j(T_w) * \overline{N_j} / R_j) * C . \qquad (4.1)$$

where $\max[(T_w - K_j(T_w) * \overline{N_j} / R_j) * C] = \overline{N_j}$, and $K_j(T_w) = \left\lfloor \frac{T_w * R_j}{\overline{N_j}} \right\rfloor$. The term $K_j(T_w)$ is the number of full bursts that can be sent within the time window T_w. The maximum number of full bursts that can be sent is $K_j(T_w)+1$, if $(T_w - K_j(T_w) * \overline{N_j} / R_j) * C = \overline{N_j}$.

From the i^{th} flow's perspective, the maximum number of packets that can arrive from all other flows before flow i starts sending in its maximum burst is

$$\overline{A_i}(T_w) = (\sum_{j=1}^{n} A_j(T_w)) - A_i(T_w). \tag{4.2}$$

Meanwhile during the time window T_w, the number of packets can be processed by the scheduler is $D(T_w) = C * T_w$. Therefore, the maximum number of packets in the FCFS queue from flows other than i is:

$$Q_i(T_w) = \overline{A_i}(T_w) - D(T_w). \tag{4.3}$$

The question now is what T_w value may lead to the largest $Q_i(T_w)$. It turns out that the largest $Q_i(T_w)$ happens when T_w equals to the time it takes for the flow with the largest burst size (among all flows except i) to send in one burst of packets. The reason for that is, when a flow is sending its burst, the packets are sent with peak rate of C. This means that as long as there is at least one flow is sending the burst, the number of packets in FCFS queue either stays unchanged or increases. The largest $Q_i(\max_{j \in J/i}(N_j)/C)$ in turn leads to the longest time window during which flow i may send in the most back-to-back packets that will be served by the FCFS scheduler. Notice that it takes

$$T_{i,1} = \frac{Q_i(\max_{j \in J/i}(N_j)/C)}{C}. \tag{4.4}$$

amount of time to serve $Q_i(\max_{j \in J/i}(N_j)/C)$ number of packets. During this time frame, flow i may send in at most $A_i(T_{i,1})$ packets. While the server is serving these $A_i(T_{i,1})$ packets, which takes $T_{i,2} = A_i(T_{i,1})/C$ amount of time, additional packets from flow i may arrive to the queue. The maximum overall number of packets that can arrive from flow i during this $T_{i,1} + T_{i,2}$ time period is again $A_i(T_{i,1} + T_{i,2})$. Continue this iterative process until there are no more packets in the queue leads to the maximum number of back-to-back packets that may be served by the FCFS scheduler for flow i. The maximum burst for flow i can be expressed by following

$$N_i^* = A_i(\sum_{k=1}^{\infty} T_{i,k}). \tag{4.5}$$

where $T_{i,k} = A_i(\sum_{m=1}^{k-1} T_{i,m})/C$ for $k \geq 2$ and $T_{i,1}$ is defined in (4.4). Notice that N_i^* is a function of C, R_j, and N_j for all $j \in J$. By setting $N_i^* = N_i$, the smallest integer $\overline{N_i}$ is what shall be used to regulate flow i before sending to the FCFS scheduler. In order to find N_i^*, the calculation would involve many iteration steps, especially for a large burst size. To simplify the calculation, (4.5) can be approximated using the following equation.

$$N_i^* = \frac{T_{i,1} * R_i * C}{C - R_i}. \tag{4.6}$$

The above equation is derived by approximating $K_j(T_w) = \frac{T_w * R_j}{\overline{N_j}}$, which leads to

$$A_j(T_w) = T_w * R_j. \tag{4.7}$$

Since $T_{i,2} = A_i(T_{i,1})/C$ and based on (4.7), $T_{i,2}$ can be rewritten as:

$$T_{i,2} = \frac{A_i(T_{i,1})}{C} = \frac{T_{i,1} * R_i}{C}. \tag{4.8}$$

By using (4.7) and (4.8), (4.5) can be written as

$$\begin{aligned} N_i^* &= A_i(\sum_{k=1}^{n} T_{i,k}) = A_i(T_{i,1} + T_{i,2} + T_{i,3} + ... + T_{i,n}) \\ &= A_i(T_{i,1} + \frac{T_{i,1} * R_i}{C} + \frac{T_{i,2} * R_i^2}{C^2} + ... + \frac{T_{i,n} * R_i^{n-1}}{C^{n-1}}) \\ &= A_i(T_{i,1} * [1 + \frac{R_i}{C} + \frac{R_i^2}{C^2} + ... + \frac{R_i^{n-1}}{C^{n-1}}]). \end{aligned} \tag{4.9}$$

Since $1 + X^2 + X^3 + ... + X^n = \frac{X(1 - X^n)}{1 - X}$, and letting $X = \frac{R_i}{C}$, one can rewrite (4.9) as

$$\begin{aligned} N_i^* &= A_i(T_{i,1} * [\frac{\frac{R_i}{C}(1 - (\frac{R_i}{C})^{n-1})}{1 - \frac{R_i}{C}}]) \\ &= A_i(T_{i,1} * [\frac{R_i(1 - (\frac{R_i}{C})^{n-1})}{C - R_i}]). \end{aligned} \tag{4.10}$$

Since $C > R_i$, $(\frac{R_i}{C})^{n-1}$ will go to zero as $n \to \infty$. So,

$$
\begin{aligned}
N_i^* &= A_i(T_{i,1} * \frac{C}{C - R_i}) \\
&= (T_{i,1} * \frac{C}{C - R_i}) * R_i \\
&= \frac{T_{i,1} * C * R_i}{C - R_i}
\end{aligned}
\tag{4.11}
$$

The approximation in (4.11) is valid when the total time to create the maximum burst size ($\sum_{k=1}^{\infty} T_{i,k}$) is larger than N/R. The larger the total time, the more number of iterations are needed to calculate the maximum number of back-to-back packets. Since the number of iterations is assumed to be infinity in (4.11), the larger the number of iterations, the more accurate the approximation of equation (4.11) would be. The following example will illustrate this:

> Assume that the total capacity of the scheduler is C = 1, the arrival rate for the first flow is R_1 = 0.7 and R_j = 0.1, for j = 2 to 4. Also assume that N = 2 for all flows. The maximum burst size for flow 1 based on (4.11) and (4.5) in this example are 14 and 14.5 packets, respectively. On the other hand, it would be 0.6 and 2 packets for flow 2 based on (4.11) and (4.5), respectively.

Even though the results calculated by (4.11) and (4.5) have a large difference for flow 2 in the above example, both results are still within the allowed burst size (N=2). When the total time ($\sum_{k=1}^{\infty} T_{i,k}$) is less than N_i/R_i, N_i^* can be approximated to be N_i.

4.1.2 Front-end Regulator for WFQ scheduler

Similar to the case of the FCFS scheduler, the FER scheme for the WFQ scheduler also consists of a dedicated buffer for each incoming traffic flow at the regulator (Figure 4.2). The WFQ scheduler consists of n classes of service. For simplicity, this thesis considers one flow per service class. When a packet is eligible for transmission, it is put into a dedicated buffer for that service class in the WFQ scheduler.

Let w_i denote the weight for service class i. For simplicity, the sum of the weights is assumed to be 1, i.e., $\sum_{i=1}^{n} w_i = 1$.

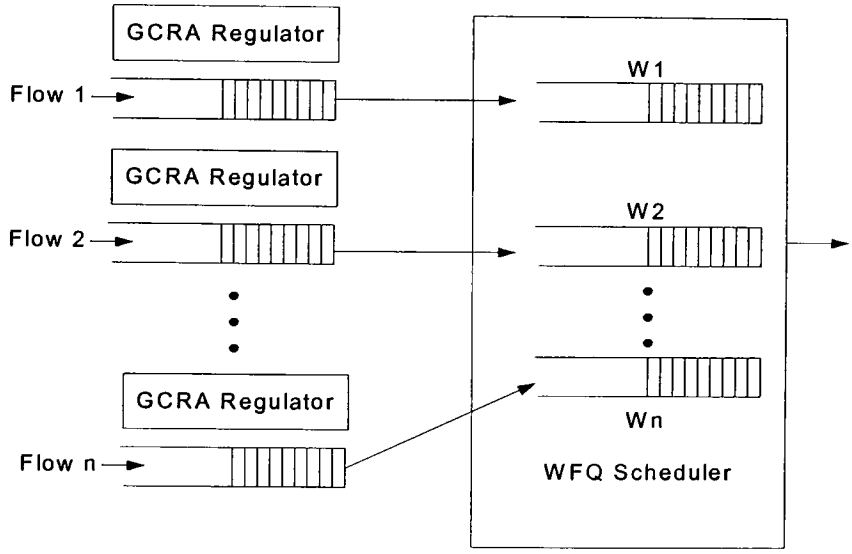

Figure 4.2: Architecture of the FER for WFQ scheduler. Incoming packets are placed in a per connection buffer until the GCRA traffic regulator determines that they are eligible for transmission (WFQ scheduler).

As discussed in the previous chapter, WFQ is a packetised version of GPS, which means that each service class would virtually have dedicated processing capacities. Since each service class only serves one traffic flow, when calculating the number of back-to-back packets for a flow, there is no need to know the information about other flows. Similar to the FCFS scheduler case, the maximum burst size that can be created for a given flow i by WFQ scheduler is the number of packets that can arrive from flow i during the longest continuous time window, within which no packets from other flows would arrive and the queue for that flow in WFQ scheduler never empties. Note that the delay experienced by a packet using the WFQ scheme is within one packet transmission time of the delay encountered if the ideal GPS scheme were used [2]. More specifically, the maximum delay experienced by the first packet from flow i is $D_{i,\max} \le t/w_i + t$, where t is the packet transmission time. It turns out that this delay bound $D_{i,\max}$ leads to the

derivation for the longest time period during which flow i may send in the most back-to-back packets. Similar to the arguments for the FCFS scheduler case, flow i may send in at most $A_i(D_i)$ packets, while the packet experiencing the longest delay is in the scheduler. Assuming no packets from other flows arriving to the scheduler, this iterative process will lead to the maximum number of back-to-back packets for the i^{th} flow. That is,

$$N_i^* = A_i(\sum_{k=1}^{\infty} D_{i,k}) + 1. \tag{4.12}$$

where $D_{i,k} = A_i(\sum_{m=1}^{k-1} D_{i,m})/C$ for $k \geq 2$ and $D_{i,1} = D_{i,\max}$. The extra one packet in (4.12) is the very first packet from i that encounters the maximum delay bound $D_{i,\max}$. By contrast to the FCFS case, the maximum burst size N_i^* for the WFQ case is a function of C, w_i, R_i, and N_i, but independent of the traffic descriptor of other flows. Similar to the derivation for the FCFS scheduler case, (4.12) can be approximated by (4.13) below:

$$N_i^* = \frac{D_{i,1} * C * R_i}{C - R_i} + 1. \tag{4.13}$$

4.2 FCFS and WFQ with Eligible Timer

The complexity of the FER scheme lies in the offline computation of the parameters, which may be executed upon connection setup. This in practice shall incur minimal overhead since the signaling for connection setup usually take much longer time than such computation. The FER scheme, however, may not guarantee 'zero' non-conforming packets for certain scenarios. An obvious example is when the GCRA descriptor of the input traffic flow has $N = 1$, i.e., the traffic flow is peak rate constrained. In this example, although no back-to-back packets are generated from the input flow, the scheduler may still produce bursts of packets. To prevent this from happening, the FER scheme will need to shape the incoming packets streams with $\overline{N} < 1$, which obviously is infeasible. Other examples with $N > 1$ but require $\overline{N} < 1$ also exist.

In order to overcome the shortcoming of the FER scheme, a new scheme is proposed to use an eligible timer for each traffic stream. The eligible timer (ET) updates the next eligible time that a packet can be serviced by the scheduler. The eligible service time is updated upon the packet departure from each flow. If k^{th} packet from flow j departs at time s_j^k, then the eligible time for the next packet of the same flow is

$$e_j^{k+1} = \max(TAT_j^{k+1}, s_j^k + \frac{1}{R_j}) - (N_j - 1) * (\frac{1}{R_j} - t). \quad (4.14)$$

where t is the packet transmission time. TAT_j^{k+1} is the theoretical time that the $(k+1)^{th}$ packet should depart, where $TAT_j^{k+1} = \max(TAT_j^k, s_j^k) + \frac{1}{R_j}$. The term $(N_j - 1) * (\frac{1}{R_j} - t)$ is actually the L value for the $GCRA(I, L)$ model. Notice that e_j^{k+1} is independent of the scheduler used. In fact, the eligible timers of the active flows only decide the packets that are eligible to be serviced. Which of the packets will actually be serviced next (if there were multiple flows have their eligible service time less than or equal to the current time) is still determined based on the scheduler used. The eligible timer approach requires one update for each flow upon every packet departure. However, one should recognize that this requires the same complexity as the WFQ scheduler that maintains a finish tag per flow.

Chapter 5: Simulator Implementation and Simulation Results

In order to find out the performance of the two proposed conformance preserving schemes, a simulator is designed and implemented for both schemes with FCFS and WFQ schedulers. The following section will briefly illustrate the implementation of the simulator followed by a detailed discussion of the simulation results.

5.1 Simulator Implementation

Five key components make up the simulator. The first component is a set of regulators, which are used to ensure that the arrival packet streams are indeed conforming to their GCRA traffic descriptors, as assumed throughout the thesis. Traffic streams passing through this component can be purposely made bursty for simulation purposes. The second component is another set of regulators that are used for the FER scheme. GCRA parameters for this set of the regulators are calculated based on (4.11) and (4.13). The rest of the simulator consists of the scheduler itself, including the eligible timers it is used, the traffic generator, and the statistics collector.

The two sets of regulators are implemented using the Token-leaky Buffers (TBF) proposed in [20]. In a TBF, a buffer is "furnished" with a token bucket that controls the service rate of the buffer as shown in Figure 5.1. The total size of the bucket is N, and is initially filled with tokens. The token is generated at a constant rate of R, and a token is removed from the bucket each time a packet leaves the buffer. When there are tokens in the bucket, a packet leaves at a rate of $v = u$, where u is the packet arrival rate to the TBF. When the bucket is almost empty, i.e., the packets arrive faster than the tokens do, then a packet leaves at a rate of $v = R < u$.

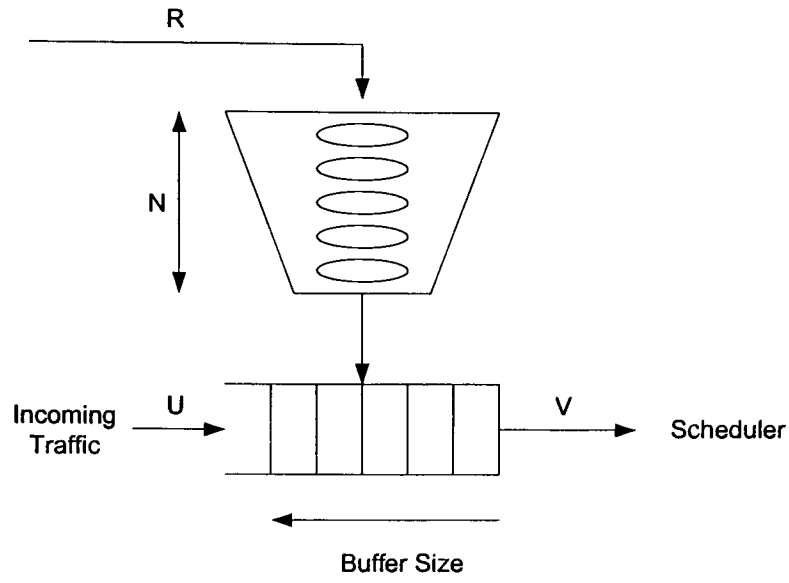

Figure 5.1: Token-Leaky Buffer (TBF)

The simulator can generate two different types of arrival patterns: burst traffic and random traffic. For the burst traffic case, each traffic stream may generate bursts of packets as shown in Figure 5.2. For example, the simulator can be configured to generate N_i number of packets at peak rate of C for flow i during the "On" state; then, wait until the token bucket is full (i.e. the "Off" state) before sending out another burst for flow i. For the random traffic case, the packets of each flow i follow an independent Poisson process with an average rate of R_i.

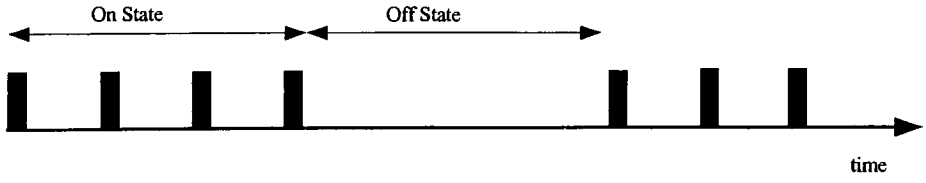

Figure 5.2: Bursty Traffic Arrivals

The simulator is an event driven system, where processes are triggered upon one of the following three different events:

1. **Arrival**: a packet arriving to the system.
2. **Token**: a new token is ready for the token bucket
3. **Departure**: a packet is ready to depart from the system.

The simulator generates events depending on which event occurs first. When the event is an arrival, a packet would be formed and sent to the scheduler if there is a token available; otherwise, it would be stored in the regulator buffer. The arrival time of the next packet is calculated depends on the type of arrivals. For bursty traffics, there can be more than one packet arrive from different flows, and all of them can be sent to the scheduler if the token buckets that are associated to the packets are not empty.

A departure event occurs when a packet can be departed before the arrival of next packet and the next token generation. When a packet is departed, a set of performance metrics such as packet delay, and number of non-conforming packets are collected for post-simulation analysis. A token event occurs when it is the time to generate a new token. Since there is a token bucket for each flow, there can be multiple token events occur at the same time. After a token is created, if there is a packet waiting in the buffer, the packet is sent to the scheduler and the token is removed from the token bucket. When the bucket is full, the newly generated token is discarded.

A general flow chart of the simulator is given in Figure 5.3. The simulator is initiated using command lines, where a set of the arguments will be given. The usage of the command line is as following:

project [#_of_runs] [#_of_packets] [link_capacity] [packet_size] [scheduling_algorithm]

project:	name of the executable.
#_of_runs:	the number of runs to simulate for the same setup, where the average results will be obtained at the end of the simulation.
#_of_packets:	the number of packets to generate for simulation for each run.
link_capacity:	processing capacity (C) in bits per second.
packet_size:	size of the packet in bits. (The simulator assumes all packets have the same size)
scheduling_algorithm:	the numerical index to indicate the scheduling algorithm to be used.

A numerical example would be: **project 10 15000 10000000 2500 1**

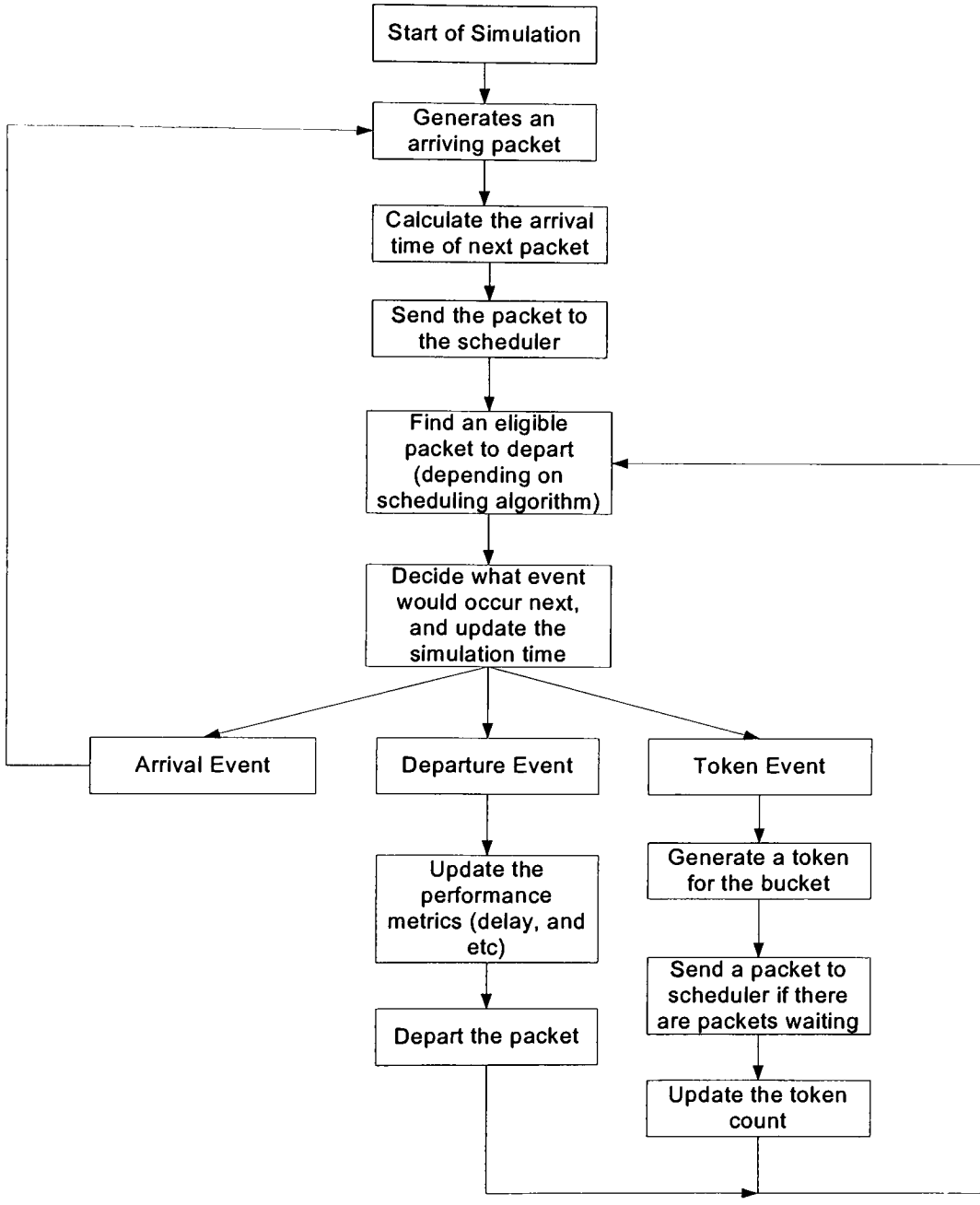

Figure 5.3: The flow-chart of the simulator

5.2 Simulation Results

This section exhibits the performance improvements achieved by using the proposed schemes for the FCFS and WFQ schedulers. The % non-conforming packets

will serve as the key performance measure, while the average packet delay experienced by each packet will also be examined. Recall that the % non-conforming packets is defined as the ratio of the number of non-conforming packets with respect to the total number of packets that enter the scheduler. Two types of traffic mixtures of four flows are considered: (1) a dominant flow contributes to the majority of traffic load, and (2) all arriving flows have different but relatively comparable average arrival rates. More specifically, the simulations investigate the cases where the ratio of average arrival rates of flow one to four is 7:1:1:1, and 4:3:2:1, respectively. The purpose of examining these two scenarios is to investigate the impact of having a dominant flow in the traffic mix. In addition to the traffic mix scenarios, the simulations shown in this section also assume a 95% traffic load. Recall that in Figure 3.2, the % non-conforming packets incurred by various schedulers is high when the traffic load is 80% or more. To exhibit that the proposed schemes work well even when the traffic load is extremely high, the experiments will be placed at the 95% traffic load region, where both FCFS and WFQ incur high percentage of non-conforming packets. All buffers that are used for storing packets in the simulation are assumed to be infinite, so there won't be any buffer overflow instants occur.

5.2.1 Performance without the proposed schemes

Before proceeding to exhibit the performance achieved by the proposed schemes, we first examine how FCFS and WFQ schedulers perform if they were serving different traffic mixture scenarios, and if the burst sizes of the flows were different. For simplicity, the burst sizes of all flows are assumed to be the same in this set of experiments. All flows start sending their bursts at the beginning of the simulation (simulation time equals to zero).

Figure 5.4 plots the % non-conforming packets incurred by the FCFS and WFQ schedulers when the arriving burst sizes of all flows increase. As shown, the % non-conforming packets experienced by all flows decrease as the burst size increases. Recall that, from Section 4.1.1, the maximum number of packets in the FCFS queue from flows other than i is Q_i. For example, assume that each flow can send a burst of 2 packets, then

Q_i would be 5 packets, which is 83.3% (5 out 6) of all packets from flows other than flow *i*. When the burst size is increased to 8 packets for each flow, the value for Q_i is 17 packets, or 71% of all packets from the other three flows. This example illustrates when the burst size increases, the maximum number of packets (in terms of percentage) in the FCFS queue from flows other than *i* actually decreases. Since Q_i has direct relationship with the maximum burst that can be created by flow *i*, the % non-conforming packets is expected to decrease as the burst size increases. When burst size gets larger, the percentage is going to converge (i.e.,70% when the burst size is 10). This explains why the % non-conforming packets settles down when the burst size gets very large for FCFS scheduler. As for the WFQ scheduler, since WFQ provides some fairness, ideally, flow 1 would process 7 packets before 1 packet can be processed from each of other three flows for the dominant flow case regardless of the burst size. When the burst size increases to 7 packets or beyond, the % non-conforming packets is expected to be small for the dominant flow case. Ideally, none of the flows can have 7 or more packets getting processed in a row at a given time, which means there won't be any bursts for a flow that come out of the scheduler is larger than 7 packets. As can be seen in Figure 5.4, the % non-conforming packets drops to about 2 or 3% when the burst size is 8 packets or larger. For the non-dominant flow case, the % non-conforming packets actually goes down very close to zero when the burst size is increased to 4 packets since none of the four flows is expected to have more than 4 packets getting processed in a row at any given times.

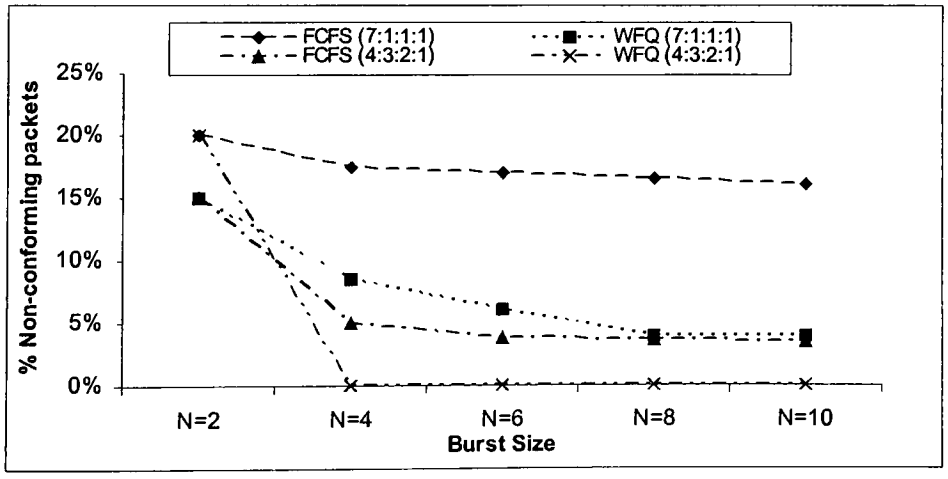

Figure 5.4: % non-conforming packets incurred by the FCFS and WFQ schedulers as the burst size increases at 95% traffic load.

Figure 5.4 also exhibits the impact of traffic mixture on the % non-conforming packets. As shown, the dominant flow case results in more non-conforming packets for FCFS, but this is not true for WFQ when the burst size is 2. For FCFS, it is harder to create large bursts for a flow in non-dominant flow case as compared to the dominant flow case, since all flows have relatively comparable average arrival rates and FCFS scheduler services packets in the order of the arrival times. For WFQ, since there is no more than one packets can be processed from flows 2 to 4 at any given times, all the non-conforming packets that are created by flow 1 contribute for 70% of the traffic load in the dominant flow case. In the non-dominant flow case, flows 1 to 3 can all have 2 or more packets getting processed in a row. This means that 90% of the traffic load can create non-conforming packets when the burst size is 2 packets. Therefore, the non-dominant flow case has higher the % non-conforming packets than the dominant flow case when the burst size is 2 packets.

For the peak-rate-constrained flows (N =1), the simulation considers random traffic. In average, random traffic generates a packet every $1/R_i$ unit of time for flow i. As shown in Figure 5.5, the non-dominant flow case actually has a higher % non-conforming packets as compared to the dominant flow case. In non-dominant flow case, 90% of the traffic load is responsible for creating the % non-conforming packets compare to 70% for the dominant flow case with the WFQ scheduler. For FCFS, more packets are likely being out of the sequence with their supposed service time for non-dominant flow case as compare to the dominant flow case. Since those are peak-rate-constrained flows, any packets that are not served with their supposed service time could cause packets to be non-conforming. By comparing Figures 3.2 and 5.5, one can see similar performance trends exhibited as the traffic load increases no matter it is bursty or peak-rate-constrained traffic. In the sequel, the focus would be on examining performance achieved by the proposed schemes assuming the dominant flow case with bursty traffic flows.

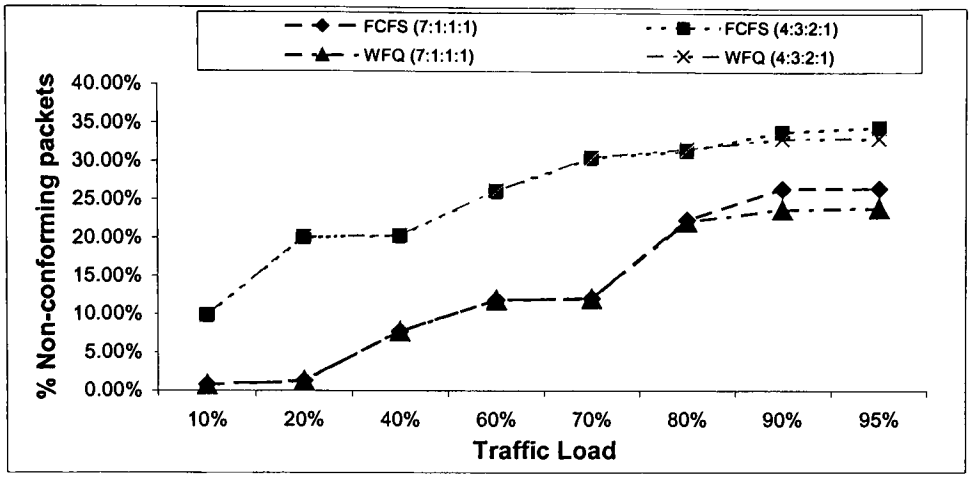

Figure 5.5: % non-conforming packets incurred by the FCFS and WFQ schedulers for peak-rate-constrained flows (N = 1)

5.2.2 Performance of the proposed scheme for the dominant flow case

This section investigates whether the proposed schemes actually achieve 'zero' non-conforming packets. Figure 5.6 shows the % non-conforming packets achieved by the proposed schemes when they are used for the FCFS scheduler, as the arriving burst size increases. Notice that while the ET scheme guarantees 'zero' non-conforming packets, the FER still incur about 20% overall non-conforming packets as if it has no effect at all. As discussed earlier, this is due to the fact that there is no valid $\overline{N_i}$ (which needs to be integer and larger or equal to 1) can be found when the arriving burst size N_i is small (N_i=2 in this example). In fact, for this experiment, non-conforming packets exist for the FER scheme even when $N_i = 6$. Due to such limitation of FER and the significant delay incurred by the ET scheme (which will be discussed next), another scheme that combines the use of FER and ET is proposed. This hybrid scheme is referred as 'ET+FER', and it simply use both the Front-end Regulator to further regulate the incoming traffic and compute the eligible timer to determine whether a packet should be served. As can be easily seen in Figure 5.6, 'ET+FER' also guarantees 'zero' non-conforming packets, as the ET scheme does.

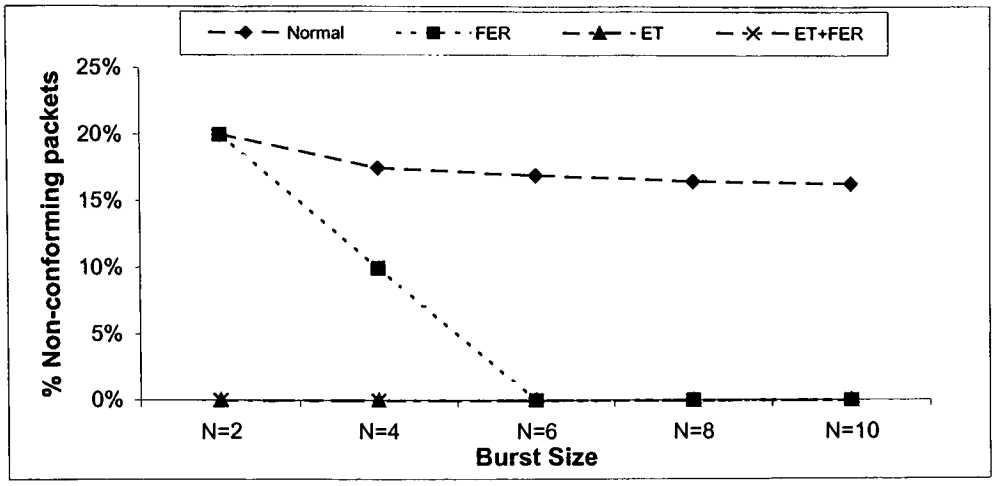

Figure 5.6: % non-conforming packets achieved by regular FCFS and FCFS with various supplemental schemes as the arriving burst size increases (dominant flow case).

Although the ET scheme achieves 'zero' non-conforming packets, the delay encountered is much greater than the FER scheme. Figure 5.7 shows the average packet delay incurred by the proposed schemes when they were used for the FCFS scheduler. Notice the significant average delay experienced by the packets when the ET scheme is applied for FCFS. This large delay even increases faster when the arriving burst size increases, as compared to the regular FCFS and other proposed schemes. This is due to the fact that any schedulers with ET are non-work conserving. By contrast, the maximum delay for the FER scheme is $(N_i - \overline{N_i})/R_i$ for flow i. The hybrid 'ET+FER' scheme, meanwhile, also incurs comparable delay to the FER scheme. Figures 5.6 and 5.7 suggest that the hybrid scheme performs well when applied to the FCFS scheduler.

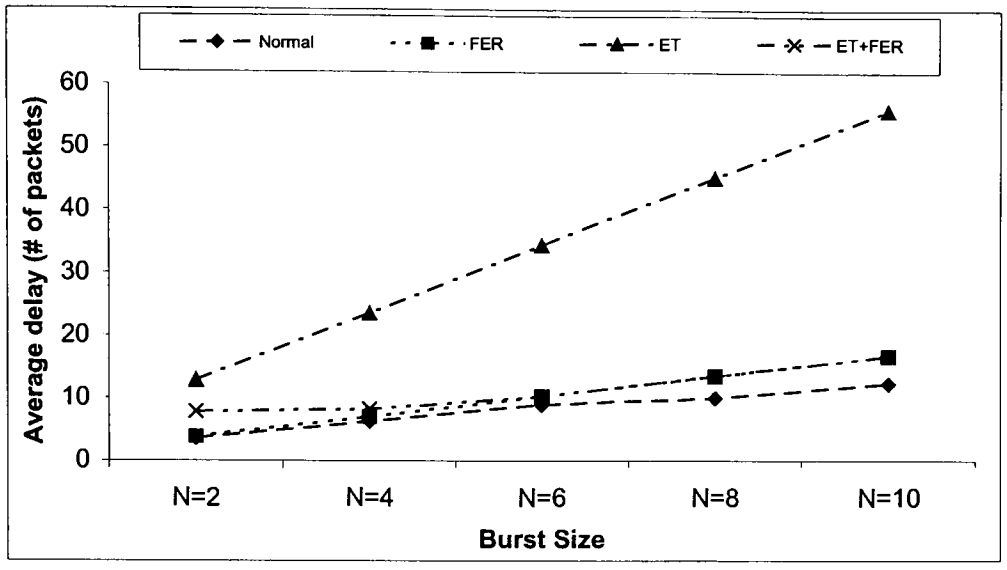

Figure 5.7: Average packet delay incurred by regular FCFS and FCFS with various supplemental schemes as the arriving burst size increases (dominant flow case).

The following examines the performance improvements of the proposed scheme when they were applied to the WFQ scheduler. Figures 5.8 and 5.9 show the % non-conforming packets and the average packet delay, respectively, incurred by the regular WFQ scheduler and those with the proposed supplement schemes. As can be seen, Figure 5.8 exhibits similar performance trends as Figure 5.6 does for the FCFS case. In Figure 5.9, however, one can observe that the all schemes achieve similar average delays. In fact, the ET scheme actually achieves better average delay than the FER scheme. This is due to that the FER is designed assuming the worst-case scenario. Therefore, the arriving traffic streams may be 'over-regulated' even though they do not produce the worst-case scenario. When the streams of packets are over-regulated, unnecessary delays will be introduced. By contrast, the ET scheme adds additional delay only when the packets are forced to wait even the scheduler is available, which occurs rarely.

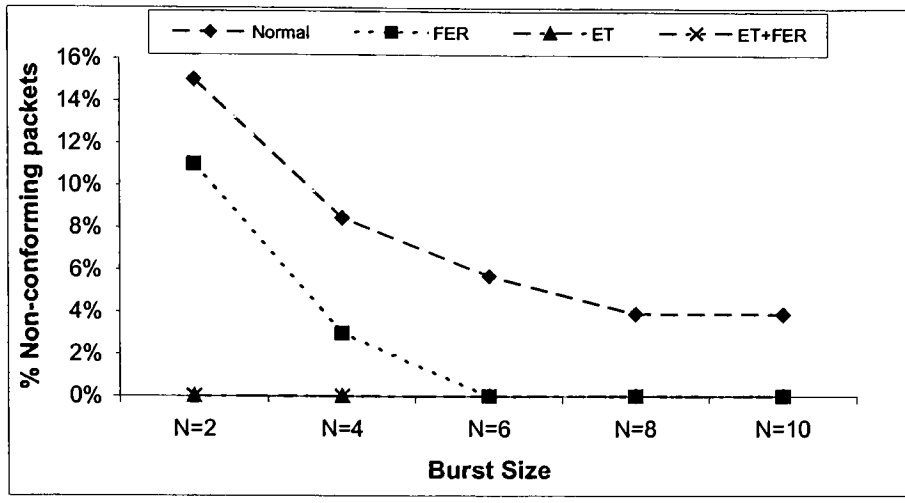

Figure 5.8: % non-conforming packets achieved by regular WFQ and WFQ with various supplemental schemes as the arriving burst size increases (dominant flow case).

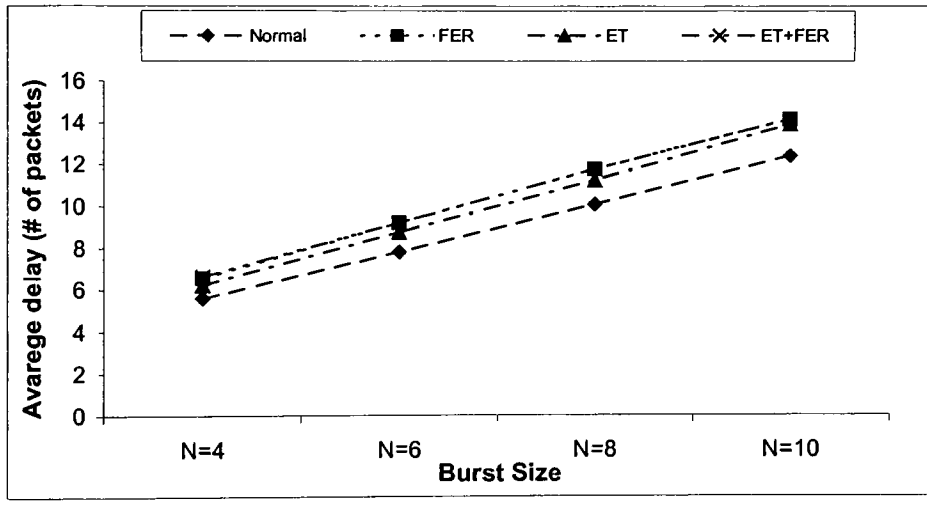

Figure 5.9: Average packet delay incurred by regular WFQ and WFQ with various supplemental schemes as the arriving burst size increases (dominant flow case).

5.2.3 Performance of the proposed scheme for the non-dominant flow case

This section examines the case where no flow significantly dominates the traffic load. Similar performance trends should be expected here as compared to the dominant flow case regardless whether it is the FCFS or WFQ scheduler being used. The

discussion on the non-dominant flow case therefore will be focused on the scenarios with the FCFS scheduler.

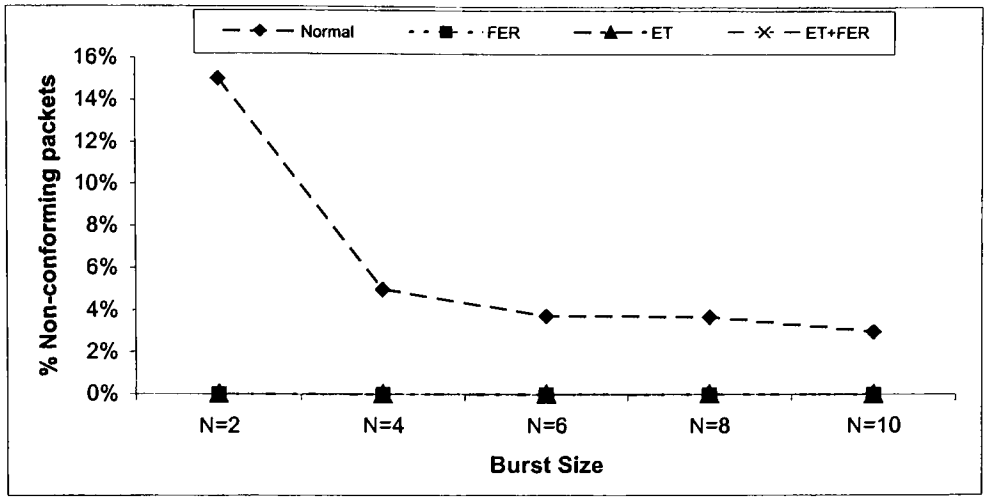

Figure 5.10: % non-conforming packets achieved by regular FCFS and FCFS with various supplemental schemes as the arriving burst size increases (non-dominant flow case).

The non-dominant flow scenario also achieves less number of non-conforming packets when N gets larger. As one can see, the proposed schemes can all achieve zero non-conforming packets. As explained above, when there is less number of non-conforming packets, the Eligible Timer scheme does provide little better delay than the FER scheme. This is still true for this scenario as shown in Figure 5.11.

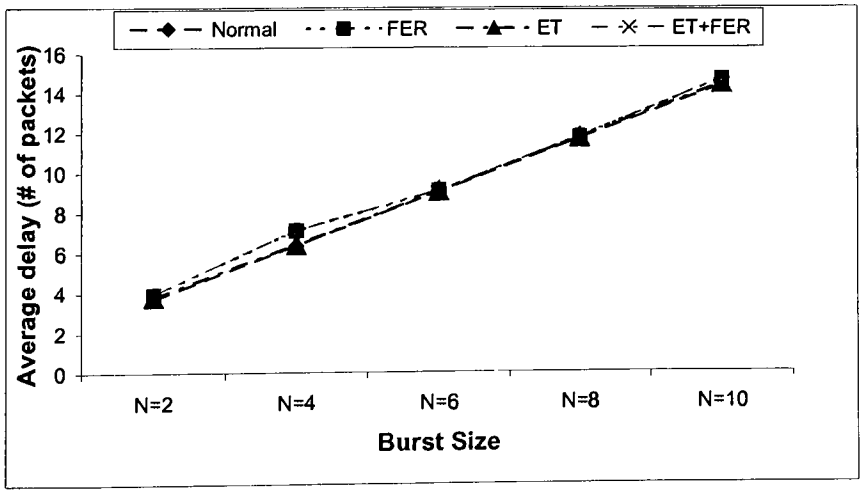

Figure 5.11: Average packet delay incurred by regular FCFS and FCFS with various supplemental schemes as the arriving burst size increases (non-dominant flow case).

As for the peak-rate-constrained flows, the FER scheme would not be able to provide better performance since the flows cannot be further regulated. On the other hand, the ET scheme can achieve zero non-conforming packets. Since there are many non-conforming packets, the delay for ET scheme is also quite significant. The performance achieved by the hybrid scheme is just the same as the ET scheme since the FER scheme has no effect on this type of traffics.

5.2.4 Summary

The discussion in the previous sections suggests that, in general, similar performance trends are exhibited for the proposed schemes no matter which schedulers (FCFS or WFQ) these supplement schemes are used for. Therefore, the decision on which supplement scheme to use depends on traffic mixtures (dominant flow case or non-dominant flow case), traffic types (bursty traffic or peak-rate-constrained traffic), and the size of the burst. Table 5.1 summarizes the best scheme to use under various scenarios.

Factors			Scheme to use
Dominant flow case	Bursty traffic	Small burst size	The 'ET+FER' scheme works best for this type of traffics. It achieves 'zero' non-conforming packets, and less queuing delay than the ET scheme.
		Large burst size	The FER scheme can achieve 'zero' non-conforming packets with little additional delay. Queuing delay incurred by a packet will be large when the ET scheme is used.
	Peak-rate-constrained traffic		The ET scheme should be used for this type of traffic. The FER scheme cannot further regulate this type of traffic. The 'ET+FER' scheme does not provide any better performance than the ET scheme.
Non-Dominant flow case	Bursty traffic	Small burst size	The ET scheme should be used. Even though the 'ET+FER' scheme can also achieve 'zero' non-conforming packets, the delay experienced by a packet can be large since the FER scheme might 'over regulate' the flows.
		Large burst size	The FER scheme can achieve 'zero' non-conforming packets with little additional delay.
	Peak-rate-constrained traffic		The ET scheme should be used for this type of traffic. Queuing delay incurred by a packet is expected to be large.

Table 5.1: Scheme to use for different traffic factors

Chapter 6: Conclusion and Future Work

Two schemes, FER and ET, and a hybrid of the two schemes are proposed in this thesis to complement packet schedulers, so as to prevent the schedulers from penalizing flows that conform to their GCRA descriptors when they enter the scheduler. The FER scheme shapes the incoming traffic aggressively to avoid building up excessive backlog in the queue. It requires only offline computational effort and incurs only slight increase in the average delay. The drawback, however, is that FER cannot guarantee zero non-conforming packets for all types of traffic flows. The ET scheme, by contrast, requires online updates as the WFQ scheduler does and may incur large additional delays simply because it is a non-work conserving policy. The advantage of the ET scheme is that it always guarantees zero non-conforming packets regardless what the traffic descriptors are. A hybrid scheme, as exhibited by simulation, possesses the advantages of both the FER and ET schemes. The choice of using which conformance preservation scheme depends on the traffic mixtures and the burstiness of the packet streams. In general, if there is only small number of non-conforming packets, the ET scheme suffices and does not cause significant additional delay. Otherwise, the FER or the hybrid scheme shall be used. It is suggested that employing the hybrid schemes will help to reduce unfair packet drops or over-conservative resource provisioning in the network core under most scenarios.

Further experiments have also shown that the proposed schemes do not provide fairness in terms of the average packet delay among flows. It will be interesting to investigate schemes that not only achieve zero non-conforming packets but also provide fairness on average packet delay among flows. One potential approach is to incorporate feedback control into the real-time scheduling algorithms to achieve predictable system performance under unpredictable network loads. One may borrow the idea from Random Early Detection (RED) queue management [21] for TCP traffic to provide the feedback control. As an overall improvement to the quality of the thesis and potential future work, an established network simulation tools, such as Network Simulator 2 (ns-2) [23] or OPNET [22] shall be used to evaluate system performance.

References:

[1] A. Hagai, "*Multiple Priority, Per Flow, Dual GCRA Rate Controller for ATM Switches*", Master Thesis, Tel Aviv University, June 5, 2000.

[2] A. Parekh, "*A Generalized Processor Sharing Approach to Flow Control in Integrated Services Network*", PhD dissertation, MIT, December 1992.

[3] "*Comparing Traffic Policing and Traffic Shaping for Bandwidth Limiting*", www.cisco.com/warp/public/105/policevsshape.html

[4] D. I. Norris, "*Introduction to the IBM Introduction to the IBM Network Processor Network Processor NP4GS3 Rev. 2.0 NP4GS3 Rev. 2.0*", IBM Microelectronics Division Education and Support.

[5] Hui Zhang, Domenico Ferrari, "*Improving Utilization for Deterministic Service In Multimedia Communication*", Proceedings of IEEE International Conference on Multimedia Computing and Systems, 1994.

[6] H. Perros and K. Elsayed, "*Call Admission Control Schemes: A Review*", IEEE Magazine on Communications, special issue on Congestion Control, Pages 82-91, November 1996.

[7] J. Qiu, and E. W. Knightly, "*Measurement-Based Admission Control with Aggregate Traffic Envelopes*", IEEE/ACM Transactions on Networking, Vol. 9, NO. 2, April 2001.

[8] J. C.R. Bennett, H. Zhang, "*WF^2Q: Worst-case Fair Weighted Fair Queuing*", Proceedings of IEEE INFOCOM, 1996.

[9] M. Bjoerkman, P. Gunningberg, "*Locking Effects in Multiprocessor Implementations of Protocols*", ACM Sigcomm, 1993

[10] M. Ritter, "*Analysis of the Generic Packet Rate Algorithm Monitoring ON/OFF-Traffic* ", University of Wurzburg, Institute of Computer Science Research Report Series, Report No. 77, Jan 1994.

[11] R. L. Cruz, "*A calculus for network delay part I: Network elements in isolation*", IEEE Transactions on Information Theory, January 1991.

[12] R. Jain, "*Congestion Control and Traffic Management in ATM Networks: Recent Advances and A Survey*", Computer Networks and ISDN Systems, Vol. 28, No. 13, Pages 1723-1738, Oct. 1996.

[13] R. G. Gallaer, A. K. Parekh, "*A generalized processor sharing approach to flow control in integrated services network-the single node case.*", IEEE/ACM Transaction on Network, June 1993

[14] R. Haas, "*Creating Advanced Functions on Network Processors: Experience and Perspectives*", IBM Microelectronics, Research Triangle Park, NC 27709, USA

[15] S.-C. Yang, "*Traffic Dispersion for Bursty Traffic on Heterogeneous Networks*", Master Thesis, The University of Texas at Austin, 1998.

[16] S. Lakshmanamurthy, K. Liu, Y.Pun, L. Huston, U. Naik, "*Network Processor Performance Analysis Methodology*", Intel Technical Journal, Vol. 6 Aug, 2002

[17] S. Sidiropoulos, M. Katevenis, C. Couroubetis, "*Efficient Fair Queuing using Deficit Round Robin*", in Proc. SIGCOMM'95, pp.231-242, September 1995

[18] The ATM Forum Technical Committee, "Traffic management specification version 4.0", April 1996, available via http://www.atmforum.com.

[19] V. Firoiu, J. L. Boudec, D. Towsley, and Z. Zhang, "*Theories and Models for Internet Quality of Service*", Proceedings of the IEEE, special issue on Internet Technology, May 2002.

[20] V. Guffens, G. Bastin, H. Mounier, "*Using token leaky bucket with feedback control for guaranteed boundedness of buffer queue*", IEEE transactions on automatic control, 2002.

[21] V. Jacobson, S. Floyd, "*Random Early Detection Gateways for Congestion Avoidance*", IEEE/ACM Transactions on Networking, 1993.

[22] OPNET Network Simulator, http://mil3.com.

[23] The Network Simulator 2, http://www.isi.edu/nsnam/ns/index.html.