Abstract

Service providers in both wireline and wireless areas now often look to consolidate multimedia traffic and data traffic onto a common platform. However, current commercial Internet services offer only best-effort services. Due to the lack of distinguishing flows or classes of packets, quantitative commitments regarding the QoS provision are not accommodated by current Internet switching nodes. But the business model of ISP now allows the change of the premise of resource allocation from the traditional best-effort to one based on user profiles or service level agreements. Many users also want and are willing to pay for preferential treatments for their traffic, especially for multimedia applications. Thus, network providers are demanded to satisfy this requirement.

To cope with the issues of accommodating QoS for multimedia traffic and data traffic simultaneously, several novel approaches for different network scenarios are proposed and studied in this dissertation. The topics studies in this dissertation include three parts. The first one is how to provide QoS for backbone networks, where the multimedia traffic and the data traffic are divided into classes. The second part is how to accommodate different QoS requirements for multimedia and general data traffic streams via a common scheduling platform over wireline access networks, where these two types of traffic streams are aggregated. Finally, we concern the issue how to provide fair channel usage and maintain high channel utilization simultaneously for wireless access networks, where the bursty channel errors and the location-dependent properties in wireless environments may lead to failure of conventional wireline scheduling algorithms.

For the first topic, a novel packet scheduling algorithm, called *Gated-Frame-Queueing* (GFQ), with low implementation complexity is presented. The GFQ algorithm not only eliminates the output sorters in its scheduler design such that the

scheduling mechanism can accommodate large number of flows but also preserve the advantages in low delay bound and fairness. Then a framework for real-time multimedia transmission in ATM networks is proposed. This framework includes an efficient traffic scheduling scheme, called *Multi-layer Gated Frame Queueing* (MGFQ), and two special cell formats for audio and video traffic streams. A hybrid design combining MGFQ and APPD (Age Priority Packet Discarding) is described to improve the packet level QoS.

In the second topic, a new traffic scheduling scheme, called *Differentiated Multi-layer Gated Frame Queueing* (DMGFQ), is proposed to provide users with differentiated QoS, jitter control and fair bandwidth sharing for real-time and non-real-time traffic simultaneously. A protection factor in DMGFQ is also included to provide the flexibility for network operators to achieve appropriate compromise between the network utilization and the service quality. As a result, DMGFQ not only supports a wide range of QoS for real-time multimedia but also achieves the goal of protecting the users conforming to their service level agreements.

In the wireless access network, the last topic of this dissertation, we propose a novel scheduling algorithm, called the *Wireless Differentiated Fair Queueing* (WDFQ) algorithm, to accommodate QoS for real-time and non-real-time traffic streams, including timely delivery of real-time traffic, virtually error-free transmission of non-real-time traffic, and fair usage of channel bandwidth among remote stations. In addition, the location-dependent channel error property, as appeared in most wireless networks, are considered in the model and the temporary short error burst are compensated by the concept of credits.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the increasing popularity of multimedia applications, the traditional role of a computer network as a means of transmitting data is changing. Currently, service providers in both wireline and wireless areas are looking to consolidate multimedia traffic and data traffic onto a common platform. Ideally, in the so-called next generation network (NGN), high-speed transport and switching of multimedia and regular Internet data traffic in an aggregated fashion can be realized using only packet-based networking technology. The advent of such high speed LAN and WAN in recent years has introduced great opportunities for many multimedia applications such as teleconferencing, distance learning, real-time video and voice, etc. These real-time or near real-time applications often have stringent performance requirements in terms of throughput, delay, delay jitter, and loss rate. However, currently, Internet offers only a best-effort service, where the performance of each flow can degrade significantly when the network is congested. In addition, most networks used First-Come-First-Served (FCFS) queues to do packet scheduling at the switching nodes. Due to the lack of distinguishing flows or classes of packets, quantitative commitments regarding the Quality-of-Service (QoS) provision are not accommo-

dated by FCFS. Finally, link speeds are experiencing dramatic increases and the number of users is growing exponentially. All these factors make the control of such an integrated-services communication network a challenging task.

But the business model of ISP now allow the change of the premise of resource allocation from the traditional best-effort to one based on user profiles or service level agreements. Specifically, the emergence of applications with various throughput, delay or jitter requirements is calling for a new network infrastructure which is capable of supporting different levels of services, as opposed to a single, best-effort type of service. Many customers want and are willing to pay for preferential treatments of their traffic, especially for multimedia applications. Thus, network providers are demanded to satisfy this requirement. And it may generate additional revenue from the same investment in network resources if appropriate mechanisms and policies for different QoS levels were in place.

The ability of satisfying QoS requirements in current wireless access network is as worried as wireline networks. The demands from customers in wireless access will lead to high volume of real-time (RT) and non-real-time (NRT) connectivity, which could often be beyond the available bandwidth. As a result, the issues in Quality of Service, fairness and pricing strategies should have expedited the emergence of service differentiation in such wireless access networks. However, to design a traffic management mechanism to support differentiated services in a wireless multimedia environment is a nontrivial task. Due to the fact that characteristics of wireless channels can be very different from the wireline links, the approaches that satisfies the needs of wireless multimedia demands forms a complete new design direction.

In most cases, the network may provide satisfactory performance for most applications. However, *congestion* may degrade its performance in either short or long intervals of time. Thus, in order for a packet-switched network to avoid degrada-

tion in performance and waste of resources, it must support some mechanisms to prevent or recover from congestion. The mechanism that determines which packet will be transmitted next on the output link is one of the most important mechanisms. This mechanism is referred as the **traffic scheduling algorithm**. Thus, this dissertation addresses several issues involved in the design and implementation of traffic scheduling algorithms, such as delays, delay jitters and residual bandwidth shares. We consider traffic scheduling algorithms for several different environments, including wireline and wireless access networks and backbone networks. The QoS requirements of these networks are different and we design a common scheduling platform. Based on this platform, we develop several scheduling algorithms for these network environments. We believe our scheduling platform can still be extended to satisfy QoS requirements of variant networks.

1.1 Service Architecture

Currently, there are two well-known service reference architectures in the communication networks: one is the *Integrated Services* (IntServ) [1]-[10] and the other is the *Differentiated Services* (diffServ) [12]-[16]. IntServ attempts to provide end-to-end QoS for each flow via dynamic resource reservation. The service models proposed in IntServ consist of the guaranteed services (GS) and controlled load services (CLS). The former provides a strict bound on the delay and loss probability for packets of a given flow, under the condition that the flow complies with a traffic contract. The latter does not define the provided service in terms of exact values for delay or loss probability. Under the CLS model the packets of a given flow will experience delays and loss comparable to a network with no load. The IntServ architecture assumes that some explicit setup mechanism is used to convey information to routers so that

they can provide requested services to flows. While RSVP [11] is the most widely known example of such a setup mechanism, the Intserv architecture is designed to accommodate other mechanisms.

DiffServ, on the other hand, is aimed at traffic aggregates that may not correspond to fine-grained flows. In a DiffServ environment, network devices take on responsibilities for classifying packets into aggregates of desired granularity, policing traffic, and allocating static or dynamic resources to satisfy QoS requirements. Within the core network, packets are forwarded according to the per-hop behavior (PHB) associated with their DiffServ code point. For *uncontrolled traffic classes*, qualitative, but no quantitative, service guarantees (priorities) are provided by PHB. Traffic is controlled at the ingress to the network; end-to-end QoS is provisioned in order to control the congestion phenomena. For *controlled traffic classes*, end-to-end service guarantees can be provided to the DiffServ core network based on per-flow admission control.

Since IntServ and DiffServ focus on reservations and scalable service differentiation, respectively, it is advantageous to combine them into an end-to-end QoS solution [10, 17, 18]. As a result, in many experimental or trial network, the concept of IntServ and DiffServ have been implemented (in parts) along with other QoS control mechanism. Because the QoS guarantees mechanism and service scalability are all required by the IntServ/DiffServ like architecture, the studies of scheduling algorithms in this dissertation can be used along with these service architecture or their various implementations.

1.2 Quality-of-Service Parameters

The most important clauses in the service level agreements, for any service architectures supporting QoS, are the specifications of performance requirements and traffic profiles. The traffic profiles, which are defined via the traffic description models, are elaborated in the next section. For the performance parameters, we follow the QoS parameters defined in [19], including the end-to-end delay bound, the jitter bound, the packet loss probability. In addition to these parameters, guaranteed minimum bandwidth and residual bandwidth shares [20] are also adopted as QoS parameters in this dissertation.

The most important QoS parameters is the end-to-end delay bound, which is the essential requirement for RT applications. However, the end-to-end delay bound is provided automatically if the granted service rate is guaranteed and the input traffic pattern satisfies the traffic profiles. Another important QoS parameter is the end-to-end delay jitter bound. The delay jitter for a packet stream is defined to be the maximum difference between delays experienced by any two packets [21]. For RT multimedia application, especially for live multimedia, the requirement of end-to-end jitter guarantee is more stringent than the end-to-end delay guarantees. The reason is that if the variant of the packet arrival times is too large, it is necessary to reserve large buffer to compensate the variant and this would result in low buffer utilization. Therefore, if a bounded delay-jitter service is accommodated by the network, the destination can reserve appropriate buffer space to eliminate the jitter before the transmission starts. This can improve the buffer usage utilization.

A third important parameter is the loss probability. Packet loss would occur due to buffer overflow or delay/jitter bound violation. In general, RT multimedia applications are able to tolerate packet loss occurrence while NRT data traffic stream expect no loss occurrence. Because of the loss tolerant ability, network operators

can overbook network resources beyond the worst case requirements to increase the overall network utilization. On the other hand, the cost of packet loss in NRT traffic streams is the retransmission, which results in the waste of bandwidth and buffer space. Hence, worst case guarantees are required for NRT traffic streams.

The last QoS parameter adopted in this dissertation is the guaranteed minimum bandwidth and the residual bandwidth share [20]. A RT multimedia traffic stream may require the minimum bandwidth to achieve the basic service quality, while the NRT applications such as TCP sessions or WWW flows also require a minimum bandwidth to maintain their basic operations. If the granted bandwidth cannot meet this requirement, the multimedia service may not be accepted and TCP sessions or WWW flows may be disconnected or may not provide services. In addition, if there is any residual bandwidth available, the residual bandwidth should be distributed fairly to required flows to provide better QoS.

1.3 Traffic Description Models

In conventional queueing networks, one of the most popular mathematic models for traffic source description is the Poisson Process, which is the arrival process of phone calls in a large public telephone networks. However, in the data networks that support data, video or other applications, the Poisson arrival process may not be appropriate for characterizing the packet level traffic patterns. Hence, more generic traffic models such as IPP (Interrupted Poisson Process), MMPP (Markov Modulated Poisson Process) [22] and other Markovian models are proposed. The MMPP model can characterize most of the traffic patterns by using limited number of states or phase. However, under the MMPP model the mathematical analysis may be intractable for some cases due to the computer hardware or software limita-

tions. In summary, these models are either too simple to characterize the important properties of the source or too complex for tractable analysis.

Recently, several new traffic models are proposed to bound the traffic rather than characterize the process exactly and they form a new class of traffic descriptor models. These bounding characterizations can either be *deterministic* or *stochastic*. A deterministic bounding characterization defines a deterministic traffic constrain function. A monotonic increasing function $b_j(\cdot)$ is called a deterministic traffic constrain function of connection j if the number of bits arrived on connection j during any interval $(t, t+u]$, $A_j(t, t+u)$, is no greater than $b_j(u)$. On the other hand, a stochastic bounding characterization defines a stochastic traffic constrain function, where the monotonic increasing function within any interval u, $b_j(u)$, is *stochastically larger* than the arrival function $A_j(t, t+u)$.

For deterministic bounding characterization, several most popular models are: $(X_{\min}, X_{\text{ave}}, I, S_{\max})$ defined by Tenet Group [23], (σ, ρ) proposed by Cruz [24], and the D-Bind model defined by Knightly and Zhang [25]. In each of the above models, the exact traffic pattern for a flow is unknown, the only requirement is that the volume of the traffic be bounded in certain ways. Such bounding models can characterize a wide variety of bursty sources. In addition, it is sufficient for resource management algorithms to allocate resources by knowing just the bounds on the traffic volume. However, the cost of the above models is the low utilization of network resource.

On the other hand, the representatives of the stochastic bounding characterization are Stochastic Bounding Interval Dependent (S-BIND) model proposed by Zhang and Knightly [26] and exponentially bounded burstiness (EBB) model proposed by Yaron and Sidi [27, 28]. Take the EBB for instance. A source is said to be EBB with parameters (ρ, A, α) if $Pr\{A(s, s+t) \geq \rho t + \sigma\} \leq Ae^{-\alpha\sigma}$ for all $\sigma \geq 0$ and

$s, t > 0$ where random variable $A(t_1, t_2)$ denotes the total number of bits generated by a source in the interval $(t_1, t_2]$. Although the stochastic traffic models can characterize practical traffic streams more exactly, however, it is difficult to estimate their parameters and these traffic models may not be implemented in practical network efficiently.

1.4 Motivations of This Dissertation

Classical queueing analyses usually study average performance for aggregated traffic, while for guaranteed performance service, for multimedia services, performance bounds need to be provided on a per-flow or per-class basis. Recently, a number of new service disciplines, such as Weighted Fair Queueing (WFQ) [31] and its extensions, that are aimed to provide per-flow or per-class performance guarantees have been proposed in the context of high-speed packet-switching networks. However, the use of these scheduling algorithms implies the upper limit of possible delay jitter only in the form of trivial bounds. As a result, these conventional scheduling algorithms inherently face the problem of the trading-off between jitter bound and statistical multiplexing gain. On the other hand, another major direction of designing scheduling algorithms such as [21] is to support both flexible delay and jitter guarantees, using nonwork-conserving discipline. However, these algorithms have either high implementation complexity, making it difficult to be realized in high-speed networks, or cannot share bandwidth fairly like WFQ.

As far as the wireless environment is concerned, two key characteristics have made the conventional wireline-based QoS-enabling algorithms inapplicable: one is the bursty channel errors, and the other is location-dependent channel capacity and errors [32, 33]. In recent years, many scheduling algorithms, such as idealized

Wireless Fair-Queueing (IWFQ) [32, 33], etc., are proposed to accommodate QoS for wireless access networks. However, they make their efforts in the fair bandwidth sharing and tightening of delay bound. The flexible delay/jitter guarantees, the most important QoS requirements for RT applications, are not provided.

From above discussion, it is difficult to design a common scheduling algorithm to satisfy all QoS requirements for all networks from wireline access networks, backbone networks, to wireless networks. Consequently, the motivation of this dissertation is providing the QoS requirements in each part of networks. For backbone networks, due to the classification of RT and NRT traffic streams, flexible delay/jitter guarantees and the control of packet loss rate are provided to RT traffic, while low implementation complexity, low delay bound and fair bandwidth sharing are provided to NRT traffic. For wireline access networks, we would like to accommodate flexible delay/jitter control and residual bandwidth sharing should be accommodated via a single scheduling platform. Last but not least, differentiated QoS in delay/jitter, link utilization and fairness should be taken into consideration simultaneously in wireless access networks.

1.5 Topics to Be Addressed in This Dissertation

The major concern in this dissertation is that how to provide the QoS guarantees for not only the NRT data traffic but also the RT multimedia traffic in the networks. To realize such an objective, different scheduling policies should be designated to different network architectures as shown in Fig. 1.1. In general, there are four types of network nodes: the Internet node, the media switch node, the wireline egress/ingress node and the wireless egress/ingress node. First, in the backbone network, including the Internet switch node and the media switch node, the RT traffic streams and

NRT traffic streams are separated. Thus, we focus on designing hardware efficient scheduler for these two types of traffic streams respectively. Secondly, for wireline egress/ingress nodes, because the NRT and RT traffic streams are integrated, we consider how to accommodate different QoS requirements for these traffic streams via a single scheduling platform. Last, as far as wireless egress/ingress nodes is concerned, because the network environment is very different from wireline networks, conventional scheduling algorithms over wireline networks cannot be applied. Hence, we concern how to provide not only QoS for RT/NRT traffic streams but also fair channel usage and maintain high channel utilization simultaneously.

1.5.1 GFQ: The Gated-Scheduling Algorithm with Fair Residual Bandwidth Sharing for Internet Switches

In this part, a novel packet scheduling algorithm with low implementation complexity is presented. Most scheduling algorithms proposed so far usually involve a sorting operation with the complexity of $O(\log N)$ per packet, where N denotes the number of flows sharing the link. To solve this problem, we propose a new scheduling algorithm with the complexity $O(1)$, implemented with only one single FIFO queue for each port in the output scheduler. The proposed scheduling algorithm makes use of the concept of gated scheduling, and thus the algorithm is called the *Gated-Frame-Queueing* (GFQ) algorithm. The key contribution of the GFQ algorithm is the successful elimination of output sorter in its scheduler design such that the scheduling mechanism can accommodate large number of flows and the preservation of the advantages in low delay bound and fairness. Both delay bound and Fairness Index for flows scheduled under this algorithm are derived and validated with simulations.

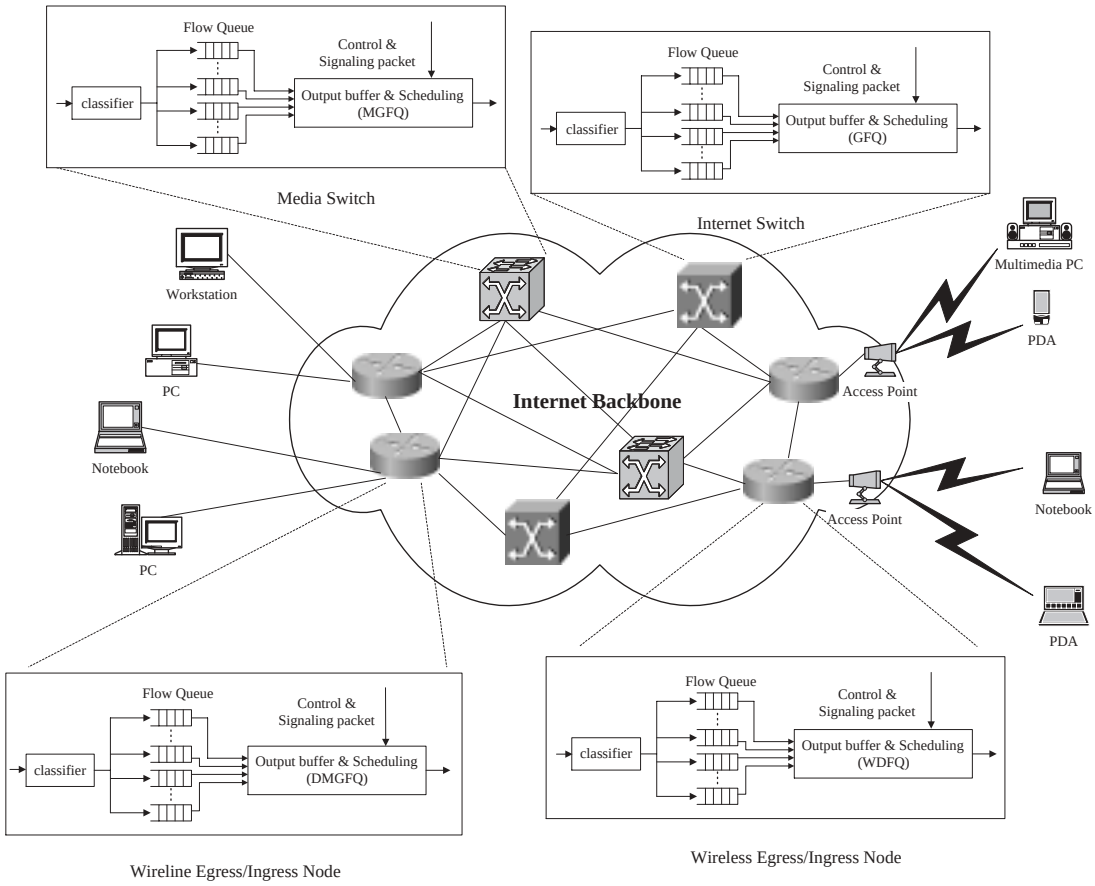


Figure 1.1: Several network scenarios discussed in this dissertation.

1.5.2 A Novel Framework and the MGFQ Scheduler with Jitter and Delay Guarantees for Media Switches

In this study, we propose a framework for RT multimedia transmission in ATM networks with an efficient traffic scheduling scheme called *Multi-layer Gated Frame Queueing* (MGFQ). MGFQ employs only one set of FIFO queues to provide a wide range of QoS for RT applications. We also propose special cell formats for RT multimedia transport and a hybrid design to allow MGFQ to combine its scheduling scheme with Age Priority Packet Discarding scheme. For this hybrid design, the cell level performance as well as the packet level QoS can be improved at the same time. Simulation results show that this hybrid design will be useful for packetized voice and progressive layer-compressed video transmission across the backbone networks. With the presented framework and the MGFQ algorithm, RT multimedia traffic streams can be much better supported in terms of cell/packet delay and jitter.

1.5.3 DMGFQ: A Delay/Jitter Guaranteed Traffic Scheduler with Fair Residual Bandwidth Sharing for Wireline Egress/Ingress Nodes

In the third study, we propose a new traffic scheduling scheme, called *Differentiated Multi-layer Gated Frame Queueing* (DMGFQ), to provide users with differentiated QoS, jitter control and fair bandwidth sharing simultaneously. DMGFQ assumes the underlying layer-2 protocol to be with fixed size PDU, and employs two extra FIFO queues for each VP to accommodate those eligible cells conforming to the guaranteed minimum bandwidth and the share weighting factor of residual bandwidth. We show that DMGFQ can achieve good performance for well-behaved flows, including delay, jitter and cell loss ratio, even under the presence of misbehaved flows. The

residual bandwidth released from other terminated sessions is also fairly distributed among all backlogged flows. In addition, a protection factor in DMGFQ provides the flexibility for network operators to achieve appropriate compromise between the network utilization and the service quality. As a result, DMGFQ not only supports a wide range of QoS for RT multimedia but also achieves the goal of protecting the users conforming to their service level agreements.

1.5.4 WDFQ: An Efficient Traffic Scheduler with Fair Residual Bandwidth Sharing for Wireless Egress/Ingress Nodes

Currently, the issues in Quality of Service, fairness and pricing strategies should have expedited the emergence of service differentiation in wireless access networks. Therefore, in the last part, we propose a novel scheduling algorithm, called the *Wireless Differentiated Fair Queueing* (WDFQ) algorithm, to accommodate such need by providing delay/jitter controls, and fair residual bandwidth sharing for RT and NRT traffic streams simultaneously. We show that the WDFQ scheme can achieve excellent performance, including timely delivery of RT traffic, virtually error-free transmission of NRT traffic, and fair usage of channel bandwidth among remote stations. In addition, the location-dependent channel error property, as appeared in most wireless networks, are considered in the model and the temporary short error bursts are compensated by credits of bandwidth. The simulation results suggest that the length of retransmission period should be adapted to the error length to achieve good performance and maintain low implementation complexity.

1.6 Thesis Organization

The rest of this dissertation is organized as follows. In Chapter 2, a survey on classical work-conserving and non-work-conserving scheduling schemes are presented. Chapter 3 proposes an efficient traffic scheduling algorithm with no need of sorting operations to provide fair residual bandwidth sharing and minimum bandwidth guarantees for NRT traffic streams. In Chapter 4, we design a scheduling algorithm to accommodate delay and jitter guarantees for RT multimedia traffic. This algorithm can be combined with the advanced packet discarding schemes to provide higher-layer QoS. In Chapter 5, a novel scheduling algorithm is proposed to satisfy different QoS requirements in NRT traffic and RT traffic simultaneously. Chapter 6 extends the algorithm proposed in Chapter 5 to support fair channel usage and maintain high channel utilization in wireless access network. Finally, conclusions are drawn in Chapter 7.

Chapter 2

Related Research Works

Many service disciplines are proposed in the literature for bandwidth allocation and transmission scheduling. In general, service disciplines can be characterized as either work-conserving or nonwork-conserving. A work-conserving server is never idle if the buffer in the system is nonempty, while a nonwork-conserving server may remain idle even if there are available packets to transmit. Nonwork-conserving algorithms, such as jitter-EDD (JEDD) [21], etc., are used to control delay jitter by delaying packets that arrive early. In this chapter, we briefly review recent scheduling algorithms, including work-conserving algorithms, non-work-conserving algorithms and scheduling algorithms in wireless access networks.

2.1 Work-Conserving Scheduling Algorithms

In this section, we will study several work-conserving disciplines for packet scheduling: Generalized Processor Sharing (GPS) [34, 35], Weighted Fair Queueing (WFQ) (also called packet-by-packet Generalized Processor Sharing (PGPS)) [34, 35], Worst-case Fair Weighted Fair Queueing (WF^2Q) [36], Self-Clocked Fair Queuing (SCFQ)

[37], Deficit Round Robin [38] and delay-Earliest-Due-Date (delay-EDD) [39]. Other extensions of these algorithms can be referred in [40, 41, 42, 43, 44].

2.1.1 Generalized Processor Sharing

A generalized processor sharing (GPS) [34] scheduler is characterized by N positive real numbers, $\phi_i, \phi_2, \cdots, \phi_N$, each of which corresponds to the bandwidth share weight of a flow. At any time t, the service rate for a backlogged flow i, denoted as $S_i(t)$, is calculated via the following equation:

$$S_i(t) = \frac{\phi_i}{\sum_{j \in B(t)} \phi_j} C, \qquad t \geq 0, \tag{2.1}$$

where $B(t)$is the set of backlogged flows at time t and C is the link speed. Thus, at each instant bandwidth is distributed among all backlogged flows in proportion to their reservations. This results in a bandwidth sharing policy, with perfect flow isolation and ideal fairness. In addition, as long as the average arrival rate is less than service rate $S_i(t)$ at any time t, the delay of a flow can be bounded as a function of the queue length of flow i, independent of the queues and arrivals of the other flows.

However, GPS is impractical because it assumes that the server can serve all backlogged flows simultaneously and that the traffic is infinitely divisible. In a more realistic packet system, only one flow can receive service at a time and an entire packet must be served before another packet can be served. Hence, its packet-by-packet version (known as PGPS or WFQ) and many extended scheduling algorithms under the same framework are proposed to accommodate some scheduling properties.

2.1.2 WFQ and WF^{2}Q

There are many different ways proposed in the past to approximate GPS in a packet switch. Among them, the most well known one should be the WFQ discipline [31], also known as PGPS proposed by Parekh and Gallager [34]. In WFQ, when the server completes the transmission of a packet at time τ, it picks, among all the packets queued in the system at τ, the first packet that would complete service in the corresponding GPS system. While WFQ uses only finish times of packets in the GPS system, WF^{2}Q, proposed by Bennett and Zhang [36], uses both start times and finish times of packets in the GPS system to achieve a more accurate emulation. In WF^{2}Q, when the next packet is chosen for service at time τ, the server only considers the set of packets that have started receiving service in the corresponding GPS system at time τ, and selects the packet among them that would complete service first in the corresponding GPS system.

In WFQ and WF^{2}Q, the updating operations of the state variables is based on a notion of virtual time. The evolution of virtual time measures the progress of the system and depends on system load. For WFQ and WF^{2}Q, the virtual time function $V(\cdot)$ during any busy period $[t_1, t_2]$ is defined as follows

$$V(t_1) = 0, \tag{2.2}$$

$$\frac{\partial V(\tau)}{\partial \tau} = \frac{1}{\sum_{j \in B_{GPS}(\tau)} \phi_j}, \quad \forall t_1 \leq \tau \leq t_2, \tag{2.3}$$

where $B_{GPS}(\tau)$ is the set of backlogged flows at time τ under the reference GPS system. Note that at time τ the backlogged set $B_{GPS}(\tau)$ may not be identical to the set of backlogged flows under WFQ or WF^{2}Q.

The virtual time of the WFQ system depends on how many other flows are active in the system. The dependency on virtual time introduces extra complexities for WFQ and WF^{2}Q since the system needs to emulate GPS and keeps track of

the number of active flows at any moment in GPS. Many other schemes have been proposed to simplify the calculation complexity of WFQ, such as [40]. However, the cost paid to reduce the calculation complexity is the degradation of the fairness and the delay bound.

2.1.3 Self-Clocked Fair Queueing

Both WFQ and WF^{2}Q need to emulate a reference GPS server. However, maintaining the reference GPS server is computationally expensive. One simpler packet approximation algorithm of GPS is Self-Clocked Fair Queueing (SCFQ) proposed by Golestani [37].

In order to reduce the complexity of computing virtual times, SCFQ adopts the virtual service time of the packet currently being serviced as the system's virtual time at any moment t to approximate the virtual time in GPS. Specifically, the virtual service time F_i^k of packet p_i^k is defined as

$$F_i^k = \max\{\hat{V}(a_i^k), F_i^{k-1}\} + \frac{L_i^k}{\phi_i}, \tag{2.4}$$

where the virtual time $\hat{V}(t)$ at time t is defined as

$$\hat{V}(0) = 0, \tag{2.5}$$

$$\hat{V}(t) = F^p, \quad s^p < t \leq f^p, \tag{2.6}$$

where F^p is the virtual service time of packet p, and s^p and f^p denote the times packet p starts and finishes service in the SCFQ systems.

Although the calculation of virtual times is simpler in SCFQ, the inaccuracy incurred results in much worse performance in SCFQ.

2.1.4 Deficit Round Robin

Although WFQ has nearly perfect isolation and fairness, unfortunately, it appears to be expensive to implement. Specifically, WFQ requires $O(N)$ work per packet to approximate the GPS system, where N is the number of packet streams that are concurrently active at the gateway or router. With a large number of active packet streams, WFQ is hard to implement at high speeds. On the other hand, ordinary round-robin service discipline has very low implementation cost. The major problem, however, is the unfairness caused by possibly different packet size used by different flows. Deficit Round Robin (DRR) propose by Shreedhar and Varghese [38] are capable of removing this flaw and still requiring only $O(1)$ implementation complexity.

DRR also uses round-robin servicing with a quantum of service assigned to service each queue. However, the difference from traditional round-robin is that if a queue was not able to send a packet in the previous round because its packet size was too large, the remainder from the previous quantum can be added to the quantum for the next round. Thus deficits are kept track off; queues that were shortchanged in the previous round are compensated in the next round.

DRR provides near-perfect isolation at very low implementation cost. However, its delay bound and fairness index is proportional to the frame size. In addition, the coupling problem of delay bound and bandwidth requirement still exists in DRR.

2.1.5 Delay-Earliest-Due-Date

Delay-earliest-due-date algorithm or delay-EDD proposed by Ferrari and Verma [39] is an extension to the classic earliest-due-date-first (EDD or EDF) scheduling [45]. A state variable called "Expected Deadline" ExD_i^k is associated each packet p_i^k in

flow i. If the minimum packet interarrival time is $X_{\min_i}$ and the local deadline bound of a flow is d_i, the expected deadline is calculated as:

$$ExD_i^k = \max(a_i^k, ExD_i^{k-1} + X_{\min_i}), \tag{2.7}$$

where a_i^k is the arrival time of p_i^k. The variable $X_{\min_i}$ captures the bandwidth allocated to flow i.

Unlike previous scheduling algorithms, delay-EDD decouple the allocation of delay bound and bandwidth. On the other hand, in WFQ, SCFQ, VC, etc., the delay bound is inversely proportional to the allocated long term average rate. This will result in a waste of resources when the low delay flow also has a low arrival rate. However, the complexity of the schedulability test of delay-EDD is very high. And, a flow, whose requested bandwidth is available, may not be accepted in the network even due to the restriction of the delay bound. Another problem of delay-EDD is that it can not provide a fair allocation of the free residual bandwidth among competing sources. Therefore, delay-EDD may not satisfy the applications that desire fair shares of free residual bandwidth.

2.2 Non-Work-Conserving Scheduling Algorithms

Although work-conserving scheduling disciplines can accommodate end-to-end delay bounds, buffer space requirements in each switching node and high link utilization, the traffic pattern is distorted inside the network due to network load fluctuation. As a result, the output traffic pattern is very difficult to predict. A typical approach to deal with the problem of traffic pattern distortions is to employ nonwork-conserving disciplines to control the distortions at each switch node.

A nonwork-conserving server may hold a packet for some purposes even when the server is idle. Although this may increase the average delay of packets and even

decrease the average throughput of the server, the burstiness of the stream is easier to be kept at the same level. Important nonwork-conserving disciplines proposed for high speed integrated services networks, such as Jitter Earliest-Due-Date (jitter-EDD or JEDD) [21], Stop-and-Go [46, 47], Rate-Controlled Static Priority (RCSP) [48], are introduced briefly in this section.

2.2.1 Jitter-Earliest-Due-Date

The jitter-EDD (JEDD) discipline proposed by Verma, Zhang, and Ferrari [21] extends delay-EDD to provide delay-jitter bounds (that is, a bound on the maximum delay difference between two packets). After a packet has been served at each server, a field in its header is stamped with the difference between its deadline and the actual transmission time. A regulator at the entrance of the next server holds the packet for this period before it is made eligible to be scheduled.

More specifically, if a packet p_i^k got served $PreAhead_{h-1}^k$ time units before its deadline in node $h-1$, then the eligible time ET_i^k, at which it is made eligible to be sent, at node h is calculated as

$$ET_i^k = a_i^k + PreAhead_{h-1}^k + D_{i,h} - J_{i,h}, \tag{2.8}$$

where a_i^k is the arrival time in node h, $D_{i,h}$ is the pre-allocated delay bound for flow i in node h, and $J_{i,h}$ is the delay jitter guaranteed for flow i in node h.

Although JEDD is able to accommodate exact delay and jitter bound for each flow or a class of streams, its implementation complexity is the same as delay-EDD. The schedulability test is also difficult to be performed.

2.2.2 Stop-and-Go

Stop-and-Go, proposed by Golestani [46, 47], divides the time axis into frames with periods of some constant length T. Packets arriving on an incoming link during a frame should always be held down until the next frame on the output link. Hence, a delay θ, where $0 \leq \theta < T$, is introduced for each packet to wait for transmission.

Suppose the link speed is r and the frame period is T. If no more than rT workload arrives during the frame, which is characterized by (r, T), Stop-and-Go ensures the delay bounds and the same traffic characterization (r, T) for the output traffic. Nevertheless, the framing strategy in Stop-and-Go introduces the problem of coupling between delay bound and bandwidth allocation granularity. The delay of any packet at a single switch is bounded by two frame times. To reduce the delay, a smaller T is desired. On the other hand, because the incoming traffic is characterized by (r, T), a larger T is preferred to accommodate flexible bandwidth allocation. Hence, there is a trade-off between low delay bound and fine granularity of bandwidth allocation.

2.2.3 Rate-Controlled Static Priority (RCSP)

A RCSP server, proposed by Zhang and Ferrari [48], is composed of a rate-controller and a static priority scheduler. Each of the flows traversing the server is assigned a regulator in the rate-controller to shape the input traffic of the corresponding flow into the desired traffic pattern. Upon arrival of each packet, an eligible time is calculated according to some algorithms and assigned to the packet by the regulator. Then, the packet is held in the regulator till its eligible time before being handled to the scheduler for transmission. The scheduler in a RCSP server has a number of priority levels with each priority level corresponding to a delay bound. Each flow is

assigned to a priority level during flow establishment time.

The key idea of RCSP is to separate rate-control and delay-control functions of the server which are performed by the rate-controllers and static priority schedulers, respectively. However, the static priority policy will result in low multiplexing gain. In addition, RCSP can only offer a fixed number of delay bounds, while JEDD is able to accommodate a continuous spectrum of delay bounds. The tradeoffs in choosing the number of priority levels and the delay bound associated with each priority level are not fully understood and need to be investigated further.

2.3 Scheduling Algorithms for Wireless Access Networks

The network environments of wireline network and wireless network are very different, due to the bursty link errors and location-dependent channel property. Thus, the conventional wireline-based QoS-enabling algorithms inapplicable in wireless access networks. In this section, we review several representative scheduling algorithms in wireless networks, including idealized Wireless Fair-Queueing (IWFQ) [32, 33], Channel-State Dependent Packet (CSDP) Scheduling [49], Effort-limited Fair (ELF) Scheduling Algorithm [50], and Channel-condition Independent packet Fair Queueing (CIF-Q) [51].

2.3.1 Wireless Fulid Fair-Queueing (Wireless FFQ)

Wireless FFQ proposed by Lu *et al.* adopts the FFQ system under error-free environment (also known as the GPS system) as its reference system [32, 33]. Specifically, given the arrival processes for each of the flows and the error patterns perceived by

each of the flows, wireless FFQ defines an error-free service as the FFQ service for the flows with identical arrival processes and completely error-free channels.

Wireless FFQ defines a flow to be lagging at any time instant if its queue length is greater than the queue length of its error-free service at the same time instant, otherwise, a flow is said to be leading. The key feature of the wireless FFQ model is to allow lagging flows to make up their lag by causing leading flows to give up their lead. By artificially bounding the amount of the lag and lead, the tradeoff can be made between long-term fairness and separation between flows.

Each arriving bit has a service tag, which is the virtual time of its error-free service. Then, the bit to be transmitted is chosen from the head of the queue of the flow with the minimum service tag among the backlogged flows which perceive a good channel. However, the service tag is also adjusted according to the predefined bounding amount of the lag and lead. Therefore, in wireless FFQ, a flow which leads by more than its upper leading bound does not have to "pay" for more than its leading bound; likewise, a flow which lags by more than its lagging bound cannot reclaim more than the lagging bound.

2.3.2 Idealized Wireless Fair-Queueing (IWFQ)

Because wireless FFQ is a fluid model, it is impractical in realistic wireless access networks. IWFQ is the packtization version of wireless FFQ algorithm. IWFQ makes two idealistic assumptions: (1) each flow knows whether it can transmit correctly in the current slot and (2) there is a perfect MAC protocol.

For error-free service, IWFQ and WFQ are identical. However, when some flows perceive short error-bursts, IWFQ performs local adjustments in channel allocation in order to compensate the flows for their channel errors. In addition to buffer overflow, packets may also be discarded in IWFQ if a flow lags too much. In order

to preserve the service precedence of a lagging flow, whose HOL packet may be forced to be discarded, IWFQ maintains a slot queue and a packet queue for each flow. By decoupling slot queues from packet queues, IWFQ can handle multiple types of delay and loss requirements for flows while still maintaining the precedence in channel access for lagging flows.

However, IWFQ does not consider the delay/jitter requirements in wireless multimedia applications. In addition, the guarantees for throughput and delay in IWFQ are tightly coupled. In many scenarios, especially multimedia applications, the decoupling of delay from bandwidth might be a more attractive approach [52].

2.3.3 Channel-State Dependent Packet (CSDP) Scheduling

The CSDP scheduler proposed by Bhagwat *et al.* has three components [49]: a set of per destination queues, a link status monitor, and a packet dispatcher. For each mobile host in the coverage area, the CSDP scheduler at the base station maintains a separate FIFO queue of packets. The packet dispatcher maintains the specific service policy among different destination queues. The state of channels between the base station and each mobile host is monitored by the link status monitor (LSM).

Many variations of the CSDP scheme are proposed to achieve different performance objectives. For example, Fragouli, Sivaraman and Srivastava propose a modified version of CBQ [53] with CSDP to achieve controlled sharing of the wireless link and improved radio channel utilization simultaneously [54]. In their scheme, the CBQ component provides the controlled sharing among multiple packet streams (fairness) while the CSDP component improves channel utilization (throughput) by taking into account the different states of the wireless link radio channel that may be seen by the different spatially distributed receivers.

CSDP mechanism avoids retransmitting the lost packet immediately following

an error transmission. Thus, in the presence of burst errors, HOL blocking is significantly reduced. This mechanism also yields better wireless link utilization at a marginal cost of software complexity. However, if the deferred period length is more than the TCP's timeout period, the source will timeout and retransmit a copy of the delayed packet, thereby unnecessarily increasing the load on the system.

2.3.4 Effort-limited Fair (ELF) Scheduling Algorithm

ELF proposed by Eckhardt and Steenkiste extends WFQ via dynamic weight adjustments and guarantees determined according to the error rate experienced by the flow [50]. The authors propose two special principles to design ELF. One is that the amount of capacity loss suffered by a flow should not be proportional to its bandwidth or its error rate, but should be configurable through administrative controls. The other is that it must be possible to administratively bound the amount of capacity that is lost due to location-dependent errors.

In order to achieve the above two principles, the air time spent on a flow and the actual useful throughput achieved by the flow must be distinguished. ELF calls the former one as the "*effort*" and the latter one as the "*outcome*." An ELF scheduler also adopt the concept of *power factor*, which is a control knob to be used to administratively implement a variety of fairness and efficiency policies, to achieve the outcome that is envisioned by users (either a specific throughput for reserved flows or a specific fraction of residual link capacity for best-effort flows).

By setting the power factor appropriately, administrators can control the degree to which the fidelity of a flow will be maintained in the presence of errors. However, again, ELF only guarantees the throughput of a flow while the flexible delay/jitter bounds are not accommodated. Hence, when fairness, differentiated QoS in delay/jitter, and link utilization are all taken into consideration, it is necessary to

redesign a new scheduling algorithm for wireless multimedia.

2.3.5 Channel-condition Independent packet Fair Queueing (CIF-Q)

The goals of the Channel-condition Independent packet Fair Queueing (CIF-Q) algorithm [51], proposed by Ng, Stoica and Zhang, are (1) delay and throughput guarantees for error-free flows, (2) long-term fairness for error flows, (3) short-term fairness for error-free flows, and (4) graceful degradation for flows that have received excess service. CIF-Q uses Start-time Fair Queueing (SFQ) [55] as its *reference* system. It provides long-term fairness via an additional parameter (called *lag*) and ensures delay and throughput guarantees for error-free flows via the concept of *forced compensation*. The parameter *lag* represents the difference between the service that a flow should receive in a reference error-free packet systems and the service it has received in the real system. An active flow is said to be *lagging* if its *lag* is positive, *leading* if its *lag* is negative and *satisfied* otherwise. Once the system becomes busy, CIF-Q select the non-leading, error-free and active flow with the minimum virtual time. This can ensures error-free non-leading flows get their fair share.

Although CIF-Q achieve the above four important properties in wireless networks. However, its implementation complexity ($O(n \log n)$) is still too high for a cost-effective implementation. In addition, the decoupling requirement of delay from bandwidth may not be achieved by CIF-Q.

Chapter 3

GFQ: The Gated-Scheduling Algorithm Over Packet Switching Networks with Fair Residual Bandwidth Sharing

3.1 Introduction

Because RT and NRT traffic streams in the backbone network are saperated for transmission, we focus on how to design an efficient scheduling algorithm for NRT traffic in this chapter. Currently, scheduling algorithms for NRT traffic streams may be classified as frame-based or sorted-priority-based [56]. In a frame-based scheduler, time is split into frames of either fixed or variable length. Reservations of flows are made in terms of the maximum amount of traffic at which the flow is allowed to transmit during a frame period. With a fixed frame size, the server may remain idle if flows transmit less traffic than their reservations over the duration of

a frame. Thus, the link utilization may be reduced. A representative of this class of algorithms is Stop-and-Go [46, 47]. In turn, most round-robin schedulers have no such drawbacks. They allow the frame size to vary, subject a maximum size. Thus if the traffic from a flow is less than its reservation, a new frame can be started early. The most famous round-robin examples are Weighted Round Robin (WRR) [57] and Deficit Round Robin (DRR) [38]. Usually, the frame-based scheduling algorithms have especially low implementation complexity. For example, DRR and WRR both achieve fair scheduling of bandwidth with $O(1)$ complexity. However, it is noted that DRR and WRR have a drawback that the end-to-end delay bound increases with the bandwidth granularity among flows and with the number of flows sharing the link [58]. Many variations of WRR try to reduce the scheduling delay of WRR. For example, URR (Urgency-based Round Robin) [42] maintains a *urgency-index table* to determine the serving sequence, but the complexity of maintaining the urgency-index table is $O(\log N)$.

On the other hand, within a sorted-priority scheduler there is always an additional global variable, usually referred to as the *virtual time*, associated with the outgoing link being scheduled. When a packet arrives, it is assigned a timestamp computed as a function of the virtual time. Packets are then sorted based on their timestamps, and are transmitted in that order. The most typical examples of the sorted-priority scheduling algorithms are the General Processor Sharing (GPS) [34, 35] and its extensions [36, 37, 60]. In contrast to the frame-based scheduling algorithms, the sorted-priority schedulers usually have smaller scheduling delay but have higher implementation complexity. The complexity of the sorted-priority scheduler comes from two aspects: timestamp calculation and output sorting operations. Taking weighted fair queueing (WFQ) [31], or packet-by-packet generalized processor sharing (PGPS) [34], as an example, the time required for selecting a packet to

be transmitted under WFQ is $O(N)$. Recently many other scheduling disciplines have been proposed to reduce the computation complexity of the *virtual time function*. These scheduling algorithms includes SCFQ (Self-Clocked Fair Queueing) [37], VirtualClock [59], FFQ (Frame-based Fair Queueing) [60], etc. These algorithms successfully reduce complexity to $O(\log N)$, which is the complexity of the output sorter. QLWFQ (Queue Length Based Weighted Fair Queueing) [61] and BSW (Binary Scheduling Wheels) [62] successfully reduces the complexity to $O(1)$, but their delay performance are sacrificed. Actually, each policy is a trade-off between implementation complexity and other desirable features such as delay characteristics, fairness, etc. A detailed comparison of various scheduling algorithms can be seen in [56].

In view of above mentioned research results, via either the frame-based or the sorted-priority approaches, to design a scheduling algorithm with both low complexity and low delay bound is not a trivial task. However, such scheduling mechanisms are especially desired in high speed backbone networks where different Quality of Services are to be supported. We believe the requirement of strict-QoS scheduling can be relaxed in backbone switch design in order to achieve large capacity in flow processing. Therefore, scheduling algorithms with high implementation complexity are not suitable to be implemented in the high speed backbone switches. In addition, above research results couple the minimum bandwidth guarantee with the bandwidth share. Thus, if a traffic stream with small burst requires larger reserved service rate, it may be assigned more bandwidth than what it needs in the conventional scheduling algorithm. This may restrict the flexibility of bandwidth usage of backbone networks.

Thus, in this chapter, we propose a new scheduling discipline, called the *Gated-Frame-Queueing* (GFQ) algorithm, which does not need output sorting circuits in its

implementation and hence becomes capable of supporting extreme large number of flows. Although its performance may not be compatible with conventional scheduling algorithm such as WFQ, it achieve the performance similar to SCFQ with much lower complexity. In addition, GFQ also decouple the bandwidth reservation and bandwidth share. Thus, variant QoS services can be accommodated by GFQ. The organization of the rest of this chapter is as follows. In section 3.2, operations of the GFQ algorithm and its queueing model are presented. Section 3.3 describes the fairness and delay bound of the GFQ algorithm. Simulation results are described in section 3.4. Finally, conclusions and future work are presented in section 3.5.

3.2 The Gated Frame Queueing (GFQ) Discipline

3.2.1 Minimum Bandwidth Guarantee for Traffic Streams

Many applications such as TCP sessions or WWW flows may require a minimum bandwidth to maintain their basic operations. If the granted bandwidth cannot meet this requirement, these TCP sessions or WWW flows may be disconnected or may not provide services. Therefore, the underlying network is responsible for providing minimum bandwidth guarantees for upper layer applications. This concept is also adopted by the ATM forum as GFR service class [20]. However, the conventional scheduling algorithms, such as WFQ, SCFQ, etc., couple the minimum bandwidth guarantees with bandwidth shares. If an application require large minimum bandwidth guarantee and small residual bandwidth share, this requirement cannot be accommodated by these conventional scheduling algorithm. In GFQ, the residual bandwidth share is decoupled from the minimum bandwidth guarantees. This mechanism provides more flexibility for bandwidth usage of applications. The following are the definitions of the required notations:

- M_i: the guaranteed minimum bandwidth of flow i when flow i is backlogged;
- ϕ_i: the share weighting factor for flow i to share the residual bandwidth;
- C: the output link service rate;
- $S_i(t)$: the granted service rate for backlogged flow i at time t;
- $B_X(t)$: the set of backlogged flows at time t under scheduling algorithm X, and it should be updated at any time instant t.

For convenience, M_i and $S_i(t)$ are all normalized by the link capacity C.

It is well known that the granted service rate of conventional scheduling algorithms for a backlogged flow i, $S_i(t)$, is often calculated and updated at any time t as t increases. For example, $S_i(t)$ of the Generalized Processor Sharing (GPS) algorithm [34] is calculated via the following equation:

$$S_i(t) = \frac{r_i}{\sum_{j \in B_{GPS}(t)} r_j}, \qquad t \geq 0, \tag{3.1}$$

where r_i is the normalized bandwidth share of flow i, $B_{GPS}(t)$ is the set of backlogged flows at time t under GPS system and it should be updated at any time instant t. Because the fluid flow model adopted by GPS cannot be applied to the practical network system, WFQ (also called PGPS) uses the *virtual time* to help to determine the service sequence of all backlogged flow. Suppose a server becomes busy at time ζ. Then, the *virtual time* of WFQ, $VT(t)$, is defined as

$$\begin{aligned} VT(\zeta) &= 0, \\ VT(t+\tau) &= VT(t) + \frac{\tau}{\sum\limits_{j \in \mathcal{B}_{GPS}(t)} r_j}, \quad t \geq \zeta. \end{aligned} \tag{3.2}$$

In eq. (3.2), WFQ has to maintain the set of backlogged flows under the GPS system, $B_{GPS}(t)$. However, this maintenance operation leads to a complexity issue. Other related scheduling algorithms, such as SCFQ [37], WRR [57], etc., all accommodate the fair bandwidth sharing via the same framework focus on reducing the complexity of WFQ. Note that in GPS and its extensions the weighting factor r_i determines the overall bandwidth share, not just on the share of residual bandwidth. Hence, these bandwidth sharing approaches are not appropriate for traffic that requires minimum bandwidth guarantee. In addition, if there is any residual bandwidth available, all flows should be provided with a better QoS via this residual bandwidth. Therefore, we suggest a bandwidth sharing criterion where a minimum bandwidth is guaranteed for a backlogged flow by the scheduler, and then the residual bandwidth is shared by all other flows that require more bandwidth following the pre-determined weighting factors (ϕ_i's).

In order to realize the above bandwidth allocation policy, and to reduce implementation complexity, we only update the backlogged set $B_{GFQ}(t)$ under the GFQ system and calculate the granted service rate of each backlogged flow only at the starting epoch of a fixed time interval, which is called the *refreshing period*. The length of the refreshing period is denoted as T and the starting epoch of n-th refreshing period is t_n, with $t_n = t_{n-1} + T$ for $n \geq 1$. Then, we suggest in GFQ that the bandwidth sharing criterion based on eq. (5.1) should be modified as

$$S_i(t) = M_i + \frac{\phi_i}{\sum_{j \in B_{GFQ}(t_n)} \phi_j}(1 - \sum_{j \in B_{GFQ}(t_n)} M_j), \qquad \text{for } t_n \leq t < t_{n+1}, n = 1, 2, \cdots, \tag{3.3}$$

where flow i is a backlogged flow. The definition of the *backlogged flow* will be described later in this section.

With only the bandwidth sharing criterion proposed in eq. (3.3), the rate control mechanism may still be difficult to be realized. Hence, we suggest the adoption of the service workload instead of the service rate. For example, it is only necessary to employ certain counters, and then whether a flow overuses its guaranteed minimum bandwidth the shares of service workload for the whole refreshing period can be determined. Before describing the calculation procedures of the service workload, we introduce the following notations. We assume the underlying (internal) switching block to be fixed and call it "cell" for simplicity.

- ϕ_i: the share weighting factor of residual bandwidth. The residual bandwidth of an output link is allocated according to ϕ_i, period by period;
- $\psi_i(t_n)$: the number of cells of flow i at the starting epoch of the n-th refreshing period;
- $W_i^r(t_n, t_{n+1})$: the *granted workload* for flow i within the n-th refreshing period, which is the service workload contributed by the guaranteed minimum bandwidth of flow i. For simplicity, we also call it *reserved workload*;
- $W_i^e(t_n, t_{n+1})$: the *granted workload extension* (which is called *extended workload* for short) for flow i within the n-th refreshing period, which is the workload contributed by the residual bandwidth observed within the n-th refreshing period and is beyond the reserved workload for $[t_n, t_{n+1})$;
- $W_i(t_n, t_{n+1})$: the *total granted service workload* for flow i within the n-th refreshing period, which is the sum of $W_i^r(t_n, t_{n+1})$ and $W_i^e(t_n, t_{n+1})$;
- $W_i^a(t_n, t_{n+1})$: the *actual service workload* for flow i within the n-th refreshing period.

For convenience, $W_i^r(t_n, t_{n+1})$, $W_i^e(t_n, t_{n+1})$, $W_i(t_n, t_{n+1})$, and $W_i^a(t_n, t_{n+1})$ are all normalized by the size of a single cell.

In order to allocate bandwidth more fairly, we classify the backlogged flows into two classes: one is *lightly backlogged* and the other is *heavily backlogged*. Their definitions are described as follows.

Definition 3.1 *A flow is called lightly backlogged within n-th refreshing period if its backlogged cells can be served by the reserved workload. On the other hand, if the backlogged cells of a flow cannot be served by the reserved workload, the flow is called heavily backlogged.*

When the n-th refreshing period starts and the integrity of a cell is taken into considerations, the reserved workload of a backlogged flow i can be calculated via the following recursive equation at t_n:

$$W_i^r(t_n, t_{n+1}) = \lfloor M_i(n+1)T \rfloor - \lfloor M_i nT \rfloor + I_i(t_n, t_{n+1}), \quad n \geq 1, \tag{3.4}$$

where $I(i, t_n)$ is an indicator function defined as follows.

$$I_i(t_n, t_{n+1}) = \begin{cases} -1, & \text{if } I_i(t_{n-1}, t_n) = 1, \\ 1, & \text{if } \sum_{j \neq i} W_j^r(t_n, t_{n+1}) < T, \\ 0, & \text{otherwise.} \end{cases} \tag{3.5}$$

However, for lightly backlogged flows, the number of cells at the starting epoch of the n-th refreshing period, $\psi_i(t_n)$, may be less than the reserved workload $W_i^r(t_n, t_{n+1})$. In order to allocate bandwidth more fairly, these residual time slots should be shared by other heavily backlogged flows. Therefore, the extended workload of a backlogged flow i is calculated at t_n:

$$W_i^e(t_n, t_{n+1}) = \left\lfloor \frac{\phi_i}{\sum_{j \in B_{GFQ}^H(t_n)} \phi_j} \left(T - \sum_j \min\{W_j^r(t_n, t_{n+1}), \psi_j(t_n)\} \right) \right\rfloor. \tag{3.6}$$

In order to evaluate the fairness of a scheduling algorithm for RT traffic, we introduce the notation of the *fairness index* within the interval $(s,t]$, $\mathcal{FRI}(s,t)$, which is defined as follows.

Definition 3.2 *Suppose flow i and flow j are any continuously backlogged flows within the n-th refreshing period. Then the Fairness Index ($\mathcal{FRI}$) is defined as*

$$\mathcal{FRI} = \max_{n,i,j} \left| \frac{W_i^a(t_n, t_{n+1}) - W_i^r(t_n, t_{n+1})}{\phi_i T} - \frac{W_j^a(t_n, t_{n+1}) - W_j^r(t_n, t_{n+1})}{\phi_j T} \right|, \quad (3.7)$$

Based on the definition provided in eq. (3.7), a smaller fairness index of a scheduling algorithm implies a better fairness level this algorithm can achieve.

3.2.2 Queueing Model of the GFQ Algorithm

The queueing model of the GFQ algorithm is shown in Fig. 3.1. This queueing model requires the underlying layer-2 protocol to employ fixed size PDU.

In GFQ, each flow is assigned a dedicated FIFO queue, which is called flow-queue. Each flow is associated with a class of services with a set of pre-determined cell-level QoS parameters, including guaranteed minimum bandwidth, maximum burst size, and residual bandwidth share. The flow processor is responsible for recording the update of flow information including admission of new flow and release of existing flows.

When a cell arrives, it is buffered in the corresponding flow-queue at first. Once the n-th refreshing period starts, the reserved workload, $W_i^r(t_n, t_{n+1})$, and the extended workload, $W_i^e(t_n, t_{n+1})$, of flow i are calculated according to eqs. (3.4) and (3.6). Then the flow processor informs all departure controllers (DCs) to open the gates and the cells conforming to the total granted service workload, $W_i(t_n, t_{n+1})$, are moved to the output queue. The output queue is a simple FIFO queue and no additional sorting operation is required. If the output queue is empty within the

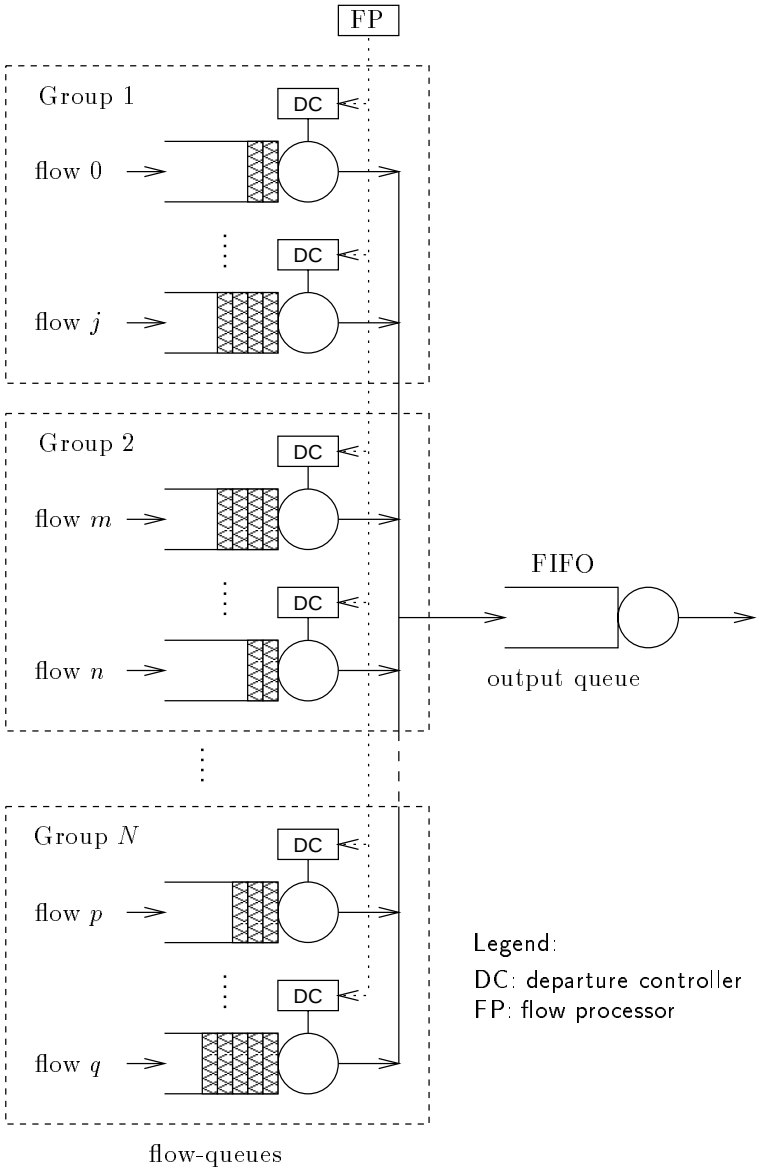


Figure 3.1: Queueing model of the GFQ algorithm.

refreshing period, the new arriving cells in the flow-queues are move to the output queue to increase the link utilization.

Based on the above queueing model and scheduling algorithm, we can observe that GFQ does not require complex virtual time calculation and sorting operations like conventional scheduling algorithms. The hardware implementation complexity is reduced significantly in GFQ. Figure 3.2 shows the pseudo code of the GFQ algorithm.

3.3 Performance Analysis

In this section, we analyze the fairness and delay bound of the proposed GFQ algorithm.

According to the definition of the Fairness Index described in eq. (3.7), the upper bound of the Fairness Index of the GFQ algorithm can be described in the following theorem.

Theorem 3.1 *The upper bound of Fairness Index of the GFQ algorithm is*

$$\mathcal{FRI} = \frac{\phi_j - \phi_i}{2\phi_i\phi_j}\left(1 - M_i - M_j + \frac{3}{T}\right) + \frac{1}{\phi_i T} + \frac{2}{\phi_j T}, \tag{3.8}$$

where i and j are always heavily backlogged within the interval $[t_n, t_{n+1})$.

Proof: Without loss of generality, we aggregate all flows other than flow i and flow j as flow k. In the following, n-th refreshing period is considered and, for simplicity, the notation of a variable within n-th refreshing period, denoted as $V(t_n, t_{n+1})$, is simplified as the notation V. Because the extended workloads of flow i and flow j, W_i^e and W_j^e, are distributed according to their residual bandwidth share, ϕ_i and ϕ_j. The most unfair condition occurs when $\psi_k = W_k^r + 1$. The reason is that the

main()
{
  while (1){
    t = system time;
    if (*cell_arrived*==TRUE){
      p = arrived cell;
      place cell p in the corresponding flow queue;
    }
    extract cell p from output buffer;
    if (*output queue is NOT empty*)
      extract HOL cell p from the output queue and transmit cell p;
    else
      extract HOL cell p from each flow queue and transmit it;
    if (*refreshing_event_occurs*==TRUE){
      refreshing_procedure(t);
      schedule next *refreshing_event* at time $t + T$;
    }
  }
}

refreshing_procedure(t)
{
  for (*each flow* i){
    calculate reserved workload $W_i^r(t, t+T) = \lfloor M_i(n+1)T \rfloor - \lfloor M_i nT \rfloor + I_i(t_n, t_{n+1})$;
    extended workload $W_i^e(t, t+T) = \left\lfloor \frac{\phi_i}{\sum_{j \in B_{GFQ}^H(t)} \phi_j} \left(T - \sum_j \min\{W_j^r(t, t+T), \psi_j(t)\} \right) \right\rfloor$;
    move $(\min\{\psi_i(t), W_i^r(t, t+T) + W_i^e(t, t+T)\})$ cells of flow i to the *output queue*;
  }
}

Figure 3.2: The pseudo code of the GFQ algorithm.

unused time slots assigned to flow k originally are distributed to flow i and flow j via round-robin scheme. The reserved workloads of flows i, j and k have following relations.

$$M_iT - 1 \leq \quad W_i^r \quad \leq M_iT + 1, \tag{3.9}$$

$$M_jT - 1 \leq \quad W_j^r \quad \leq M_jT + 1, \tag{3.10}$$

$$M_kT - 1 \leq \quad W_k^r \quad \leq M_kT + 1. \tag{3.11}$$

For simplicity of notations in following derivation process, we denote $T - W_i^r - W_j^r - W_k^r$ as Δ. Then three flows are granted extended workloads as follows:

$$W_i^e = \left\lfloor \frac{\phi_i \Delta}{\phi_i + \phi_j + \phi_k} \right\rfloor, \tag{3.12}$$

$$W_j^e = \left\lfloor \frac{\phi_j \Delta}{\phi_i + \phi_j + \phi_k} \right\rfloor, \tag{3.13}$$

$$W_k^e = \left\lfloor \frac{\phi_k \Delta}{\phi_i + \phi_j + \phi_k} \right\rfloor. \tag{3.14}$$

We know that under worst case, the actual service workload of flow k, W_k^a, is $W_k^r + 1$. And flow i and flow j are always heavily backlogged. Therefore, the actual workload of flow j is

$$W_j^a = W_j^r + W_j^e + \left\lfloor \frac{\Delta - W_i^e - W_j^e - 1}{2} \right\rfloor. \tag{3.15}$$

Then we have

$$W_j^e + \frac{\Delta - W_i^e - W_j^e - 1}{2} - 1 \leq W_j^a - W_j^r \leq W_j^e + \frac{\Delta - W_i^e - W_j^e - 1}{2}. \tag{3.16}$$

Similarly, we have

$$W_i^a = W_i^r + W_i^e + \left((\Delta - W_i^e - W_j^e - 1) - \left\lfloor \frac{\Delta - W_i^e - W_j^e - 1}{2} \right\rfloor \right), \tag{3.17}$$

and

$$W_i^e + \frac{\Delta - W_i^e - W_j^e - 1}{2} \leq W_i^a - W_i^r \leq W_i^e + \frac{\Delta - W_i^e - W_j^e - 1}{2} + 1. \quad (3.18)$$

Therefore, we may derive

$$\begin{aligned} &\frac{W_i^a - W_i^r}{\phi_i} - \frac{W_j^a - W_j^r}{\phi_j} \\ &\leq \frac{\Delta + W_i^e - W_j^e + 1}{2\phi_i} - \frac{\Delta - W_i^e + W_j^e - 3}{2\phi_j} \\ &= \frac{\Delta}{2}\left(\frac{1}{\phi_i} - \frac{1}{\phi_j}\right) + \frac{W_i^e - W_j^e}{2}\left(\frac{1}{\phi_i} + \frac{1}{\phi_j}\right) + \frac{1}{2\phi_i} + \frac{3}{2\phi_j}. \end{aligned} \quad (3.19)$$

According to eqs. (3.12) and (3.13), we have

$$\frac{\Delta(\phi_i - \phi_j)}{\phi_i + \phi_j + \phi_k} - 1 \leq W_i^e - W_j^e \leq \frac{\Delta(\phi_i - \phi_j)}{\phi_i + \phi_j + \phi_k} + 1. \quad (3.20)$$

Hence, we have

$$\frac{W_i^a - W_i^r}{\phi_i} - \frac{W_j^a - W_j^r}{\phi_j} \leq \frac{\Delta(\phi_j - \phi_i)}{2\phi_i\phi_j}\left(1 - \frac{\phi_j + \phi_i}{\phi_i + \phi_j + \phi_k}\right) + \frac{1}{\phi_i} + \frac{2}{\phi_j}. \quad (3.21)$$

It can be easily verified that if ϕ_k is much greater than ϕ_i and ϕ_j, and M_k is as small as possible, the worst case of fairness would be achieved. Hence, we can derive the Fairness Index ($\mathcal{FRI}$) as

$$\mathcal{FRI} = \frac{\phi_j - \phi_i}{2\phi_i\phi_j}\left(1 - M_i - M_j + \frac{3}{T}\right) + \frac{1}{\phi_i T} + \frac{2}{\phi_j T}. \quad (3.22)$$

Q.E.D

Then, we analyze the delay bound of the GFQ algorithm in three parts: the first part is the delay bound due to waiting for a new refreshing period, denoted as D_1, the second part is the delay bound in waiting for moving to the output queue, denoted as D_2, and last part is the queueing delay in the output buffer, denoted as D_3. In the following, we present the delay bound due to waiting for the new refreshing period at first.

Theorem 3.2 *The maximum delay bound, D_1, contributed from waiting for the new refreshing period is T.*

Proof: If a cell arrives just after the refreshing period, the cell has to wait for the next refreshing period to grant the reserved workload and extended workload. In this case, the maximum delay bound contributed from waiting for the new refreshing period is the length of refreshing period T.

Q.E.D.

Next, we describe the second part of delay bound in the following theorem.

Theorem 3.3 *Suppose the traffic of flow i satisfies the traffic descriptors (λ_i, B_i) as it enters the flow-queues, where λ_i is the average arrival rate and B_i is the maximum burst size. And the guaranteed minimum bandwidth is M_i, $M_i \geq \lambda_i$. Then the delay bound in the flow-queue waiting for moving to the output buffer is*

$$D_2 \leq \left\lceil \frac{B_i}{M_i T} \right\rceil T. \tag{3.23}$$

Proof: Let $A_i(0,t)$ denote the total arrival cells within the interval $[0,t]$ and $S_i(0,t)$ denote the total amount of cells moved to the output buffer within the interval $[0,t]$. Without loss of generality, we assume the whole refreshing procedures begins at time 0, i.e., $t_0 = 0$. Suppose the cells arriving at time t require n refreshing periods to be moved to the output buffer. Then, we have

$$A_i(0,t) \leq S_i(0, t + nT). \tag{3.24}$$

Therefore, we can derive

$$\lambda_i t + B_i \leq M_i(t + nT), \tag{3.25}$$

and then

$$n \geq \frac{B_i + (\lambda_i - M_i)t}{M_i T}. \tag{3.26}$$

Because $\lambda_i \leq M_i$, the minimum integer n is $\left\lceil \frac{B_i}{M_i T} \right\rceil$. Then we can derive the delay bound D_2 as

$$D_2 = \left\lceil \frac{B_i}{M_i T} \right\rceil T. \tag{3.27}$$

Q.E.D.

Last but not least, the last part of delay bound is the maximum queueing delay in the output buffer. This part of delay bound is described as follows.

Theorem 3.4 *The maximum queueing delay in the output buffer, D_3, is T.*

Proof: Because we consider the worst case delay bound, we assume all flows are always backlogged. At the starting epoch of the n-th refreshing period t_n, the total reserved workload of all flows is

$$\sum_i W_i^r(t_n, t_{n+1}) = \sum_i \left(\lfloor M_i(n+1)T \rfloor - \lfloor M_i n T \rfloor + I_i(t_n, t_{n+1}) \right). \tag{3.28}$$

Because the extended workload $W_i^e(t_n, t_{n+1})$ is contributed from the residual time slots within the refreshing period, $T - \sum_i W_i^r(t_n, t_{n+1})$, it is well enough to prove that if $\sum_i W_i^r(t_n, t_{n+1})$ is less than or equal to T, the delay bound D_3 is T. However, according to the definition of the reserved workload in eq. (3.4), we know that if there are positive number of indicator functions to be 1, then $\sum_i W_i^r(t_n, t_{n+1}) \leq T$.

We use mathematical induction to prove this argument. For $n = 0$, we have

$$\sum_i \lfloor M_i T \rfloor \leq T. \tag{3.29}$$

Therefore, there are $(T-\sum_i \lfloor M_i T \rfloor)$ indicator functions to be 1 within 0-th refreshing period. Hence, $\sum_i W_i^r(t_0, t_1) \leq T$ and the argument that the maximum queueing delay D_3 is T holds.

For $n = 1$, we know there are $(T-\sum_i \lfloor M_i T \rfloor)$ indicator functions to be -1 in the 1-st refreshing period. Hence, within 1-st refreshing period, the number of indicator functions whose values are 1 is as

$$T - \left(\sum_i \left(\lfloor M_i 2T \rfloor - \lfloor M_i T \rfloor - I_i(t_0, t_1) \right) \right) = 2T - \sum_i \lfloor M_i 2T \rfloor \geq 0. \tag{3.30}$$

Hence, $\sum_i W_i^r(t_1, t_2) \leq T$ and the argument that the maximum queueing delay D_3 is T holds.

Assume for $n = k$, $\sum_i W_i^r(t_k, t_{k+1}) \leq T$ and the argument that the maximum queueing delay D_3 is T holds. Because the server is work-conserving, We know that the number of the indicator functions whose values are 1 is $(k+1)T - \lfloor \sum_i M_i (k+1) T \rfloor$. Therefore, for $n = k+1$, the number of indicator functions whose values are 1 is as

$$\begin{aligned} & T - \left(\sum_i \left(\lfloor M_i (k+2) T \rfloor - \lfloor M_i (k+1) T \rfloor - I_i(t_k, t_{k+1}) \right) \right) \\ & = (k+2)T - \sum_i \lfloor M_i (k+2) T \rfloor \geq 0. \end{aligned} \tag{3.31}$$

Hence, $\sum_i W_i^r(t_{k+1}, t_{k+2}) \leq T$ and the argument that the maximum queueing delay D_3 is T holds for $n = k+1$.

According to the mathematical induction, we know the maximum queueing delay in the output buffer D_3 is T.

Q.E.D.

Then, the following theorem describes the overall delay bound of the GFQ algorithm.

Theorem 3.5 *Suppose the traffic of flow i satisfies the traffic descriptors* (λ_i, B_i) *as it enters the flow-queues, where* λ_i *is the average arrival rate and* B_i *is the maximum burst size. And the guaranteed minimum bandwidth is* M_i, $M_i \geq \lambda_i$. *Then the overall delay bound of flow i,* D_i^*, *under the GFQ algorithm is*

$$D_i^* = \left\lceil \frac{B_i}{M_i T} \right\rceil T + 2T. \tag{3.32}$$

Proof: According to Theorem 3.2 to Theorem 3.4, we know the worst case delay bound of GFQ is the sum of D_1, D_2 and D_3. Thus, the overall delay bound of flow i is

$$D_i^* = \left\lceil \frac{B_i}{M_i T} \right\rceil T + 2T. \tag{3.33}$$

Q.E.D.

Table 3.1 shows the delay bounds and implementation complexity of several scheduling algorithms. WFQ has the smallest delay bound and the highest implementation complexity. On the other hand, DRR has the lowest implementation complexity while its delay bound depends on the frame size F, which depends on the bandwidth allocation granularity and the number of flows. The GS algorithm provides a good compromise between delay bound and implementation complexity.

3.4 Simulation Analysis

In this section, simulation results are presented to illustrate delay and fairness performance of the GFQ algorithm. The performance metrics adopted in the experiments

Scheduling Algorithm	Delay Bound	Complexity
WFQ	$\frac{B_i+1}{r_i} + 1$	$O(N)$
SCFQ	$\frac{B_i+1}{r_i} + (N-1)$	$O(\log N)$
DRR	$\frac{B_i}{r_i} + 3F - \phi_i + 1$	$O(1)$
GFQ	$\left\lceil \frac{B_i}{M_i T} \right\rceil T + 2T$	$\approx O(1)$

Table 3.1: Delay bounds and implementation complexity of several work-conserving scheduling algorithms. Note that the reserved rate r_i, the guaranteed minimum bandwidth M_i, the frame size F and maximum burst size B_i are all normalized by the size of a single packet and the link capacity. The time unit in the table is "slot."

include average delay, maximum delay and Fairness Index experienced by individual traffic flows. Each flow is derived from a replay of LAN traffic trace obtained from the Lawrence Berkeley National Laboratory[63], with equally separated starting points within the 500000 trace positions. The average rate is 1.375 Mbps. Before the traffic stream enters the system, it is shaped by the leaky bucket with parameters (λ_i, B_i), where λ_i is the average arrival rate of flow i andB_i is the maximum burst size.

3.4.1 Experiment 1: Performance comparison between GFQ and other algorithms

In this simulation experiment, 25 flows share the same outgoing link. In order to simplify the presentation of simulation results, 25 flows are classified into five groups, where two of them, 4 flows, are misbehaving groups, attempting to transmit more

than their reservations. Detailed configuration of each group is shown in Table 3.2. WFQ [34], SCFQ [37] and DRR [38] are adopted to be compared with GFQ in this experiment. The deficit count of DRR is set as 100. In this simulation experiment, the refreshing period of the GFQ algorithms is set as 30 slot times, and the total simulation time is 10^7 slot times.

Group ID	Flow ID	Arrival Rate (λ_i)	Reserved Bandwidth	Bandwidth Share
0	0 – 1	0.1000	0.05	0.05
1	2 – 6	0.0498	0.05	0.05
2	7 – 8	0.1000	0.04	0.04
3	9 – 16	0.0398	0.04	0.04
4	17 – 24	0.0312	0.03125	0.03125

Table 3.2: Simulation configuration of Experiment 1. It is noted that the fields "Reserved Bandwidth" and "Bandwidth Share" are equivalent to "guaranteed minimum bandwidth" and "residual bandwidth share" in the GFQ algorithm.

In the first simulation scenario, the maximum burst size B_i is set to be 5. Figure 3.3 shows the average delay with 99% confidence intervals of four scheduling algorithms. In this scenario, it is observed that WFQ still has the smallest scheduling delay. This is achieved by the cost of high implementation complexity. We also can observe that average delay of the GFQ algorithm is similar to that of SCFQ. The maximum scheduling delays of four scheduling algorithms experienced by each flow and their corresponding delay bounds are shown in Fig. 3.4. From Fig. 3.4, we can observe again that SCFQ and the GFQ algorithm all have experienced almost similar maximum delays. On the other hand, although the complexity of DRR is

$O(1)$ and is slightly better than that of GFQ, but both average delay and maximum experienced delay of DRR are worse than the GFQ algorithm.

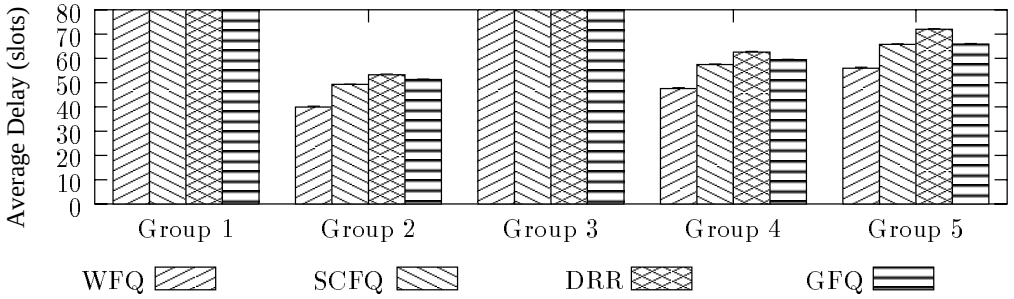


Figure 3.3: Average scheduling delay with 99% confidence intervals of four scheduling algorithms. (Burstiness of each traffic source B_i is 5 cells.)

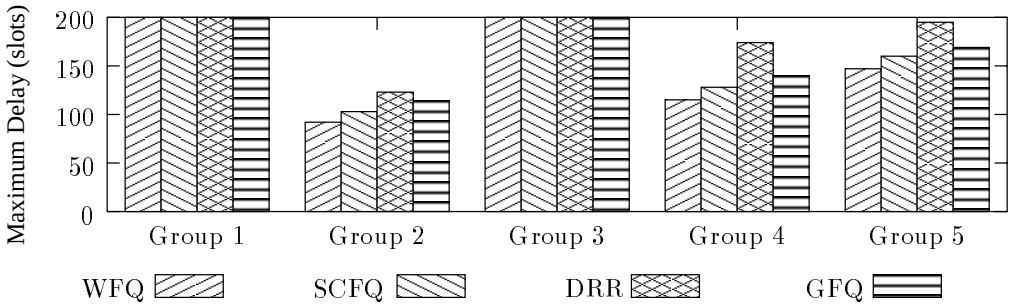


Figure 3.4: Maximum scheduling delay experienced by three behaved groups of flows. (Burstiness of each traffic source B_i is 5 cells.)

In order to examine the performance of variant scheduling algorithms in details, we present the theoretical delay bounds, simulated maximum delays of three well-behaved groups of four scheduling algorithms in Table 3.3. We also list the Fairness Index for WFQ, SCFQ, DRR and GFQ. From Table 3.3, we could observe that the

fairness achieved by our GFQ algorithm following (3.7) is close to those of WFQ and SCFQ. Again, DRR is much more unfair than other three scheduling algorithms. Last but not least, we observed that the delay bound of SCFQ is less than GFQ. However, according to the delay bound in Table 3.1, the delay bound of SCFQ increases with the number of flows. Therefore, if the number of flows becomes larger and larger, the performance of SCFQ becomes worse than GFQ.

Group ID	WFQ		SCFQ		DRR		GFQ	
	Theo. (slots)	Sim. (slots)	Theo. (slots)	Sim. (slots)	Theo. (slots)	Sim. (slots)	Theo. (slots)	Sim. (slots)
2	121	92	144	103	390	123	180	114
4	151	115	174	128	417	174	210	140
5	193	147	216	160	454	235	240	169
$\mathcal{FRI}$	4.267		4.267		7.467		4.267	

Table 3.3: Theoretical delay bounds, simulated maximum delays and fairness indices of different scheduling algorithms under maximum burst size 5 cells.

In order to emulate bursty network traffic, we also increase the bucket depth B_i to 50 for each flow to evaluate the performances of four scheduling algorithms, and the total simulation time is also 10^7 slot times. Figure 3.5 and Fig. 3.6 show the average delay with 99% confidence intervals and maximum experienced delay of four scheduling algorithms. We can observe that even under bursty network traffic, the GFQ algorithm still has similar performance to SCFQ.

The simulation results of delay and fairness verify the point we make in the beginning of the chapter that the GFQ algorithm achieves similar, even better, performance to SCFQ with much lower implementation complexity. This property

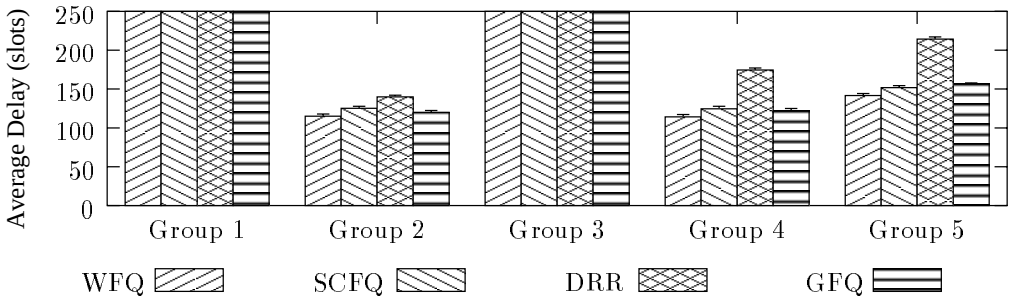


Figure 3.5: Average scheduling delay with 99% confidence intervals of four scheduling algorithms. (Burstiness of each traffic source B_i is 50 cells.)

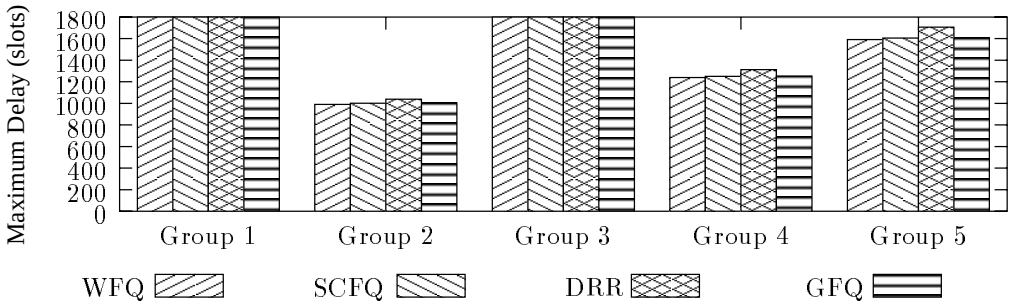


Figure 3.6: Maximum scheduling delay experienced by three behaved groups of flows. (Burstiness of each traffic source B_i is 50 cells.)

Group ID	WFQ		SCFQ		DRR		GFQ	
	Theo. (slots)	Sim. (slots)	Theo. (slots)	Sim. (slots)	Theo. (slots)	Sim. (slots)	Theo. (slots)	Sim. (slots)
2	1021	990	1044	1001	1290	1038	1080	1011
4	1276	1239	1299	1250	1542	1313	1320	1263
5	1633	1592	1656	1606	1894	1707	1680	1621
$\mathcal{FRI}$	6.667		5.333		12.000		4.267	

Table 3.4: Theoretical delay bounds, simulated maximum delays and fairness indices of different scheduling algorithms under maximum burst size 50 cells.

makes the GFQ algorithm capable of accommodating large number of flows in very high speed networks.

3.4.2 Experiment 2: Properties of guaranteed minimum bandwidth and residual bandwidth share in GFQ

In this simulation scenario, we examine the transient behavior of the GFQ algorithm for its special property, minimum bandwidth guarantee and residual bandwidth share. Four flows are employed in this simulation experiment and their configurations are shown in Table 3.5. The source state diagram of all flows is shown in Fig. 3.7. Flow 0 is activated at 3 sec. Flow 1 is activated at 0 sec and deactivated at 2 sec, and then re-activated at 3 sec. Flow 2 is activated at 2 sec, while flow 3 is activated at 3 sec. The traffic sources are also derived from the LAN traffic trace obtained from the Lawrence Berkeley National Laboratory[63]. In order to observe the behaviors of bandwidth sharing more clearly, we overload the whole

system and make all flows always heavily backlogged. The output link bandwidth is also assumed 45 Mbps.

	Arrival Rate (Mbps)	Reserved BW M_i (Mbps)	Max. Burst Size B_i (cells)	Residual BW Share ϕ_i
Flow 0	40.5	13.5	50	0.3
Flow 1	40.5	4.5	50	0.5
Flow 2	40.5	4.5	50	0.1
Flow 3	40.5	22.5	50	0.1

Table 3.5: Simulation configuration for Experiment 2.

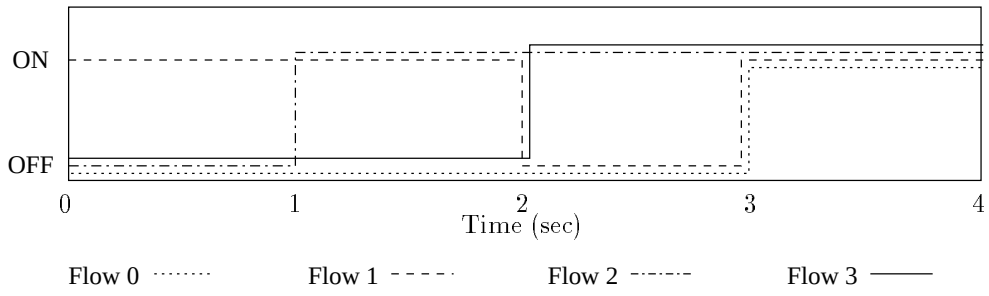


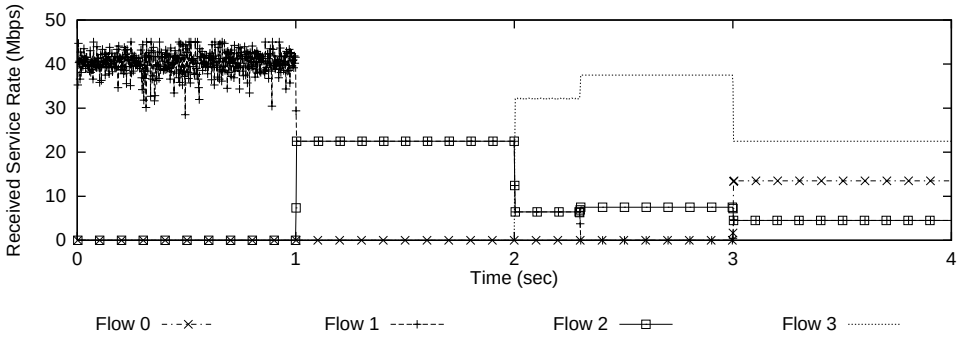
Figure 3.7: The state diagram of all traffic sources in Experiment 2.

Before 1 sec, only flow 1 is activated and it is granted all bandwidth, 45 Mbps. During the interval $[1, 2]$ sec, flow 1 and flow 2 are activated flows. The granted-service rate of flow 1 should be $4.5 + (45 - 4.5 - 4.5)\frac{0.5}{0.5+0.1} = 34.5$ Mbps while the granted-service rates of flow 2 should be $4.5 + (45 - 4.5 - 4.5)\frac{0.1}{0.5+0.1} = 10.5$ Mbps. However, it can be observed that WFQ and DRR guarantee the minimum bandwidth

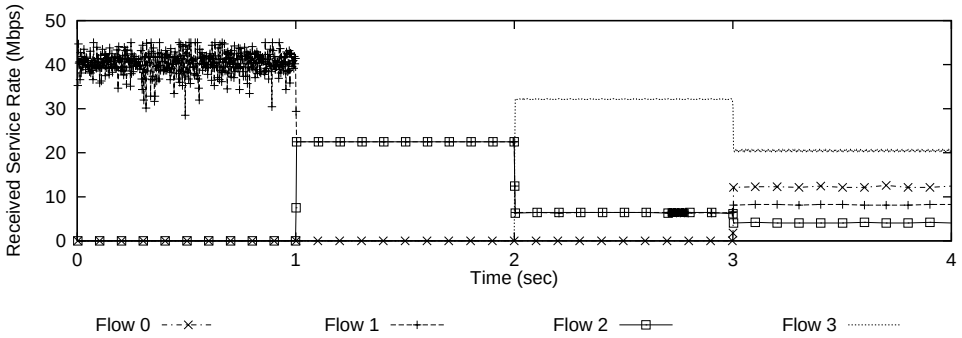
but they cannot distribute the residual bandwidth according to the bandwidth share factors (see Fig. 3.8(a) and (b)). Next, during the interval $[2, 3]$ sec, flow 1 is deactivated and flow 3 is activated. However, during the interval $[2, 2.15]$ sec, flow 1 is still backlogged due to queueing cells arriving within $[1, 2]$ sec. After 2.15 sec, flow 2 and 3 are backlogged flows and their granted service rates are 13.5 and 31.5 Mbps. Again, WFQ and DRR cannot assign bandwidth according to guaranteed minimum bandwidth and residual bandwidth share. Last, after 3 sec, all four flows are all activated and backlogged. The granted service rates of four flows are the same as the reserved bandwidth in Table 3.5. We can observe that WFQ assigns bandwidth fairly but DRR does not. However, from Fig. 3.8(c), we can observe that GFQ assigns bandwidth according to specified guaranteed minimum bandwidth and residual bandwidth share. The simulation results validate GFQ can decouple the guaranteed minimum bandwidth and residual bandwidth share easily, but most existing scheduling algorithms cannot.

3.5 Concluding Remarks

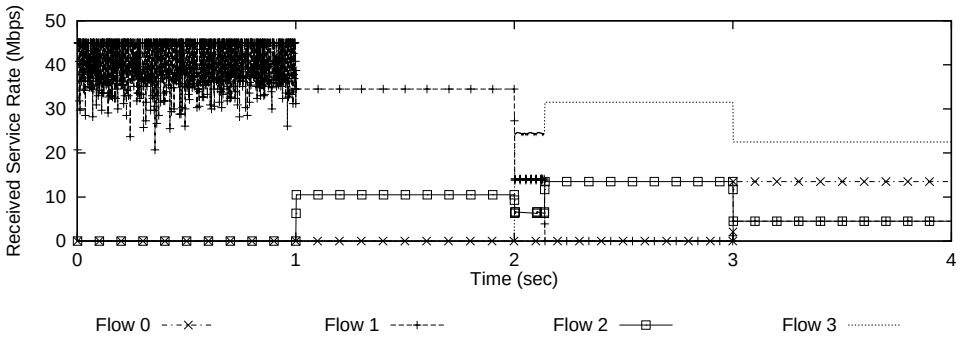
In this chapter we propose the GFQ algorithm at first. The GFQ implements a simple FIFO queue in its output buffer. The sorting operation is not needed in GFQ, therefore, the operation speed of the sorter is no longer the bottleneck of the scheduler. Hence, different QoS requirements, even with large number of flows, can be supported by the GFQ algorithm. Then, we show that the fairness is achieved and delay is bounded. In the simulation, the performance of the GFQ algorithm is examined by applying the leaky-bucket shaped traffic sources derived from the LAN trace. The results are compared with WFQ, SCFQ, and DRR. The simulation results show that though the GFQ algorithm performs slightly worse than WFQ,



(a) The WFQ algorithm.



(b) The DRR algorithm.



(c) The GFQ algorithm.

Figure 3.8: Transient bandwidth sharing behaviors of variant scheduling algorithms.

but it is comparable to SCFQ. Hence, the GFQ algorithm is able to achieve similar performances to SCFQ via much lower implementation complexity. Although DRR has slightly better complexity than GFQ, but its performance is much worse than GFQ.

In summary, the key contribution of the GFQ algorithm is the successful elimination of the output sorter in its designs while the scheduling mechanism can still accommodate large number of flows. We believe there are other scheduling approaches that are capable of handling large number of flows, without involving sorting procedures. But factors such as the FIFO constraint, Fairness Index, decoupling of minimum bandwidth guarantee and residual bandwidth share, and delay performance requirements will still affect the design direction of scheduling algorithms.

Chapter 4

A Novel Framework and the MGFQ Scheduler for Real-Time Multimedia Transport with Jitter and Delay Guarantees

4.1 Introduction

In Chapter 3, we propose an efficient scheduling algorithm GFQ with fair residual bandwidth share for NRT traffic streams over backbone networks. In this chapter, we pay our attention to providing QoS for RT traffic streams.

Many scheduling algorithms, such as Weighted Fair Queueing (WFQ) [31], Weighted Round-Robin (WRR) [57], etc., have been proposed for general data communications. However, these algorithms simply deal with the reduction of implementation complexity and the improvement of packet delay bound and fairness. In other words, they are not designed to meet the requirements of real-time traffic streams. For ex-

ample, one may not need to reduce the cell delay bound as small as possible when real-time streams are conveyed. Instead, one can choose to increase the statistical multiplexing gain as large as possible and meet the delay/jitter constraints at the packet level at the same time. Currently, nearly all data communication scheduling algorithms adopt work-conserving disciplines. As a result, they can only limit the CDV to trivial bounds. As is known, scheduling algorithms, such as WFQ and its extensions, inherently face the problem of trading off between jitter bound and statistical multiplexing gain. In other words, the duration over which the statistical multiplexing gain is performed must be restricted if a tight jitter bound is desired. Conversely, if the multiplexing gain is to be maximized, then the jitter bound must be relaxed and this may lead to the need for transmission overhead for source clock recovery.

A typical representative of the scheduling algorithm supporting both flexible delay and jitter guarantees is the jitter-Earliest-Due-Date (JEDD) [21] proposed by Verma *et al.* After each packet is served and prepared to transmit to its downstream node, the *due-date*, which is the difference between its local transmission deadline and actual transmission time, is inserted into a field of packet header. A regulator at the ingress of the next node holds the packet for a period before it is made eligible to be scheduled without violating the jitter bound. The node then transmits those eligible packets in an increasing order of their due-dates. However, the complexity of "winner selection" or "queue insertion" operation in the JEDD algorithm makes it difficult to be realized by a cost-effective hardware implementation. It is proved that the advanced buffer management mechanisms, such as the PPD [64] and the APPD [65, 66] schemes, can avoid the waste of network resources and improve the quality of service in the TCP or application layer at the receiving node. However, the implementation cost for such buffer management to be applied in combination with

the JEDD algorithm could be very high because of the winner selection operations of the JEDD in the output buffer.

In order to reduce implementation complexity, Pocher *et al.* introduces the delayed frame queueing (DFQ) service discipline [67] and adopts the concept of rotating-priority-queue (RPQ) [68] proposed by Liebeherr *et al.* to achieve the delay and jitter guarantees. The service queue of each link is organized as a sequential row of several FIFO buffers. Service priority is given to the cells buffered in the FIFO queues with the smallest index value. Via employing RM cells, DFQ discipline can support a variety of delay and jitter bound combinations without sacrificing the fair distribution of QoS violations among the traffic streams. However, a trade-off exists between the scheduling performance and transmission overhead when DFQ scheme is employed. In addition, the number of the FIFO-queue sets in DFQ scheme must be the same as the number of the supported jitter levels. This implementation cost leads to the limitation on the scalability and granularity of jitter level for DFQ.

Hence, in this chapter we propose a framework for multimedia transmission using a novel traffic scheduling scheme, called Multi-layer Gated Frame Queueing (MGFQ) for real-time traffic [69]. The rationale of the MGFQ algorithm is to accommodate an arriving cell into the proper FIFO queues according to its due-date in the current node, where the due-date is calculated based on the previous due-date passed over from its upstream node. The goal of the MGFQ algorithm is to provide efficient real-time traffic scheduling, with a minimum level of processing, and yet satisfies different QoS including jitter and delay of various scales and granularities. The framework is supplemented with special cell formats for real-time voice and video streams. In addition, we also propose a hybrid design, called MGFQ with APPD [65, 66], to combine scheduling scheme and packet discarding scheme. With this hybrid design, cell discarding does not follow the so called "tail-drop" policy and

the loss ratio is improved directly at the packet level.

The organization of this chapter is as follows. In section 4.2, the special cell formats for real-time transport and the proposed traffic scheduler combined with selective packet discarding schemes is presented. The due-date calculation procedure is described in section 4.3. And the implementation complexity is discussed in section 4.4. Simulation results of voice and video traffic are shown in section 4.5 and section 4.6, respectively. And finally, in section 4.7, we draw our conclusions.

4.2 The Framework for Supporting QoS Real-Time Traffic

In this section, we first describe the cell format for MGFQ scheme, which carries necessary due-date information without incurring significant protocol overhead. And, we illustrates how deu-date is carried in voice and video streams. Then, the MGFQ operations in an ATM switch are described.

4.2.1 Cell Format for Real-Time Transport

We adopt AAL2 defined in ITU-T recommendation I.363.2[70] as our ATM adaptation layer protocol for voice and make some modifications to AAL2 in order to carry necessary due-date information. The resulting protocol stack of Voice over ATM and cell format is shown in Fig. 4.1. In order to improve network utilization, we assume one or multiple voice calls can be carried in one ATM virtual channel (VC). A new field to support MGFQ, i.e., the due-date field, is assigned behind the *Start field* (STF) and is allocated 2 bytes for each ATM cell. In the *due-date* field, if we denote 12 bits as x and other 4 bits as y, then this *due-date* field represents $x \cdot 2^y$

time slots. The definitions and formats of other fields follow the definitions in [70].

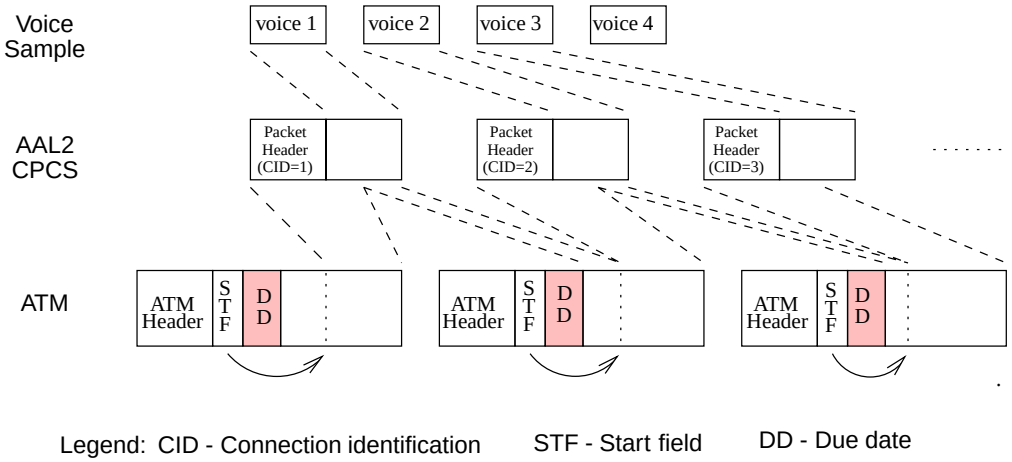


Figure 4.1: The cell format for voice traffic in the MGFQ algorithm.

Next, we introduce the protocol stack for transport video over ATM with MGFQ. We recommend the use of AAL5 and also adopt RTP [71] and the principle of Application Level Framing [72] to minimize the impact on the receiver's frame-level QoS degradation due to cell losses. With the protocol stack of RTP/UDP/IP/AAL5, the header information of all layers can be accommodated in the first two cells of a video frame. We use an additional *DD* (*due-date*) field in AAL5 overhead to carry necessary information for MGFQ as shown in Fig. 4.2. An optional RTP header extension is also suggested. The first field, *FT*, denotes the type of the video frame, such as I-frames,P-frame and B-frames. The field is useful to enforce selective discarding to further improve video frame playback performance. The *frame sequence number* and *frame size* (in cells) should be helpful to additional buffer management and error control. The *timing field* carries timing information needed in the RTP and applications. If this RTP header extension is adopted and is passed over to the AAL layer, the timing field of RTP header extension can be

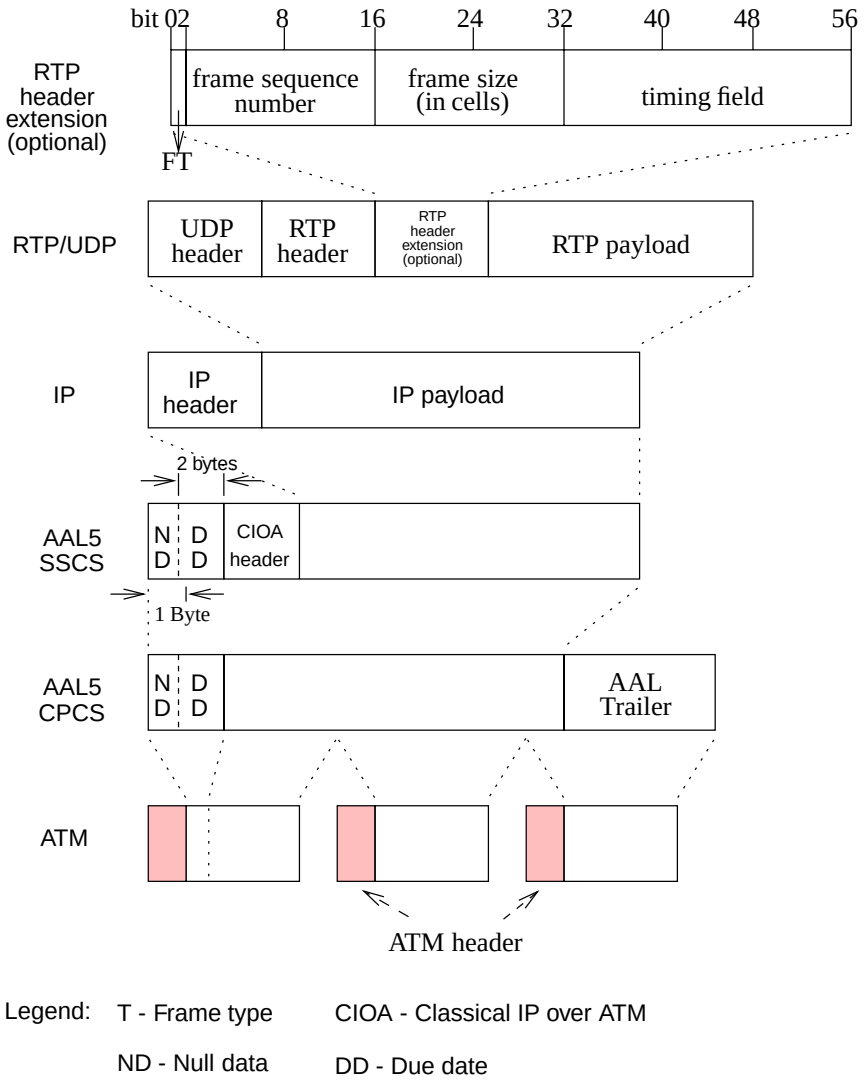


Figure 4.2: The protocol stack and cell format for video traffic.

extracted and transformed into *DD* field in AAL5. Otherwise, the due-date can be calculated with the timing information that is passed over directly from Application Programming Interface, in addition to Application PDU. Overhead of Classical IP over ATM [73] is then included. In the Service Specific Convergence Sublayer (SSCS) of AAL5, we assign a 1-byte dummy data and a 2-byte due-date field before the regular video data[1]. After an SSCS PDU pass through Common Part Convergence Sublayer (CPCS) and is segmented by SAR of ATM, the due-date field of the first cell of a video frame should be at the same position as the voice cell. Therefore, the operations of the ATM switch are similar for both voice and video cells. As long as the switch can detect whether an incoming cell is a BOM (Beginning of Message) cell, the switch always extracts the correct due-date information of a video frame packet.

4.2.2 Operations of the MGFQ Algorithm

The queueing model of MGFQ is shown in Fig. 4.3. Each virtual path (VP) is assigned a dedicated FIFO queue. We assume each virtual path is dedicated to a class of services with a set of pre-determined cell-level QoS parameters, including delay, jitter, and cell loss ratio, etc. Thus, the cells in the same VP queue can be served by FCFS discipline. In addition, VPs of the physical link are organized as several groups according to VPs' jitter bounds. The jitter bounds of all VPs in Group i are within $[(i-1)T+1, iT]$ slot times. Thus, T decides the granularity of the jitter bounds. T also denotes the length of the period that parameters in the scheduling operations are updated. This period is called *refreshing-period*, and is explained in

[1]It is noted that if the switch is able to perform different processing on voice and video cells according to their VPIs and VCIs respectively, then the *null data* field in the video cell can be eliminated.

further details later. Without loss of generality, we assume the assignment of the group identifier is in the increasing order of the jitter bounds. In other words, Group i is assigned the tighter jitter bound than Group $i+1$. Some FIFO queues called *temporary-queues* in this scheme are also dedicated for each group. The function of *temporary-queue* of Group i is to buffer the cells which were eligible in Group $i+1$ during the last refreshing-period. Here, a cell is called *eligible* if it does not violate the nodal delay bound and nodal jitter bound. In other words, the *temporary-queue* i buffers the cells whose due-dates were within the interval $[iT+1, (i+1)T]$ in the last refreshing-period. Next, the flow processor (FP) informs the due-date departure-controllers (DDCs) to open the "gate" with period T. When DDCs of Group i open their gates, eligible cells belonging to Group i are moved to the temporary queue $i-1$. In order to reduce the implementation complexity, the jitter bound of each VP has to be ceiled as the integer multiple of T. In order to improve QoS regarding packet loss ratio, the output buffer can employ a FIFO queue combined with two packet discarding schemes: Partial Packet Discarding scheme (PPD) [64] and Aged Priority Packet Discarding scheme (APPD) [65, 66].

The operations of the MGFQ algorithm are described as follows. Suppose the nodal jitter bounds of all VPs in this node is within the interval $[0, NT]$, then N temporary queues are dedicated to buffer eligible cells. Each time when a cell arrives, the initial nodal due-date and the eligible time of the cell are calculated. If the cell is for voice, the initial nodal due-date is calculated directly based on its own *DD* field. For video cells, such calculation is based on the *DD* field of the BOM cell. If this arriving cell is not eligible right now, it is attached to its own VP queue. Otherwise, it is put into the corresponding temporary-queue according to its due-date. When the flow processor informs all DDCs to open the "gate," the eligible cells in the temporary queue i are moved to the temporary-queue $i-1$.

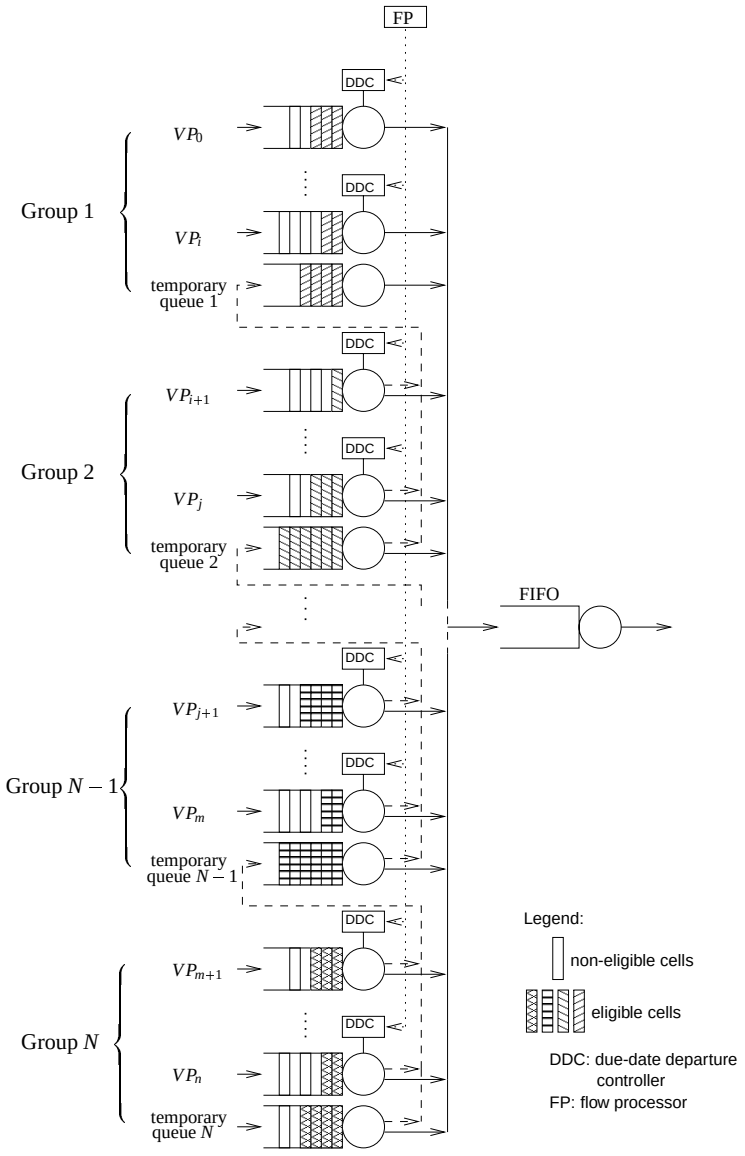


Figure 4.3: Queueing model of the MGFQ algorithm for real-time traffic.

Next, eligible cells originally belonging to Group i are also moved to the temporary-queue $i-1$. And, the cells whose due-dates are within $[(i-1)T+1, iT]$ are marked eligible. After above operations, the eligible cells in Group 1 are moved to the output buffer during the period. And eligible cells in Groups 2, 3, 4 and so on, can be sent to output buffer during this period, if the high priority Groups are empty. We called this procedure as the "*refreshing procedure*." At the same time, APPD and PPD mechanisms can be also applied to arrange eligible cells into the output buffer to avoid unnecessary waste of transmitting the cells that can not be re-assembled into a useful data unit in the receiver due to cell overdue. The detailed operations of APPD and PPD mechanisms are available in [65] and [66], respectively. In order to reduce the implementation complexity, we perform the refreshing procedure only at the starting epoch of refreshing-periods. If the output buffer is empty during the refreshing-period, then the eligible cells are served in the increasing order of group number. If the output buffer is not empty at the starting epoch of the refreshing-period, the remaining cells in the output buffer are discarded because of their violations of delay bounds. In order to more precisely describe the operation of the MGFQ algorithm, in Fig. 4.4 and Fig. 4.5, we present the pseudo code for an implementation of MGFQ.

4.3 The Due-Date Calculation Procedure

Before describing the calculation procedure of the due-date of every arriving cell, we introduce the following notations. Without loss of generality, our due-date calculation procedure is focused on the virtual channel j (VC_j) of virtual path i (VP_i). Hence, in the following discussion, we omit the subscripts i and j which represents VC_j and VP_i. Three delay and jitter parameters are essential:

```
main()
{
  while (1) {
    t = system time;
    if (cell_arrival_event_occurs == TRUE) {
      p = arrived cell;
      calculate initial nodal due-date and eligible time of p;
      enqueue p into its VP queue;
    }
    if (refreshing_event_occurs == TRUE) {
      discard all cells in the output buffer;
      refreshing_procedure();
      schedule next refreshing_event at time t + T;
      α = 2;
    }
    if (cell_output_event_occurs == TRUE) {
      if (output_buffer != EMPTY)
        p = HOL cell of the output buffer;
      else {
        do {
          p = HOL eligible cell in Group α;
          if (p == NULL)
            α = α + 1;
        } while((p == NULL) and (α ≤ N))
      }
      if (p != NULL) {
        calculate the due-date of p;
        transmit p;
      }
      schedule next cell_output_event at time t + 1;
    }
  }
}
```

Figure 4.4: Pseudo code of the MGFQ algorithm applied to the real-time traffic streams.

```
refreshing_procedure()
{
  for (i = N; i ≥ 2; i − −){
    move eligible cells in Group i to temporary queue i − 1;
    mark cells in VP queues of Group i whose due-dates are
     within the range [(i − 1)T + 1, iT] as eligible cells;
  }
  mark cells in VP queues of Group 1 whose due-dates are
   within the range [0, T] as eligible cells;
  move eligible cells in Group 1 to the output buffer
   with enforcing APPD and PPD procedure;
}
```

Figure 4.5: Pseudo code of the refreshing procedure in the MGFQ algorithm.

- ND^h: the nodal cell delay bound assigned to VP_i at node h, $h = 1, 2, ..., H$;
- J^h: the nodal cell jitter bound assigned to VP_i at node h, $h = 1, 2, ..., H$;
- $LD^{h-1,h}$: the propagation delay of the link between node $h-1$ and node h, $h = 2, ..., H$.

Here, we simply assume node 1 and node H are the ingress node and the egress node of the network, respectively. Although the assigned nodal delay bound ND^h and nodal jitter bound J^h are ND^h and J^h could be arbitrary, one should note that the actual nodal delay bound and jitter bound provided by the MGFQ scheduler are $\left\lceil \frac{ND^h}{T} \right\rceil T$ and $\left\lceil \frac{J^h}{T} \right\rceil T$, respectively.

Because the approaches of carrying due-dates for voice traffic and video traffic are different, the notations and the procedure of updating due-date for voice cells and video cells must be presented separately. Notice that the calculations are based on the modified cell format introduced and the operation algorithm mentioned in section 4.2.

4.3.1 Voice Traffic Streams

Additional definitions of notations to calculate the due-date information for voice traffic streams are as follows.

- P_k^A: the k-th voice cell of VP_i, where the superscript "A" stands for "audio;"
- $AT_k^{A,h}$: the arrival time of voice cell P_k^A at node h;
- $LTT_k^{A,h}$: the latest transmission time of voice cell P_k^A at node h;
- $ET_k^{A,h}$: the eligible time of voice cell P_k^A at node h;
- $DT_k^{A,h}$: the departure time of voice cell P_k^A at node h;
- $J_k^{A,h}$: the cell delay jitter of voice cell P_k^A at node h;
- $IND_k^{A,h}$: the initial nodal due-date of voice cell P_k^A at node h;
- $DD_k^{A,h}$: the due-date of voice cell P_k^A when it leaves node h.

In the above notations, the latest transmission time at a node means the latest time epoch (or so-called deadline) at which a cell transmission still does not violate its nodal delay bound. Therefore, $LTT_k^{A,h}$ can be obtained via

$$\begin{aligned} LTT_k^{A,1} &= AT_k^{A,1} + ND^1, \\ LTT_k^{A,h} &= AT_k^{A,1} + \sum_{j=1}^{h} ND^j = LTT_k^{A,h-1} + ND^h, h > 1. \end{aligned} \tag{4.1}$$

In other words, $LTT_k^{A,h}$ can be calculated recursively.

If the voice cell P_k^A arrives at node h at time $AT_k^{A,h}$, then the initial nodal due-date, $IND_k^{A,h}$, is designated as

$$IND_k^{A,h} = LTT_k^{A,h} - AT_k^{A,h}. \tag{4.2}$$

We also know that the due-date of a cell at its departure time is the difference between the latest transmission time and the departure time. Therefore, when the cell P_k^A departs for node $h+1$, its due-date is calculated via

$$DD_k^{A,h} = LTT_k^{A,h} - DT_k^{A,h}. \tag{4.3}$$

Combining eq. (4.1), eq. (4.2) and adopting the property $AT_k^{A,h} = DT_k^{A,h-1} + LD^{h-1,h}$, we can derive

$$IND_k^{A,h} = ND^h + LTT_k^{A,h-1} - DT_k^{A,h-1} - LD^{h-1,h}. \tag{4.4}$$

Hence, according to eq. (4.3) we derive the recursive formulas for the initial nodal due-date as

$$IND_k^{A,1} = ND^1, \tag{4.5}$$

$$IND_k^{A,h} = DD_k^{A,h-1} + ND^h - LD^{h-1,h}, h > 1. \tag{4.6}$$

Following to the mathematically definition of delay jitter in [19], the cell delay jitter $J_k^{A,h}$ is given by

$$J_k^{A,h} = \left| \left(DT_k^{A,h} - AT_k^{A,1} \right) - \left(DT_{k-1}^{A,h} - AT_{k-1}^{A,1} \right) \right|. \tag{4.7}$$

Now, it is easy to derive from eq. (4.1), (4.3) and (4.7) that $J_k^{A,h}$ can also be calculated using

$$J_k^{A,h} = \left| DD_k^{A,h} - DD_{k-1}^{A,h} \right|. \tag{4.8}$$

In other words, the notion of jitter described in Sec. 4.2.1 is actually consistent with the definition in eq. (4.7).

As we know, the eligible time of a cell is the time when that cell can be transmitted immediately without jitter bound violation. In the following, we show that

$J_k^{A,h}$ is bounded by

$$-J^h \leq J_k^{A,h} \leq J^h \tag{4.9}$$

if we set the eligible time, $ET_k^{A,h}$, as

$$\begin{aligned} ET_k^{A,h} &= LTT_k^{A,h} - J^h \\ &= AT_k^{A,h} + IND_k^{A,h} - J^h, h \geq 1. \end{aligned} \tag{4.10}$$

By combining eq. (4.3) and (4.8), one can write

$$J_k^{A,h} = \left| LTT_k^{A,h} - LTT_{k-1}^{A,h} - \left(DT_k^{A,h} - DT_{k-1}^{A,h} \right) \right|. \tag{4.11}$$

Because $DT_k^{A,h}$ has to satisfy the relationship $ET_k^{A,h} \leq DT_k^{A,h} \leq LTT_k^{A,h}$, we can argue that the difference between $DD_k^{A,h}$ and $DD_{k-1}^{A,h}$ satisfies

$$ET_{k-1}^{A,h} - LTT_{k-1}^{A,h} \leq DD_k^{A,h} - DD_{k-1}^{A,h} \leq LTT_k^{A,h} - ET_k^{A,h}, \tag{4.12}$$

which is equivalent to eq. (4.9).

Therefore, when cell P_k^A is allowed to depart for the node $h+1$, it violates neither the delay bound nor the jitter bound of node h in the most strict sense. When P_k^A departs from node h, its due-date $DD_k^{A,h}$ updated via eq. (4.3) also carries the necessary due-date information.

By summarizing the whole due-date calculation procedure, the computation involves only 5 additions per cell. Since the due-date of a cell does not have to be updated slot-by-slot and only has to be updated when that cell leaves a node, such computation complexity should not be difficult to handle.

In the following, we use an example as shown in Fig. 4.6 to illustrate the operations of MGFQ algorithm when a voice cell is processed. We consider a VP whose assigned nodal delay bound is 12 slots and jitter bound is 9 slots. The length of the refreshing-period, T, is assumed as 3 time slots. Then, this VP is assigned to Group

3. We assume voice cell p arrives at time $t = 0$, and then its latest transmission time is set to be 12 based on eq. (4.1) and its eligible time is 3 according to eq. (4.10). Suppose the "gate" opens at time $t = 2$, 5, 8, 11, and 14 as shown in Fig. 4.6. When the "gate" opens at time $t = 5$, p is marked as eligible cell because its due-date is within $[7, 9]$. In the same way, p is moved to temporary-queue 2 and 1 at time $t = 8$ and 11, respectively. If cell p is still not transmitted until time 14, then p will be discarded because of its overdue. On the other hand, if p can be transmitted within the time interval $[5, 14]$, the jitter of p will conform to eq. (4.9).

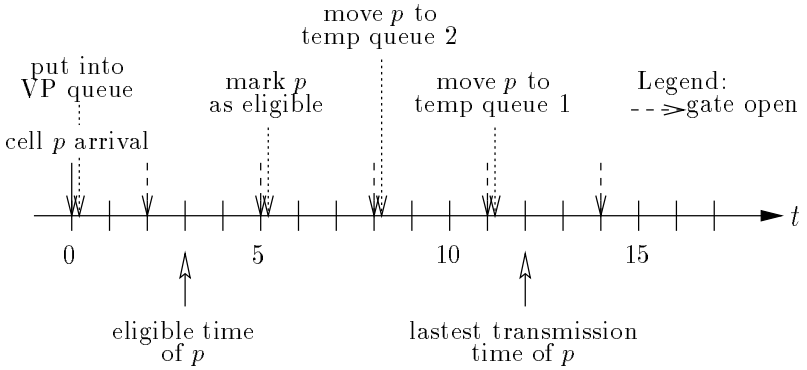


Figure 4.6: The example of the operations of MGFQ algorithm.

4.3.2 Video Traffic Streams

Additional notations to calculate the due-date for video traffic streams are as follows.

- X^V: the peak cell rate of the considered VC, VC_j of VP_i, of the video traffic, where the superscript "V" stands for "video;"
- $P^V_{k,l}$: the l-th cell of k-th video frame of the considered VC, VC_j of VP_i;
- $AT^{V,h}_{k,l}$: the arrival time of video cell $P^V_{k,l}$ at node h;

- $LTT_{k,l}^{V,h}$: the latest transmission time of video cell $P_{k,l}^{V}$ at node h;
- $ET_{k,l}^{V,h}$: the eligible time of video cell $P_{k,l}^{V}$ at node h;
- $DT_{k,l}^{V,h}$: the departure time of video cell $P_{k,l}^{V}$ at node h;
- $IND_{k,l}^{V,h}$: the initial nodal due-date of video cell $P_{k,l}^{V}$ at node h;
- $DD_{k,l}^{V,h}$: the due-date of video cell $P_{k,l}^{V}$ when it departs from node h.

Suppose the video cell $P_{k,l}^{V}$ arrives at node h at time $AT_{k,l}^{V,h}$. Therefore, the initial nodal due-date calculation formulas for the video cells are as

$$IND_{k,1}^{V,1} = ND^{1}, \tag{4.13}$$

$$IND_{k,l}^{V,h} = DD_{k,l}^{V,h-1} + ND^{V,h} - LD^{h-1,h}, h > 1. \tag{4.14}$$

For BOM (Begin-Of-Message) cells (i.e., $l = 1$), their due-dates is extracted from the *DD* field. Otherwise, for the cells other than BOM cells, their initial nodal due-dates are derived via the following procedure:

We know $DD_{k,l}^{V,h-1}$ $(l > 1)$ can be obtained from

$$\begin{aligned} DD_{k,l}^{V,h-1} &= LTT_{k,l}^{V,h-1} - DT_{k,l}^{V,h-1}, \\ &= LTT_{k,l}^{V,h-1} - (AT_{k,l}^{V,h} - LD^{h-1,h}). \end{aligned} \tag{4.15}$$

Because the delay and jitter distributions for cells in a virtual path are all the same, the relationship

$$LTT_{k,l}^{V,h-1} - LTT_{k,1}^{V,h-1} = AT_{k,l}^{V,h-1} - AT_{k,1}^{V,h-1} = \left\lceil \frac{l-1}{X^{V}} \right\rceil \tag{4.16}$$

should hold. Then, from eq. (4.14) and eq. (4.15), one can derive

$$\begin{aligned} IND_{k,1}^{V,h} &= DD_{k,1}^{V,h-1} + ND^{V,h} - LD^{h-1,h}, \\ &= LTT_{k,1}^{V,h-1} - (DT_{k,1}^{V,h-1} + LD^{h-1,h}) + ND^{V,h}, \\ &= LTT_{k,1}^{V,h-1} - AT_{k,1}^{V,h} + ND^{V,h}. \end{aligned} \tag{4.17}$$

Combining eq. (4.14), (4.15), (4.16) and (4.17), we obtain $IND_{k,l}^{V,h}$ as

$$IND_{k,l}^{V,h} = IND_{k,1}^{V,h} + \left(AT_{k,1}^{V,h} + \left\lceil \frac{l-1}{X^V} \right\rceil - AT_{k,l}^{V,h} \right). \tag{4.18}$$

Hence, we obtain a recursive formula for computing the initial nodal due-dates of video cells:

$$IND_{k,1}^{V,h} = DD_{k,1}^{V,h-1} + ND^{V,h} - LD^{h-1,h}, h > 1, \tag{4.19}$$

$$IND_{k,l}^{V,h} = IND_{k,1}^{V,h} + \left(AT_{k,1}^{V,h} + \left\lceil \frac{l-1}{X^V} \right\rceil - AT_{k,l}^{V,h} \right), l > 1, h > 1, \tag{4.20}$$

with initial condition listed in eq. (4.13). With eq. (4.20), the corresponding $DD_{k,l}^{V,h-1}$ in eq. (4.14) is simply $IND_{k,l}^{V,h} - ND^{V,h}$.

Similar to the voice connections, the eligible time $ET_{k,l}^{V,h}$ for video cells is calculated via

$$ET_{k,l}^{V,h} = AT_{k,l}^{V,h} + IND_{k,l}^{V,h} - J^h, \qquad l \geq 1, h \geq 1. \tag{4.21}$$

Different from voice traffic, we only have to update the due-date of the BOM cell of every video frame. Suppose the first cell of a video frame k departs from node h at time $DT_{k,1}^{V,h}$, then its due-date $DD_{k,1}^{V,h}$ is updated via

$$DD_{k,1}^{V,h} = IND_{k,1}^{V,h} - \left(DT_{k,1}^{V,h} - AT_{k,1}^{V,h} \right) \qquad h \geq 1. \tag{4.22}$$

The total complexity of the DD calculation involves at most 7 additions and 1 multiplication per cell for video.

4.3.3 Design Issues

Because the propagation delay of a link between two nodes is fixed, the effect of the propagation delay on the due-date calculation procedures can be combined with the effect on the nodal delay. Therefore, in the following discussions, we neglect the propagation delay temporarily.

According to the operations of the MGFQ algorithm, we know that the jitter bound of a VP is constrained by the egress node of the network. Suppose the local jitter bound assigned to VP_i at the egress node H is J^H. Then, the end-to-end transmission delay (CTD) of VP_i is

$$\left(\sum_{h=1}^{H} ND^h\right) - \left\lceil \frac{J^H}{T} \right\rceil T \leq CTD \leq \left(\sum_{h=1}^{H} ND^h\right) + T. \tag{4.23}$$

Therefore, the end-to-end jitter bound (or called Cell Delay Variation (CDV)) is $\left\lceil \frac{J^H}{T} \right\rceil T + T$.

In addition, we do not expect cell losses to have any impact on scheduling performance. For voice traffic, each cell carries the *due-date* information. Therefore, the due-date calculations are mutually independent. For video traffic, the due-date calculations depend on the BOM cell. If the BOM cells can not be distinguished, the due-date calculations will be in error until the next distinguished BOM cell. However, because the PPD mechanism is applied, these cells, whose due-date are in error, will all discarded by the ATM switch. Therefore, scheduling performance should not be affected by the cell loss for video traffic. Hence, MGFQ is robust against cell loss events if PPD is used.

4.4 Discussions on Implementation Complexity for MGFQ

Since maintaining a sorted priority queue often introduces significant processing overhead, much emphasis on QoS scheduler design is put on methods to simplify the task of maintaining a sorted priority queue. However, the implementation complexity will be an important metric to evaluate the worth of the schedulers. Hence, we investigate the implementation complexity for MGFQ and make a comparison

with other scheduling algorithms, such as JEDD [21], DFQ [67], RPQ [68], RPQ$^+$ [74], etc., in this section.

As is well known, the algorithmic complexity for maintaining a sorted priority queue with N arbitrary entries is $O(\log N)$ in the worst case. The cost to maintaining the sorted queue usually is due to the queue insertion operation upon each cell arrival. Alternatively, a "winner selection" procedure to select the cell with the shortest due-date in an unsorted queue can be applied. Via either the "winner selection" or "queue insertion" operation to select the cell with minimum due-date in JEDD, the implementation cost is still not easy to reduce, especially in the large-scale switches. Meanwhile, schedulers designed to operate with lower complexity have proposed. For example, Liebeherr and Wrege have proposed an approach that attempts to approximate a sorted priority queue at an output-buffered switch with significant complexity reduction [74]. In the following, we provide a comparison of the implementation complexity on those schemes employing the technique of approximating a sorted priority queue, such as MGFQ, DFQ, RPQ and RPQ$^+$ [74].

Given the due-date supported by RPQ is in the range $[0, NT]$, RPQ employs extra N FIFO queues with one pointer-operation to achieve the sorting operation. Nevertheless, RPQ will cause the problem of *rotation anomaly* [74]. To solve the rotation anomaly, RPQ$^+$ employs extra N FIFO queues and increase the number of FIFO queues to $2N$. In addition, extra N pointer-operations are needed to concatenate the cells in FIFO i^+ into FIFO i [74]. An alternative approach, called DFQ [67], also adopts the concept of RPQ to achieve low-complexity traffic scheduler with delay/jitter guarantees. However, as mentioned in section 4.1, the number of the FIFO-queues set is increased linearly proportional to the supported jitter levels. In other words, there is a trade-off between the supported jitter levels and implementation complexity for the DFQ scheme.

In contrast, MGFQ needs N FIFO queues (called *temporary queues* in this chapter) to accommodate the delay bound in the range $[0, NT]$. For each refreshing-period, MGFQ needs N pointer-operations to move cells from temporary queue i to temporary queue $i-1$. However, these pointer-operations also can be achieved via rotating the FIFO queues. Therefore, the implementation complexity of MGFQ is higher than RPQ while it is lower than RPQ$^+$. Afterward, we will show MGFQ can be combined easily with advanced buffer management schemes, such as APPD and PPD via simulations. Hence, the packet level QoS in terms of packet loss ratio can be improved at the same time via employing MGFQ algorithm. In Table 4.1, we summarize the implementation complexity in terms of the number of FIFO queues employed in different schedulers and the scheduling complexity in terms of the number of pointer operations[2].

Scheduler	DFQ	RPQ	RPQ$^+$	MGFQ
No. of FIFO Queues	NJ	N	$2N$	N
No. of Pointer Operations	J	1	$N+1$	$\leq N$

Table 4.1: Comparison of implementation complexity among various scheduling algorithms, where J is the number of delay jitter levels provided by the scheduler.

[2]Note that the number of respective VP queues and the number of pointer-operations between VP queues and scheduler FIFO queues are not included in the table.

4.5 Simulation of Voice Traffic Streams

Next, we evaluate the performance of MGFQ scheme for voice traffic streams. The examined QoS parameters include cell delay, cell jitter distribution and cell discarding ratio. Here, the cell discarding ratio only accounts for those cells discarded due to delay or jitter violations. We employ FCFS and JEDD as baseline comparisons. The assumed queueing model of JEDD is shown in Fig. 4.7. The arriving cells of each virtual path are buffered in the corresponding regulators until they do not violate their jitter bounds, i.e., when they become eligible. Then, eligible cells are moved to the output buffer. The scheduling algorithm of the output buffer for JEDD algorithm is Earliest Due-Date (EDD). Here, we also assume the instantaneous movements of cells from the input regulators to the output buffer. Therefore, the JEDD algorithm adopted in this chapter is considered as ideal cases for illustrating baseline performance.

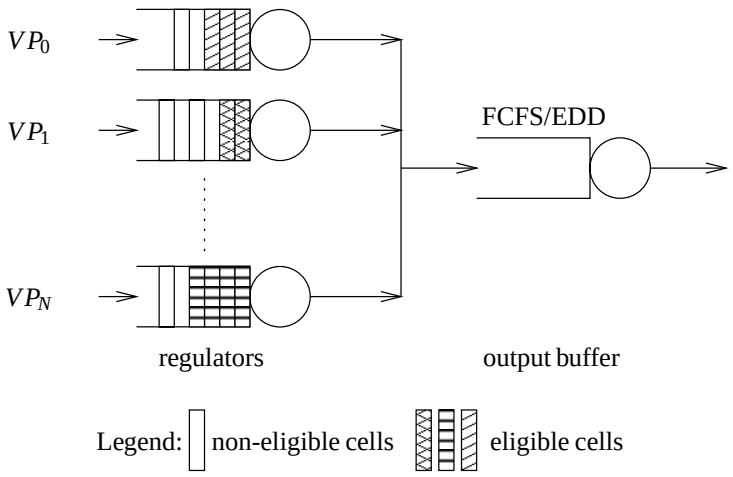


Figure 4.7: Queueing model of JEDD algorithm in the simulation experiments.

The simulation model shown in Fig. 4.8 consists of a three-node network and

1500 voice streams in each node. All voice streams are assumed to follow ITU-T G.764 voice packetization recommendation [75] and the silence suppression mechanism is implemented. Therefore, the voice stream can be modeled as an ON-OFF traffic source.

Suppose the link bandwidth is 45 Mbps and VP_0 consists of 300 virtual connections (VCs). These VCs are assigned nodal delays of 6.0 ms, 6.0 ms, and 0.5 ms at nodes one, two, and three, respectively. VP_1, VP_2 and VP_3 serve as competing cross traffic and each of them contains 1200 VCs. The nodal delay assigned to cross traffic are all 6ms. The ON/OFF duration of all these voice connections are with exponential distribution with mean 1.5 and 2.25 seconds, respectively. While ON, a voice source transmits one cell every 703 slot times, which is sufficient to support a 64 Kbps stream with silence suppression and necessary compression. Therefore, the average bottleneck link utilization is about 0.85. In order to avoid man-made simultaneous arrivals of cell bursts at the multiplexer, the starting epoch of each voice source is uniformly distributed over the 3.75 sec interval. The refreshing-period is set to be 0.5 ms and all simulations last for 10^8 time slots.

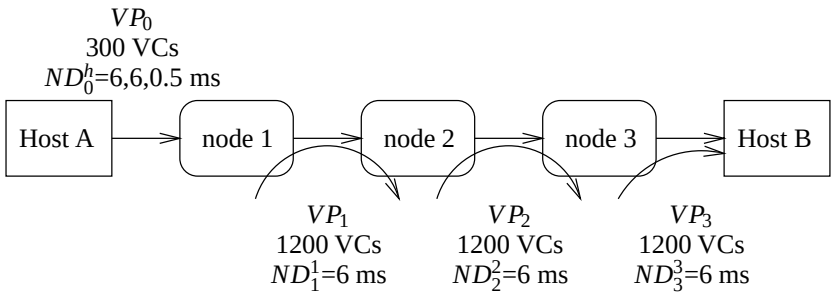


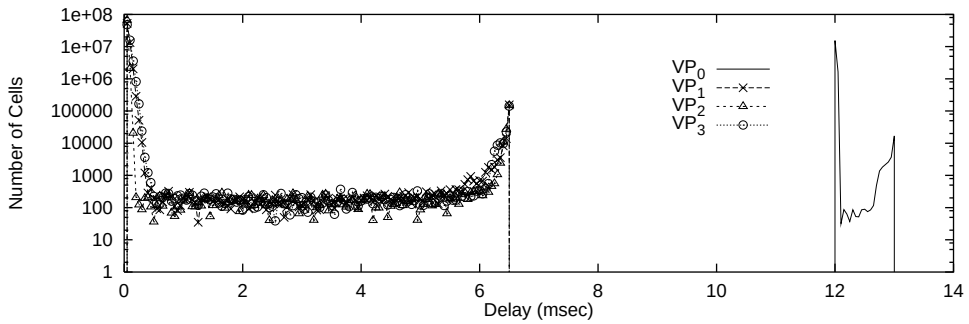
Figure 4.8: Simulation model of MGFQ network for voice traffic.

A. Cell Delay Distribution

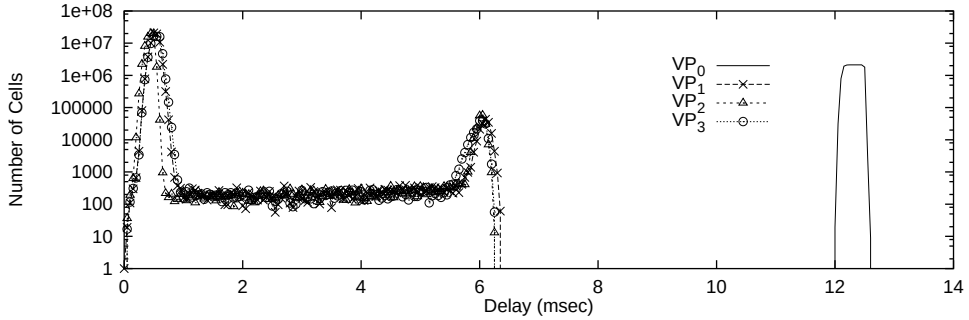
According to eq. (4.23), it can be estimated that the guaranteed upper bounds on queueing delay are 13 ms for VP_0, and 6.5 ms for VP_1 to VP_3. Two classes of guaranteed jitter bound are simulated for VP_0: a tight bound of 1 ms and a loose bound of 13 ms. We assume only a loose jitter bound of 6.5 ms is guaranteed for the cross traffic. Note that our MGFQ algorithm requires only one set of FIFO queues in node 3 to provide 2 classes of jitter bounds, while two sets of FIFO queues are needed in DFQ [67]. Figures 4.9 (a) and (b) show the delay distributions of various VPs under the JEDD and the MGFQ algorithms, where the jitter bound of VP_0 is 1 ms. On the other hand, Figs. 4.10 (a) and (b) show the delay distributions of various VPs under different scheduling algorithms, where the jitter bound of VP_0 is 13 ms. The cell delay distribution under FCFS discipline, i.e., without any control mechanisms and regulators, is shown in Fig. 4.10 (c) for the baseline comparison. All switch nodes in this baseline system perform nothing except forwarding the cells. The cells with delay bound violations are discarded only by the receiver. The cell delay distributions of all VPs spread over a wide range, and a large shadow part in Fig. 4.10 (c) represents the cells of VP_0 with delay beyond 13 ms. We can observe that if no control mechanism is adopted, the cell delay distributions of all VPs are beyond control and a large portion of cells violate their delay constraints. Hence, in the following simulation studies for voice traffic, statistics of the FCFS case are not included.

B. Cell Delay Metrics

Subsequently, in Table 4.2 we illustrate the cell delay performances of different scheduling algorithms. First, we observe that the mean queueing delay for the cross traffic is less than 0.5 ms, and is also less than a frame period in the DFQ algorithm.

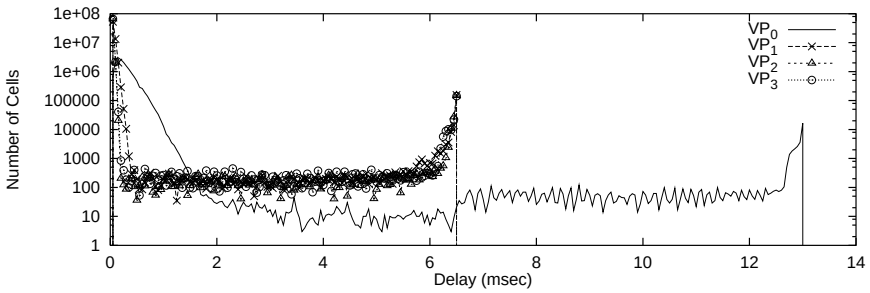


(a) The JEDD Algorithm

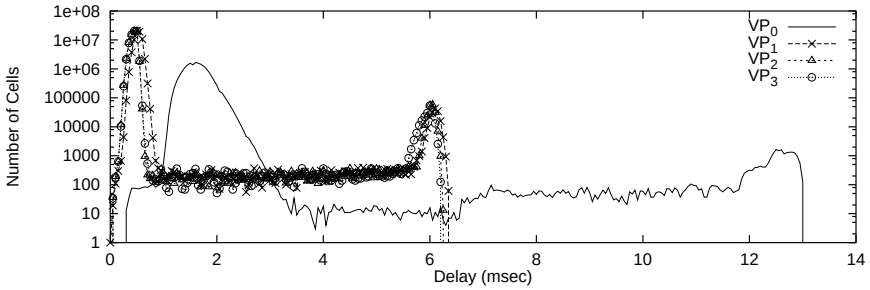


(b) The MGFQ Algorithm

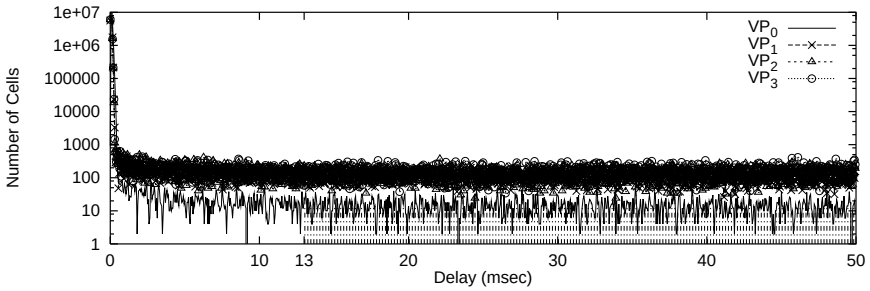
Figure 4.9: The cell delay distributions of JEDD and MGFQ for voice traffic. The jitter bound of VP_0 is 1 ms.



(a) The JEDD Algorithm



(b) The MGFQ Algorithm



(c) FCFS without any control mechanism

Figure 4.10: The cell delay distributions for voice traffic. The jitter bound of VP_0 is 13 ms. The shadow part of (c) represents the discarded cells of VP_0 due to violations of delay constraints.

Thus, the transmission of multiple RM cells during a single frame period is required to improve the performance if DFQ is employed [67]. But note that this operation will increase the overhead of network. Secondly, the mean delay of the cross traffic in the MGFQ algorithm is larger than that in the JEDD scheme. This is because the MGFQ algorithm is a sub-optimal scheduling algorithm. In the MGFQ algorithm, the eligible cells belonging to the same temporary queue have the same priority, regardless of the order of their eligible times and departure times, while the JEDD algorithm schedules the eligible cells according their departure times. Hence, the coarse granularity in the service order of the MGFQ algorithm increases mean delay slightly. However, the maximum delay and maximum jitter experienced by all VPs still conforms the delay and jitter bounds.

C. Cell Overdue Ratio

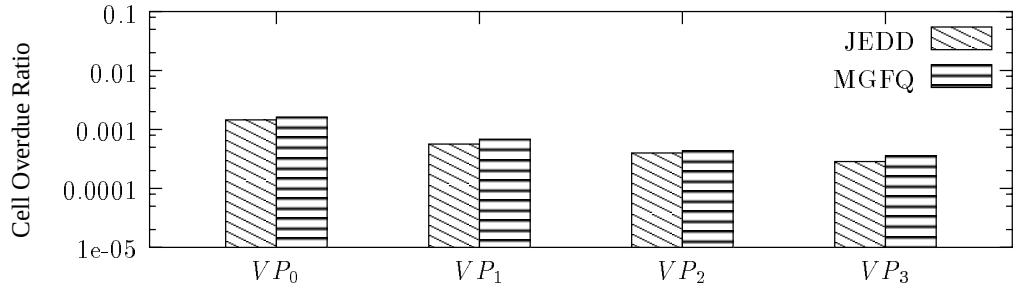
Figure 4.11 shows the cell overdue ratios of JEDD and MGFQ algorithms under two different jitter constraints. It can be observed that the cell overdue ratio under our MGFQ algorithm for VP_0 can achieve a performance level close to that of the JEDD algorithm. For handling cross traffic, though the cell overdue ratio of the MGFQ algorithm is slightly higher than that of the JEDD algorithm, we have to note that the implementation complexity of the MGFQ algorithm is much lower than JEDD.

D. The Impact of Congestion

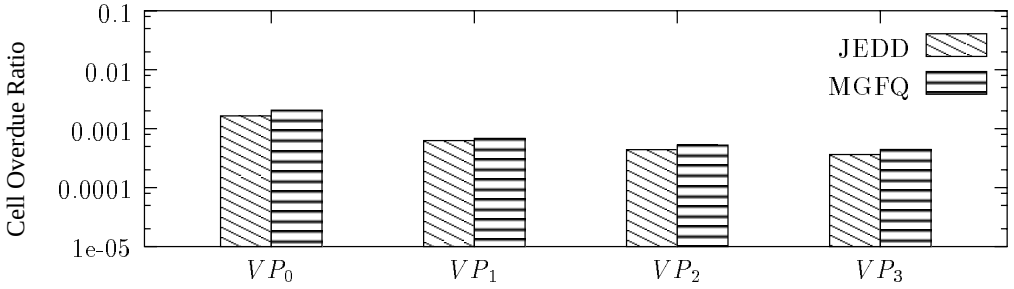
This simulation scenario illustrates the impact on the cell loss ratio among all connections during congestion periods. The simulation configuration consists of only one switching node and two VPs, VP_1 and VP_2. VP_1 contains 1100 VCs while the number of VCs in VP_2 is increased from 100 to 1100. Hence, the total traffic load

	Virtual Path	Scheduling Algorithm	Max. Delay (msec)	Min. Delay (msec)	Avg. Delay (msec)	Jitter (msec)
No Control	VP_0	FCFS	1091.26	0.03	11.47	12.92
	VP_1		499.07	0.03	4.91	6.40
	VP_2		350.69	0.03	3.33	6.44
	VP_3		290.29	0.03	2.90	6.49
Tight Jitter Bound (VP_0)	VP_0	JEDD	13.00	12.00	12.01	1.00
		MGFQ	13.00	12.00	12.29	0.97
	VP_1	JEDD	6.50	0.03	0.05	6.47
		MGFQ	6.46	0.03	0.51	6.41
	VP_2	JEDD	6.50	0.03	0.04	6.47
		MGFQ	6.50	0.03	0.43	6.44
	VP_3	JEDD	6.50	0.03	0.06	6.47
		MGFQ	6.48	0.03	0.53	6.45
Loose Jitter Bound (VP_0)	VP_0	JEDD	13.00	12.00	0.26	12.87
		MGFQ	13.00	12.00	1.61	12.98
	VP_1	JEDD	6.50	0.03	0.05	6.47
		MGFQ	6.46	0.03	0.51	6.41
	VP_2	JEDD	6.50	0.03	0.04	6.47
		MGFQ	6.50	0.03	0.43	6.44
	VP_3	JEDD	6.50	0.03	0.04	6.47
		MGFQ	6.48	0.03	0.43	6.45

Table 4.2: Cell delay metrics of multiple delay/jitter bounds for various scheduling disciplines. For tight jitter control, jitter bound of VP_0 is 1 ms; while for loose jitter control, jitter bound of VP_0 is 13 ms.



(a) Tight Jitter Control (VP_0)



(b) Loose Jitter Control (VP_0)

Figure 4.11: Cell overdue ratios of multiple delay/jitter bounds for JEDD and MGFQ. For tight jitter control, jitter bound of VP_0 is 1 ms; while for loose jitter control, jitter bound of VP_0 is 13 ms.

is increased from 0.683 to 1.252 when the number of VCs in VP_2 is increased. We assume the delay bounds of two VPs are all equal to 6.5 ms. The jitter bound of VP_1 is set to 1 ms while the jitter bound of VP_2 is 6.5 ms.

From the simulation results shown in Fig. 4.12, we find that the performances of JEDD and MGFQ are very close under all levels of traffic loads. In other words, the CLR achieved by MGFQ is very similar to JEDD, but with much lower complexity.

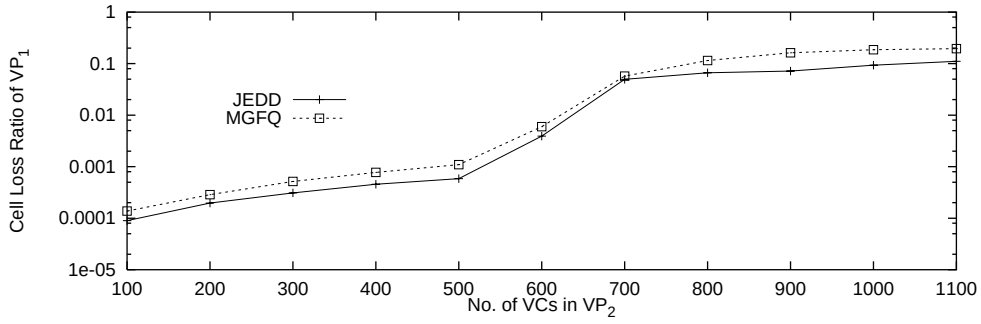


Figure 4.12: Cell loss ratios of VP_1 for JEDD and MGFQ under various traffic loads.

4.6 Simulation of Video Traffic Streams

In this simulation scenario, video traces are applied to investigate the performance of MGFQ algorithm supporting real-time MPEG video over ATM. In this simulation, not only the cell level performance is shown, but also the frame level QoS, such as frame delay distribution and frame discarding ratio are presented. Notice that any cell is discarded while it violates the delay constraint and that a video frame is discarded if any cell of the frame is discarded.

The video traffic simulation model, which is similar to the model illustrated in Fig.4.8, is shown in Fig. 4.13. A 45 Mbps link bandwidth is still assumed. The target

virtual path, VP_0, consists of 10 VCs. These VCs are also assigned nodal delays of 6.0 ms, 6.0 ms, and 0.5 ms at nodes i (where $i = 1 \sim 3$) respectively. VP_1 to VP_3 serve as competing cross traffic and each of them contains 45 VCs. The nodal delays assigned to cross traffic are all 6 ms. Each VC carries a video stream and each video stream is a replay of "James Bond: Gold finger" MPEG-1 video trace obtained from University Wuerzburg [76], with equally separated starting points within the 39996 frame positions. Since the frame rate is 24 frames/sec, each stream is equivalent to a video of the length 1666.5 seconds. Again, in order to avoid simultaneous arrivals of cell bursts at the multiplexer at the beginning, the starting epoch of each cell stream is uniformly distributed over the 1 sec interval. All simulations last for 10^8 cell slot periods. Related statistical information of the video trace is listed in Table 4.3. When a VC has a video frame to send, it uses peak rate to transmit the cell burst of the video frame. In this simulation experiment, we assume the peak rate of each VC is 15 Mbps. The average bottleneck link utilization is 0.72 in this simulation scenario.

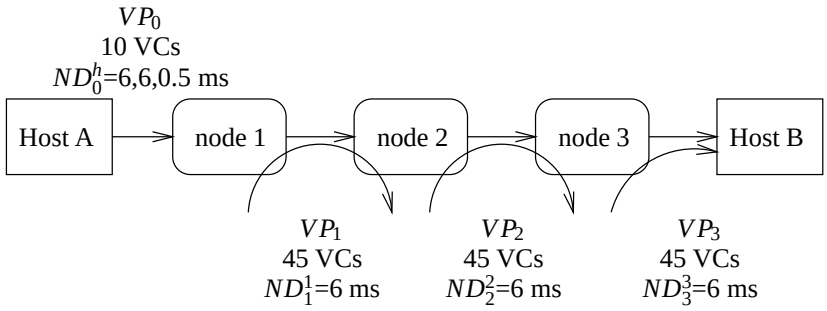


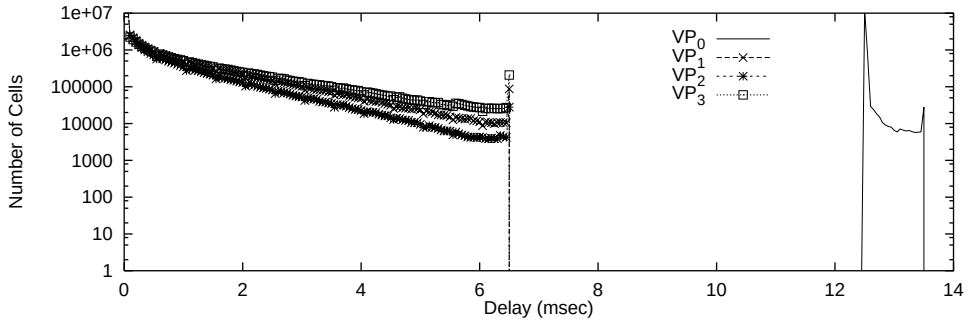
Figure 4.13: Simulation model of the MGFQ network for video traffic.

A. Cell Delay Distribution and Frame Delay Distribution

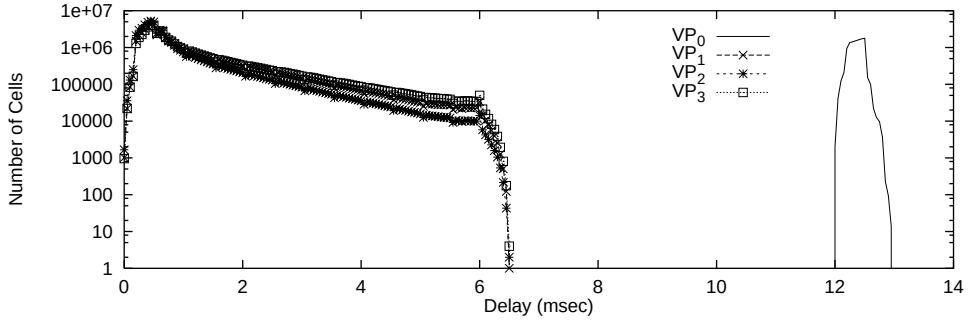
	I-Frame	P-Frame	B-Frame
Mean frame length (cells)	217.401	108.376	27.868
Variance of frame length (cell^2)	4609.706	2757.451	119.330
Maximum frame length (cells)	637	543	145
Overall mean rate	587.881 Kbps		

Table 4.3: General information of the MPEG video trace in simulations.

Figures 4.14 (a), (b) and Figs. 4.16 (a), (b) show the cell delay distributions and frame delay distributions of the JEDD and the MGFQ algorithms for video traffic under two different jitter constraints, respectively. The cell delay distribution and frame delay distribution under FCFS discipline, i.e., without any control mechanism and regulators, are also shown in Fig. 4.15 (c) and Fig. 4.17 (c) for the baseline comparison. Again, all switch nodes in this baseline system perform nothing except forwarding the cells. The receiver of host B is responsible for discarding the cells with delay constraint violations. Again the cell distributions of all VPs spread over a wide range. The ratio of cells of VP_0 with delay beyond 13 ms, indicated by the shadow in Fig. 4.15 (c), is significant. In a precise fashion, the frame delay is defined as the time interval between the time when the first cell of the frame is transmitted and the time when the last cell is received by the receiver. According to this definition, it is not trivial to control the delay jitter at the frame level. Nevertheless, we find that if the cell delay jitter is under control, then the frame delay jitter becomes small, as illustrated by the simulation results. Therefore, it is possible for the receiver to allocate smaller buffer to compensate the frame delay jitter if MGFQ traffic scheduler is implemented in the network. In turn, the receiver must allocate very large buffer to compensate the disturbed cell arrivals under FCFS.



(a) The JEDD Algorithm

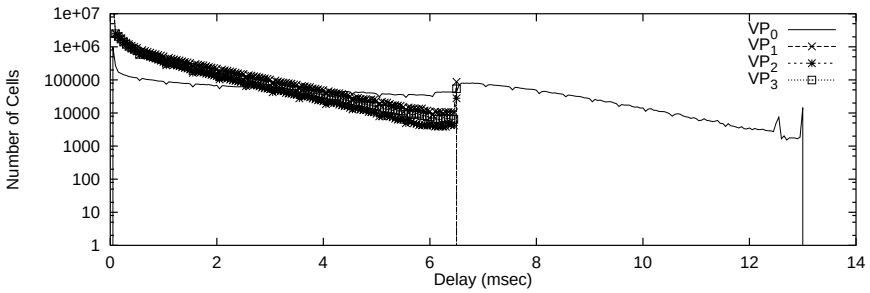


(b) The MGFQ Algorithm

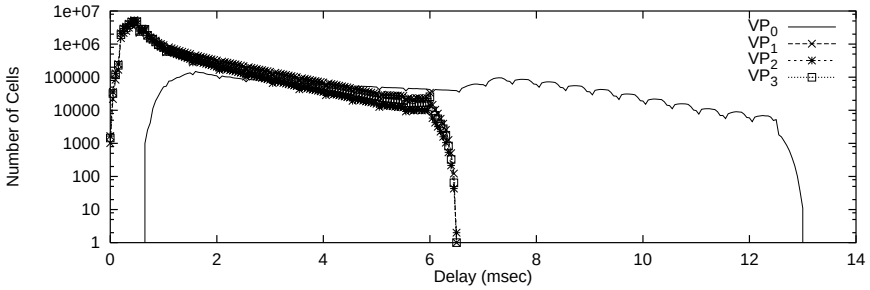
Figure 4.14: The cell delay distributions of JEDD and MGFQ for video traffic. The jitter bound of VP_0 is 1 ms.

B. Frame Discarding Ratio

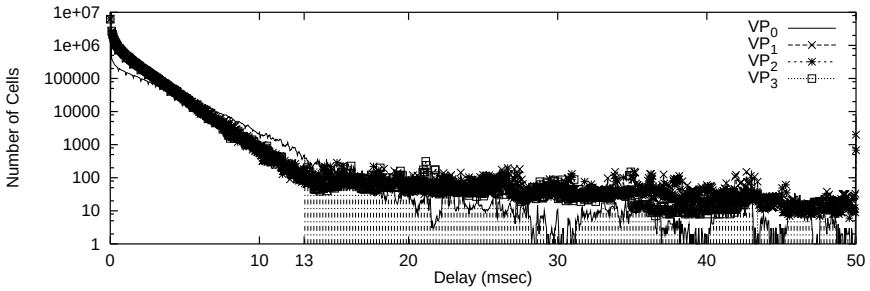
Figures 4.18 and 4.19 show the frame discarding ratios of each frame type of 4 VPs under JEDD, MGFQ, and MGFQ combined with APPD and PPD, which



(a) The JEDD Algorithm

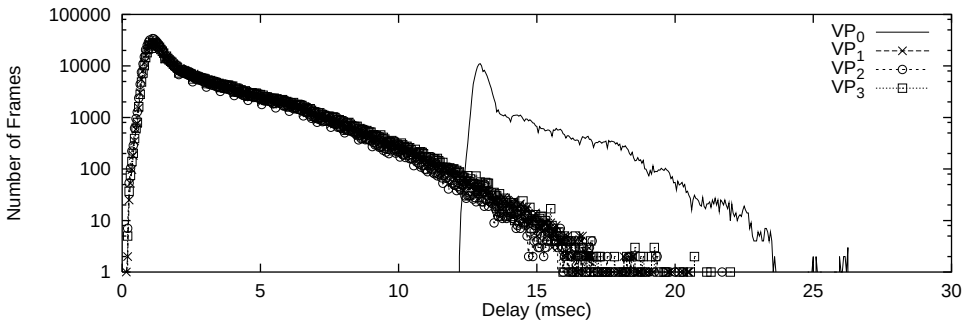


(b) The MGFQ Algorithm

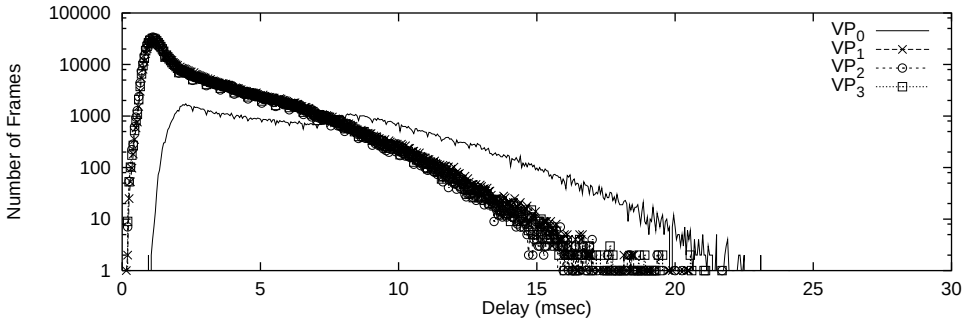


(c) FCFS without any control mechanism

Figure 4.15: The cell delay distributions for video traffic. The jitter bound of VP_0 is 13 ms. The shadow part of (c) represents the discarded cells of VP_0 due to violations of delay constraints.

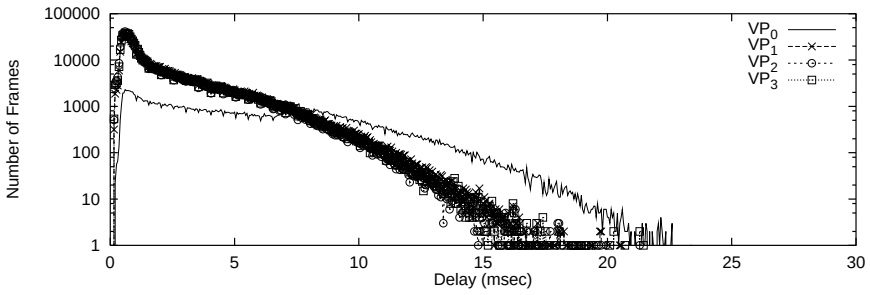


(a) The JEDD Algorithm

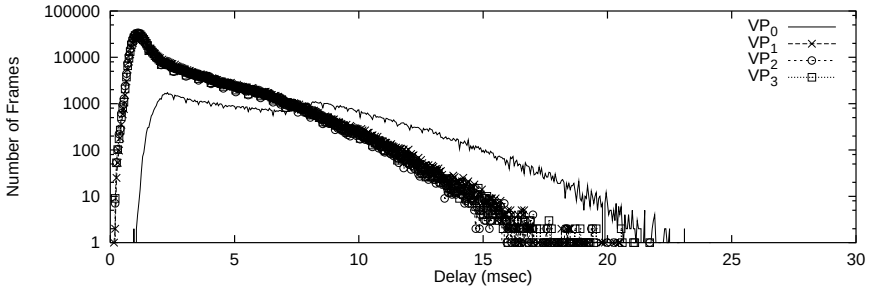


(b) The MGFQ Algorithm

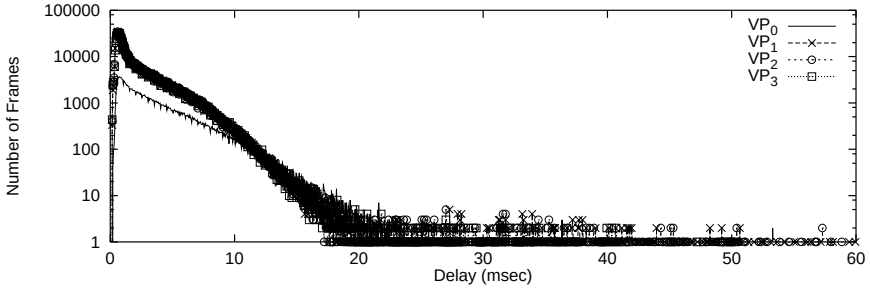
Figure 4.16: The frame delay distributions for video traffic simulation, where the cell delay jitter bound of VP_0 is 1 ms.



(a) The JEDD Algorithm



(b) The MGFQ Algorithm



(c) FCFS without any control mechanism

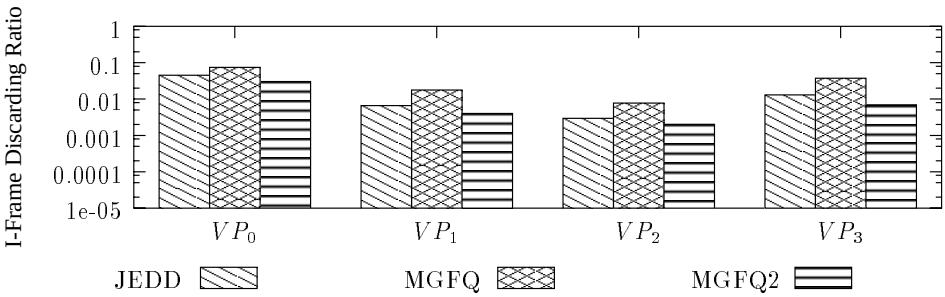
Figure 4.17: The frame delay distributions under difference scheduling algorithms for video traffic simulation, where the cell delay jitter bound of VP_0 is 13 ms.

is denoted as MGFQ2 in the figures. Although JEDD algorithm has the smallest frame discarding ratio for VP_0, the implementation complexity is the major cost. The reason that frame discarding ratio of I-frame is higher than B-frame or P-frame under JEDD and pure MGFQ algorithm, as observed from the simulation results, is due to large cell bursts of I-frames. Meanwhile, if the MGFQ algorithm is combined with APPD and PPD scheme, there are significant improvements in terms of fairness among different types of frames. Although the frame discarding ratios of P-frame and B-frame are higher than other schemes, when APPD and PPD schemes are adopted, the frame playback performance is not expected to degrade seriously since the layering codec technique is used. Therefore, we believe the MGFQ algorithm should be an excellent candidate to be used with advanced buffer management schemes. In contrast, this feature has not been well investigated in other scheduling algorithms.

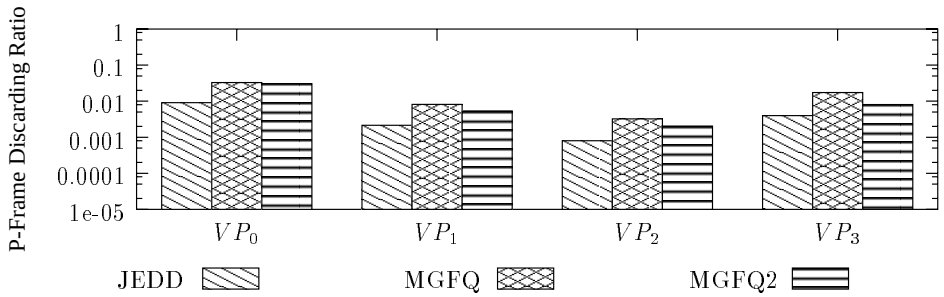
C. The Impact of Congestion

This simulation scenario describes the impact on the frame loss ratios under heavy loaded conditions. Similar to the simulation experiment of voice traffic, only one switching node and two VPs, VP_1 and VP_2, are included in the simulation configuration. VP_1 contains 40 video VCs while the number of VCs in VP_2 is increased from 5 to 40. Each VC carries a video stream as mentioned in Sec. 4.6. Hence, the total traffic load ranges from 0.588 to 1.045 when the number of VCs in VP_2 is increased. We assume the delay bounds of two VPs are all equal to 6.5 ms. The jitter bound of VP_1 is set to 3.5 ms while the jitter bound of VP_2 is 6.5 ms.

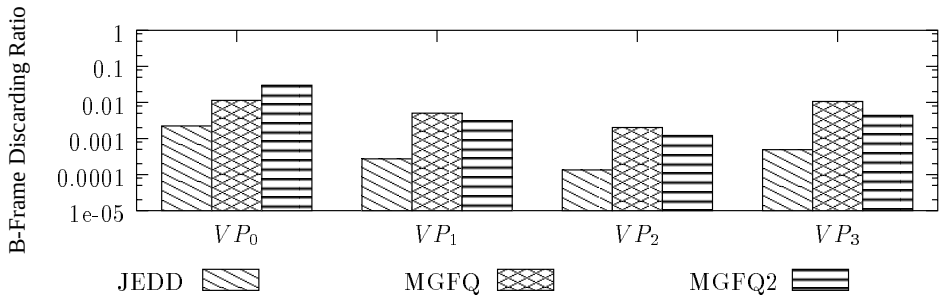
The frame discarding ratios of each frame type for VP_1 under JEDD, MGFQ, and MGFQ combined with APPD and PPD, which is denoted as MGFQ2 in the figures, are shown in Fig. 4.20. When the traffic load increases (see Fig. 4.20), the



(a) I-Frame

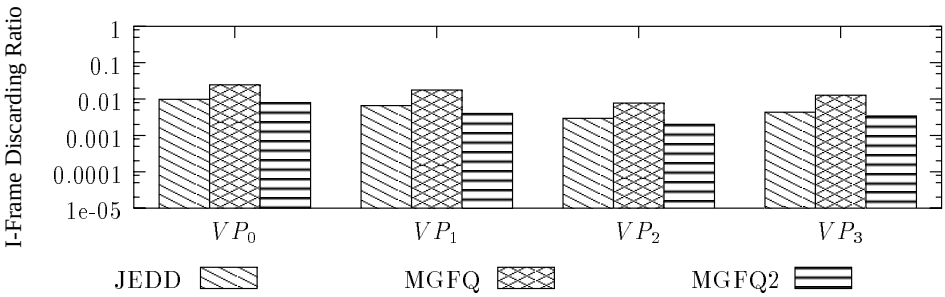


(b) P-Frame

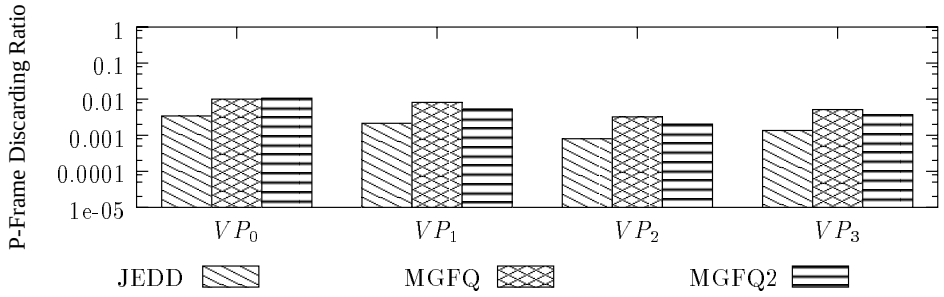


(c) B-Frame

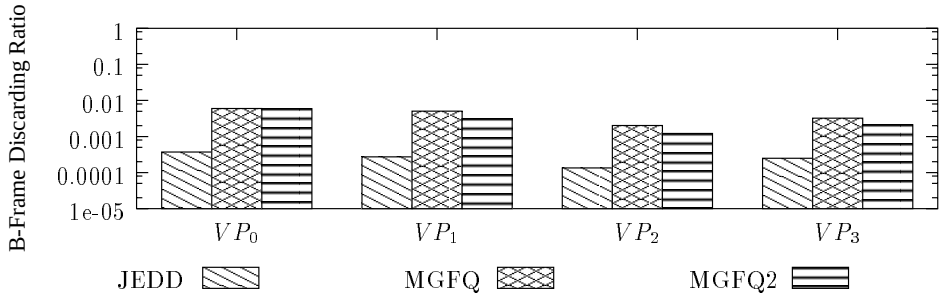
Figure 4.18: Frame discarding ratios of various scheduling algorithms, where MGFQ2 represents the MGFQ algorithm combined with APPD and PPD schemes. The cell delay jitter bound of VP_0 is 1 ms.



(a) I-Frame

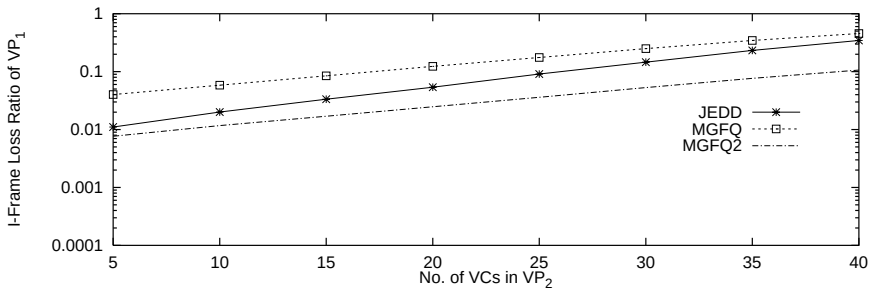


(b) P-Frame

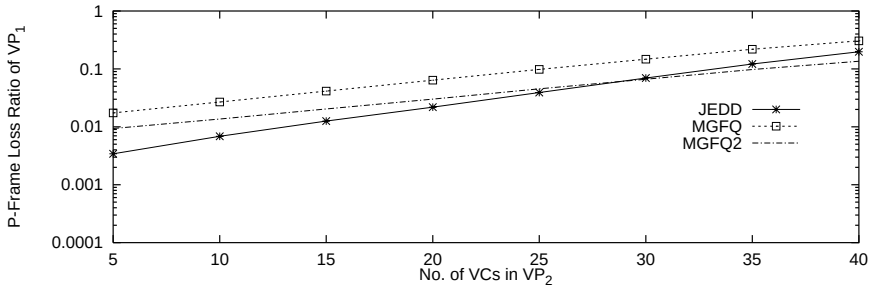


(c) B-Frame

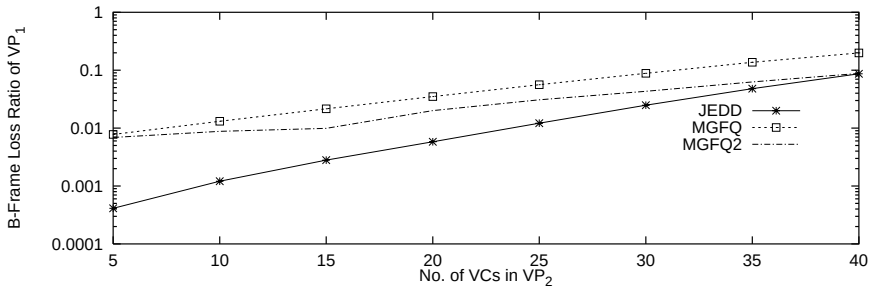
Figure 4.19: Frame discarding ratios of various scheduling algorithms, where MGFQ2 represents the MGFQ algorithm combined with APPD and PPD schemes. The cell delay jitter bound of VP_0 is 13 ms.



(a) I-frame discarding ratio.



(b) P-frame discarding ratio.



(c) B-frame discarding ratio.

Figure 4.20: Frame discarding ratios of VP_1 versus traffic loads under different scheduling disciplines.

I-frame discarding ratio of VP_1 under MGFQ2 is better than JEDD at the expense of increasing the frame discarding ratios of P-frame and B-frame. Since the error in the I-Frame could propagate and influences the quality for a sequence of frames, I-Frame is considered more important. Meanwhile, the loss of B-Frame is expected to have only limited impact. Hence, we believe it does not have much impact on the perceived visual quality. Last but not least, the MGFQ algorithm, which is a suboptimal scheduling discipline compared to JEDD, can still accommodate a good frame level performance even under heavy loaded conditions if it combines with advanced buffer management schemes such as APPD and PPD.

4.7 Concluding Remarks

In this chapter, we first designed two ATM cell formats for carrying timing information in the upstream node to the downstream node along the transmission path for voice and video traffic, respectively. Based on these special cell formats, we presented a framework which includes an efficient scheduling algorithm called MGFQ for transporting real-time traffic over ATM networks with minimum processing and protocol overhead. Unlike previous studies [67], MGFQ employs only one set of FIFO queues to provide a wide range of QoS for real-time applications. Thus, it not only reduces the hardware implementation complexity significantly but also achieves high multiplexing gain. In addition, we had shown it can be combined easily with advanced buffer management schemes, such as APPD and PPD. Hence, both the cell level performance and the packet level QoS can be improved.

From the simulation results, we found that MGFQ can provide much better control of delay and jitter, and yet improve cell/packet discarding ratio. Because MGFQ allows the target traffic stream be granted higher priority than an interfering

traffic stream, it may even accomplish better performance than JEDD by slightly degrading the performance of cross traffic. With the help of APPD and PPD, MGFQ can improve packet level QoS significantly in term of frame discarding ratio for video traffic. Although many people believe it is difficult to efficiently control packet level QoS, such as frame delay jitter, using pure cell level QoS mechanisms, we showed that effective QoS control mechanisms at the cell level (especially jitter) should be able to achieve a commensurate packet level QoS. With MGFQ, a receiver can allocate less resources, such as buffers, to compensate the disturbed cell arrivals.

To summarize, the presented framework, with MGFQ, provides a novel approach to implement real-time multimedia transport and the efficient traffic scheduling with flexible jitter and delay guarantees. Various kinds of customer requirements, such as flexible end-to-end jitter constraints, transmission via AAL1/2/5, and adaptive playout, etc., can be achieved by employing MGFQ-enabled switches. We believe that the MGFQ scheme should be able to support a wide range of other jitter and delay sensitive applications for multimedia communications.

Chapter 5

DMGFQ: A Novel Traffic Scheduler with Differentiated QoS Guarantee for Internet Multimedia Services

5.1 Introduction

In Chapter 3 and Chapter 4, we discuss the approaches providing QoS for RT and NRT traffic over backbone networks. In this chapter, we take the wireline access networks into consideration. In the access networks, RT and NRT traffic streams are aggregated. Thus one should design a universal scheduling platform to accommodate QoS requirements of RT and NRT traffic streams, simultaneously.

For RT traffic, one may not wish to reduce the cell delay bound as small as possible. Instead, network managers can choose to increase the statistical multiplexing gain as long as the delay/jitter constraints at the packet level can be met at the

same time. Hence, conventional scheduling algorithms designed for bandwidth allocation only, such as Weighted Fair Queueing (WFQ) [31], Self-Clocked Fair Queueing (SCFQ) [37], Weighted Round-Robin (WRR) [57], etc., may not satisfy the requirements of NRT traffic and RT streaming data traffic simultaneously. In many cases, the use of these scheduling algorithms implies the upper limit of possible Cell Delay Variation (CDV) only in the form of trivial bounds. As a result, these conventional scheduling algorithms, such as WFQ and its extensions, inherently face the problem of the trading-off between jitter bound and statistical multiplexing gain.

Another approach of designing scheduling algorithms is to support both flexible delay and jitter guarantees, and one typical representative is the jitter-Earliest-Due-Date (JEDD) algorithm [21]. In JEDD, the required information, *due-date*, which is the difference between its local deadline and actual transmitting time, is required to be inserted into a field of packet header. However, the complexity of required "winner selection" or "queue insertion" operation in the JEDD scheduling algorithm should have made it difficult to be realized by a cost-effective hardware implementation. In [67], the delayed frame queueing (DFQ) service discipline adopts the concept of rotating-priority-queue plus (RPQ^+) [74] to achieve the delay and jitter guarantees for RT traffic in ATM networks. Again, a tradeoff exists between the scheduling performance and transmission overhead when the DFQ scheme is employed via the RM cell transmissions. In addition, the number of the FIFO-queue sets in DFQ must be the same as the number of the supported jitter levels. This implementation cost leads to the limitation on the scalability of jitter level for DFQ. The dynamic-R&S scheme proposed in [77] extends the design of Rate Controlled Static Priority (RCSP) [48] and provides substantially better jitter control and higher statistical multiplexing gain. The improvement of dynamic-R&S is achieved by employing regulators, schedulers and a feedback mechanism from the scheduler to regulators

to adjust the packet release from regulators. However, such feedback mechanism increases its implementation complexity. And its design does not support 100% delay and jitter guarantees for streaming data. Although dynamic-R&S can be used effectively as a bandwidth allocation mechanism for traffic streams within traffic contracts, the allocation of residual bandwidth is not taken into considerations.

In turn, the Multi-layer Gated Frame Queueing (MGFQ) algorithm in Chapter 4 employs a set of multiple FIFO queues, each associated with a level of due-date, and accommodates arriving cells into the most appropriate one according to their delay/jitter requirements in a switching node. Here, the due-dates are calculated based on the previous due-dates passed over from their upstream nodes. MGFQ provides a new direction to implement the low-complexity traffic scheduler for flexible delay and jitter guarantees. However, current design of MGFQ is not targeted at guaranteeing the QoS of well-behaved traffic streams under the presence of misbehaved flows. Additional improvement of MGFQ is required if the residual bandwidth is to be allocated fairly and precisely under all traffic conditions.

Recently, several studies, such as RIO (RED with In/Out) [29] and LQD [30], have paid attentions to protect the in-profile portion of traffic flows. But those schemes mainly focus on solving the problem of NRT traffic, such as TCP flows. In addition, the per-flow requirement usually encounters the problem of scalability. Therefore, we propose an enhanced version of the MGFQ algorithm, called the *Differentiated Multi-layer Gated Frame Queueing* (DMGFQ) algorithm [78], to accommodate delay/jitter controls and fair residual bandwidth sharing for RT and NRT traffic streams simultaneously. The rationale of DMGFQ is that the eligible packets from misbehaved users will be scheduled to gain service only with the lower priority. DMGFQ not only possesses the advantages of low complexity, but also provides the capability of allocating bandwidth to differential user groups following

the pre-determined bandwidth sharing factors.

The organization of this chapter is as follows. In section 5.2, the proposed traffic scheduler with fair bandwidth sharing enhancement is presented. Simulation results of voice traffic under various experiments are shown in section 5.3. In section 5.4, we discuss the influence from its design factor and the selection of traffic profile parameters. And, finally, our conclusions are drawn in section 5.5.

5.2 Differentiated Multi-layer Gated Frame Queueing Discipline

5.2.1 Minimum Bandwidth Guarantee for RT and NRT Traffic Streams

One of the design goals of the DMGFQ algorithm is to achieve QoS guarantees to RT and NRT traffic streams conforming to the pre-determined traffic contracts. Therefore, it is necessary to define the conforming criteria for all traffic streams. The following are the definitions of the required notations:

- M_i: the guaranteed minimum bandwidth of VP i when VP i is backlogged;
- ϕ_i: the share weighting factor for VP i to share the residual bandwidth;
- C: the output link service rate;
- $S_i(t)$: the granted service rate for backlogged VP i at time t;
- $B(t)$: the backlogged VP set at time t;
- $A_i(s,t)$: the average arrival rate of VP i within the interval $(s,t]$;

- $R_i(s,t)$: the average output rate of VP i within the interval $(s,t]$.

In order to reduce implementation complexity, DMGFQ adopts the concept introduced in Chapter 4 to calculate the granted service rate $S_i(t)$ of each backlogged VP i only at the starting epoch of fixed time intervals, which are called the *refreshing period*. The length of the refreshing period is denoted as T and the starting epoch of n-th refreshing period is t_n, with $t_n = t_{n-1} + T$ for $n \geq 1$. For convenience, M_i, $S_i(t)$, $A_i(s,t)$ and $R_i(s,t)$ are all normalized by the link capacity C.

It is well known that the granted service rate of conventional scheduling algorithms for a backlogged VP i, $S_i(t)$, is often calculated and updated at any time t as t increases. For example, $S_i(t)$ of the Generalized Processor Sharing (GPS) algorithm [34] is calculated via the following equation:

$$S_i(t) = \frac{\phi_i}{\sum_{j \in B(t)} \phi_j} C, \qquad t \geq 0. \tag{5.1}$$

Other related scheduling algorithms, such as SCFQ [37], WRR [57], etc., all accommodate the fair bandwidth sharing via the same framework. In eq. (5.1), the maintenance of $B(t)$ leads to a complexity issue. Note that in GPS and its extensions the weighting factor ϕ_i determines the overall bandwidth share, not just on the share of residual bandwidth. Hence, these bandwidth sharing approaches are not appropriate for traffic that requires minimum bandwidth guarantee. A RT multimedia VP may require the minimum bandwidth to achieve the basic service quality. If the service rate for a multimedia application is less than the required minimum bandwidth, this multimedia service may not be accepted at all. In addition, if there is any residual bandwidth available, all VPs should be provided with a better quality via this residual bandwidth. Therefore, we suggest a bandwidth sharing criterion for RT multimedia applications where a minimum bandwidth is guaranteed for a backlogged VP by the scheduler, and then the residual bandwidth is shared by all

backlogged VPs following the pre-determined weighting factors (ϕ_i's).

In order to realize the above bandwidth allocation policy, and to reduce implementation complexity, it is necessary to update the backlogged set $B(t)$ and to calculate the granted service rate of each backlogged VP only at the starting epoch of each refreshing period, i.e., at t_n. Therefore, we suggest in DMGFQ that the bandwidth sharing criterion based on eq. (5.1) should be modified as

$$S_i(t) = M_i + \frac{\phi_i}{\sum_{j \in B(t_n)} \phi_j}(C - \sum_{j \in B(t_n)} M_j), \qquad \text{for } t_n \leq t < t_{n+1}, n = 1, 2, \cdots, \tag{5.2}$$

where VP i is backlogged. In DMGFQ, a VP is said to be a backlogged VP within a refreshing period if it has any cells to transmit during the refreshing period.

With only the bandwidth sharing criterion proposed in eq. (5.2), the rate control mechanism may still be difficult to be realized. Hence, we suggest the adoption of the service workload instead of the service rate. For example, to improve the efficiency of calculation, DMGFQ can employ certain counters to determine whether a VP has overused its reserved bandwidth, and then directly determines their shares of service workload for the whole refreshing period. Before describing the calculation procedures of the service workload, we introduce the following notations. We assume the underlying layer-2 PDU to be fixed and call it "cell" for simplicity.

- (B_i, M_i, ϕ_i): the traffic profile of VP i used in the traffic contract, consisting of the maximum burst size B_i, guaranteed minimum bandwidth M_i and the share weighting factor of residual bandwidth ϕ_i. Note that when a leaky-bucket policer [79] is used, M_i is equivalent to the *average rate* in the leaky-bucket policing algorithm, and B_i is equivalent to the *bucket depth* in the policer. If the arrived traffic load of VP i within $(0, t]$ is denoted as $N_i(t)$, then $\min\{N_i(t), M_i t + B_i\}$ is guaranteed for transmission within delay/jitter

constraints, regardless of residual bandwidth. Residual bandwidth is then allocated according to ϕ_i, period by period;

- $\psi_i(t_n, t_{n+1})$: the number of eligible cells of VP i within the n-th refreshing period;

- $W_i^r(t_n, t_{n+1})$: the *granted workload* for a backlogged VP i within the n-th refreshing period, which is the service workload satisfying the traffic profile of VP i. For simplicity, we also call it *reserved workload*;

- $W_i^e(t_n, t_{n+1})$: the *granted workload extension* (which is called *extended workload* for short) for a backlogged VP i within the n-th refreshing period, which is the workload contributed by the residual bandwidth observed within the n-th refreshing period and is beyond the reserved workload for $[t_n, t_{n+1})$;

- $W_i(t_n, t_{n+1})$: the *total granted service workload* for a backlogged VP i within the n-th refreshing period, which is the sum of $W_i^r(t_n, t_{n+1})$ and $W_i^e(t_n, t_{n+1})$.

For convenience, $W_i^r(t_n, t_{n+1})$, $W_i^e(t_n, t_{n+1})$ and $W_i(t_n, t_{n+1})$ are all normalized by the size of a single cell.

When the n-th refreshing period starts and the integrity of a cell is taken into considerations, the reserved workload of a backlogged VP i can be calculated via the following recursive equation at t_n:

$$W_i^r(t_n, t_{n+1}) = \min\left\{ \lfloor M_i T + B_i \rfloor, \left\lfloor (n+1) M_i T + B_i - \sum_{j=0}^{n-1} \min\{W_i^r(t_j, t_{j+1}), \psi_i(t_j, t_{j+1})\} \right\rfloor \right\}, \quad n \geq 1. \tag{5.3}$$

It can be shown that eq. (5.3) is equivalent to the result of the leaky-bucket policing algorithm in [79]. However, the number of eligible cells, $\psi_i(t_n, t_{n+1})$, may be less than

the reserved workload $W_i^r(t_n, t_{n+1})$. In order to allocate bandwidth more fairly, these residual time slots should be shared by other backlogged VPs. Therefore, the extended workload of a backlogged VP i is calculated at t_n:

$$W_i^e(t_n, t_{n+1}) = \left\lfloor \frac{\phi_i}{\sum_{j \in B(t_n)} \phi_j} \left(CT - \sum_{j \in B(t_n)} \min\{W_j^r(t_n, t_{n+1}), \psi_j(t_n, t_{n+1})\} \right) \right\rfloor . \tag{5.4}$$

In order to evaluate the fairness of a scheduling algorithm for RT traffic, we introduce the notation of the *fairness index* within the interval $(s, t]$, $\mathcal{FRI}(s, t)$, which is defined by

$$\mathcal{FRI}(s, t) = \max_{i,j} \left| \frac{\frac{R_i(s,t) - M_i}{C - \sum_{k \in B(t_n)} \min(A_k(s,t), M_k)}}{\phi_i} - \frac{\frac{R_j(s,t) - M_j}{C - \sum_{k \in B(t_n)} \min(A_k(s,t), M_k)}}{\phi_j} \right|, \quad 0 \le s < t, \tag{5.5}$$

where VP i and VP j are any continuously backlogged VPs within the interval $(s, t]$. Based on the definition provided in eq. (5.5), a smaller fairness index of a scheduling algorithm implies a better fairness level this algorithm can achieve.

5.2.2 Queueing Model of the DMGFQ Algorithm

The queueing model of the DMGFQ algorithm is shown in Fig. 5.1. This queueing model is an enhanced version of MGFQ in Chapter 4, and requires the underlying layer-2 protocol to employ fixed size PDU. For easy illustration, we assume ATM as the chosen layer-2 protocol and the fixed size cell as its PDU. In this model, it is required that the virtual connections belonging to the same virtual path always share the same nodal QoS parameters. Hence, we use a dedicated virtual path to support each VP or a class of VPs with the same QoS. For simplicity, we consider a

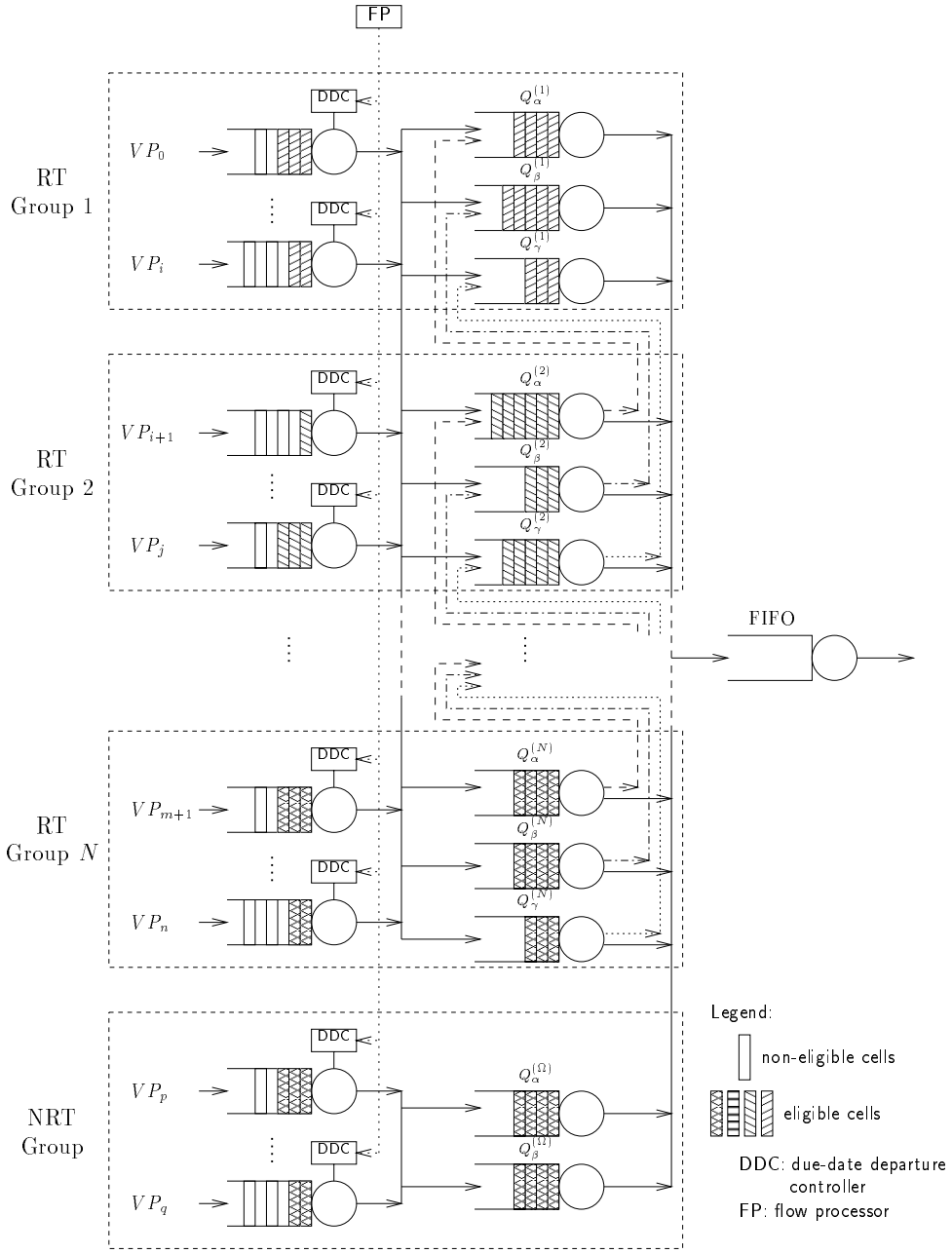


Figure 5.1: Queueing model of the DMGFQ algorithm.

VP stream, regardless of its virtual channel (VC) components, as a single connection (flow) supported by the DMGFQ scheduler.

In DMGFQ, each VP is assigned a dedicated FIFO queue. Each VP is associated with a class of services with a set of pre-determined cell-level QoS parameters, including delay, jitter, and cell loss ratio, residual bandwidth share, etc[1]. In addition, VPs of the physical link are organized as several groups according to VPs' jitter constraints. The jitter bounds of all VPs in the Group i are within $((i-1)T, iT]$ slot times, where $i = 1, \cdots, N$. NRT (NRT) traffic is assigned a special dedicated group, called Group Ω. The Head-of-Line (HOL) cell of a VP queue is called an eligible cell if it can be transmitted immediately without violating its delay bound and cell jitter constraint. Once the HOL cell has become eligible, it can be moved to the output queue at any stage of the scheduling process. For RT traffic, three FIFO queues at the output port, called α-*queue*, β-*queue* and γ-*queue*, are also dedicated for providing different services in the DMGFQ algorithm. The function of the α-*queue* of Group i (denoted as $Q_{\alpha}^{(i)}$) is to buffer the eligible cells belonging to the reserved workload of the corresponding VPs in Group i and the cells in queue $Q_{\alpha}^{(i+1)}$ during the last refreshing period. On the other hand, the β-*queue* of Group i (denoted as $Q_{\beta}^{(i)}$) buffers the eligible cells accounting for the workload beyond the reserved workload and being served with current residual bandwidth, $C - \sum_{j \in B(t_n)} M_j$. These cells belong to the extended workload. And the γ-*queue* of Group i (denoted as $Q_{\gamma}^{(i)}$) buffers the eligible cells exceeding the sum of the reserved workload and extended workload of the corresponding VPs. Next, the flow processor (FP) is responsible for calculating the reserved workload and extended workload of each VP and to inform the due-date departure-controllers (DDCs) to open their *gate*s with period T. When DDCs of Group i open their gates, eligible cells belonging to Group i are moved to

[1]For NRT traffic, its delay and jitter limits are assigned infinite.

the corresponding queues $Q_{\alpha}^{(i-1)}$, $Q_{\beta}^{(i-1)}$ and $Q_{\gamma}^{(i-1)}$. The jitter bound of each VP has to be ceiled as an integer multiple of T. On the other hand, for NRT traffic, only two extra FIFO queues, $Q_{\alpha}^{(\Omega)}$ and $Q_{\beta}^{(\Omega)}$, are needed. Each time when DDCs of Group Ω opens their gates, the cells whose total traffic load is within the *reserved workload* of each NRT VP are moved to queue $Q_{\alpha}^{(\Omega)}$ while the workload beyond the reserved bandwidth are moved to queue $Q_{\beta}^{(\Omega)}$. The cell moving procedure among α-, β- and γ-queues does not apply to NRT group.

In order to support effective delay and jitter control for multimedia traffic associated with each multimedia cell, additional control parameters are required in the DMGFQ operations. Like JEDD, the field of *Due-Date* must be included in the cell. Examples of such cell formats are available in Chapter 4. Before describing the operations of the DMGFQ algorithm, we introduce the following notations for delay and jitter controls when voice is assumed the major application.

- ND_i^h: the nodal cell delay bound assigned to virtual path i (VP_i) at node h, $h = 1, 2, ..., H$;
- J_i^h: the nodal cell jitter bound assigned to VP_i at node h, $h = 1, 2, ..., H$;
- $P_{i,k}^{vo}$: the k-th voice cell of VP_i;
- $AT_{i,k}^{vo,h}$: the arrival time of voice cell $P_{i,k}^{vo}$ at node h;
- $ET_{i,k}^{vo,h}$: the eligible time of voice cell $P_{i,k}^{vo}$ at node h, which is the time epoch the cell become eligible;
- $DT_{i,k}^{vo,h}$: the departure time of voice cell $P_{i,k}^{vo}$ at node h;
- $IND_{i,k}^{vo,h}$: the initial nodal due-date of voice cell $P_{i,k}^{vo}$ at node h;
- $DD_{i,k}^{vo,h}$: the due-date of voice cell $P_{i,k}^{vo}$ when it leaves node h.

Suppose the nodal jitter bounds of all RT flows in this node is within the interval $[0, NT]$, then total $3N$ queues (N FIFO queues for each set of α-, β-, and γ-*queue*s) are dedicated to buffer eligible cells. Each time when a cell arrives, the *initial nodal due-date* (IND) and the *eligible time* (ET) of the cell are calculated. DMGFQ can then provide delay and jitter guarantees via IND and ET. The calculation procedures of INDs and ETs for voice traffic are described briefly as follows.

For voice traffic, when a cell $P_{i,k}^{vo}$ arrives node h, its IND and ET are calculated via following equations:

$$\begin{cases} IND_{i,k}^{vo,1} &= ND_i^1, \\ IND_{i,k}^{vo,h} &= DD_{i,k}^{vo,h-1} + ND_i^h, & h > 1, \\ ET_{i,k}^{vo,h} &= AT_{i,k}^{vo,h} + IND_{i,k}^{vo,h} - J_i^h, & h \geq 1. \end{cases} \tag{5.6}$$

However, in DMGFQ the due-date of a cell does not have to be updated slot-by-slot. The due-date of a cell only has to be updated via following equation when that cell departs node h:

$$DD_{i,k}^{vo,h} = IND_{i,k}^{vo,h} - (DT_{i,k}^{vo,h} - AT_{i,k}^{vo,h}). \tag{5.7}$$

Note that above notations and operations can be easily extended to support video cells. Corresponding cell formats and due-date calculation procedures have been derived in Chapter 4. In addition, the above notations and operations are not required for NRT traffic.

After finishing the calculation procedures of IND and ET, this cell is buffered in the corresponding VP queue. At t_n, the flow processor clears all eligible cells in the output buffer and three FIFO queues in Group 1, including queue $Q_\alpha^{(1)}$, $Q_\beta^{(1)}$ and $Q_\gamma^{(1)}$. Then the flow processor calculates the reserved workload and the extended workload of each VP and informs each VP of these two parameters. Next, the flow processor informs all DDCs to open their gates and to move the eligible cells from

Group i to higher level groups. The eligible cells originally belonging to the VP queues of Group i are moved to the appropriate α-*queue*s according to their due-dates if these cells are within the reserved workloads. If the number of the eligible cells of a VP in Group i exceeds the reserved workload, the excess portion of the eligible cells are moved to the appropriate β-*queue*s. However, if the excess portion still exceeds the extended workload, the rest eligible cells are moved to the γ-*queue*s. The α-*queue* is assigned the highest service priority while the γ-*queue* is assigned the lowest service priority. Then, the cells in Group i whose due-dates are less than or equal to iT are marked eligible. The sequence of operation follows the increasing order of the group index. During the transmission stage, cells in queue $Q_\alpha^{(i)}$ are always moved before $Q_\alpha^{(i+1)}$. This is to assure the cell sequence integrity.

Before the above operations, the scheduler can determined in advance an integer $1 \leq \kappa \leq N$ such that all eligible cells in queue $Q_\alpha^{(1)}$, $Q_\alpha^{(2)}, \cdots, Q_\alpha^{(\kappa)}$ are guaranteed the highest priority to be moved to the final FIFO queue during a refreshing period. Here, we call the factor κ as the *protection factor*, which is related to the desired fairness. If there are residual time slots in a refreshing-period after serving queue $Q_\alpha^{(\kappa)}$, then the cells in queue $Q_\alpha^{(\Omega)}$ are served. In turn, we use the residual time slots to serve the cells in β-*queue*s, following the sequence $1, 2, \cdots, \kappa$ and the cells in queue $Q_\beta^{(\Omega)}$. If there is still some residual time slots after serving queue $Q_\beta^{(\Omega)}$ in the refreshing-period, then γ-*queue*s are granted the permission to transmit. The larger value κ is, the more eligible cells belonging to reserved workload can be transmitted. As a result, with larger κ, more well-behaved flows are *protected*, and better fairness can be achieved. On the other hand, the residual bandwidth available to casually misbehaved flows will be limited by large κ. For network managers who wish to increase link bandwidth utilization at the same time, extremely large κ may not be able to serve their goals.

Based on the required queueing and scheduling operations, the sorting overhead of DMGFQ is at the same order of RPQ$^+$ [74], while the due-date calculation is at the same order of JEDD [21]. Therefore, DMGFQ accommodates both RT streaming data and best-effort traffic with fair residual bandwidth share via low processing overhead.

5.2.3 The Concept of Virtual Cells

According to the operation described in the previous section, we observe that the condition of out-of-sequence transmission may occur. More specifically, because the eligible cells of VP i are sequencially moved to the $\alpha/\beta/\gamma$-queues. Hence, the sequence numbers of the cells belonging a flow in queue $Q_{\alpha}^{(i+1)}$ are larger than those cells in queues $Q_{\beta}^{(i)}$ and $Q_{\gamma}^{(i)}$. However, due to the mechanism of the protection factor κ, the cells in queue $Q_{\alpha}^{(i+1)}$ may be transmitted before queues $Q_{\beta}^{(i)}$ and $Q_{\gamma}^{(i)}$. Once this condition occurs, cell transmission are out of sequence and these cells will be discarded in the destination nodes.

In order to maintain the cell transmission sequence, we propose the virtual cell operation to preserve the advantages of the protection factor and maintain the cell transmission sequence simultaneously. When a new cell arrives, the following actions are taken: (1) the cell joins the corresponding share memory, which is called *flow queue*; (2) a new virtual cell with the same information such as VPI, VCI, due-date, etc., is created and its initial due-date, eligible time are calculated according to the algorithm described in section 5.2.2; (3) the virtual cell joins the corresponding VP queue. Any time when a virtual cell is chosen to transmit, then the HOL cell of the corresponding VP queue is picked and the due-date information of the virtual cell is assign to that HOL cell. If a virtual cell is discarded due to delay/jitter violation, the HOL cell of the corresponding VP queue is also discarded. The shift operations

of the virtual cells from VP queues to $\alpha/\beta/\gamma$-queues, and from Group i to Group $i-1$ are the same as what described in section 5.2.2.

Because the effective information required in virtual cell operations is only information in the cell headers and the actual cell payload is not required in the virtual cells, the hardware cost in memory is very limited. Thus, we believe this virtual cell operation will not become another scheduling bottleneck for DMGFQ.

5.2.4 Feasibility of Hardware Implementation

When the implementation complexity is limited, the value of $\psi_i(t_n, t_{n+1})$ might not be available to the scheduler at t_n. Hence, we propose the following flexible implementation approach to reduce hardware implementation complexity.

Suppose the switching node needs Δ refreshing periods (where $\Delta \geq 1$ is an integer) to compute the number of eligible cells of all VPs within a refreshing period $[t_n, t_{n+1})$. Implicitly, the additional requirement is that the minimal nodal delay and jitter bounds have to be greater than or equal to $(\Delta+1)T$ time slots. We then have to add additional $(\Delta+1)L$ counters to register $\psi_i(t_n, t_{n+1})$, $\psi_i(t_{n+1}, t_{n+2}), \cdots$, $\psi_i(t_{n+\Delta}, t_{n+\Delta+1})$, $i = 1, \cdots, L$, where L is the maximum number of VPs of a switching node. At time t_n, the switching node operation, additional to the original DMGFQ approach, is to compute the variable $\psi_i(t_{n+\Delta}, t_{n+\Delta+1})$ for all VPs. In this case, extra buffer size needed for no loss guarantee in the worst case is $C\Delta T$. When all additional requirements in delay/jitter bounds are met, extra buffers and counters are available, the use of eq. (5.3) and (5.4) can accommodate many flexible hardware realization approaches without sacrificing the performance of DMGFQ.

In order to more precisely describe the operation of the DMGFQ algorithm using eq. (5.3) and (5.4), in Fig. 5.2 we present the pseudo code for an implementation of the DMGFQ scheduler when RT traffic is supported. When NRT traffic exists,

the pseudo code is almost the same except the delay/jitter bounds for NRT streams should be set equivalent to infinite. The operational parameter Δ for flexible hardware implementation is also included in the pseudo code. Note that these codes can be further optimized for efficient execution, such as in parallel mode. We prefer this version because it is easier to read.

5.3 Performance Analysis of Non-Real-Time Traffic Streams

In this section, we analyze the delay bound of the proposed DMGFQ algorithm. In the following, we describe the definition of *well-behaved* flows and *misbehaved* flows at first.

Definition 5.1 *Suppose the average arrival rate of flow i is ρ_i and the maximum input burst size is σ_i. Assume the negotiated service level agreements of flow i are (M_i, B_i), where M_i is the guaranteed minimum bandwidth and B_i is the maximum burst size. Then, this traffic stream is called a well-behaved traffic stream if (1) $\rho_i \leq M_i$, and (2) $\sigma_i \leq B_i$; otherwise it is called a misbehaved traffic stream.*

Then, we present the delay bound of the DMGFQ algorithm for NRT traffic streams in the following theorem.

Theorem 5.1 *Suppose that there are N_R RT traffic flows, denoted from flow 1 to flow N_R, and N_N NRT traffic flows, denote from flow $N_R + 1$ to flow $N_R + N_N$ in the system, and that flow i is with QoS parameter (M_i, B_i). Then the worst case delay bound of the well-behaved NRT traffic streams is $\frac{\sum_{i=1}^{N_R+N_N}(M_iT+B_i)}{C-\sum_{i=1}^{N_R+N_N}M_i} + T$.*

Before describing the proof of Theorem 5.1, we introduce following lemmas at first.

main()
{
  while (1){
    t = system time;
    if (*cell_arrived*==TRUE){
      p = arrived cell;
      calculate the due-date of cell p;
      place cell p in its VP queue;
    }
    extract cell p from output buffer, α-*queue*s, β-*queue*s, or γ-*queue*s;
    if (p != *NULL*){
      update due-date of cell p;
      extract cell p from output buffer and transmit it;
    }
    if (*refreshing_event_occurs*==TRUE){
      discard all cells in the output buffer;
      discard all cells in queue $Q_\alpha^{(1)}$, $Q_\beta^{(1)}$-, and $Q_\gamma^{(1)}$;
      refreshing_procedure(t);
      schedule next *refreshing_event* at time $t+T$;
      reset output process;
    }
  }
}
refreshing_procedure(t)
{
  for ($i = N; i \geq 2; i--$){
    move cells from $Q_\alpha^{(i)}$ to $Q_\alpha^{(i-1)}$;
    move cells from $Q_\beta^{(i)}$ to $Q_\beta^{(i-1)}$;
    move cells from $Q_\gamma^{(i)}$ to $Q_\beta^{(i-1)}$;
  }
  for (*each backlogged virtual path* i){
    calculate reserved workload, $W_i^r(t,t+T)$, via leaky-bucket algorithm with parameters (B_i, M_i);
    extended workload $W_i^e(t,t+T) = \left\lfloor \frac{\phi_i}{\sum_{j\in B(t)}\phi_j}\left(CT - \sum_{j\in B(t)} \min\{W_j^r(t,t+T), \psi_j(t,t+T)\}\right)\right\rfloor$;
    if ($\psi_i(t,t+T) \leq W_i^r(t,t+T)$)
      move ($\psi_i(t,t+T)$) eligible cells of VP i to corresponding α-*queue*s;
    else if ($W_i^r(t,t+T) < \psi_i(t,t+T) \leq W_i^r(t,t+T) + W_i^e(t,t+T)$){
      move ($W_i^r(t,t+T)$) eligible cells of VP i to corresponding α-*queue*s;
      move residual ($\psi_i(t,t+T) - W_i^r(t,t+T)$) eligible cells of VP i to corresponding β-*queue*s;
    }
    else{
      move ($W_i^r(t,t+T)$) eligible cells of VP i to corresponding α-*queue*s;
      move residual ($W_i^e(t,t+T)$) eligible cells of VP i to corresponding β-*queue*s;
      move residual ($\psi_i(t,t+T) - W_i^r(t,t+T) - W_i^e(t,t+T)$) eligible cells of VP i to corresponding γ-*queue*s;
    }
  }
  move cells in queue $Q_\alpha^{(1)}$ to the output buffer;
  for (*each VP* i)
    calculate the variable $\psi_i(t+\Delta T, t+(\Delta+1)T)$;
}

Figure 5.2: Pseudocode of DMGFQ algorithm applied to the RT traffic streams.

Lemma 5.1 *Suppose there are N_R well-behaved RT traffic flows, denoted from flow 1 to flow N_R, in node h, and flow i is with QoS parameter (M_i, B_i, J_i^h, D_i^h). Then, there exists at least one arrival pattern such that no cell loss occurs during the server busy period.*

Proof: Suppose the whole system become busy at the time epoch t_1. We designate the arrival patterns of flow i at the starting epoch of j-the refreshing period as $M_iT + B_i^j$, $j \geq 1$.

From t_1 to t_{n_1}, where $n_1 = \left\lceil \frac{B_1}{\left\lfloor \left(1-\sum_{i=1}^{N_R} M_i\right)T \right\rfloor} \right\rceil$, we may assign B_i^j as follows:

$$B_i^j = \begin{cases} \left\lfloor \left(1 - \sum_{i=1}^{N_R} M_i\right) T \right\rfloor, & \text{for } i = 1, j < n_1; \\ B_1 - (n-1) \left\lfloor \left(1 - \sum_{i=1}^{N_R} M_i\right) T \right\rfloor, & \text{for } i = 1, j = n_1; \\ 0, & \text{for } i \neq 1, j \leq n_1. \end{cases} \tag{5.8}$$

And after t_{n_1}, $B_1^j = 0$, then no cell losss will occur for flow 1.

Similarly, from $t_{n_1} + 1$ to $t_{n_1} + t_{n_2}$, where $n_2 = \left\lceil \frac{B_2}{\left\lfloor \left(1-\sum_{i=1}^{N_R} M_i\right)T \right\rfloor} \right\rceil$, we can assign B_i^j as follows:

$$B_i^j = \begin{cases} \left\lfloor \left(1 - \sum_{i=1}^{N_R} M_i\right) T \right\rfloor, & \text{for } i = 2, n_1 < j < n_1 + n_2, \\ B_2 - (n_2 - 1) \left\lfloor \left(1 - \sum_{i=1}^{N_R} M_i\right) T \right\rfloor, & \text{for } i = 2, j = n_1 + n_2, \\ 0, & \text{for } i \neq 2, n_1 < j \leq n_1 + n_2. \end{cases} \tag{5.9}$$

And after $t_{n_1} + t_{n_2}$, $B_2^j = 0$, then no cell losss will occur for flow 2.

Following the above procedure, we may designate B_i^j as follows:

$$B_i^j = \begin{cases} \left\lfloor \left(1 - \sum_{i=1}^{N_R} M_i\right) T \right\rfloor, & \text{for } i = m, \sum_{k=1}^{m-1} n_k < j < \sum_{k=1}^{m} n_k, \\ B_m - (n_m - 1) \left\lfloor \left(1 - \sum_{i=1}^{N_R} M_i\right) T \right\rfloor, & \text{for } i = m, j = \sum_{k=1}^{m} n_k, \\ 0, & \text{for } i \neq m, \sum_{k=1}^{m-1} n_k < j \leq \sum_{k=1}^{m} n_k, \end{cases} \tag{5.10}$$

where $n_m = \left\lceil \frac{B_m}{\left\lfloor \left(1-\sum_{i=1}^{N_R} M_i\right) T \right\rfloor} \right\rceil$. Again, after $\sum_{k=1}^{m} t_{n_k}$, $B_m^j = 0$, then no cell losss will occur for flow m.

Hence, there exists at least one arrival pattern such that no cell loss occurs during the server busy period.

Q.E.D.

Lemma 5.2 *The maximum delay of a well-behaved NRT traffic stream incurred by the α-queue is* $\frac{\sum_{i=1}^{N_R+N_N}(M_i T+B_i)}{C-\sum_{i=1}^{N_R+N_N} M_i}$.

Proof: Because there are delay and jitter constraints for RT traffic streams, the cells violating these constraints are discarded. In addition, the cells of RT traffic streams belonging to α-queues have higher transmission priority than the cells of NRT traffic streams. Therefore, the worst case delay of NRT traffic streams occurs when no RT cell is discarded.

We assume the cells moved to the NRT α-queue at time $(n-1)T$ meets the worst case delay D_N and the system becomes busy at time 0. Then

$$A_{RT}(0, \left\lceil \frac{D_N}{T} \right\rceil + (n-1)T) + A_{NRT}(0, nT) = S(0, (n-1)T + D_N), \tag{5.11}$$

where $A_{RT}(0,t)$ and $A_{NRT}(0,t)$ represent the total workload of RT and NRT traffic streams moved to α-queues within the interval $[0,t]$, respectively. And $S(0,t)$ denotes the total service workload within the interval $[0,t]$. Hence, we have

$$\sum_{i=1}^{N_R} M_i \cdot \left(\left\lceil \frac{D_N}{T} \right\rceil T + (n-1)T \right) + \sum_{i=1}^{N_R} B_i + \sum_{i=N_R+1}^{N_R+N_N} M_i nT + \sum_{i=N_R+1}^{N_R+N_N} B_i = C(D_N + (n-1)T). \tag{5.12}$$

Then we can simplify eq. (5.12) as

$$CD_N - \sum_{i=1}^{N_R} M_i \left\lceil \frac{D_N}{T} \right\rceil T = \sum_{i=1}^{N_R+N_N} B_i + \sum_{i=N_R+1}^{N_R+N_N} M_i T + \left(\sum_{i=1}^{N_R+N_N} M_i - C \right) (n-1)T. \tag{5.13}$$

Because $\lceil \frac{D_N}{T} \rceil < \frac{D_N}{T} + 1$, eq. (5.13) can be simplified further as

$$D_N \left(C - \sum_{i=1}^{N_R} M_i \right) \leq \sum_{i=1}^{N_R+N_N} (M_i T + B_i) + \left(\sum_{i=1}^{N_R+N_N} M_i - C \right) (n-1)T. \tag{5.14}$$

Because $\sum_{i=1}^{N_R+N_N} M_i \leq C$, therefore, we can derive the worst case delay bound of NRT traffic streams as

$$D_N \leq \frac{\sum_{i=1}^{N_R+N_N} (M_i T + B_i)}{C - \sum_{i=1}^{N_R} M_i}. \tag{5.15}$$

Q.E.D.

Lemma 5.3 *The maximum delay of a well-behaved NRT traffic stream contributed by waiting for movement from flow queues to corresponding α-queues is T.*

Proof: If a cell arrives just after the refreshing period, the cell has to wait for the next refreshing period to grant the reserved workload and extended workload. In this case, the maximum delay bound contributed from waiting for the new refreshing period is the length of refreshing period T.

Q.E.D.

Proof of Theorem 5.1: From Lemma 5.1 to Lemma 5.3, it is easy to derive that the worst case delay bound of well-behaved NRT traffic streams is $\frac{\sum_{i=1}^{N_R+N_N} (M_i T + B_i)}{C - \sum_{i=1}^{N_R+N_N} M_i} + T$.

Q.E.D.

5.4 Simulation Results

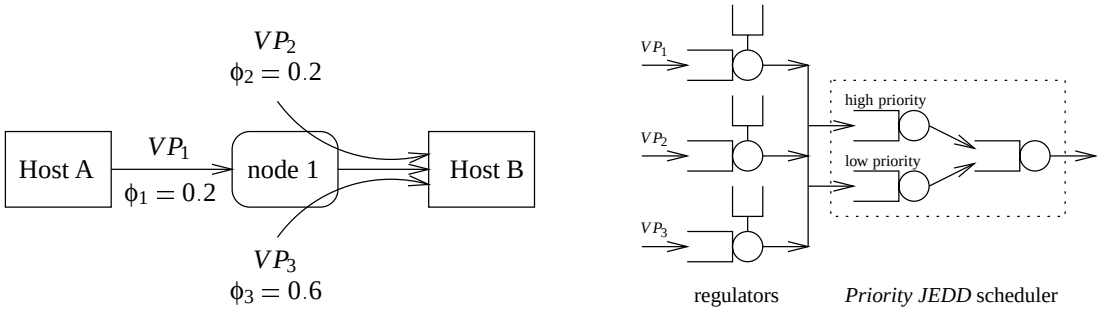
In this section, we evaluate the performance of the DMGFQ scheme for voice traffic streams. The examined performance metrics include the cell loss ratio, the shared bandwidth and Fairness Index. Here, the cell loss ratio only accounts for those cells discarded due to delay or jitter violations. Since buffer size is assumed infinite, no cell losses result from buffer overflow. We also employ FCFS and JEDD for baseline comparisons.

Suppose the link bandwidth is 45 Mbps. All voice streams are assumed to follow ITU-T G.764 voice packetization recommendation [75] and the silence suppression mechanism is implemented. Therefore, a voice stream is modeled as an ON-OFF traffic source in the simulation. The ON/OFF durations of all these voice streams follows the exponential distribution with mean 1.5 and 2.25 seconds, respectively. While ON, a voice source transmits one cell every 703 slot times, corresponding to a 64 Kbps stream with silence suppression. In order to avoid man-made simultaneous arrivals of cell bursts at the multiplexer, the starting epoch of each voice source is uniformly distributed over the 1.5 sec interval. The refreshing period is set to be 0.5 ms and all simulations lasts for 10^8 time slots. Both the target delay constraints and jitter constraints are flexible.

5.4.1 Experiment 1: The Transient Behaviors of Priority JEDD and DMGFQ for RT Traffic Streams

In this simulation experiment (see Fig. 5.3 (a)), we examine the transient behaviors of the DMGFQ algorithm and the *Priority JEDD* algorithm. The *Priority JEDD* algorithm is an extended version of the Jitter-EDD algorithm [21], where the eligible cells conforming to the traffic contracts are assigned the high service priority, and

non-conforming eligible cells are assigned the low priority. The queueing model of the *Priority JEDD* algorithm is shown in Fig. 5.3 (b). Each scheduler queue serves eligible cells in the increasing order of their due-dates. The introduction of *Priority JEDD* is to allow a fair comparison between JEDD and DMGFQ algorithms. There are three CBR flows and their configurations are shown in Table 5.1. VP_1 is activated at 0.2 sec and is deactivated at 0.5 sec. The nodal delays and jitter constraints of three VPs are all 6.5 msec.



(a) Network configuration.

(b) Internal architecture for node 1 under *Priority JEDD*.

Figure 5.3: Simulation model for Experiment 1.

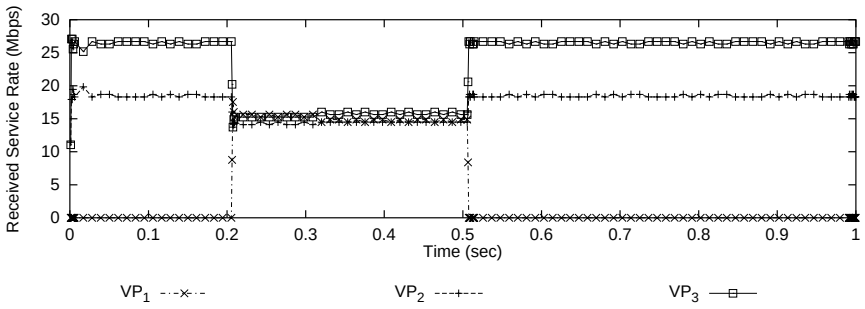
Figure 5.4 shows the transient bandwidth sharing behaviors of VP_1, VP_2 and VP_3 for two algorithms, respectively. Ideally, before VP_1 is activated, VP_2 shall be granted the service rate $13.5 + (45 - 13.5 - 13.5)\frac{0.2}{0.2+0.6} = 18$ Mbps, and the service rate of VP_3 shall be 27. From Fig. 5.4 (a) and (b), we can observe that the granted service rates of VP_2 and VP_3 approach 18 Mbps and 27 Mbps under DMGFQ, respectively. When VP_1 is activated at 0.2 sec, the service rates of VP_1 and VP_2 both approach their ideal granted service rate $13.5 + (45 - 13.5 \times 3)\frac{0.2}{0.2+0.2+0.6} = 14.4$

	Arrival Rate (Mbps)	Reserved BW M_i (Mbps)	Max. Burst Size B_i (cells)	Residual BW Share ϕ_i
VP_1	31.5	13.5	50	0.2
VP_2	31.5	13.5	50	0.2
VP_3	31.5	13.5	50	0.6

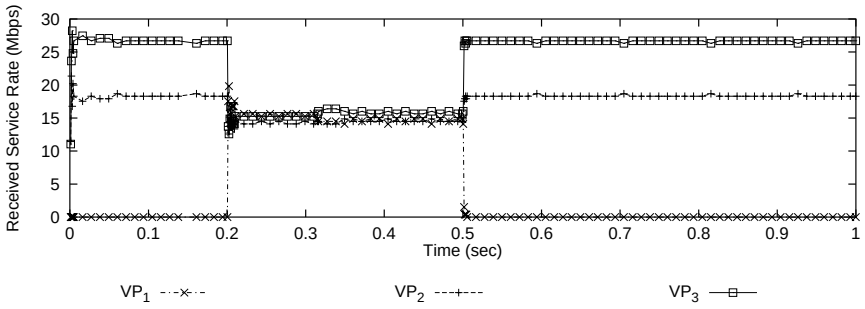
Table 5.1: Simulation configuration for Experiment 1.

Mbps and the service rate of VP_3 is close to 16.2 Mbps. As one can easily observe, all VPs are granted service rates according to their traffic profiles under DMGFQ. Therefore, DMGFQ indeed provides the capability of allocating bandwidth to different user groups following the pre-determined bandwidth sharing factor. Under *Priority JEDD*, VP_1 can be provided with guaranteed bandwidth according to its reservation, as shown in Fig. 5.4 (c). However, the *Priority JEDD* algorithm cannot distribute the residual bandwidth fairly to differentiated users. When VP_1 is OFF, both VP_2 and VP_3 obtain the same bandwidth regardless of their designated residual bandwidth shares. Therefore, the *Priority JEDD* algorithm can protect the well-behaved streams but it cannot differentiate users based on different traffic profiles. In addition, we compare the fairness indices as defined in (5.5) for various scheduling algorithms in three different time intervals. The simulation results of the fairness indices are shown in Table 5.2. We can observe that even when DMGFQ under the most unfair condition ($\kappa = 1$), DMGFQ still provides much better fairness performance than *Priority JEDD*. We also find that the larger the protection factor κ is, the more fairly DMGFQ performs.

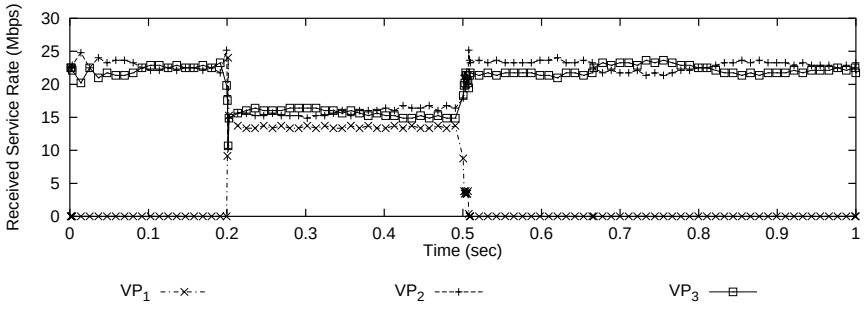
If we compare Fig. 5.4 (a) and (b) in details, we can observe that when someone activates VP_1, the reaction times for the DMGFQ scheduler to adjust the service



(a) DMGFQ with protection factor $\kappa = 1$.



(b) DMGFQ with protection factor $\kappa = 12$.



(c) The *Priority JEDD* algorithm.

Figure 5.4: Transient bandwidth sharing behaviors of the *Priority JEDD* algorithm and the DMGFQ algorithm with two different protection factors ($\kappa = 1, 12$).

Algorithm	Fairness Index ($\mathcal{FRI}$)		
	$0 \sim 0.2$ sec	$0.2 \sim 0.5$ sec	$0.5 \sim 1.0$ sec
Priority JEDD	1.77	2.81	1.82
DMGFQ ($\kappa = 1$)	0.37	0.73	0.18
DMGFQ ($\kappa = 6$)	0.23	0.58	0.13
DMGFQ ($\kappa = 12$)	0.09	0.25	0.13

Table 5.2: The Fairness Indices of various scheduling algorithms within different time intervals.

rate of VP_1 in these two figures are different. When the protection factor κ is 1, the eligible cells of VP_1 have to wait for transmission until the eligible cells, including eligible cells in α-, β- and γ-*queue*s of high priority groups are served. On the other hand, when the protection factor κ is 12, the eligible cells of VP_1 which satisfy the reserved workload only have to wait for all eligible cells in α-*queue*s of high priority groups to finish transmission. Hence, the larger the protection factor κ is set, the faster the DMGFQ algorithm reacts the change of the traffic load.

Another advantage of DMGFQ to be noticed is that it implies no punishment mechanism as the VirtualClock [59] algorithm. After VP_1 is activated at 0.2 sec, VP_2 and VP_3 just only release the bandwidth they have overused. They are never punished by DMGFQ for the bandwidth they legally overused in previous periods. On the other hand, the credit of bandwidth usage for any VP is not accumulated in any condition. For example, after VP_1 is activated, VP_1 is only granted the bandwidth it reserved. VP_1 cannot ask for the bandwidth it does not make use of before 0.2 sec. This feature makes DMGFQ a promising candidate as a fair scheduling algorithm.

5.4.2 Experiment 2: The Transient Behavior of DMGFQ for NRT Traffic Streams

In this simulation scenario (see Fig. 5.5), we examine the transient behavior of the DMGFQ algorithm for NRT traffic streams. Two CBR VPs (VP_1 and VP_2) are employed to model the RT traffic streams and their configurations are shown in Table 5.3. Similar to Experiment 1, VP_1 is activated at 0.2 sec and is deactivated at 0.5 sec, while VP_2 is activated and deactivated at 0 sec and 0.75 sec, respectively. The nodal delays and jitter constraints of these two VPs are all 6.5 msec. VP_3 carries the NRT traffic stream and consists of 25 VCs. The traffic in each VC is a replay of LAN traffic trace obtained from the Lawrence Berkeley National Laboratory [63]. In total, the original trace is used to derive a trace with average traffic load equal to 36 Mbps. The output link bandwidth is also assumed 45 Mbps.

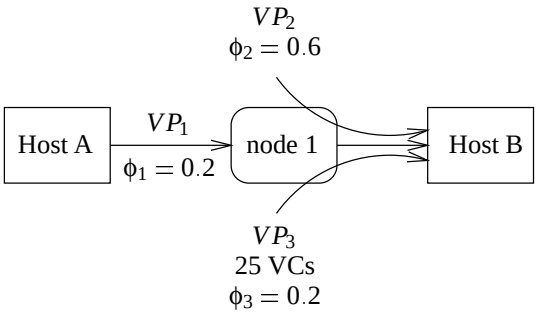


Figure 5.5: Simulation model for Experiment 2.

In this experiment, FCFS is employed as the baseline comparison. The switching node in this baseline system perform nothing except forwarding the cells. The cells with delay constraint violations are discarded only by the receiver. Figure 5.6 shows the bandwidth sharing behaviors of VP_1, VP_2 and VP_3 for FCFS and DMGFQ

	Arrival Rate (Mbps)	Reserved BW M_i (Mbps)	Max. Burst Size B_i (cells)	Residual BW Share ϕ_i
VP_1	31.5	9.0	20	0.2
VP_2	31.5	9.0	20	0.6
VP_3	36.0	9.0	20	0.2

Table 5.3: Simulation configuration for Experiment 2.

with two protection factors during the time interval $(0, 1]$ sec. We can observe obviously that bandwidth is not allocated according to the traffic profiles of each VP. Therefore, in the following discussions, the behavior of FCFS is not included.

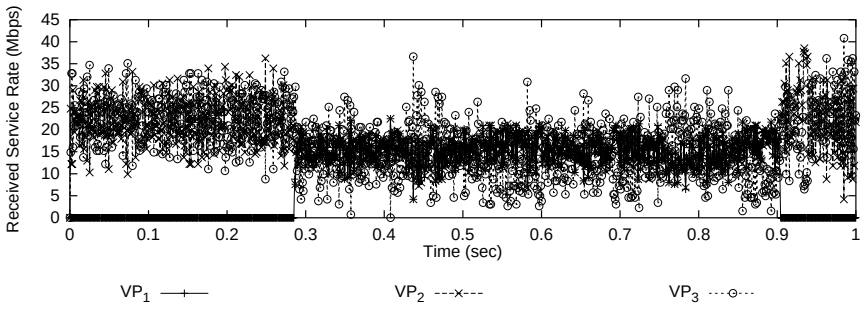
Before VP_1 is activated, the ideal granted-service rates of VP_2 and VP_3 shall be $9.0 + (45 - 9.0 - 9.0)\frac{0.6}{0.2+0.6} = 29.25$ Mbps and 15.75 Mbps. From Fig. 5.6 (b) and (c), we can observe that the average service rates of VP_2 and VP_3 both approach their ideal granted-service rates under DMGFQ. Note that the bursty behaviors shown in Fig. 5.6 is resulted from the burstiness of LAN traffic traces. Upon VP_1 is activated at 0.2 sec, the service rates of VP_1 and VP_3 both approach their ideal granted service rates $9.0 + (45 - 9.0 \times 3)\frac{0.2}{0.2+0.2+0.6} = 12.6$ Mbps, and the service rate of VP_2 is close to 19.8 Mbps. As one can observe easily, all VPs are granted service rates according to their traffic profiles under DMGFQ, even for the low prioritized NRT traffic streams. After VP_1 is terminated at 0.5 sec, the average service rates of VP_2 and VP_3 approach 29.25 Mbps and 15.75 Mbps again. After VP_2 is deactivated at 0.75 sec, VP_3 uses the total link bandwidth and is granted the service rate 45 Mbps.

The detailed simulation statistics are listed in Table 5.4. In general, the behavior among VPs are pretty similar for two extreme κ ($\kappa = 1, 12$), which indicates that

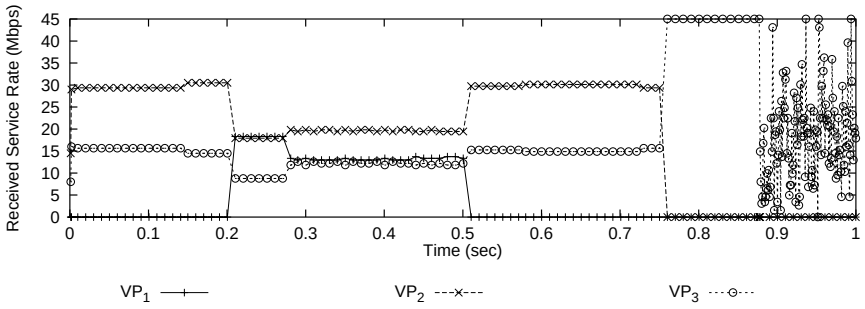
DMGFQ should be effective against bursty NRT traffic for a wide range of parameter κ. Here, we divide the granted service workload by the length of the observed interval to derive the average service rate. It is easy to verify that DMGFQ is able to adapt to the traffic conditions and traffic profiles to adjust its bandwidth sharing while FCFS always share the bandwidth equally. And DMGFQ can achieve satisfactory bandwidth sharing fairness in spite of the heavy burstiness of NRT traffic. Based on the simulation results obtained in Experiment 1 and Experiment 2, we conclude that DMGFQ not only provides QoS guarantees for RT traffic streams but also guarantees the bandwidth usage of NRT user groups following pre-determined traffic profiles.

5.4.3 Experiment 3: The Performance Impact on the Well-behaved User with Tighter Jitter Constraints

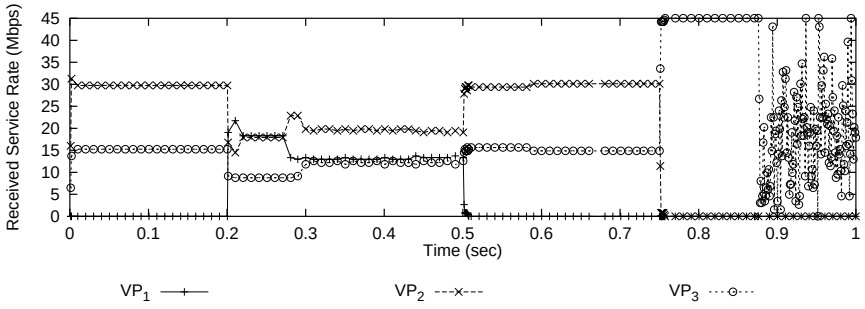
In the third experiment, we evaluate the cell loss ratio of various scheduling algorithms. The network configuration model is shown in Fig. 5.7 (a). VP_1 consists of 700 voice streams and the number of voice streams of VP_2 varies from 700 to 1200. The simulation configuration and delay/jitter constraints of two VPs are shown in Table 5.5. Because the jitter bounds of VP_1 and VP_2 are different and the performances of various scheduling algorithms must be compared fairly, regulators are required by the scheduler FIFO to accommodate different jitter constraints. These regulators, which serve as the front end cell processor, perform nothing except preventing non-eligible cells from moving to the scheduler buffer. The cells with delay constraint violations are discarded only by the receiver in the scheduler FIFO. The queueing model of the scheduler FIFO or JEDD is shown in Fig. 5.7 (b). In order to have a fair comparison between JEDD and DMGFQ algorithm, the *Priority JEDD* scheduler shown in Fig. 5.3 (b) is also employed in this simulation experiment. For the DMGFQ algorithm, we adjust the protection factor to two critical points,



(a) FCFS without any control mechanism.



(b) DMGFQ with protection factor $\kappa = 1$.



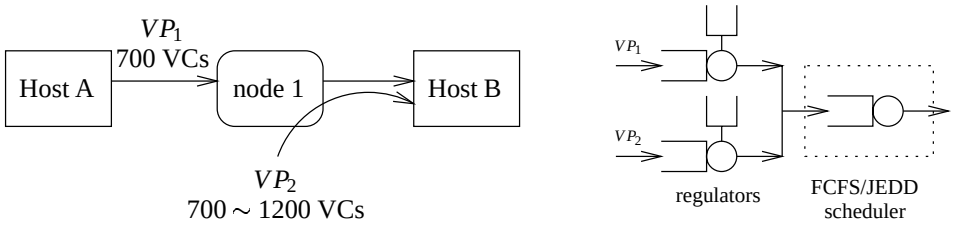
(c) DMGFQ with protection factor $\kappa = 12$.

Figure 5.6: Transient bandwidth sharing behaviors of FCFS and DMGFQ with two protection factors ($\kappa = 1, 12$) for NRT traffic streams.

Observed Interval (sec)			$0 \sim 0.2$	$0.2 \sim 0.5$	$0.5 \sim 0.75$	$0.75 \sim 1.0$
Activated VP			VP_2, VP_3	VP_1, VP_2, VP_3	VP_2, VP_3	VP_3
FCFS	Avg. Service Rate	VP_1	0	10.679	15.873	9.113
		VP_2	22.140	16.919	15.871	18.060
		VP_3	22.833	17.402	13.256	17.828
	$\mathcal{FRI}$		1.751	1.868	1.273	2.421
DMGFQ ($\kappa = 1$)	Avg. Service Rate	VP_1	0	14.262	0.319	0
		VP_2	28.804	18.011	28.593	0.588
		VP_3	16.072	12.727	16.088	34.333
	$\mathcal{FRI}$		0.087	0.627	0.103	**N.A.**
DMGFQ ($\kappa = 12$)	Avg. Service Rate	VP_1	0	13.512	0.120	0
		VP_2	28.821	18.765	28.685	0.558
		VP_3	16.067	12.723	16.088	34.333
	$\mathcal{FRI}$		0.085	0.349	0.097	**N.A.**

Table 5.4: The average service rates and Fairness Indices of FCFS and DMGFQ within different time intervals. The unit of average service rate is Mbps. The notation **N.A.** represents "Not Available."

i.e., the maximum value 6 and the minimum value 1, respectively. Details of the DMGFQ scheduler is referred to Fig. 5.1.



(a) Network configuration.

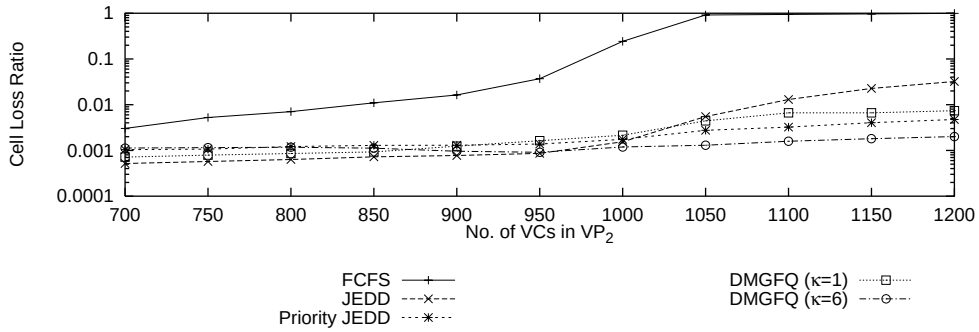
(b) Internal architecture for node 1 under FCFS or JEDD.

Figure 5.7: Simulation model for Experiment 3.

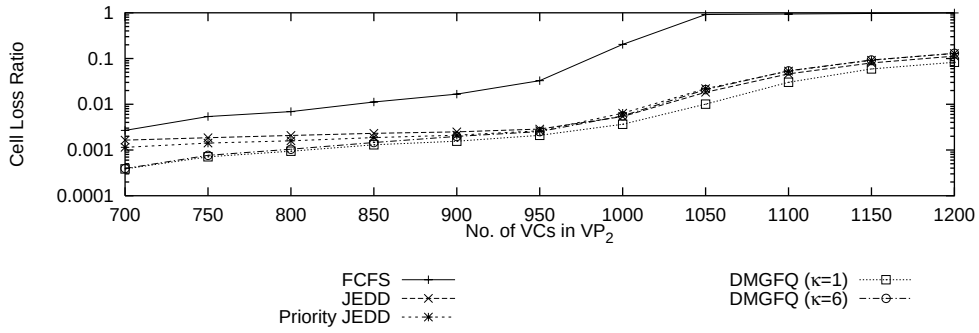
	Average Arrival Rate (Mbps)	Reserved BW M_i (Mbps)	Max. Burst Size B_i (cells)	Residual BW Share ϕ_i	Delay Bound (msec)	Jitter Bound (msec)
VP_1	17.91	18.9	100	0.5	6.5	3.5
VP_2	17.91 $\sim$ 30.74	18.9	100	0.5	6.5	6.5

Table 5.5: Detailed information of VP_1 and VP_2 for Experiment 3.

The simulation results for DMGFQ, *Priority JEDD*, JEDD and FCFS, are all shown in Fig. 5.8. We can observe that if no control mechanism is adopted, the performance of two virtual paths is worse under FCFS than under other scheduling algorithms in this comparison. Hence, in the following discussions, statistics of the FCFS case are not included. For the JEDD algorithm, when the traffic load of



(a) Cell Loss Ratio of VP_1



(b) Cell Loss Ratio of VP_2

Figure 5.8: Cell loss ratios of VP_1 and VP_2 for difference scheduling algorithms.

VP_2 is under its reserved bandwidth, the performance of VP_1 is the best among all scheduling algorithms. This is because the JEDD algorithm does not perform the function of input traffic policing and all eligible cells are scheduled optimally. On the other hand, *Priority JEDD* polices the eligible cells of all VPs. Some eligible cells of VP_1 might violate the traffic profile of VP_1 temporarily and they are assigned to the low-priority queue. Therefore, the performance of VP_1 under *Priority JEDD* is worse than JEDD under light traffic load. For DMGFQ with $\kappa = 1$, the serving sequence is in the order of $Q_\alpha^{(1)}$, $Q_\beta^{(1)}$, $Q_\gamma^{(1)}$, $Q_\alpha^{(2)}, \cdots$, etc. On the other hand, before *Priority JEDD* begins to serve eligible cells in the low-priority queue, it has to serve eligible cells in the high-priority queue first. Hence, the performance of VP_1 for DMGFQ with $\kappa = 1$ is slightly better than *Priority JEDD* under light traffic load. Last but not least, we observe that the performance of DMGFQ $\kappa = 6$ is even better than *Priority JEDD* under heavy traffic load. This is because *Priority JEDD* has to serve high-priority cells first even when the due-dates of low-priority cells are much tighter than high-priority cells. Hence, if the burst of VP_1 are assigned to low priority queue, they have higher probability to be discarded under the *Priority JEDD* algorithm than that under the DMGFQ algorithm with $\kappa = 6$. From the simulation results, DMGFQ is shown to be a good scheduling algorithm to provide RT QoS metrics with the protection of well-behaved traffic streams.

5.4.4 Experiment 4: The Performance of DMGFQ in the Multi-node Environment

In this simulation scenario, we present the simulation experiment of a multiple node environment. The simulation model is shown in Fig. 5.9. VP_0 consists of 300 voice connections via different VCs, and they are assigned nodal delays of 6 ms, 6 ms, and 1 ms at nodes 1, 2, and 3, respectively. The jitter bound of VP_0 is 1 ms. VP_1 to VP_9

serve as competing cross traffic and three of them, VP_1 to VP_3, are well-behaved VPs while other VPs belong to misbehaved traffic streams. Each well-behaved VP also contains 300 VCs and each misbehaved VP consists of 1000 VCs. The nodal delays, the delay bounds and the jitter bounds assigned to cross traffic (VP_1 to VP_9) are all 6.5 ms. The detailed configurations of traffic streams are shown in Table 5.6.

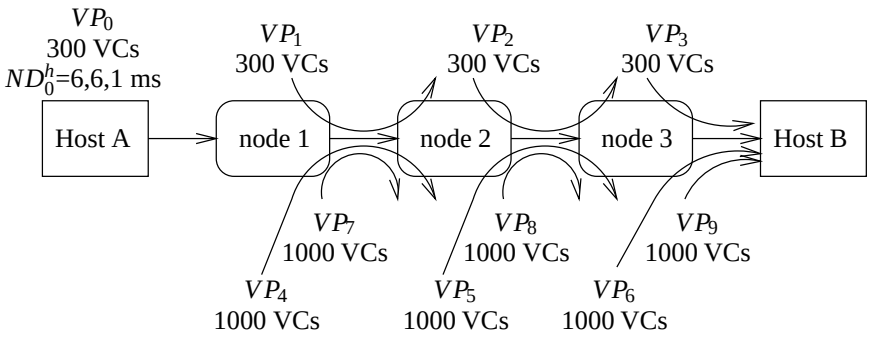


Figure 5.9: Simulation model for Experiment 4.

	Average Arrival Rate (Mbps)	Reserved BW M_i (Mbps)	Max. Burst Size B_i (cells)	Residual BW Share ϕ_i	End-to-end Delay Bound (ms)	Jitter Bound (ms)
VP_0	7.66	9.00	50	0.2	13	1
$VP_1 \sim VP_3$	$7.71 \sim 7.74$	9.00	50	0.2	6.5	6.5
$VP_4 \sim VP_6$	$25.70 \sim 25.74$	9.00	50	0.2	6.5	6.5
$VP_7 \sim VP_9$	$25.70 \sim 25.79$	9.00	50	0.4	6.5	6.5

Table 5.6: Simulation configuration for Experiment 4.

In Table 5.7, we list the simulation results with respect to two different protection

factor (i.e., $\kappa = 1$ and 12). Here, the infinite buffer size is assumed and the cell loss ratio (CLR) only accounts for those cells discarded due to delay/ jitter violations. In addition, R_{α_i}, R_{β_i}, and R_{γ_i} in Table 5.7 denote the average output rates of VP_i contributed by α-, β- and γ-*queue*s, respectively. Because of the operations under different selections of protection factor in the DMGFQ algorithm, a virtual channel might have some small-sequence-number cells in the low-priority queues while large-sequence-number cells in the high-priority queues. Therefore, the cell delivery of a virtual channel may be out of sequence. As we can observe later in this experiment, when compared with the CLRs of misbehaved VPs, the *Ratio of out-of-sequence cells* (ROS) is quite negligible. In addition, for simplicity, we only measure the overall fairness indexes of all switching nodes within the whole simulation interval, $(0, 10^8]$. The simulation results show the relationship between the fairness index and the protection factor. It also can be observed from Table 5.7, column of R_{β_i}, that DMGFQ not only protects well-behaved traffic streams but also allocates the residual bandwidth to backlogged flows ($VP_4 \sim VP_9$) more precisely according to ϕ_i especially for $\kappa = 12$.

In Table 5.7, we observe that the performance of the well-behaved VPs, VP_0 to VP_3, are well protected regardless the bursty cell arrivals from misbehaved VPs. Especially, when κ is set larger, more eligible cells in α-*queue*s are served during one refreshing-period and then cell loss ratios of VP_0 to VP_3 are reduced. Note that individual reserved service rate of VP_i matches the average output rate (i.e., R_{α_i}) from the α-*queue* of the corresponding group exactly. Hence, this result shows the DMGFQ algorithm provides the bandwidth reservation guarantees for individual VPs. As far as the sharing of residual bandwidth is concerned, the simulation results show that the fairness indices ($\mathcal{FRI}$) are degraded when the larger value of protection factor is employed. In other words, the DMGFQ algorithm allocates the

residual bandwidth more fairly when the value of the protection factor κ is larger.

According to Table 5.7, we also find that though there exists out-of-sequence cells, but *ROS* can be controlled to be within a satisfactory range as long as κ is not set exceedingly large. In other words, there is a controllable trade-off between the fairness of the residual bandwidth sharing and the cell level performance.

5.5 Discussions

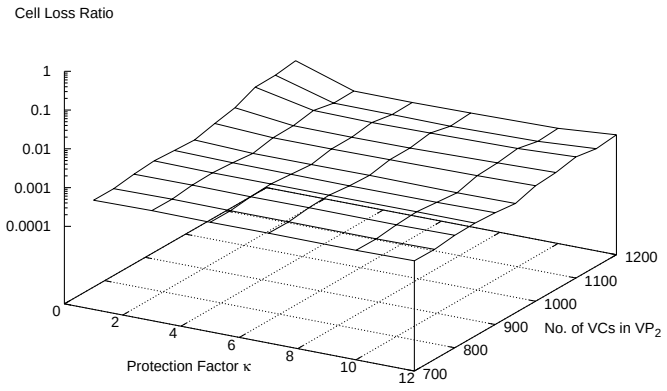
5.5.1 The Influence of the Protection Factor

In this section, we conduct an experiment to investigate the influence of the protection factor (κ) for DMGFQ. The network configuration model is the same as section 5.4.3 (as shown in Fig. 5.7 (a)) and consists of only one switching node and two virtual paths. The delay bounds and jitter bounds of two VPs are all 6.5 ms. The virtual path VP_1 consists of 700 voice streams. VP_2 serves as the competing cross traffic and consists of 700 to 1200 voice streams. We also vary the protection factor κ from 1 to the maximum value 12 to verify the relation among the cell loss ratio, the traffic load and the protection factor. The detailed information of two VPs is shown in Table 5.8.

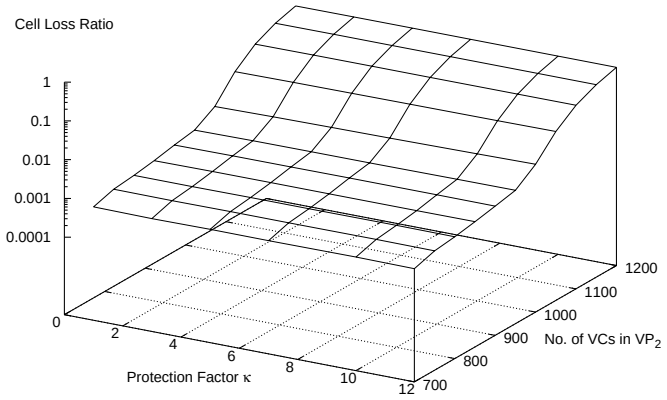
Figure 5.10 illustrates the simulation results regarding the impact on cell loss ratio from protection factor κ. If we adjust the protection factor to the maximum value 12, i.e., the well-behaved traffic are assigned high priority and misbehaved traffic are with low priority, the eligible cells belonging to the α-*queue*s are always transmitted first. Hence, the cell loss ratio of VP_1 increases only slightly even when the traffic load of VP_2 increases significantly. On the other hand, if the protection factor is assigned the minimum value 1, then the service sequence of eligible cells should be in the order of $Q_{\alpha}^{(1)}$, $Q_{\beta}^{(1)}$, $Q_{\gamma}^{(1)}$, $Q_{\alpha}^{(2)}$,..., etc. Therefore, the well-behaved

κ	Node	VP	Avg. Arr. Rate (Mbps)	CLR	R_{α_i} (Mbps)	R_{β_i} (Mbps)	R_{γ_i} (Mbps)	$\mathcal{FRI}$
1	$1 \sim 3$	0	7.66	8.3×10^{-3}	7.65	7.0×10^{-4}	0	0.28
	1	1	7.74	2.4×10^{-4}	7.74	3.5×10^{-3}	0	
		4	25.70	0.48	9.00	4.00	0.29	
		7	25.79	0.37	9.00	7.30	1.2×10^{-3}	
	2	2	7.71	2.3×10^{-4}	7.70	3.7×10^{-3}	0	0.46
		5	25.72	0.47	9.00	4.00	0.58	
		8	25.75	0.38	9.00	7.04	6.1×10^{-3}	
	3	3	7.71	3.0×10^{-4}	7.70	5.0×10^{-3}	0	0.62
		6	25.74	0.46	9.00	3.96	0.88	
		9	25.70	0.39	9.00	6.77	2.6×10^{-2}	
12	$1 \sim 3$	0	7.66	4.1×10^{-4}	7.65	1.5×10^{-3}	0	0.10
	1	1	7.74	6.5×10^{-5}	7.74	4.0×10^{-3}	0	
		4	25.70	0.49	9.00	4.02	2.5×10^{-4}	
		7	25.79	0.36	9.00	7.57	4.5×10^{-7}	
	2	2	7.71	4.8×10^{-5}	7.70	3.7×10^{-3}	0	0.10
		5	25.72	0.43	9.00	4.03	1.3×10^{-4}	
		8	25.75	0.42	9.00	7.60	5.9×10^{-6}	
	3	3	7.71	1.0×10^{-4}	7.70	5.8×10^{-3}	0	0.10
		6	25.74	0.49	9.00	4.03	1.6×10^{-4}	
		9	25.70	0.35	9.00	7.60	4.5×10^{-7}	

Table 5.7: Simulation results for Experiment 4.



(a) Cell Loss Ratio of VP_1



(b) Cell Loss Ratio of VP_2

Figure 5.10: Cell loss ratio for various protection factor and various VP_2 traffic load.

	Average Arrival Rate (Mbps)	Reserved BW M_i (Mbps)	Max. Burst Size B_i (cells)	Residual BW Share ϕ_i
VP_1	17.91	18.90	100	0.5
VP_2	$17.91 \sim 30.74$	18.90	100	0.5

Table 5.8: Detailed characteristics of VP_1 and VP_2.

traffic are affected more seriously by the misbehaved traffic, as illustrated in Fig. 5.10 (a).

5.5.2 The Selection of Traffic Profile Parameters

Ideally the delay bound of a scheduling algorithm should be independent of the number of connections [31, 34], though this goal is difficult to achieve for many schedulers [19]. Such goal is even more difficult to achieve when there are several parameters involved in the traffic profile. For example, the delay bound of one connection will often be influenced by other connections with different burst sizes if their traffic profile parameters consist of burst size and guaranteed minimum bandwidth (i.e., if $(B_i, M_i, 0)$ is used) and a regular FCFS multiplexer is employed. Since there is always a probability that two bursts of different connections arrive at the multiplexing node or a scheduler simultaneously, the delay bound for either stream may often be increased, compared with the single connection case.

Fortunately, the DMGFQ scheduler provides the flexibility to cope with the above issue. Consider the simplest case of employing only M_i as the traffic profile (i.e., with $(0, M_i, 0)$ as traffic profile), and assume the operations of DMGFQ follow the procedures described in Section 5.2.2. Then, the reserved workload defined in the DMGFQ scheme is $M_i t$ instead of $B_i + M_i t$. In this case, delay bounds of all

connections of DMGFQ under this scenario can be calculated and be maintained independently. In a more sophisticated scenario, if $B_i < \frac{\phi_i}{\sum_{j \in B(t)} \phi_j} \left(C - \sum_{j \in B(t)} M_j \right) T$ can be controlled to hold in all refreshing periods, and an appropriate CAC algorithm is applied, we expect the burst size can be accommodated by the residual bandwidth within a refreshing period. This implies that delay bounds of different connections can be maintained to be independent, even under the presence of small bursts. In contrast, many current scheduling disciplines, such as WFQ [31], SCFQ [37], etc., are not equipped with the same flexibility as that of DMGFQ. Because of the flexibility of the DMGFQ algorithm, our scheduling policy can satisfy many versatile traffic environments.

5.6 Concluding Remarks

The proposed DMGFQ scheduling algorithm have been designed for transporting both RT streaming data and best-effort traffic over ATM networks with processing overhead at the same order of RPQ$^+$ [74] in sorting or queue insertion operations, and with the same due-date calculation complexity at the same order of JEDD [21]. We have illustrated that DMGFQ not only can provide a wide range of delay/jitter controls for RT applications, but also can achieve the goal of protecting the users conforming to their service level agreements. From the simulation results, we find that DMGFQ achieves much better performance, in terms of delay, jitter, cell loss ratio, and fairness in residual bandwidth allocations, in comparison with the JEDD algorithm under the presence of misbehaved traffic. Notice that employing the DMGFQ scheme allows a late-activated flow to acquire its pre-determined bandwidth share easily in competition with other early-activated flows immediately. Via the DMGFQ scheme, the residual bandwidth released from other terminated flows is

fairly distributed to other active flows according to their pre-determined weighting factors ϕ_i as soon as such bandwidth is available. Meanwhile, the protection factor κ in DMGFQ provides the flexibility of setting the scheduling performance objective in either optimizing performance or optimizing the fairness. Nevertheless, how to precisely determine QoS parameters from the end-to-end requirement to a nodal requirement, and how to balance between system-wise throughput versus fairness does require further investigation.

To summarize, DMGFQ provides a novel approach to implement efficient traffic schedulers with residual bandwidth sharing, while guarantee minimum bandwidth, delay and jitter constraints, simultaneously. The rise in usage and the popularity of Internet coupled with new multimedia applications has fueled network operator to offer different levels of treatment to users based on their QoS requirements. We believe that this goal is easier to achieve by employing DMGFQ-enabled switches.

Chapter 6

WDFQ: An Efficient Traffic Scheduler with Fair Bandwidth-Sharing for Wireless Multimedia Services

6.1 Introduction

In Chapter 5, we consider how to provide QoS over wireline access networks. However, the wireless network environment is very different from the wireline networks. Hence, in this chapter we focus on how to accommodate fair bandwidth usage and delay guarantees for RT and NRT traffic streams.

In conventional wireline networks, QoS guarantees if implemented are achieved via a combination of resource reservation mechanisms, such as RSVP, and packet scheduling schemes that determine the bandwidth share such as WFQ, etc. However, in a wireless environment, user mobility and wireless channel errors make it

very difficult to perform either resource reservation or fair packet scheduling. Even though resource reservation may be achieved by implementation of some selected resource reservation mechanisms such as Integrated Services [80], while how to provide minimum QoS guarantees, such as delay/jitter guarantees and minimum bandwidth guarantees, for RT multimedia traffic streams, and assure fair bandwidth sharing for NRT regular packet streams via the same scheduling platform has remained largely unaddressed.

Many scheduling algorithms, such as Weighted Fair Queueing (WFQ) [31], Self-Clocked Fair Queueing (SCFQ) [37], Weighted Round-Robin (WRR) [57], etc., have been proposed for regular data communications. However, what these algorithms achieve is to provide weighted bandwidth share while they also targets at low implementation complexity and the improvement in packet delay bound and fairness. In other words, they are not specifically designed to meet the requirements of RT traffic streams, such as jitter guarantees. On the other hand, algorithms such as jitter-Earliest-Due-Date (JEDD) [21], delayed frame queueing (DFQ) [67], dynamic-R&S [77], etc., do accommodate flexible delay and jitter guarantees. The cost of jitter control in their schemes is usually high complexity and they may not protect well-behaved flows against mis-behaving flows. Recently, the new trend in traffic management allows residual bandwidth to be fairly shared by existing flows such as GFR service in ATM traffic management [20]. The DMGFQ scheme in Chapter 5 is a possible scheduling solution for this policy. Note that all these algorithms including DMGFQ assume an error-free channel, or assume that all backlogged flows can be scheduled to transmit. However, in the real world of wireless communications, two key characteristics have made the conventional wireline-based scheduling algorithms inapplicable: one is the bursty channel errors, and the other is location-dependent channel capacity and errors [32, 33]. Therefore, even though packets of

a well-behaved flow may be successfully scheduled to transmit, it is still difficult to guarantee its QoS due to possible channel errors, which may dynamically change as time evolves.

A representative of the scheduling algorithms that specifically design for wireless access and to handle location-dependent error bursts is the idealized Wireless Fair-Queueing (IWFQ) algorithm proposed by Lu, Bharghavan and Srikant [32, 33]. The key rationale of IWFQ is to allow *lagging* flows to make up their *lag* by causing *leading* flows to give up their *lead*. However, IWFQ does not consider the delay/jitter requirements in wireless multimedia applications. In addition, the guarantees for throughput and delay in IWFQ are tightly coupled. In many scenarios, especially multimedia applications, the decoupling of delay from bandwidth might be a more attractive approach [52]. The Channel-condition Independent packet Fair Queueing (CIF-Q) algorithm [51], proposed by Ng, Stoica and Zhang, uses Start-time Fair Queueing (SFQ) [55] as its *reference* system. It provides long-term fairness via an additional parameter (called *lag*) and ensures delay and throughput guarantees for error-free flows via the concept of *forced compensation*. However, its implementation complexity ($O(n \log n)$) is still too high for a cost-effective implementation. In addition, the decoupling requirement of delay from bandwidth may not be achieved by CIF-Q. Fragouli *et al.* proposed an enhanced Class-Based-Queueing (enhanced CBQ) scheme [54], which is composed of a modified version of CBQ [53] and Channel-State Dependent Packet Scheduling (CSDPS) [49], to transport multimedia traffic across wireless networks. However, flexible delay/jitter guarantee is still not a direct QoS target of enhanced CBQ. Another important related design is the effort-limited fair (ELF) scheduling algorithm proposed by Eckhardt and Steenkiste [50]. ELF extends WFQ via dynamic weight adjustments and guarantees determined according to the error rate experienced by the flow. Again, ELF

only guarantees the throughput of a flow while the flexible delay/jitter bounds are not accommodated. Hence, when fairness, differentiated QoS in delay/jitter, and link utilization are all taken into consideration, it is necessary to redesign a new scheduling algorithm for wireless multimedia.

In this chapter, we propose a novel scheduling algorithm for wireless access networks, called the *Wireless Differentiated Fair Queueing* (WDFQ) algorithm [81], to accommodate delay/jitter controls and fair residual bandwidth sharing for RT and NRT traffic streams simultaneously. The goals of WDFQ are: 1) timely delivery of delay/jitter constrained RT traffic with controlled packet losses; 2) virtually error-free transmission of NRT traffic if buffer is sufficient; 3) shared utilization of the residual bandwidth for both RT and NRT traffic streams; 4) fair usage of channel bandwidth among NRT streams on different remote stations. In addition, the implementation of such traffic scheduler shall do not require the use of sorter circuit and thus involve very limited complexity issues. The organization of this chapter is as follows. In section 6.2, the proposed traffic scheduler with fair residual bandwidth sharing is presented. Simulation results of RT and NRT traffic under various scenarios are shown in section 6.3. Our concluding remarks are drawn in section 6.5.

6.2 Wireless Differentiated Fair Queueing Discipline

In order to achieve the above mentioned design goals, the following design rules are considered in WDFQ: 1) the channel is fairly accessed by all backlogged flows; 2) the location-dependent channel error property is used to accommodate only short error bursts; 3) a slotted channel is assumed and the scheduler maintains a *credits*

system to compensate or to render back slots, up to a threshold, due to various error conditions in different channels; 4) since errors are bursty, flows which are under error-prone channels should not be granted high opportunities to transmit; instead the time slots allocated to error-prone flows, when the channel is found to be under errors, should be redistributed to flows with error-free channels; and 5) the scheduling algorithm should be simple such that all computations can be done timely.

6.2.1 Minimum Bandwidth Guarantee for RT and NRT Traffic Streams

In Chapter 5, we have verified the features of DMGFQ scheme in its capability of delay/jitter guarantee and fair bandwidth sharing. The concept of minimum bandwidth guarantees and residual bandwidth sharing is still adopted in WDFQ. Here, we employ the concept of credit of bandwidth to compensate the loss of bandwidth due to channel errors. Since the conventional rate control mechanism may be difficult to be realized, we suggest the adoption of the service workload instead of the service rate. In order to reduce complexity, on a period-by-period basis we employ certain counters to determine whether a flow has overused its reserved bandwidth, and then directly determines their granted service workload for the whole period. Before describing the calculation procedures to determine the order of packet transmissions, we define the following notations. Note that underlying layer-2 PDU is assumed to be fixed and is called the *air packet* for simplicity in the following context. The time unit in WDFQ is "slot" which is the time interval to transmit an air packet.

- (B_i, M_i, ϕ_i): the traffic profile[1] of flow i used in the service level agreement with respect to the air interface, consisting of the maximum burst size B_i, guaranteed minimum bandwidth M_i and the share weighting factor of residual bandwidth ϕ_i. If the arrived traffic load of flow i within $(0, t]$ is denoted as $N_i(t)$, then the service workload $\min\{N_i(t), M_i t + B_i\}$ is guaranteed for transmission within its delay bound and jitter bound, regardless of residual bandwidth. The residual bandwidth is then allocated to flow i according to ϕ_i, period by period;

- T: the length of a refreshing period, which is the period for service workload calculation;

- t_n: the starting epoch of n-th refreshing period, and $t_n = t_{n-1} + T$ for $n \geq 1$;

- $W_i^r(t_n, t_{n+1})$: the *reserved workload* for a backlogged flow i within the n-th refreshing period, which is the service workload satisfying the traffic profile of flow i;

- $W_i^e(t_n, t_{n+1})$: the *extended workload* for a backlogged flow i within the n-th refreshing period, which is the workload contributed by the residual bandwidth observed within the n-th refreshing period;

- $W_i(t_n, t_{n+1})$: the *total granted workload* for a backlogged flow i within the n-th refreshing period, which is the sum of $W_i^r(t_n, t_{n+1})$ and $W_i^e(t_n, t_{n+1})$;

- $W_i^E(t_n, t_{n+1})$: the *extra workload* for a backlogged flow i, which is contributed by the total granted workloads of the flows under bad channel states at the starting epoch of the refreshing period;

[1]Note that when a leaky-bucket policer[79] is used, M_i is equivalent to the *average rate* in the leaky-bucket policing algorithm, and B_i is equivalent to the *bucket depth* in the policer.

- $\psi_i(t_n, t_{n+1})$: the number of eligible air packets of flow i within the n-th refreshing period.

For convenience, $W_i^r(t_n, t_{n+1})$, $W_i^e(t_n, t_{n+1})$, $W_i^E(t_n, t_{n+1})$ and $W_i(t_n, t_{n+1})$ are all normalized by the size of a single air packet.

When the n-th refreshing period starts and the integrity of an air packet is taken into considerations, the reserved workload of a backlogged flow i can be calculated via the following recursive equation at t_n:

$$W_i^r(t_n, t_{n+1}) = \min\left\{ \lfloor M_i T + B_i \rfloor, \left\lfloor (n+1) M_i T + B_i - \sum_{j=0}^{n-1} \min\{W_i^r(t_j, t_{j+1}), \psi_i(t_j, t_{j+1})\} \right\rfloor \right\}, \quad n \geq 1. \tag{6.1}$$

It can be shown that eq. (6.1) is equivalent to the result of the leaky-bucket policing algorithm in [79]. However, the actual number of eligible air packets of flow i, $\psi_i(t_n, t_{n+1})$, may be less than the reserved workload $W_i^r(t_n, t_{n+1})$. In order to allocate bandwidth more fairly, these residual time slots should be shared by other backlogged flows. Therefore, the extended workload of a backlogged flow i at t_n is:

$$W_i^e(t_n, t_{n+1}) = \left\lfloor \frac{\phi_i}{\sum_{j \in B(t_n)} \phi_j} \left(C \cdot T - \sum_{j \in B(t_n)} \min\{W_j^r(t_n, t_{n+1}), \psi_j(t_n, t_{n+1})\} \right) \right\rfloor, \tag{6.2}$$

where $B(t_n)$ is the set of backlogged flows at time t_n.

However, if flow j is found to be with the bad channel state at t_n, the workload it granted, $W_j^r(t_n, t_{n+1})$ and $W_j^e(t_n, t_{n+1})$, should be distributed fairly to backlogged flows under good channel states, in order to achieve higher link utilization. Hence, backlogged flow i with good channel state can obtain extra workload, $W_i^E(t_n, t_{n+1})$,

which is expressed as

$$W_i^E(t_n, t_{n+1}) = \left\lfloor \frac{\phi_i}{\sum_{\substack{j \in B(t_n) \\ j \in G(t_n)}} \phi_j} \sum_{j \in E(t_n)} \left(W_j^r(t_n, t_{n+1}) + W_j^e(t_n, t_{n+1}) \right) \right\rfloor, \tag{6.3}$$

where $G(t_n)$ and $E(t_n)$ are the sets of flows under good channel state and under bad channel state at time t_n, respectively.

As concerns the *fairness index* within the interval $(s,t]$, $\mathcal{FRI}(s,t)$, conventional scheduling algorithms compare the granted service rate among all backlogged flows as the fairness index. However, it is meaningless to compare the granted service rates of two backlogged flows when any of them is with bad channel state. Hence, we redefine the fairness index as

$$\mathcal{FRI}(s,t) = \max_{i,j \in G(s,t)} \left| \frac{\frac{R_i(s,t) - M_i}{C - \sum_{\forall k} \min(A_k(s,t), M_k)}}{\phi_i} - \frac{\frac{R_j(s,t) - M_j}{C - \sum_{\forall k} \min(A_k(s,t), M_k)}}{\phi_j} \right|, \quad 0 \le s \le t, \tag{6.4}$$

where flow i and flow j are any continuously backlogged flows and $G(s,t)$ is the set of flows with good channel state within the interval $(s,t]$. $A_i(s,t)$ and $R_i(s,t)$ are the average arrival rate and the average output rate of flow i within the interval $(s,t]$, respectively. Based on the definition provided in eq. (6.4), a smaller fairness index of a scheduling algorithm implies a better fairness level this algorithm can achieve.

6.2.2 Queueing Model of the WDFQ Algorithm

In this chapter, we assume important mechanisms in wireless networks, such as *acknowledgement*, *channel state detection*, etc., are fully supported by the underlying MAC protocol. The queueing model to illustrate the WDFQ algorithm is shown in Fig. 6.1. In this WDFQ, both RT multimedia services and regular data services are

supported. In addition, we assume WDFQ accommodates two classes of services for RT multimedia applications: one is the *premium* RT multimedia service and the other is the *regular* RT multimedia service. The premium multimedia service has the requirements of tight delay/jitter constraint and higher weight in residual bandwidth sharing while the regular service has loose delay/jitter constraint and less weight for residual bandwidth sharing. In this chapter, we follow the definition of *jitter* described in [21], where the jitter of a flow (or a connection) is defined by the maximum absolute difference in the delays experienced by any two packets on that flow. Usually, NRT traffic also can be divided into two classes: the *premium* NRT service and the *regular* NRT service. The premium NRT service is allowed to use residual bandwidth share to provide better QoS but the regular service cannot. However, only the parameters regarding residual bandwidth share are adjusted then these two NRT services can be provided via the same queueing model. Hence, in the following description we do not distinguish NRT traffic between premium and regular services specifically. The mapping of classes of services and queueing groups are listed in Table 6.1.

Class of Services	Queueing Group	Residual BW Share ϕ_i	Jitter Bound	Delay Bound
Premium RT	Group RT-1	≥ 0	T	$\geq T$
Regular RT	Group RT-2	≥ 0	NT	$\geq NT$
Premium NRT	Group NRT	> 0	**N.A.**	**N.A.**
Regular NRT	Group NRT	$= 0$	**N.A.**	**N.A.**

Table 6.1: The mapping of classes of services and queueing groups, where **N.A.** stands for "not available.".

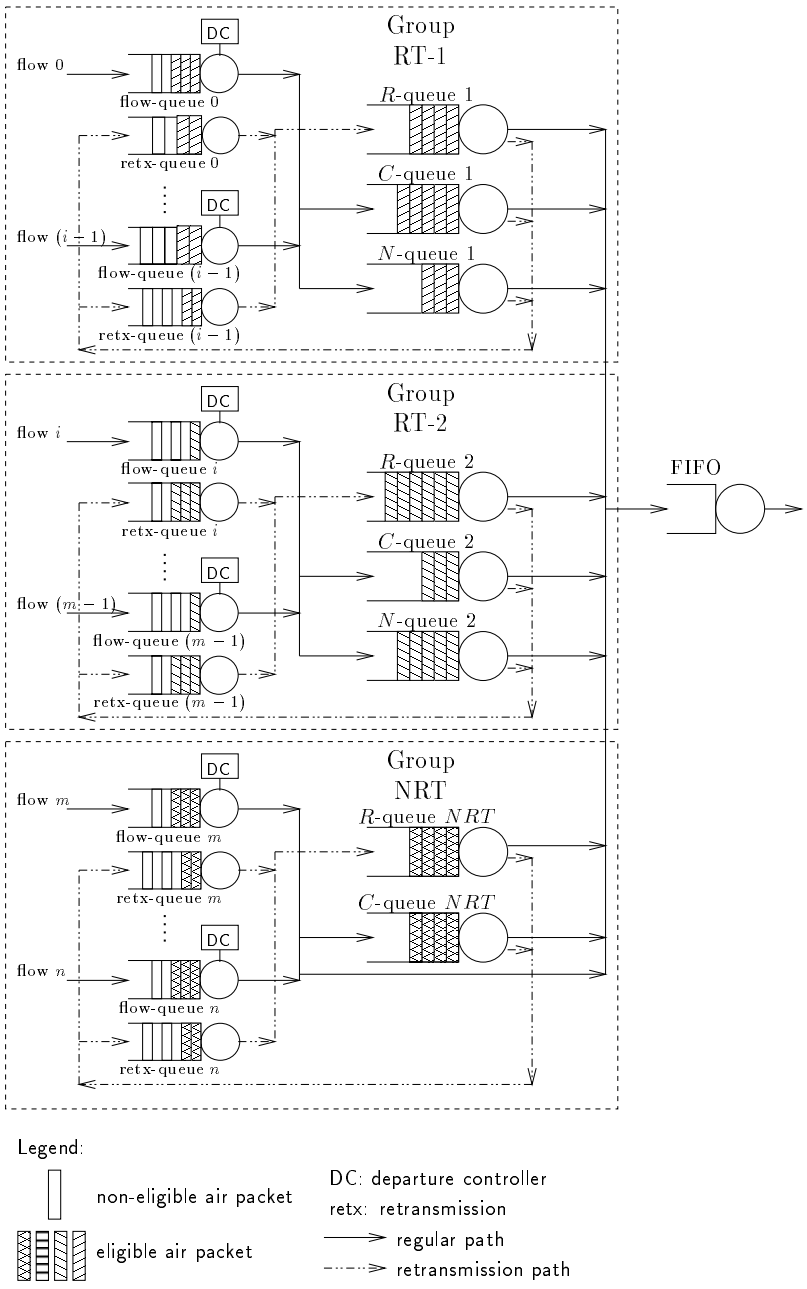


Figure 6.1: The queueing model of the WDFQ algorithm.

In WDFQ, each flow is associated with a class of service with a set of predetermined air-packet level QoS parameters, including delay/jitter, packet loss ratio, and residual bandwidth share, etc[2]. We also know that the length of refreshing period is just necessary to satisfy the requirement of the minimum jitter bound to reduce operation complexity. Therefore, the jitter bounds of all flows in the Group 1 are T slot times while the jitter bound of Group 2 is NT slot times. In this model, non-real-time (NRT) traffic is assigned a special dedicated group, called Group NRT. The Head-of-Line (HOL) packet of a flow queue is called an eligible packet if it can be transmitted immediately without violating its delay bound and packet jitter constraint. Each flow i is assigned two dedicated FIFO queues, called *flow-queue* i and *retx-queue* i. The function of *flow-queue* i is to buffer the arriving air packets until they become eligibility, while *retx-queue* i buffers the eligible air packets whose channel states observed by the scheduler are under the "error" state now.

As far as RT traffic is concerned, three FIFO queues are required at the output port, called *C-queue*, *N-queue* and *R-queue*, which are dedicated for two RT groups to provide different services in the WDFQ algorithm, where C, N and R stand for *conforming*, *nonconforming* and *retransmission*, respectively. The function of *C-queues* of Group i, denoted as $Q_C^{(i)}$, is to buffer the eligible air packets belonging to the reserved workload and extended workload, which is the workload beyond the reserved workload and being served with current residual bandwidth, $C - \sum_{j \in B(t_n)} M_j$, of the corresponding flows in Group i. In other words, $Q_C^{(i)}$ buffers the air packets conforming to their service level agreements and this is why we call it C-queue. On the other hand, the *N-queue* of Group i, denoted as $Q_N^{(i)}$, buffers

[2]For NRT traffic, its delay and jitter limits are assigned infinite. Equivalently, they are listed as **N.A.** in Table 6.1.

the eligible air packets exceeding the service level agreements, including the reserved workload and extended workload, of the corresponding flows. In other words, the *nonconforming* air packets are accommodated. In turn, the *R-queue* of Group i (denoted as $Q_R^{(i)}$) buffers the eligible air packets whose flows encountered bad channel states previously and are ready to retransmit. The function and operation of $Q_R^{(i)}$ will be described in details later. When DCs of Group i open their gates with period T, eligible air packets belonging to Group i are moved to the output queue following a specified priority. The delay bound and jitter bound of each RT flow hence has to be ceiled as an integer multiple of T.

On the other hand, for NRT traffic, only two extra FIFO queues, $Q_C^{(NRT)}$ and $Q_R^{(NRT)}$, are needed. Nonconforming air packets of NRT traffic are still buffered in the corresponding flow queues. Hence, the N-queue is not necessary in Group NRT. This special mechanism for NRT traffic is to assure the air packet sequence integrity of each flow. Each time when DCs of Group NRT opens their gates, the air packets whose total traffic load is within the *reserved workload* and the *extended workload* of each NRT flow are then moved to queue $Q_C^{(NRT)}$, before their transmission.

In order to support nontrivial delay and jitter control for multimedia traffic associated with each multimedia air packet, additional control parameters are required in the WDFQ operations. Each time when an air packet P_i^k, the k-th air packet of flow i, arrives, its *eligible time* (ET_i^k) are calculated as follows.

$$\begin{cases} ET_i^k &= AT_i^k + ND_i^k - J_i, & \text{for RT flows,} \\ ET_i^k &= AT_i^k, & \text{for NRT flows,} \end{cases} \tag{6.5}$$

where ND_i is the nodal delay bound of flow i, J_i is the jitter bound requested by flow i, $J_i = T$ or NT, and AT_i^k is the arrival time of air packet P_i^k. Usually $ND_i > J_i$ holds. Based on the information and the derived ET, WDFQ can schedule the air packets properly so as to provide both delay and jitter guarantees. Due-date calcula-

tion operations are unnecessary for NRT traffic. Note that for RT multimedia traffic the retransmission time due to channel error is involved in the nodal delay bound. In other words, RT traffic can always perform retransmission before its transmission deadline expires. Although in practice, an upper bound of retransmission times for NRT traffic may be specified, and the actual nodal delay of NRT traffic is not infinite, for the convenience of discussion we still assume NRT traffic can be always retransmitted until successful transmission.

6.2.3 Operations of the WDFQ Algorithm

Before describing the detailed operations of the WDFQ algorithm, we briefly depict how WDFQ deal with the retransmission mechanism under *bad* channel states.

Suppose that an air packet belonging to flow i is picked from certain *C-queue* or *R*-queue to transmit when the current channel state is bad. Then, this air packet is moved back to *retx-queue* i and the retransmission timer of flow i with period T_i^r is started. Before the retransmission timer expires, all eligible air packets belonging to flow i are moved to *retx-queue* i. Therefore, it is not necessary for the scheduler to check the channel state slot by slot. Once the retransmission timer expires, the channel state of flow i is updated and the eligible air packets in *retx-queue* i are then moved to the corresponding *R-queues*. Air packets in *R*-queues are the eligible and conforming air packets that were not be served due to the bad channel states in previous refreshing periods. Hence, the *R-queue* is assigned highest service priority in a group to compensate their loss in bandwidth share. Last but not least, we have to note that the eligible air packets in *N-queues* are the air packets violating the traffic contracts. Thus, air packets in *N*-queues are of the lowest priority and may be subjected to packet discarding if conforming flow must be protected.

In the following, we first describe the operations of WDFQ when channel error

is not considered. At each starting epoch of the refreshing period t_n, the reserved workload and the extended workload of each flow are calculated. Next, all DCs belonging to RT groups open their gates and move the eligible air packets from each flow queue to the corresponding *C-queues* and *N-queues* according to their reserved workloads and the extended workloads. The eligible air packets conforming to traffic contracts are moved to *C-queues* while nonconforming air packets are moved to *N-queues*. For NRT traffic, once DCs open their gates, conforming NRT packets are moved to Q_C^{NRT} while nonconforming packets are buffered in the flow queues. The operation of WDFQ with channel error is considered as an extension of the above procedure which includes the operations in *C-queues*, *N-queues* and *R-queues*.

To summarize the operation of WDFQ, the eligible packets are moved around C-queues, R-queues and N-queues and serviced in the sequence of $Q_R^{(1)}$, $Q_C^{(1)}$, $Q_R^{(2)}$, $Q_C^{(2)}$, $Q_R^{(NRT)}$, $Q_C^{(NRT)}$, $Q_N^{(1)}$, $Q_N^{(2)}$. Via this service sequence, the delay/jitter requirements and the residual bandwidth shares can be accommodated simultaneously to RT multimedia streams.

According to the operations of the WDFQ algorithm, we know that the transmission delay of flow i (TD_i) is

$$ND_i - J_i \leq TD_i \leq ND_i + T. \tag{6.6}$$

Therefore, the actual jitter bound provided to flow i is $J_i + T$. Since $J_i = T$ or NT, the actual upper limit of jitter variation is either $2T$ or $(N+1)T$, $N \geq 2$, depending on the class of service.

Based on the required queueing and scheduling operations, the sorting overhead of WDFQ is only at the same order of RPQ$^+$ ($O(1)$) [74] and thus does not require a sorter circuit, while the due-date calculation is at the same order of JEDD [21]. Therefore, WDFQ accommodates both RT streaming data and best-effort traffic with fair residual bandwidth share at the cost of low processing overhead.

6.2.4 Credit Calculation

As mentioned above, two major goals of WDFQ are utilizing the location-dependent channel error property and compensating or to rendering back slots due to different channel states to achieve high fairness. Therefore, we introduce the concept of "credit" to manage which flow should render back its reserved time slots to compensate the time slots it overused before and which flow should be granted extra time slots to compensate the loss of bandwidth it could not use before due to channel errors. In this section, we describe the credit calculation procedures. At first we introduce some notations as follows.

- channel period: the period from the epoch of a good channel state is detected to the epoch that the channel state is detected to be back to *GOOD* state from *BAD* state, as shown in Fig. 6.2. The channel state is said in BAD state if the air packet failed to be transmitted successfully, while the channel state is said to be in GOOD state if the air packet can be transmitted successfully. In addition, we assume that if the channel state is detected to be in the BAD state, the packet transmission will be failed with probability 1. The condition that MAC layer may use the approach of adaptive error coding to improve successful transmission is not considered in this chapter.

- ΔCrd_i: the change of the credit of flow i while an air packet belonging to flow i is processed during the channel period of flow i;

- Crd_i^L: credit limit of flow i, $Crd_i^L \geq 0$;

- Crd^T: total time slots which can not be transmitted successfully for all conforming flows due to BAD channel state and should be compensated in the future.

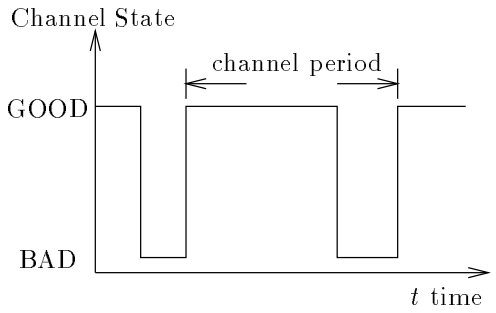


Figure 6.2: Illustration of the *channel period*.

Suppose that current time is t which is within n-th refreshing period, i.e., $t_n \leq t < t_{n+1}$. An air packet p belonging to flow i was selected for transmission. Then, the credit of flow i during the channel period is calculated as follows.

(A) If the channel state of flow i is detected to be *BAD* and the transmission has failed:

(1) *If p is from a C-*queue *or an R-*queue *then*

$$\Delta Crd_i = \min\left\{Crd_i^L - Crd_i, \min\{t_{n+1} - t - Crd^T, 1\}\right\}, \quad (6.7)$$

$$Crd_i \leftarrow Crd_i + \Delta Crd_i, \quad (6.8)$$

$$Crd^T \leftarrow Crd^T + \Delta Crd_i. \quad (6.9)$$

Next, the air packet p is moved back to retx-queue i *and the retransmission timer of flow i is activated with expired period T_i^r.*

(2) *If p is from an N-*queue*, then discard air packet p.*

The rationale of credit calculation in step (1) can be explained as follows. Because the air packet p is within the reserved workload and extended workload, we should reserve the time slot for it even if the channel state of flow i is BAD.

In order to maintain the fairness, these time slots should be compensated in the future. In step (2), we discard p because flow i has overused its specified bandwidth and air packet p should not be compensated in the future.

(B) If the channel state of flow i is detected to be *GOOD* and the transmission has succeeded:

(1) *If p is from a C-*queue*, then keep Crd_i and Crd^T unchanged*;

(2) *If p is from an R-*queue*, then*

$$\Delta Crd_i = -1 + I(-Crd_i^L, Crd_i), \tag{6.10}$$

$$Crd_i \leftarrow Crd_i + \Delta Crd_i, \tag{6.11}$$

$$Crd^T \leftarrow Crd^T + \Delta Crd_i, \tag{6.12}$$

where $I(-Crd_i^L, Crd_i)$ is an indicator function with definition

$$I(-Crd_i^L, Crd_i) = \begin{cases} 0, & Crd_i \leq -Crd_i^L, \\ 1, & otherwise. \end{cases} \tag{6.13}$$

(3) *If p is from an N-*queue *and $(t_{n+1} - t) \leq Crd^T$, then*

$$\Delta Crd_i = -1 + I(-Crd_i^L, Crd_i), \tag{6.14}$$

$$Crd_i \leftarrow Crd_i + \Delta Crd_i, \tag{6.15}$$

and keep Crd^T unchanged, where $I(-Crd_i^L, Crd_i)$ is an indicator function with definition as eq. (6.13).

(4) *If p is from an N-*queue *and $(t_{n+1} - t) > Crd^T$, then keep Crd_i and Crd^T unchanged.*

As shown in step (2), if an air packet is served from an *R-queue*, it represents a time slot reserved for the corresponding flow has been given back right

now. Therefore, the credit should be decrease. Note that the relationship $(t_{n+1} - t) \leq Crd^T$ shown in step (3) represents the remaining time slots in the n-th refreshing period cannot satisfy the requirement of reserved time slots due to BAD channel state. Therefore, if an air packet belonging to an *N-queue* uses the time slot, the corresponding flow should reimburse other flows with the same amount of bandwidth to maintain fairness. In contrary, the relationship $(t_{n+1} - t) > Crd^T$ shown in step (4) represents the remaining time slots in the n-th refreshing period can accommodate some residual bandwidth to serve misbehaved traffic. Therefore, the credit of flow i is unchanged.

(C) If the channel state of flow i is detected to be *GOOD* and the transmission has failed, then the channel state of flow i is updated as *BAD* and the credit calculation follows the operations described in Case A.

(D) If the channel state of flow i is detected to be *BAD* and the transmission has succeeded, then the channel state of flow i is updated as *GOOD* and the operations listed in Case B are applied.

When a channel period ends, the accumulated credit of flow i, Crd_i, is added to its reserved workload, $W_i^r(t_{n+1}, t_{n+2})$, if the ending epoch of the channel period of flow i is within the interval $[t_n, t_{n+1})$. Next, Crd_i should be reset by the scheduler, and Crd^T is updated as

$$Crd^T \leftarrow \max\{0, Crd^T - \max\{0, Crd_i\}\}. \tag{6.16}$$

In order to more precisely describe the operation of the WDFQ algorithm, especially the credit calculation procedure, we present the pseudo code for an implementation of the WDFQ scheduler in Fig. 6.3 to Fig. 6.5. The operational parameter Δ for flexible hardware implementation is also included in the pseudo code. Note

that these codes can be further optimized for efficient execution, such as in parallel mode. We prefer this version because it is easier to read.

6.3 Simulation Results

In this section, we evaluate the performance of the WDFQ scheme for RT and NRT traffic streams. The examined performance metrics include the packet loss ratio, the bandwidth usage and Fairness Index. Note that the packet loss ratio only accounts for those packets discarded due to delay or jitter violations. Because the buffer size is assumed infinite, no packet losses are due to buffer overflow.

Here, we assume the wireless channel follows IEEE 802.11[82] with link bandwidth is 10 Mbps at MAC layer. According to [82], we assume the overhead of MAC layer is 30 bytes in all simulation experiments. In addition, we assume the payload of the air packet is 128 bytes, which includes RTP header 12 bytes, UDP header 8 bytes, IP header 20 bytes and video frame data 88 bytes. Each video stream is a replay of "Star Wars" MPEG-1 video trace obtained from University Wuerzburg[76], with equally separated starting points within the 39996 frame positions. Since the frame rate is 24 frames/sec, each stream is equivalent to a video of the length 1666.5 seconds. As for the statistical information of the video script trace are shown in Table 6.2. In order to avoid man-made simultaneous arrivals of air packet bursts at the multiplexer, the starting epoch of each video stream is uniformly distributed over the 1 sec interval. The refreshing period is set to be 1.0 ms and all simulations lasts for 5×10^7 time slots. We will show various of target delay constraints and jitter constraints can be supported easily. We have to note that as we mention the "bandwidth" or arrival/departure "rate," the protocol overheads from RTP layer to MAC layer are included.

```
main()
{
  while (1){
    t = system time, and t_n ≤ t < t_{n+1};
    if (cell_arrived==TRUE){
      p = arrived cell;
      if (p belongs to RT traffic)
        calculate the due-date of cell p;
      place cell p in its flow queue;
    }
    if (retransmission timer expire == TRUE){
      if (channel state of flow i is GOOD){
        move cells in flow-queue-prime i to corresponding R-queues;
        W_i^r(t_{n+1}, t_{n+2}) ← W_i^r(t_{n+1}, t_{n+2}) + Crd_i;
        Crd_i ← 0;
        Crd^T ← max{0, Crd^T − max{0, Crd_i}};
      }
      else
        restart retransmission time with expired period T_i^r;
    }
    extract cell p with sequence Q_R^1, Q_C^1, Q_R^2, Q_C^2, Q_R^Ω, Q_C^Ω, Q_N^1, Q_N^2;
    probe channel state of p;
    if ((p != NULL)&&(corresponding channel state is GOOD)){
      if (p belongs to RT traffic)
        update due-date of cell p;
      extract cell p from output buffer and transmit it;
      credit_calculation_procedure(t,"GOOD");
    }
    else if ((p != NULL)&&(corresponding channel state is BAD)){
      move p back to flow-queue-prime i;
      credit_calculation_procedure(t,"BAD");
      start retransmission timer of flow i with period T_i^r;
    }
    if (refreshing_event_occurs==TRUE){
      discard all cells in the output buffer;
      discard all cells in queue Q_C^(1), Q_N^(1), and Q_R^(1);
      refreshing_procedure(t);
      schedule next refreshing_event at time t + T;
      t_{n+1} ← t + T;
      reset output process;
    }
  }
}
```

Figure 6.3: Pseudo code of the WDFQ algorithm.

refreshing_procedure(t)
{
  for (*each backlogged flow* i){

$$W_i^r(t,t+T) = \min\left\{ \lfloor M_i T + B_i \rfloor, \left\lfloor (n+1)M_i T + B_i - \sum_{j=0}^{n-1} \min\{W_i^r(t_j,t_{j+1}), \psi_i(t_j,t_{j+1})\} \right\rfloor \right\};$$

$$W_i^e(t,t+T) = \left\lfloor \frac{\phi_i}{\sum_{j\in B(t)} \phi_j} \left(CT - \sum_{j\in B(t)} \min\{W_j^r(t,t+T), \psi_j(t,t+T)\} \right) \right\rfloor;$$

$$W_i^E(t,t+T) = \left\lfloor \frac{\phi_i}{\sum_{\substack{j\in B(t) \\ j\in G(t)}} \phi_j} \sum_{j\in E(t)} \left(W_j^r(t,t+T) + W_j^e(t,t+T) \right) \right\rfloor;$$

    if $(\psi_i(t,t+T) \le W_i^r(t,t+T) + W_i^e(t,t+T) + W_i^E(t,t+T))$
      move $(\psi_i(t,t+T))$ eligible cells of flow i to corresponding C-*queue*s;
    else{
      move $(W_i^r(t,t+T) + W_i^e(t,t+T) + W_i^E(t,t+T))$ eligible cells of flow i to corresponding C-*queue*s;
      move residual $(\psi_i(t,t+T) - W_i^r(t,t+T) - W_i^e(t,t+T) - W_i^E(t,t+T))$ eligible cells of flow i to corresponding N-*queue*s;
    }
  }
}

Figure 6.4: Pseudo code of the refreshing procedure in WDFQ.

```
credit_calculation_procedure(t, channel_state)
{
  if (channel_state == GOOD){
    if (p is extracted from C-queues)
      keep Crd_i and Crd^T unchanged;
    else if (p is extracted from R-queues){
      ΔCrd_i ← −1 + I(−Crd_i^L, Crd_i);
      Crd_i ← Crd_i + ΔCrd_i;
      Crd^T ← Crd^T + ΔCrd_i;
    }
    else if (p is extracted from N-queues){
      if ((t_{n+1} − t) ≤ Crd^T){
        ΔCrd_i ← −1 + I(−Crd_i^L, Crd_i);
        Crd_i ← Crd_i + ΔCrd_i;
        keep Crd^T unchanged;
      }
      else{
        keep Crd_i unchanged;
        keep Crd^T unchanged;
      }
    }
  }
  else{
    ΔCrd_i ← min {Crd_i^L − Crd_i, min{t_{n+1} − t − Crd^T, 1}};
    Crd_i ← Crd_i + ΔCrd_i;
    Crd^T ← Crd^T + ΔCrd_i;
  }
}
```

Figure 6.5: Pseudo code of the credit calculation procedure in WDFQ.

	I-Frame	P-Frame	B-Frame
Mean frame length (air packets)	279.01	64.61	30.04
Variance of frame length (air packet^2)	8000.22	3651.21	729.25
Maximum frame length (air packets)	790	429	330
Overall mean rate	1.802 Mbps		

Table 6.2: The general information of the MPEG video trace in simulations. All statics have involved RTP, UDP and IP layer overheads.

The error characteristic of the wireless channel is modeled by a 2-state Markov chain as shown in Fig. 6.6. The channels of any two different remote stations are assumed to be independent. A transition between the GOOD and BAD states can happen at channel slot boundaries with the transition probability $P_{G,B}$ ($P_{B,G}$) from the GOOD to BAD states (from BAD to GOOD states). Hence, the average time duration T_G (T_B) at the GOOD (BAD) state is $1/P_{G,B}$ ($1/P_{B,G}$). If the channel state changes from GOOD state to BAD state suddenly during the air packet transmission period, the packet is received in error. The packet is received correctly otherwise. Every air packet received in error is assumed to be detected by the decoder.

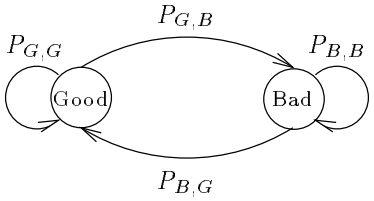


Figure 6.6: Two-state Markov chain modeled channel.

In addition, we adopt the *Priority FIFO* algorithm as the baseline comparison, whose queueing model is shown in Fig. 6.7. The regulators, which serve as the front end packet processor, perform nothing except forwarding packets conforming the traffic profile (B_i, M_i) to the high-priority output queue and forwarding nonconforming packets to the low-priority output queue, where B_i is the maximum burst size and M_i is the guaranteed minimum bandwidth.

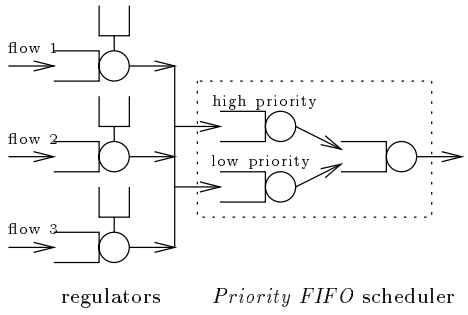


Figure 6.7: The queueing model of the *Priority FIFO* algorithm.

6.3.1 Experiment 1: Integrated Services with RT Traffic Streams and NRT Traffic Streams

In this simulation scenario (see Fig. 6.8), we examine the transient behavior of the WDFQ algorithm for NRT traffic streams. Two CBR flows (flow 1 and flow 2) are employed to model the RT traffic streams and their configurations are shown in Table 6.3. Flow 3, serving as the background traffic, carries the NRT traffic stream which is a replay of LAN traffic trace obtained from the Lawrence Berkeley National Laboratory[63]. In total, the original trace is used to generate the traffic with average load equal to 7.2 Mbps. The output link bandwidth is also assumed 10 Mbps. We assume that the average time duration T_G (T_B) at the GOOD (BAD) state is 10000 (1000) time slots. The retransmission periods of three flows are all 10 time slots and credit limits are 20 (air packets). In order to observe the behavior of bandwidth sharing more clearly, we set the average arrival rate of Flow 2 and Flow 3 much higher than their guaranteed minimum bandwidth M_i, and the packet loss ratios for two RT flows are not considered in this simulation scenario.

If the channel states of three flows are all GOOD, the granted-service rates of

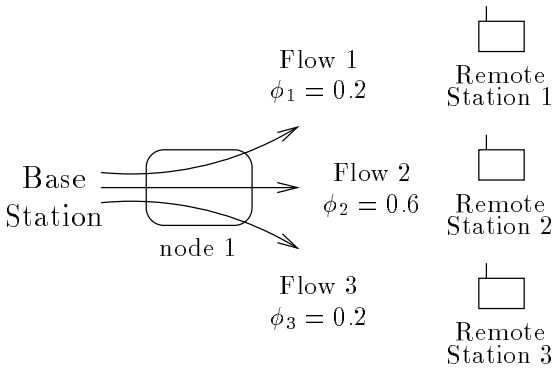


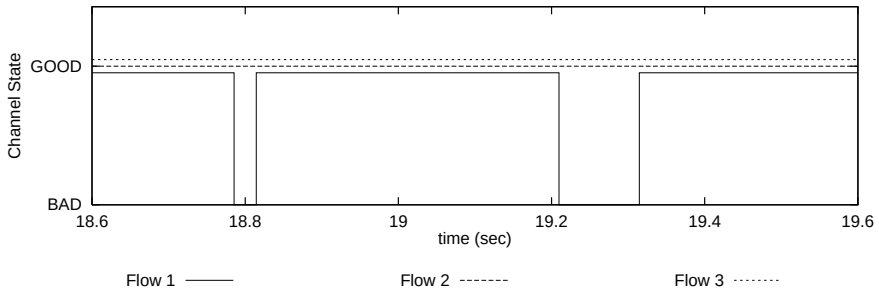
Figure 6.8: Simulation model for Experiment 1.

	Arrival Rate (Mbps)	Reserved BW M_i (Mbps)	Max. Burst Size B_i (air packets)	Residual BW Share ϕ_i	Delay Bound (msec)	Jitter Bound (msec)	Class of Service
Flow 1	2.2	2.0	20	0.2	24	6	Premium RT
Flow 2	5.0	2.0	20	0.2	24	24	Regular RT
Flow 3	9.2	2.0	20	0.6	**N.A**	**N.A**	Premium NRT

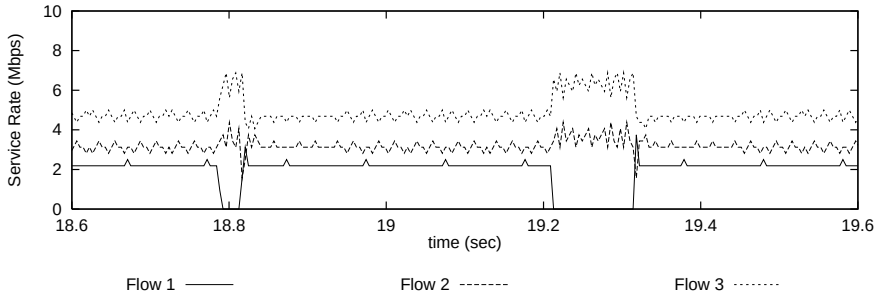
Table 6.3: Simulation configuration for Experiment 1, where **N.A** stands for "not available."

flow 1 and flow 2 should be $2.0+(10-2.0-2.0-2.0)\frac{0.2}{0.2+0.6+0.2} = 2.8$ Mbps while the granted-service rates of flow 3 should be $2.0+(10-2.0-2.0-2.0)\frac{0.6}{0.2+0.6+0.2} = 4.4$ Mbps. However, the average arrival rate of flow 1 is only 2.8 Mbps. Therefore, the overall ideal granted-service rate of flow 2 and flow 3 should be $2.8+(2.8-2.2)\frac{0.2}{0.2+0.6} = 2.95$ Mbps and $4.4+(2.8-2.2)\frac{0.6}{0.2+0.6} = 4.85$ Mbps. From Fig. 6.9 (a), we can observe that the channel states of three flows are all under GOOD states during the interval $[18.8, 19.2]$ sec. And Fig. 6.9 (b) shows that three flows approaches their ideal granted-service rates under WDFQ during this interval. During the interval $[19.2, 19.3]$ seconds, flow 1 enters BAD channel state. Hence, the ideal granted-service rates of flow 2 and flow 3 during within this interval should be $2.8+2.8\frac{0.2}{0.2+0.6} = 3.5$ Mbps and $4.4+2.8\frac{0.6}{0.2+0.6} = 6.5$ Mbps, respectively. Simulation results shown in Fig. 6.9 (b) verify that the granted-service rate of flow 1 is distributed fairly to flow 2 and flow 3 when flow 1 is under BAD channel state. After time 19.3 sec, the channel state of flow 1 becomes "GOOD" once again and it receives its granted-service rate right away. Other two flows also release their bandwidth granted from flow 1 when flow 1 was in BAD channel states. On the other hand, Fig. 6.9 (c) shows that although *Priority FIFO* can guarantee the minimum bandwidth, it cannot distribute the residual bandwidth to every backlogged flow fairly.

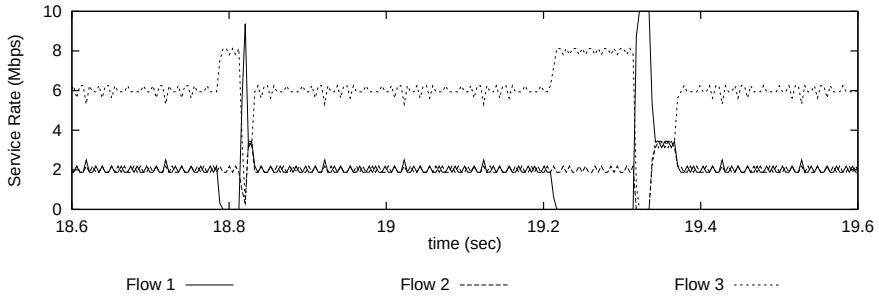
The detailed simulation statistics are listed in Table 6.4. Here, we divide the granted service workload by the length of the observed interval to derive the average service rate. The transient behaviors resulting from the change of channel states are excluded in the fairness index statistics. WDFQ can achieve satisfactory bandwidth sharing fairness in spite of the heavy burstiness of NRT traffic. Based on the simulation results obtained in Experiment 1, we conclude that WDFQ not only provides QoS guarantees for RT traffic streams but also guarantees the bandwidth



(a) Channel state diagram of all flows.



(b) Transient bandwidth sharing behaviors of WDFQ with retransmission period 10 slots.



(c) Transient bandwidth sharing behaviors of *Priority FIFO*, assuming perfect channel knowledge.

Figure 6.9: Simulation results of Experiment 1.

usage of NRT user groups following pre-determined traffic profiles.

Observed Interval (sec)			19.00 ~ 19.20	19.22 ~ 19.30	19.40 ~ 19.60
WDFQ ($T_i^r = 10$)	Avg. Service Rate	Flow 1	2.200	0	2.200
		Flow 2	3.095	3.507	3.119
		Flow 3	4.706	6.123	4.681
	$\mathcal{FRI}$		0.241	0.166	0.282
FIFO (ideal)	Avg. Service Rate	Flow 1	1.996	0	2.003
		Flow 2	2.003	2.005	2.003
		Flow 3	6.001	7.995	5.995
	$\mathcal{FRI}$		1.663	2.492	1.661

Table 6.4: The average service rates and Fairness Indices of WDFQ within different time intervals, where "FIFO (ideal)" stands for the *Priority FIFO* algorithm with full channel knowledge. The unit of average service rate is Mbps.

6.3.2 Experiment 2: The Bandwidth Sharing Behaviors of RT Services, Premium NRT Services and Regular NRT Services

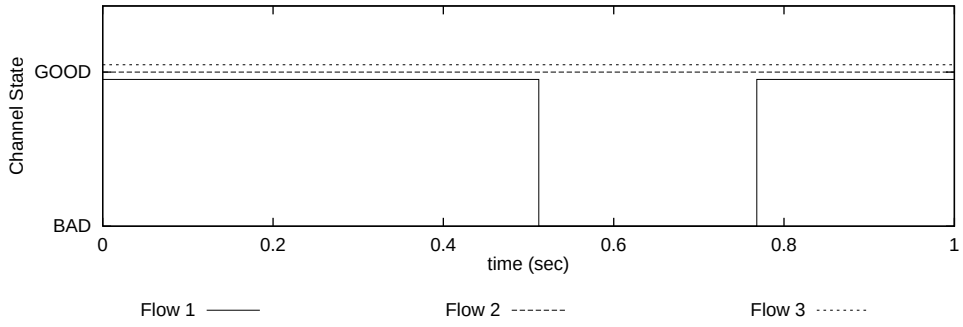
In this section, we verify the different bandwidth requirements can be accommodated via WDFQ simultaneously. The simulation model is similar to Fig. 6.8, but the configuration parameters are different and shown in Table 6.5. In this simulation experiment, the retransmission periods and the credit limits of all flows are set to be 10 slots and 20, respectively.

Flow 1 is a RT traffic stream while flow 2 and flow 3 are both aggregated NRT

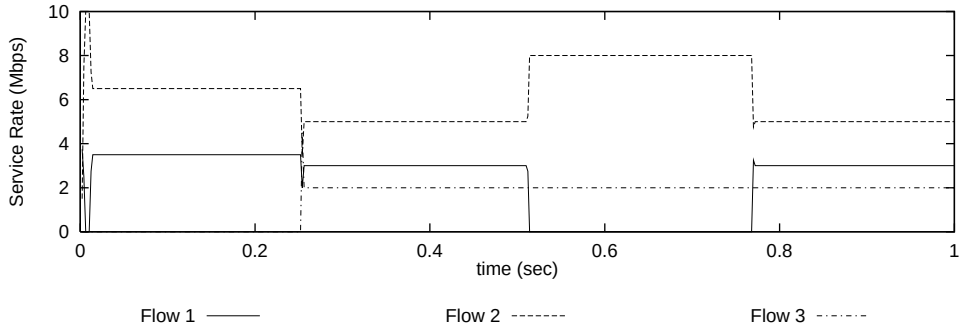
	Arrival Rate (Mbps)	Reserved BW M_i (Mbps)	Max. Burst Size B_i (air packets)	Residual BW Share ϕ_i	Delay Bound (msec)	Jitter Bound (msec)	Class of Service
Flow 1	7.0	2.0	20	0.2	13	13	Regular RT
Flow 2	7.2	2.0	20	0.6	**N.A.**	**N.A.**	Premium NRT
Flow 3	7.2	2.0	20	0	**N.A.**	**N.A.**	Regular NRT

Table 6.5: Simulation configuration for Experiment 2, where **N.A.** stands for "not available."

traffic streams. Flow 2 is assigned as premium service and flow 3 is assigned as regular service. Before flow 3 is on at time 0.254 s, the granted service rates for flow 1 and flow 2 are $2 + \frac{0.2}{0.2+0.6}(10 - 2 - 2) = 3.5$ Mbps and 6.5 Mbps, respectively. We can observe that the service rate received by flow 1 and flow 2 follows the specified minimum guaranteed service rates and residual bandwidth shares. After time 0.254 s, flow 3 is activated and the service rate received by flow 3 is 2 Mbps, which satisfies its requirement of minimum bandwidth guarantee. Note that flow 3 is regular NRT traffic, then it should not receive any other extra bandwidth except its minimum guaranteed bandwidth. Figure 6.10 (b) shows that after flow 3 is on, it only receives the bandwidth it reserved and no residual bandwidth is assigned to flow 3 even when flow 1 releases its bandwidth due to channel errors during the time interval $[0.514, 0.768]$ s. This simulation experiment shows that different service requirements in NRT traffic can be accommodated by WDFQ via the same scheduling platform. This feature reduces the implementation cost significantly.



(a) Channel state diagram of all flows.



(b) Transient bandwidth sharing behaviors of WDFQ with retransmission period 10 slots.

Figure 6.10: Transient behaviors of premium NRT traffic and regular NRT traffic in Experiment 2.

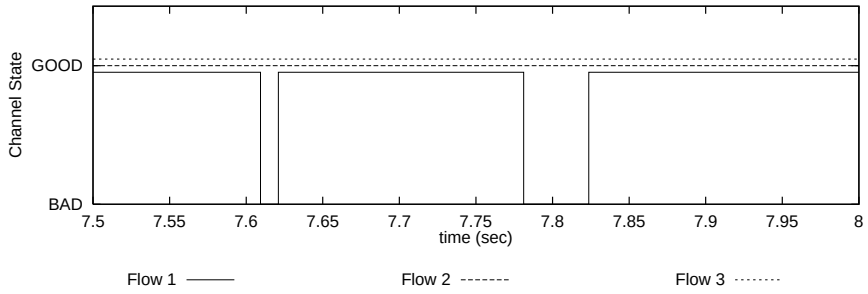
6.3.3 Experiment 3: The Influence of the Credit Limit

In this section, we investigate the influence of the credit limit. The simulation model is also the same as Fig. 6.8 and the configuration parameters are listed in Table 6.6. We assume flow 1 and flow 2 are RT traffic streams and flow 3 serves as the background NRT traffic.

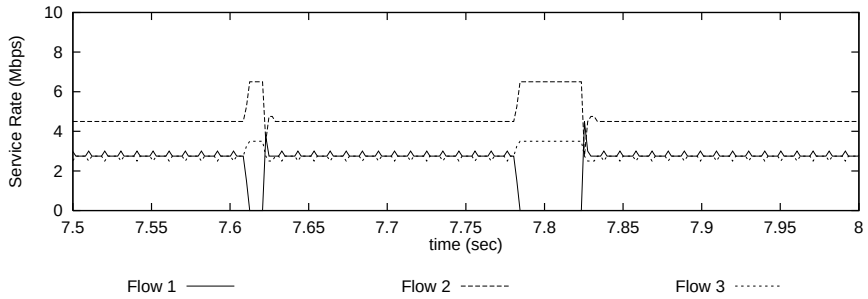
	Arrival Rate (Mbps)	Reserved BW M_i (Mbps)	Max. Burst Size B_i (air packets)	Residual BW Share ϕ_i	Delay Bound (msec)	Jitter Bound (msec)	Class of Service
Flow 1	2.5	2.0	20	0.2	13	3	Premium RT
Flow 2	7.2	2.0	20	0.6	13	13	Regular RT
Flow 3	7.2	2.0	20	0.2	**N.A.**	**N.A.**	Premium NRT

Table 6.6: Simulation configuration for Experiment 3, where **N.A.** represents "not available."

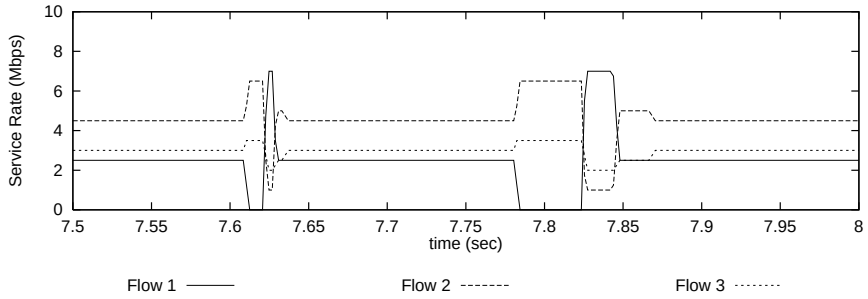
Simulation results are shown in Fig. 6.11. When the channel state of flow 1 goes back to "GOOD" state from "BAD" state at time 7.83 sec, flow 1 is not allowed to request for too much bandwidth it had lost during the "BAD" channel period if its credit limit is small. This phenomena is shown in Fig. 6.11 (b). On the other hand, when the credit limit of flow 1 is large (for example, 200), as shown in Fig. 6.11 (c), then flow 1 is granted larger service rate as its channel state goes back to "GOOD" state at time 7.83 sec. However, if the fairness index shown in eq. (6.4) is taken into consideration, we can observe that there is an obvious trade-off between the credit limit and fairness.



(a) Channel state diagram of all flows.



(b) The credit limit of flow 1 is 5.



(c) The credit limit of flow 1 is 200.

Figure 6.11: Transient behaviors of a RT traffic stream with different credit limits.

6.3.4 Experiment 4: The Influence of the Length of the Retransmission Period

In this section, we study the influence of the length of the retransmission period to find a best point to achieve the best performance via minimum processing overhead.

The simulation model and configuration parameters are shown in Fig. 6.12 and Table 6.7, respectively. The average lengths of "GOOD" period and "error" period are fixed as 1000 slots and 25 slots, respectively. The length of the retransmission period varies from 5 slots to 50 slots to investigate the influence of the length of the retransmission period. Flow 1 is a test video stream as in section 6.3 and flow 2 is an aggregated regular NRT flow driven by a LAN traffic trace.

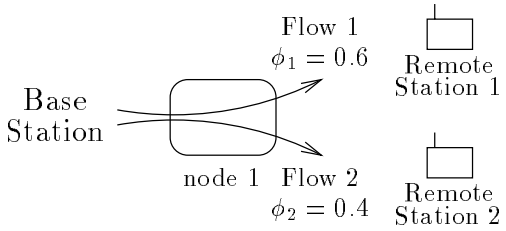


Figure 6.12: Packet loss ratio of flow 1 under various lengths of retransmission periods.

Simulation results are shown in Fig. 6.13. We can observe that if no control mechanism is adopted, the performance of the flow 1 is much worse under *Priority FIFO* than under WDFQ in this comparison. Hence, in the following discussions, the performance of *Priority FIFO* are not included. On the other hand, in WDFQ the packet loss ratio due to channel errors increases slightly as the retransmission period is less than 25 slots, the average length of the error period. Therefore, we conclude that if we set the retransmission period too small compared to the average

	Arrival Rate (Mbps)	Reserved BW M_i (Mbps)	Max. Burst Size B_i (air packets)	Residual BW Share ϕ_i	Delay Bound (msec)	Jitter Bound (msec)	Class of Service
flow 1	3.34	2.0	10	0.6	24	24	Premium RT
flow 2	6.66	2.0	10	0.4	**N.A.**	**N.A.**	Premium NRT

Table 6.7: Simulation configuration for Experiment 4, where **N.A.** represents "not available."

length of error period, the processing overhead will be high and the performance enhancement will not be sufficient. On the other hand, if the retransmission period is set too large, the performance degradation due to error period becomes significant. Hence, we recommend that the retransmission period should be set close to the observed average length of the error period as much as possible.

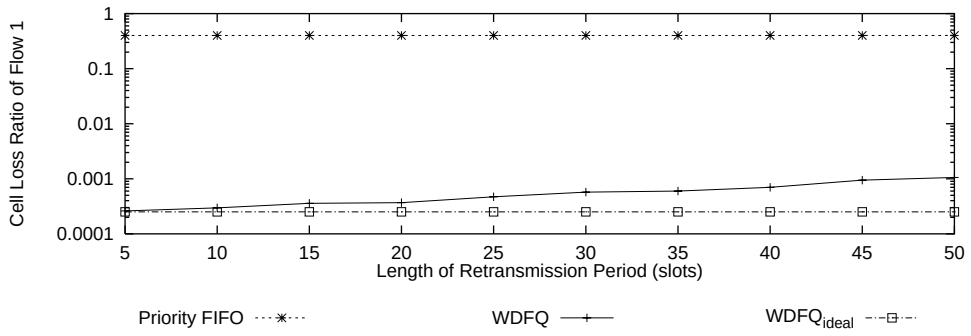


Figure 6.13: The loss ratio of air packets in WDFQ and *Priority FIFO* under various retransmission periods, where "WDFQ_ideal" stands for the WDFQ algorithm with full channel knowledge.

6.4 Discussion

Because the backbone network may accommodate wide range QoS for RT and NRT applications, the arrival traffic from backbone network to the base station may contain several QoS levels. Therefore, to schedule multi-level jitter/delay traffic to remote station hosts is inevitable. Hence, the queueing model propose in section 6.2.2 may be too simplified to satisfy future requirements. In this section, we extend the queueing model in section 6.2.2 to a generalized version, called Generalized WDFQ (GWDFQ) [83]. In this section, we briefly discuss the queueing model and the operation procedures that can accommodate N jitter levels for RT services.

The queueing model of the GWDFQ algorithm is shown in Fig. 6.14. Note that, in GWDFQ, the concept of the *virtual cell* described in Chapter 5 is also adopted. In the following, all operations of the air packet is based on the virtual cell concept. Flows of the physical link are classified as several groups according to flows' jitter constraints. The jitter limits of all flows in the Group i are within $((i-1)T, iT]$ slot times, where $i = 1, \cdots, N$. In this model, non-real-time (NRT) traffic is still assigned to dedicated group, called Group NRT. Two dedicated FIFO queues, called *flow-queue* i and *retx-queue* i, are assigned to flow i, where $i = 1, \cdots, N$. Besides, three FIFO queues at the output port, called *C-queue*, *N-queue* and *R-queue*, are also dedicated for each group. The function of *flow-queue* i, *retx-queue* i, and *R-/C-/N-queue*s are all the same as the simplified queueing model. However, the operation procedures of GWDFQ is slightly more complex than WDFQ. As far as the operations of GWDFQ is concerned, we describe them as follows in detail.

Each time at the starting epoch of the refreshing period, t_n, the flow processor clears all eligible air packets in the output buffer and three FIFO queues in Group 1, including queue $Q_C^{(1)}$, $Q_N^{(1)}$ and $Q_R^{(1)}$, due to violation of their nodal delay bounds. Then the flow processor calculates the reserved workload and the extended workload

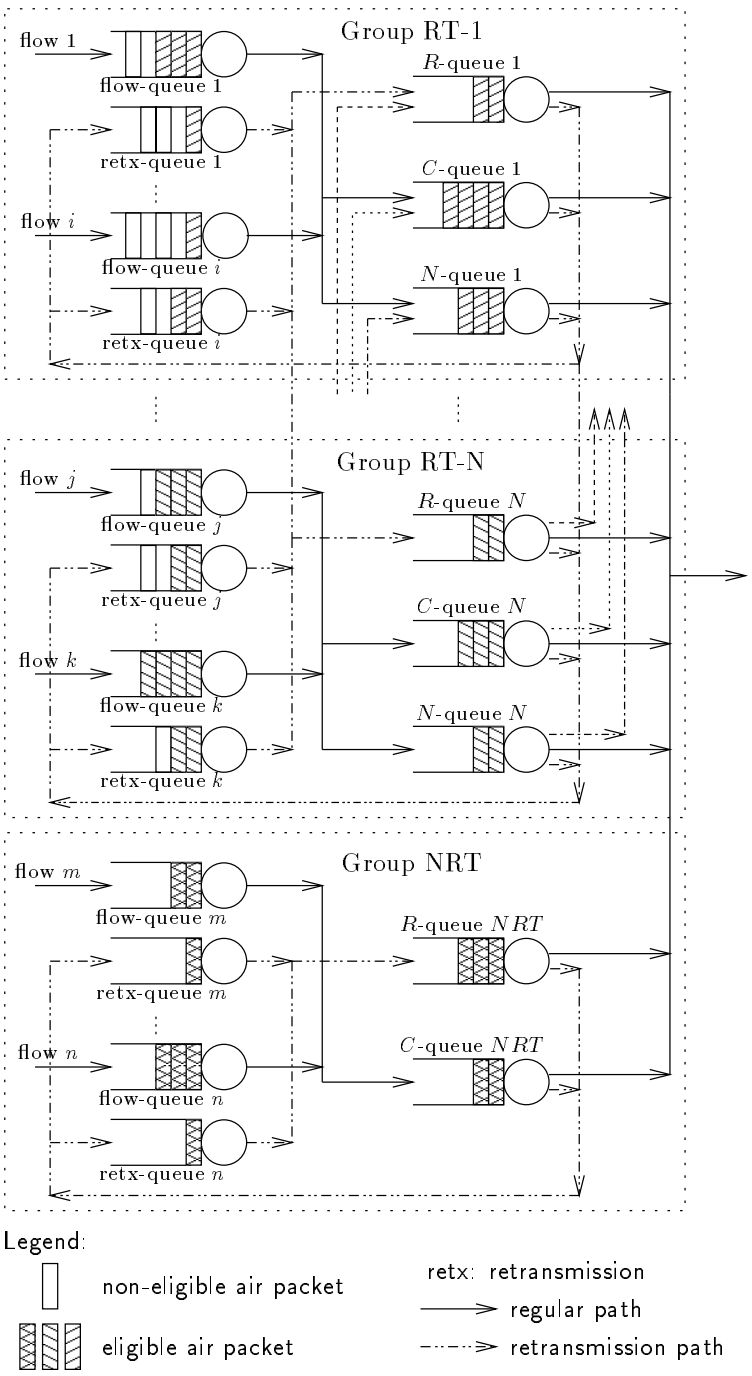


Figure 6.14: Queueing model of the GWDFQ algorithm.

of each flow and informs each flow of these two parameters. In addition to moving eligible air packets from their flow queues to corresponding *R-/C-/N-queues*, the GWDFQ scheduler also requires to move eligible air packets in *R-/C-/N-queue* i to *R-/C-/N-queue* $i-1$ during the refreshing period.

The GWDFQ scheduler can also select an integer κ in advance, where $1 \leq \kappa \leq N$, such that all eligible air packets in queue $Q_R^{(1)}$, $Q_C^{(1)}$, $Q_R^{(2)}, \cdots, Q_C^{(\kappa)}$ are guaranteed the highest priority to be served during a refreshing period. Here, we call the factor κ as the *protection factor*, which is related to the desired fairness as described in Chapter 5. If there are residual time slots in a refreshing-period after serving queue $Q_C^{(\kappa)}$, then the air packets in queue $Q_R^{(NRT)}$ and $Q_C^{(NRT)}$ would have an opportunity to be served. In turn, we use the residual time slots to serve the air packets in *N-queues*, following the sequence $1, 2, \cdots, \kappa$ and the air packets in queue $Q_N^{(NRT)}$. In other words, the eligible air packets are serviced in sequence $Q_R^{(1)}$, $Q_C^{(1)}, \cdots, Q_R^{(\kappa)}$, $Q_C^{(\kappa)}$, $Q_R^{(NRT)}$, $Q_C^{(NRT)}$, $Q_N^{(1)}, \cdots, Q_N^{(\kappa)}$, $Q_R^{(\kappa+1)}$, $Q_C^{(\kappa+1)}$, $Q_N^{(\kappa+1)}, \cdots$. The larger value κ is, the more eligible air packets belonging to reserved workload and extended workload can be transmitted. As a result, with larger κ, more well-behaved flows are *protected*, and better fairness can be achieved. On the other hand, the residual bandwidth available to casually misbehaved flows will be limited by large κ. For network managers who wish to increase link bandwidth utilization at the same time, extremely large κ may not be able to serve their goals.

The GWDFQ algorithm provides a wide range of delay/jitter controls for RT applications and achieve the goal of protecting the users conforming to their service level agreements at the cost of high hardware implementation complexity. In addition, the protection factor κ in WDFQ provides the flexibility of setting the scheduling performance objective in either optimizing performance or optimizing the fairness. However, if the broadband wireless network can be implemented and

the channel quality can be improved, the GWDFQ scheduler indeed provides a novel approach to implement efficient traffic schedulers with residual bandwidth sharing, while guarantee minimum bandwidth, delay and jitter constraints, simultaneously.

6.5 Concluding Remarks

The proposed WDFQ scheduling algorithm has been designed for transporting both RT streaming data and NRT traffic over wireless networks, and this algorithm can accommodate two different service levels (premium or regular) for RT or NRT traffic streams. As a result, the premium RT/NRT service and regular RT/NRT service can be accommodated simultaneously via the same scheduler architecture. We have also illustrated that the WDFQ scheduler can provide fair access of residual bandwidth among all backlogged flows. As for supporting multiple service levels for RT traffic streams, we believe the WDFQ scheme can be easily extended to accommodate this requirement without increase too much implementation cost. With WDFQ, one can cope with the location-dependent channel error property in wireless networks, since the concept of credits compensation is shown able to eliminate the impact of temporary short error bursts. We also use a controlled retransmission mechanism to limit the unnecessary channel access. The simulation results suggest that the length of retransmission period should be adapted to the error length to achieve good performance.

Chapter 7

Conclusions

7.1 Summary of the Works

In this dissertation, we have studied several traffic scheduling mechanisms diverse network environments, from wireline access network to wireless access networks. We first consider the scenario that the RT traffic and NRT traffic are classified. The issue addressed here is how to reduce hardware implementation complexity without degrading the QoS experienced by RT and NRT traffic streams. For NRT traffic, we propose a new scheduling algorithm GFQ with the complexity $O(1)$, implemented with only one single FIFO queue for each port in the output scheduler. We also derive the worst case delay bound and fairness of GFQ. For the previous scheduling algorithms with complexity $O(1)$ such as WRR, DRR, etc., their delay bounds depend on the frame sizes, in other words, depend on the number of flows. Unlike these studies, the delay bound of GFQ depends on the length of the refreshing period, which is dependent on the hardware operation speed and is independent of the number of flows. For RT traffic, we propose the MGFQ algorithm, which employs only one set of FIFO queues to provide a wide range of delay/jitter control for

real-time applications. The special cell formats for real-time multimedia transport are also presented and thus MGFQ is able to combined with the advanced packet discarding scheme, APPD, to accommodate packet level QoS.

Next, we consider the wireline access network, where the RT and NRT traffic streams are multiplexed. In this scenario, we focus on how to accommodate different QoS requirements for these traffic streams via a single scheduling platform. A novel traffic scheduling scheme, called *Differentiated Multi-layer Gated Frame Queueing* (DMGFQ) is proposed to provide users with differentiated QoS, delay/jitter control for RT traffic streams and fair bandwidth sharing for NRT traffic stream, simultaneously. The underlying layer-2 protocols adopted in DMGFQ are those supporting fixed size PDU. DMGFQ combines the concepts of GFQ and MGFQ, and employs a set of FIFO queues to accommodate minimum bandwidth guarantees and fair residual bandwidth share.

Last, the wireless access network is taken into account. How to provide fair channel usage and maintain high channel utilization simultaneously is the concern in this network scenario. We propose a novel scheduling algorithm, called the *Wireless Differentiated Fair Queueing* (WDFQ) algorithm, to accommodate delay/jitter controls, and fair residual bandwidth sharing for real-time and non-real-time traffic streams simultaneously in wireless access networks. The concept of credits compensation is applied in WDFQ to cope with the properties of location-dependent channel error and the temporary short error bursts in wireless networks. The simulation results suggest that the length of retransmission period should be adapted to the error length to achieve good performance and maintain low implementation complexity.

7.2 Contributions of the Dissertation

In this dissertation, several techniques related to traffic scheduling mechanisms have been presented to improve the performance of the high-speed networks, which support multimedia traffic with QoS requirements. We summarize the most important contributions as follows.

1. **Scheduling Algorithm with Residual Bandwidth Sharing for NRT Traffic:** In this research, we propose the GFQ algorithm, whose complexity is $O(1)$. The key contribution of the GFQ algorithm is the successful elimination of output sorter in its scheduler design such that the scheduling mechanism can accommodate large number of flows. However, the advantages in low delay bound and fairness are still preserved in GFQ. In addition, both delay bound and Fairness Index for flows scheduled are derived and validated with simulations. We believe factors such as no sorting operation, the FIFO constraint, Fairness Index, and delay performance requirements are the design direction of the future scheduling algorithms. However, we observe that the current design of GFQ satisfies all these requirements.

2. **Framework for RT Multimedia Transport with Jitter and Delay Guarantees:** The main contributions of this study is that we propose a framework for real-time multimedia transmission in ATM networks, which includes cell formats for real-time multimedia transport, an efficient traffic scheduling scheme called *Multi-layer Gated Frame Queueing* (MGFQ), and a hybrid design to combine scheduling algorithm and the advanced buffer management mechanism. The special cell formats for voice and video traffic streams can help the underlying scheduling algorithm to accommodate the delay/jitter control more efficiently. In MGFQ, only one set of FIFO queues is employed, but

a wide range of QoS for real-time applications is still accommodated. Not only the hardware implementation complexity is reduced significantly but also high multiplexing gain is achieved by MGFQ. The hybrid design further help MGFQ to improve packet level QoS significantly in term of frame discarding ratio for video traffic. With MGFQ, a receiver can allocate less resources, such as buffers, to compensate the disturbed cell arrivals.

To summarize, the presented framework provides a novel approach to implement real-time multimedia transport and the efficient traffic scheduling with flexible jitter and delay guarantees. Various kinds of customer requirements, such as flexible end-to-end jitter constraints, transmission via AAL1/2/5, and adaptive playout, etc., can be achieved by employing MGFQ-enabled switches. We believe that this framework should be able to support a wide range of other jitter and delay sensitive applications for multimedia communications.

3. **Scheduling Algorithm with Delay/Jitter Control and Fair Bandwidth-Sharing:** In this part, a novel traffic scheduling scheme, called *Differentiated Multi-layer Gated Frame Queueing* (DMGFQ), is proposed. The key contribution of the DMGFQ algorithm is providing a novel approach to implement efficient traffic schedulers with fair residual bandwidth sharing, while guaranteeing minimum bandwidth, delay and jitter constraints, simultaneously. More specifically, the contributions of DMGFQ are described as follows. (1) DMGFQ transports both RT streaming data and best-effort traffic over ATM networks with processing overhead at the same order of RPQ$^+$ [74] in sorting or queue insertion operations, and with the same due-date calculation complexity at the same order of JEDD [21]. (2) With the presence of misbehaved traffic, DMGFQ not only can provide a wide range of delay/jitter controls for RT applications, but also can achieve the goal of protecting the users, both RT

and NRT applications, conforming to their service level agreements. (3) Once a flow terminated, DMGFQ distributes its granted bandwidth fairly to other active flows according to their pre-determined weighting factors. (4) With the designated protection factor κ, DMGFQ is able to provide the flexibility of setting the scheduling performance objective in either optimizing performance or optimizing the fairness. To summarize, the rise in usage and the popularity of Internet coupled with new multimedia applications has fueled network operator to offer different levels of treatment to users based on their QoS requirements. We believe that this goal is easier to achieve by employing DMGFQ-enabled switches.

4. **Scheduling Algorithm with Fair Bandwidth-Sharing for Wireless Multimedia Services:** In the last study, we propose WDFQ scheduling algorithm to transport both RT streaming data and NRT traffic over wireless networks. The WDFQ scheduler can provide fair access of residual bandwidth among all backlogged flows. In addition, via the concept of credits compensation, WDFQ can cope with the location-dependent channel error property and eliminate the impact of temporary short error bursts. We also use a controlled retransmission mechanism to limit the unnecessary channel access. To summarize, timely delivery of RT traffic streams and virtually error-free transmission of NRT traffic are all well supported by WDFQ. We believe that wireless multimedia is easier to achieve by employing WDFQ-enabled switches or base stations.

7.3 Future Works

We think there still exists some related issues to be further studied in this dissertation. For example, in Chapter 3, we believe there are other scheduling approaches that are capable of handling large number of flows, without involving sorting procedures. How to design a more efficient scheduler with smaller delay bound than GFQ and satisfy factors such as the FIFO constraint, Fairness Index, and delay performance requirements is an interesting area for further investigation.

In Chapter 4, we combine the scheduler and the advanced packet discarding mechanisms to improve packet level QoS. Nevertheless, how to precisely map QoS parameters from the frame level or the packet level to the cell level needs further investigation. Next, DMGFQ accommodate QoS requirements for RT and NRT traffic streams simultaneously. however, how to precisely determine QoS parameters from the end-to-end requirement to a nodal requirement, and how to balance between system-wise throughput versus fairness does require further investigation. In Chapter 6, timely delivery of RT traffic streams and virtually error-free transmission of NRT traffic are all well supported by WDFQ. However, how to modify the WDFQ algorithm to apply to adaptive channel error correcting code and how to balance between error burst compensation versus fairness does require further investigation. Last but not least, it is difficult to perform a schedulability test for EDD-like schedulers, including MGFQ, DMGFQ and WDFQ. Hence, how to design an efficient CAC mechanism such that packet loss ratios of RT traffic streams can be controlled accurately is another important research issue.

Bibliography

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

[2] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource Reservation Protocol (RSVP) - Version 1 Functional Specification," RFC 2205, Sep. 1997.

[3] J. Wroclawski, "The use of RSVP with IETF Integrated Services," RFC 2210, Sep. 1997.

[4] J. Wroclawski, "Specification of the Controlled-Load Network Element Service," RFC 2211, Sep. 1997.

[5] S. Schenker, C. Partridge, and R. Guerin, "Specification of Guaranteed Quality of Service," RFC 2212 Sep. 1997.

[6] S. Shenker, and J. Wroclawski, "General Characterization Parameters for Integrated Service Network Elements," RFC 2215, Sep. 1997.

[7] R. Yavatkar, D. Hoffman, Y. Bernet, F. Baker, and M. Speer, "SBM (Subnet Bandwidth Manager): A Protocol for Admission Control over IEEE 802-style Networks," RFC 2814, May 2000.

[8] M. Seaman, A. Smith, E. Crawley, and J. Wroclawski, "Integrated Service Mappings on IEEE 802 Networks," RFC 2815, May 2000.

[9] A. Ghanwani, W. Pace, V. Srinivasan, A. Smith, and M. Seaman, "A Framework for Providing Integrated Services Over Shared and Switched LAN Technologies," RFC 2816, May 2000.

[10] Y. Bernet, P. Ford, R. Yavatkar, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski, and E. Felstaine, "A Framework for Integrated Services Operation over Diffserv Networks," RFC 2998, Nov. 2000.

[11] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource Reservation Protocol (RSVP) - Version 1 Functional Specification," RFC 2205, Sep. 1997.

[12] K. Nichols, V. Jacobson, and L. Zhang, "A Two-bit Differentiated Services Architecture for the Internet," ftp://ftp.ee.lbl.gov/papers/dsarch.pdf, Nov. 1997.

[13] K. Nichols, S. Blake, F. Baker, and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," RFC 2474, Dec. 1998.

[14] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," RFC 2475, Dec. 1998.

[15] D. Black, "Differentiated Services and Tunnels," RFC 2983, Oct. 2000.

[16] K. Nichols and B. Carpenter, "Definition of Differentiated Services Per Domain Behaviors and Rules for their Specification," RFC 3086, Aprl 2001.

[17] J. Wroclawski and A. Charny, "Integrated Service Mappings for Differentiated Services Networks," IETF Internet Draft: draft-ietf-issll-ds-map-01.txt, Feb. 2001.

[18] G. Eichler, H. Hussmann, G. Mamais, I. Venieris, C. Prehofer, and S. Salsano, "Implementing Integrated and Differentiated Services for the Internet with ATM Networks: A Practical Approach," *IEEE Commun. Magazine*, Vol. 38, No. 1, pp. 132-141, Jan. 2000.

[19] H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks," *Proceedings of the IEEE*, Vol. 83, No. 10, pp. 1374-1396, Oct. 1995.

[20] ATM Forum Technical Committee, "Traffic Management Specification Version 4.1," AF-TM-0121.000, Mar. 1999.

[21] D. Verma, H. Zhang, and D. Ferrari, "Guaranteeing Delay jitter Bounds in Packet Switching Networks," *Proc. Tricomm '91*, Chapel Hill, NC, pp. 35-46, Apr. 1991.

[22] W. Fisher and K. Meier-Hellstern, "The Markov-Modulated Poisson Process (MMPP) Cookbook," *Performance Evaluation*, Vol. 18, pp. 149-171, 1992.

[23] D. Ferrari and D. Verma, "A Scheme for Real-Time Channel Establishment in Wide-Area Networks," *IEEE J. Select. Areas Commun.*, Vol. 8, No. 4, pp. 368-379, Apr. 1990.

[24] R. Cruz, "A Calculus for Network Delay, Part I: Network Elements in Isolation," *IEEE Trans. Inform. Theory*, Vol. 37, No. 1, pp. 114-121, Jan. 1991.

[25] E. Knightly and H. Zhang, "Traffic Characterization and Switch Utilization Using Deterministic Bounding Interval Dependent Traffic Models," in *Proc. IEEE INFOCOM '95*, pp. 1137-1145, Apr. 1995.

[26] H. Zhang and E. Knightly, "Providing End-to-End Statistical Performance Guarantees with Interval Dependent Stochastic Models," in *Proc. ACM SIGMETRICS '94*, pp.211-220, May 1994.

[27] O. Yaron and M. Sidi, "Calculating Performance Bounds in Communication Networks," in *Proc. IEEE INFOCOM '93*, pp. 539-546, Apr. 1993.

[28] O. Yaron and M. Sidi, "Performance and Stability of Communication Networks via Robust Exponential Bounds," *IEEE/ACM Trans. Networking*, Vol. 1, No. 3, pp. 372-385, Jun. 1993.

[29] D. Clark and W. Fang, "Explicit Allocation of Best Effort Packet Delivery Service," *IEEE/ACM Trans. Networking*, Vol. 6, No. 4, pp. 362-373, Aug. 1998.

[30] B. Suter, T. V. Lakshmanan, D. Stiliadis, and A. K. Choudhary, "Design Considerations for Supporting TCP with Per-flow Queueing," *Proc. IEEE INFOCOM '98*, March 1998.

[31] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," *Proc. ACM SIGCOMM'89*, pp. 1-12, 1989.

[32] S. Lu, V. Bharghavan, and R. Srikant, "Fair Scheduling in Wireless Packet Networks," *Proceedings of ACM SIGCOMM '97*, pp. 63-74, 1997.

[33] S. Lu, V. Bharghavan, and R. Srikant, "Fair Scheduling in Wireless Packet Networks," *IEEE/ACM Trans. Networking*, Vol. 7, No. 4, pp. 473-489, Aug. 1999.

[34] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case," *IEEE/ACM Trans. Networking*, Vol. 1, No. 3, pp. 344-357, Jun. 1993.

[35] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple Node Case," *IEEE/ACM Trans. Networking*, Vol. 2, No. 2, pp. 137-150, Apr. 1994.

[36] J.C.R.Bennett and H. Zhang, "WF^2Q: Worst-Case Fair weighted Fair Queueing," *Proc. IEEE INFOCOM'96*, San Francisco, USA, pp. 120-128, Apr. 1996.

[37] S.J.Golestani, "A Self-Clocked Fair Queueing Scheme for Broadband Applications," *Proc. IEEE INFOCOM'94*, Toronto, Canada, pp. 636-646, Jun. 1994.

[38] M. Shreedhar and G. Varghese, "Efficient Fair Queueing Using Deficit Round Robin," *IEEE/ACM Trans. Networking*, Vol. 4, No. 3, pp. 375-385, Jun. 1996.

[39] D. Ferrari and D. Verma, "A Scheme for Real-Time Channel Establishment in Wide-Area Networks," *IEEE J. Select. Areas Commun.*, Vol. 8, No. 4, pp. 368-379, Apr. 1990.

[40] D.Stiliadis and A. Varma, "Efficient Fair Queueing Algorit hms for Packet-Switched Networks," *IEEE/ACM Trans. Networking*, Vol. 6, No. 2, pp. 175-185, April 1998.

[41] S. Suri, G. Varghese, and G. Chandranmenon, "Leap Forward Virtual Clock: A New Fair Queueing Scheme with Guaranteed Delays and Throughput Fairness," *Proc. IEEE INFOCOM'97*, 1997.

[42] O. Altintas, Y. Atsumi, and T. Yoshida, "Urgency-Based Round Robin: A New Scheduling Discipline for Multiservice Packet Switching Networks," *IEICE Trans. Commun.*, Vol. E81-B, No. 11, pp. 2013-2021, Nov. 1998.

[43] O. Altintas, Y. Atsumi, and T. Yoshida, "Urgency-Based Round Robin: A New Scheduling Discipline for Multiservice PAcket Switching Networks," *IEICE Trans. Commun.*, Vol. E81-B, No. 11, pp. 2013-2022, Nov. 1998.

[44] J. A. Cobb, M. G. Gouda, and A. El-Nahas, "Time-Shift Scheduling–Fair Scheduling of Flows in High-Speed Networks," *IEEE/ACM Trans. Networking*, Vol. 6, No. 3, pp. 274-285, June 1998.

[45] C. Liu and J. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," *J. ACM*, Vol.20, pp. 46-61, Jan. 1973.

[46] S. Golestani, "A Stop-and-Go Queueing Framework for Congestion Management," *Proc. ACM SIGCOMM '90*, pp. 8-18, Sept. 1990.

[47] S. Golestani, "A Framing Strategy for Congestion Management," *IEEE J. Select. Areas Commun.*, Vol. 9, No. 7, pp. 1064-1076, Sept. 1991.

[48] H. Zhang and D. Ferrari, "Rate-Controlled Static-Priority Queueing," *Proc. IEEE INFOCOM '93*, pp. 227-236, Sept. 1993.

[49] P. Bhagwat, P. Bhattacharya, A. Krishna, and S. K. Tripathi, "Enhancing Throughput over Wireless LANs using Channel State Dependent packet Scheduling," *Proceedings of IEEE INFOCOM '96*, Vol. 3, pp. 1133-1140, Mar, 1996.

[50] D. A. Eckhardt and P. Steenkiste, "Effort-limited Fair (ELF) Scheduling for Wireless Networks," *Proceedings of IEEE INFOCOM 2000*, pp. 1097-1106, 2000.

[51] T. S. E. Ng, I. Stoica, and H. Zhang, "Packet Fair Queueing Algorithms for Wireless Networks with Location-Dependent Errors," *Proceedings of IEEE INFOCOM '98*, Mar. 1998.

[52] I. Stoica, H. Zhang, and T. S. E. Ng, "A Hierarchical Fair Service Curve Algorithm for Link-Sharing, Real-Time and Priority Services," *ACM Comput. Commun.*, Vol. 27, pp. 247-262, Oct. 1997.

[53] S. Floyd, and V. Jacobson, "Link-Sharing and Resource Management Models for Packet Networks, *IEEE/ACM Trans. Networking*, Vol. 3, No. 4, pp. 365-386, Aug. 1995.

[54] C. Fragouli, V. Sivaraman, and M. B. Srivastava, "Controlled Multimedia Wireless Link Sharing via Enhanced Class-Based Queueing with Channel-State-Dependent Packet Scheduling," *Proceedings of IEEE INFOCOM '98*, Vol. 2, pp. 572-580, Mar. 1998.

[55] P. Goyal, H. M. Vin, abd H. Chen, "Start-time Fair Queueing: A Scheduling Algorithm for Integrated Services," *Proceedings of ACM SIGCOMM '96*, pp. 157-168, Aug. 1996.

[56] A. Varma and D. Stiliadis, "Hardware Implementation of Fair Queueing Algorithms for Asynchronous Transfer Mode Networks," *IEEE Commun. Magazine*, Vol. 35, No. 12, pp. 54-68, Dec. 1997.

[57] M. Katevenis, S. Sidiropoulos, and C. Courcoubetis, "Weighted Round-Robin Cell Multiplexing in a General-Purpose ATM Switch Chip," *IEEE J. Select. Areas Commun.*, Vol. 9, No. 8, pp. 1265-1279, Oct. 1991.

[58] D.Stiliadis, "Traffic Scheduling in Packet-Switched Networks: Analysis, Design, and Implementation," Ph.D

dissertation of Computer Engineering, University of California, Santa Cruz, ftp://ftp.cse.ucsc.edu/pub/hsnlab/dimitrios_dissertation.ps.Z, June 1996.

[59] L. Zhang, "VirtualClock: A New Traffic Control Algorithm for Packet Switching Networks," *Proc. ACM SIGCOMM'90*, pp. 19-29, 1990.

[60] D.Stiliadis and A. Varma, "Efficient Fair Queueing Algorithms for Packet-Switched Networks," *IEEE/ACM Trans. Networking*, Vol. 6, No. 2, pp. 175-185, April 1998.

[61] Y. Ohba, "QLWFQ: A Queue Length Based Weighted Fair Queueing Algorithm in ATM Networks," *Proc. IEEE INFOCOM'97*, Kobe, Japan, pp. 567-576, Apr. 1997.

[62] Y. Chen and J. S. Turner, "Design of a Weighted Fair Queueing Cell Scheduler for ATM Networks," *Proc. IEEE GLOBECOM'98*, 1998.

[63] W. Leland and D. Wilson, ftp://ita.ee.lbl.gov/traces/BC-pAug89.TL.Z, 1989.

[64] A.E.Kamal, "A Performance Study of Selective Cell Discarding Using the End-of-Packet Indicator in AAL Type 5," *Proc. IEEE INFOCOM '95*, pp. 1264-1272, 1995.

[65] H-B. Chiou, Z. Tsai, "An Age Priority Packet Discarding Scheme for ATM Switches on Internet Backbone," *IEICE Trans. Commun.*, Vol.E81-B No.2 pp. 380-391, Feb. 1998.

[66] H-B. Chiou and Z. Tsai, "Performance of ATM Switches with Age Priority Packet Discarding under the ON-OFF Source Model," *Proc. IEEE INFOCOM '98*, San Francisco, USA, March 31-April 2, 1998.

[67] H. L. Pocher, V. C. M. Leung, and D. W. Gillies, "An Efficient ATM Voice Service with Flexible Jitter and Delay Guarantees," *IEEE J. Select. Areas Commun.*, Vol. 17, No. 1, pp. 51-62, Jan. 1999.

[68] J. Liebeherr, D. E. Wrege and D. Ferrari, "Exact Admission Control for Networks with a Bounded Delay Service," *IEEE/ACM Trans. Networking*, Vol. 4, No. 6, pp. 885-901, Dec. 1996.

[69] F-M. Tsou, H. B. Chiou and Z. Tsai, "Design and Simulation of an Efficient Real-Time Traffic Scheduler with Jitter and Delay Guarantees," *IEEE Trans. Multimedia*, Vol. 2, No. 4, pp. 255-266, Dec. 2000.

[70] B-ISDN ATM Adaption Layer Specification: Type 2 AAL, ITU-T Recommendation I.363.2, Sep. 1997.

[71] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," Internet Engineering Task Force, Audio-Video Transport Working Group, RFC 1889, Jan. 1996.

[72] A. Tanenbaum, Computer Networks, 3rd edition, Prentice Hall, 1996.

[73] J. Heinanen, "Multiprotocol Encapsulation over ATM Adaption Layer 5," Internet Engineering Task Force, Network Working Group, RFC 1483, July 1993.

[74] J. Liebeherr and D. E. Wrege, "Priority Queue Schedulers with Approximate Sorting in Output Buffered Switches," *IEEE J. Select. Areas Commun.*, Vol. 17, No. 6, pp. 1127-1144, June 1999.

[75] ITU-T Recommendation G.764, General Aspects of Digital Transmission Systems – Packetization Guide, Nov. 1995.

[76] O. Rose, *MPEG-1 Video Trace, James Bond: Goldfinger*, Institute of Computer Science III, University Wuerzburg, ftp://ftp-info3.informatik.uni-wuerzburg.de/ pub/MPEG/traces, 1995.

[77] S. Iatrou and I. Stavrakakis, "A Dynamic Regulation and Scheduling Scheme for Real-Time Traffic Management," *IEEE/ACM Trans. Networking*, Vol. 8, No. 1, pp. 60-70, Feb. 2000.

[78] Hong-Bin Chiou, Fu-Ming Tsou, and Zsehong Tsai, "DMGFQ: ANovel Traffic Scheduler with Differentiated QoS Guarantee for Internet Multimedia Services," IEEE ICC '2001, Helsinki, Finland, June 11-14, 2001.

[79] ITU-T Recommendation I.371, Traffic Control and Congestion Control in B-ISDN, Aug. 1996.

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

[81] Fu-Ming Tsou, Hong-Bin Chiou, Zsehong Tsai, "WDFQ: An Efficient Traffic Scheduler with Fair Bandwidth-Sharing for Wireless Multimedia Services," *IEICE Trans. Commun.*, Vol.E84-B No.4 pp.823-835, Apr. 2001.

[82] IEEE 802.11: IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, June 26, 1997.

[83] Fu-Ming Tsou, Hong-Bin Chiou, and Zsehong Tsai, "Design of an Efficient Traffic Scheduler with Fair Bandwidth-Sharing for Wireless Multimedia Services," IEEE ICC '2001, Helsinki, Finland, June 11-14, 2001.