Abstract

SRIDHARAN, SAVITHA. Fair Queueing Algorithms for QoS Support in Packet Switched Networks. (Under the direction of Dr. Yannis Viniotis, in association with Agere Systems.)

The rapid growth of telecommunications has led to a demand for convergence of voice, video and data integration over traditional data networks. This has strained the current packet-switched networks which are not very well-suited for voice and video applications. Quality-of-service-aware, high-speed packet switches are being designed to alleviate the problems in traditional networks. The key motivation of this thesis was to analyze scheduling algorithms in order to guarantee Quality of Service for high-speed packet networks, particularly for voice applications.

A behavioral model of the Agere Network Processor's scheduler, *Sched-650* was implemented in C. Using this simulator, various options to modify scheduling algorithms and understand their implementation issues were analyzed. The results were then used to modify Agere's Java-based simulator. The *Active List* scheduling algorithm was modified with different scheduling policies to understand its effect on *bandwidth sharing* and *jitter* of different traffic flows. The fairness of a scheduling algorithm was determined based on the bandwidth allocated to all the competing flows. Experiments were also conducted to study the impact of varying packet size and number of flows for the different scheduling algorithms.

Fair Queueing Algorithms for QOS support in Packet Switched Networks

by

Savitha Sridharan

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Electrical Engineering

Raleigh, North Carolina

2006

Approved By:

[illegible]

Dr. Mike Devetsikiotis

[illegible]

Dr. Gregory Byrd

[illegible]

Dr. Yannis Viniotis
Chair of Advisory Committee

To my parents and my brother

Biography

Savitha Sridharan was born in Bangalore, India in June 1979. She graduated from Indian Institute of Technology, Madras, with a Bachelors degree in Electrical Engineering in July 2001. After undergraduate studies, she worked with Aural Networks, Bangalore, for three years. In August 2004, she joined North Carolina State University as a graduate student. In Summer and Fall of 2005, she interned at Agere Systems, Austin, in the Network Processor division. While working towards the Masters degree, she worked on her thesis with Agere Systems under the guidance of Dr. Yannis Viniotis.

Acknowledgement

I sincerely thank my advisor, Dr.Yannis Viniotis for his invaluable guidance and support through my graduate studies. I would like to sincerely thank him for encouraging me in pursuing my interests in the area of Network Processor Architecture with Agere Systems for my thesis. I thank him for his constant support and encouragement.

I am equally thankful to Curtis Lee Hillier and Mark Emery, my supervisors at Agere Systems, Austin, during my internship in Summer 2005 for all their encouragement and for offering me an opportunity to work at Agere Systems for my thesis. I would like to sincerely thank Paulus Sastrodjojo, David Sonnier, Rob Munoz and Balakrishnan Sundararaman for all the support and instant answers to all the questions I had during my research at Agere.

I am very grateful to Dr. Devetsikiotis and Dr. Byrd for agreeing to be on my thesis committee and for the valuable feedback regarding the thesis document.

I am greatly indebted to my parents and my brother for everything they have done for me. Without their love and support, I would never have been able to succeed in my endeavors. My sincere thanks to Bhaskar for helping me and providing me help through my graduate school here.

Many thanks to my friends Sandeep, Anand and Sudarshan from IIT and my friends Sandhya, Yasaswini, Payal , Arundathi and Sanchith for helping me through my tough times and making my stay here worthwhile. I would also like to thank Vasu, Sandhya, Anita and Aswini who made my stay in Austin so comfortable.

Contents

List of Tables **vii**

List of Figures **viii**

1 Introduction **1**

1.1 Motivation . . . 1

1.2 Outline of Thesis . . . 2

2 Fair Packet Scheduling **3**

2.1 Introduction . . . 3

2.1.1 Packet Scheduling Algorithms . . . 5

2.1.2 Packet Switch Architectures . . . 5

2.2 Scheduling Output Buffers . . . 8

2.2.1 Classifying Schedulers . . . 8

2.2.2 Shapers and Schedulers . . . 10

2.3 Requirements of a Packet Scheduler . . . 12

2.4 Complexity of a Scheduling Algorithm . . . 14

2.4.1 Design Complexity . . . 14

2.4.2 Implementation Complexity . . . 16

2.5 Applications of a Fair Packet Scheduler . . . 16

3 Understanding QOS in Packet Switched Networks **18**

3.1 Classifying Packet Networks . . . 18

3.1.1 Classification by Packet Size . . . 18

3.1.2 Classification by Packet Route . . . 20

3.2 Traffic Types in Packet Networks . . . 21

3.3 Achieving QoS . . . 22

3.4 Metrics for QOS Guarantee . . . 26

3.4.1 Bandwidth Sharing . . . 27

3.4.2 Packet Delay . . . 28

3.4.3 Delay Jitter . . . 29

3.4.4 Packet Loss . . . 31

3.5 Fairness Index 33
3.5.1 Jain's Fairness Index 34

4 Packet Scheduling Algorithms 36
4.1 Background Work 36
4.1.1 Generalized Processor Sharing (GPS) 36
4.2 Weighted Jitter Deadline Scheduling 38
4.2.1 Category 1 : Simple WJ-EDF 39
4.2.2 Category 2: Shaped WJ-EDF 41

5 Simulation Study 44
5.1 Architecture Overview 44
5.2 Simulation Environment 47
5.2.1 Setup 1: Behavioral C model 48
5.2.2 Setup 2: Agere's SDE 50
5.3 Metrics and Simulation Variables 52
5.4 Observations and Performance Analysis 54
5.4.1 Fairness Index Analysis 55
5.4.2 Jitter Analysis 60
5.5 Weighted-Jitter EDF 65

6 Conclusion and Future Work 69
6.1 Findings and Conclusion 69
6.2 Future Plans 70

Bibliography 71

A Tables 77

List of Tables

2.1 Comparison of Work Conserving and Non Work Conserving Schedulers 9
2.2 Comparison of sorted and frame-based schedulers 10

5.1 T for different arrival rates and packet sizes (IETF). 61
5.2 d_{avg} for different arrival rates and packet sizes (ITU-T). 61

A.1 Single Traffic Flow fed to OC-3 Scheduler (Sample Size n=10) 77
A.2 Two Traffic Flows fed to OC-3 Scheduler (Sample Size n=10) 78
A.3 Four Traffic Flows fed to OC-3 Scheduler (Sample Size n=10) 79
A.4 Comparison of Scheduling Algorithms for traffic flow feeding the OC-3 Scheduler (Sample Size n=10) 79
A.5 Comparison of Scheduling Algorithms for two traffic flows feeding the OC-3 Scheduler (Sample Size n=10) 80
A.6 Comparison of Scheduling Algorithms for two traffic flows feeding the OC-3 Scheduler (Sample Size n=10) 80
A.7 Comparison of Scheduling Algorithms for four traffic flows feeding the OC-3 Scheduler (Sample Size n=10) 81
A.8 Deadlines applied to packets of different sizes 82
A.9 Deadlines applied to packets arriving at different input rates 83
A.10 Algorithm Legend 83
A.11 1-Point CDV for Single Input Flow 84
A.12 2-Point PDV for Single Input Flow 84
A.13 Average 1-Point CDV for Four Input Flows 85
A.14 Average 2-Point PDV for Four Input Flows 85

List of Figures

2.1 Basic Architectural Components of a Packet Switch 6
2.2 Kleinrock's Delay Conservation law for work-conserving schedulers. . 8
2.3 Packet Processing Elements in a network device. 11
2.4 An example of a Hierarchical Scheduler 12

3.1 Example of bandwidth sharing. 27
3.2 Understanding Delay and Delay Jitter. 31
3.3 Loss graph 32

4.1 Simple WJ-EDF scheduler 39
4.2 Shaped WJ-EDF scheduler structure. 42

5.1 APP650 Architecture: Block Diagram. 45
5.2 Sched-650 Architecture: Block Diagram. 46
5.3 Behavioral C Model: Block Diagram 48
5.4 Testcase Data File format. 48
5.5 Result File: Jitter Measurements. 50
5.6 Agere SDE's Java-based GUI. 50
5.7 Behavior of packets from two traffic flows at the output of Sched-650. 55
5.8 Single flow: Output rate of a scheduler operating at OC-3 for various input rates. 56
5.9 Two traffic flows: Output rate of a scheduler operating at OC-3, for various input rates. 57
5.10 Four traffic flows: Output rate of a scheduler operating at OC-3, for various input rates. 58
5.11 Two input traffic flows: Comparison of scheduling algorithms. 59
5.12 Four input traffic flows: Comparison of scheduling algorithms. 60
5.13 Comparison of 1-Point CDV with number of flows for Packet Size = 1 Cell 62
5.14 Comparison of 2-Point PDV with number of flows for Packet Size = 1 Cell 62

5.15 Comparison of 1-Point CDV with number of flows for Packet Size = 2 Cells 63
5.16 Comparison of 2-Point PDV with number of flows for Packet Size = 2 Cells 63
5.17 Comparison of 1-Point CDV with number of flows for Packet Size = 3 Cells 64
5.18 Comparison of 2-Point PDV with number of flows for Packet Size = 3 Cells 64
5.19 Comparison of 1-Point CDV with number of flows for for Packet Size = 8 Cells 65
5.20 Comparison of 2-Point PDV with number of flows for Packet Size = 8 Cells. 65
5.21 Comparison of 1-Point CDV with packet size for Single Flow. 66
5.22 Comparison of 2-Point PDV with packet size for Single Flow. 66
5.23 Comparison of 1-Point CDV with packet size for four Flows. 67
5.24 Comparison of 2-Point PDV with packet size for four Flows. 67
5.25 Deadlines to packet departures under EDF and WJ-EDF for different packet sizes. 68
5.26 Deadlines to packet departures under EDF and WJ-EDF for different input rates. 68

Chapter 1

Introduction

1.1 Motivation

Recent trends in telecommunication, computing and entertainment along with faster networks have led to a demand for technology convergence. The idea of integrating voice, data and video over a single network, which is easily accessible for a large number of users is gaining popularity. These integrated networks help simplify consumer needs as well as reduce maintenance and support costs for their operator companies. Traditional circuit-switched networks are not suitable for these applications and have led to the development of Packet Switched Networks or PSN.

Telephone systems and virtually all data communication networks mostly used circuit-switched networks during the late 1960s. Circuit-switched networks *preallocate* transmission bandwidth for an entire call or session, hence guaranteeing capacity for the user. Though dedicating resources to facilitate a single call ensures quality of service (QoS), it is a costly proposition, because the resources are not utilized completely during the duration of the call or session. In the beginning of the 1970s, a competing approach of *dynamic allocation* of bandwidth was introduced in building communication networks, now popularly called as *packet switched networks.*

Packet Switched Networks (PSN) break the triple-play (voice, video and data) information to be communicated into smaller blocks, append each block with control information and a destination address to form packets. These packets are then transmitted through network nodes until they reach the destination, taking (hopefully) the most expedient route. The receiver uses the destination address in the packet to identify its blocks and reassemble the blocks into the original information. Thus PSNs are able to share network resources, and provide a cost-effective, flexible and reliable technology. However, it is not always possible to guarantee bandwidth in a packet switched network. Due to the stochastic nature of packet queuing in network nodes, delay varies from packet to packet based on the network traffic load. Even under lightly-loaded network conditions, delays are typically larger than in the circuit switched networks.

Packet switched networks are good candidates for a converged network. Transmitting voice and constant bit-rate (CBR) traffic on a packet-switched network is challenging because of the long transmission delay and delay variations associated with these networks. In order to alleviate these problems, Quality of Service-aware, high-speed packet switched networks are being designed to reduce end-to-end transmission delays. Analysis and design of scheduling algorithms to guarantee QoS in high-speed packet networks is the key motivation of this thesis.

1.2 Outline of Thesis

The thesis concentrates on packet scheduling algorithms employed for QoS guarantees in multi-service packet networks. In Chapter 2, we discuss the background work in this area. In Chapter 3, we cover the range of packet networks this thesis focuses on, followed by defining the metrics to characterize the performance of an algorithm and the suitability of an algorithm for QoS support. In Chapter 4, we briefly cover GPS, the ideal scheme proposed for the design of a fair packet scheduler, with emphasis on their support to guarantee QoS for packet traffic. Research conducted on Earliest Deadline First scheduling is also covered in this section. In Chapter 5, we present the simulation study of the scheduling algorithms for Constant Bit Rate traffic on a behavioral C model of Agere's APP650 scheduler, *Sched-650*, followed by the study on their SDE. Chapter 6 concludes with a discussion on the simulation results and scope for future work.

Chapter 2

Fair Packet Scheduling

2.1 Introduction

Packet Switching is a method in which data is divided into units called packets before being transmitted to the destination. The packets can arrive at the destination through completely different routes. The networks which use this method to transfer data are known as Packet Switched Networks [1]. The other kind of networks are circuit-switched networks, where a physical connection is present between the source and destination when transmitting data.

Packet Switched Networks has proved to be good candidates for *network convergence*, that is integrating voice, video and data applications. This is because as long as the packets adhere to the network protocol the underlying data can be of different types. Many different kinds of networks inter-operate to support many different kinds of services ranging from real-time voice and video applications like voice-over-IP and videoconferencing, streaming video, interactive applications like web browsing and gaming, and background applications like remote access applications, file transfer and e-mail.

As user demands and bandwidth grow, there is a need to control the available resources and distribute them among all these users. This, however, is challenging in packet switched networks, as it is harder to estimate the peak bandwidth utilization.

In traditional circuit switched networks, the participating flows (or users) would be aware of their peak bandwidth utilization during the connection setup phase. A new flow requesting connection may be admitted into the network until the sum of the maximum bandwidth requirements of the already connected flow is less than the link capacity under consideration. Consider a node in a network that can support a total bandwidth, C_{total}. Let us consider the case of a new flow k requesting this node a bandwidth of C_k at an instant when the node is already supporting n connections, each utilizing C_1, C_2,..., C_n. Flow k will be allowed to share the resources of this node only if,

$$C_k \leq C_{total} - \sum_{j=1}^{n} C_j \tag{2.1}$$

Dedicating resources to facilitate a selected number of users is a *costly* proposition, because the resources are not utilized completely during the duration of the connection. In the case of a packet switched network, a technique called *dynamic allocation* of resources for flows is used. The common dynamic resource allocation approaches include statistical multiplexing and packet scheduling algorithms. Statistical multiplexing is one of the most common techniques in network system design primarily because of simplicity of design and implementation.

In statistical multiplexing, utilization of link capacity is done by allocating the *average* bandwidth requirements for each flow instead of the *peak* bandwidth. This allows a greater number of flows to simultaneously share a link and increase the efficiency of network resource utilization. In most cases, statistical multiplexers [2] work efficiently. However, performance degrades when packets are lost due to queue overflow, when all competing flows transmit bursty traffic at the same time. Users may utilize bandwidth greater than their reserved (i.e., average) bandwidth requirements, resulting in congestion issues. In order to avoid congestion problems, networking devices must be designed with more sophisticated mechanisms to control the distribution of available bandwidth to different connections. Packet scheduling algorithms are algorithms used to achieve this.

2.1.1 Packet Scheduling Algorithms

One of the mechanisms employed to control the resources allocated to packets from different connections by deciding which packet is to be transmitted next on the output link is called a *packet scheduling algorithm*. A packet scheduler forms a small but a very critical part of network system design. The efficiency and fairness (Refer to section 3.4) achieved by a scheduling algorithm is dependent on a number of factors such as number of queues that are serviced by the algorithm, bandwidth required by each flow, arrival pattern of traffic in each of the flows, length of the queues to hold these flows and size of packets arriving into the queues.

Packet scheduling algorithms may be implemented in hardware or in software. The scheduling algorithm must also take into consideration the kind of traffic that it must handle, ranging from small fixed-size packets like ATM cells carrying real-time voice to large variable-size packets like Jumbo Ethernet frames carrying best-effort data. Therefore, the scheduling algorithms must be designed in such a way that they can satisfy both the strict performance needs of real-time applications like voice and video, as well as the fair network resource sharing needed for best-effort traffic. Meeting these multiple restrictions makes design of a scheduling algorithm a challenging task. These algorithms are further constrained by the underlying packet switch architectures and buffering mechanisms.

2.1.2 Packet Switch Architectures

Packet switches are designed to serve two main functions:

1. **Routing**: When a packet arrives at the input port, the packet switch must take a decision about its next hop towards its destination. Once the decision is made, the packet is routed to the appropriate output port.

2. **Buffering**: Output port contentions are resolved by buffering. An arbitrator decodes when a waiting packet must leave the switch towards its next hop. The scheduling algorithm, as discussed in Section 2.1.1, is implemented in this arbitrator to assure that packet switches meet the performance requirements

of their applications. The performance criteria may vary from application to application, ranging from the highest possible utilization of the expensive resources, the lowest possible latency, fair allocation of resources to competing users (QoS guarantees) or combinations of these.

The basic architectural components of a packet switch, keeping these functions in mind, have been captured in Figure 2.1.

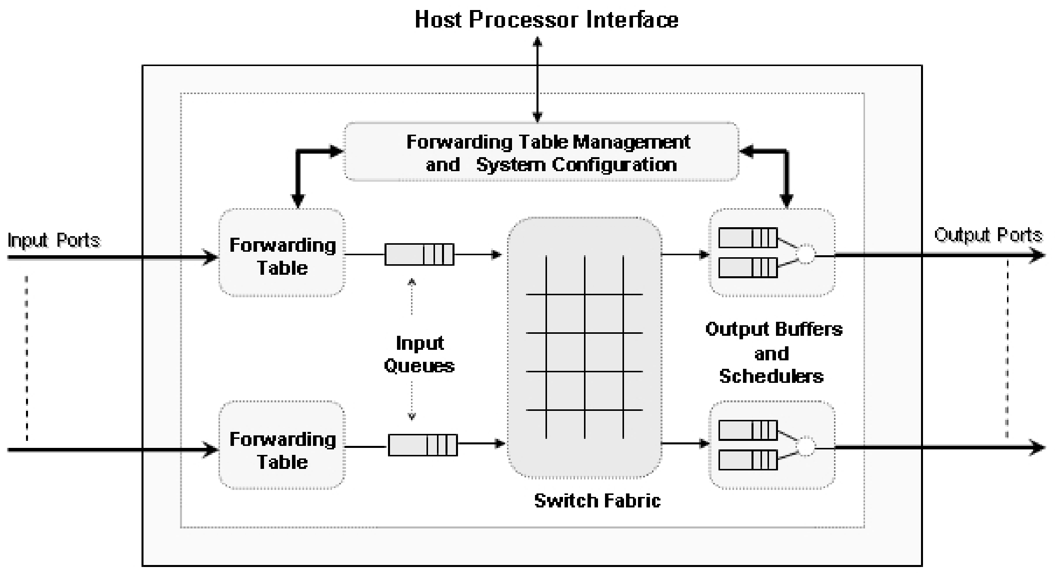


Figure 2.1: Basic Architectural Components of a Packet Switch

1. **Forwarding Engine and Policer**: The implementation of this component in switches depends on the function of the switch. The forwarding engine is also commonly referred to as the *classifier*. For example, let us consider a Ethernet Layer 2 Switch. When a packet arrives, its destination MAC address is looked up in a forwarding table. If the address is found, the packet is sent to an appropriate output port after updating its next-hop MAC address. The switch can also be programmed to discard corrupted or non-conforming packets.

   Some designs also implement traffic policing in this block, i.e., the output rate is controlled by marking or dropping those non-conforming packets when the traffic rate exceeds the configured maximum output rate (peak). In most cases, the policing block does not contain buffers and hence avoids delays due to queuing. However, bursty traffic is propagated through this block, and hence does not help in smoothening the traffic flow.

2. **Input buffers**: This may be a single FIFO or multiple FIFOs per input port in an input-buffered switch. Multiple FIFOs are used to eliminate head-of-line blocking, discussed a little later in this section. A scheduler may be needed to arbitrate packets from multiple queues [3].

3. **Switch Fabric**: This is also commonly referred to as *backplane*. The common fabric types include shared memory, bus, crossbar, ring structure and multistage networks. The packets may be buffered while the packet waits its turn to be transferred across the backplane.

4. **Output buffers and scheduler**: Packets may be buffered as they wait for their turn to be transmitted out of the output port. Schedulers can have simple algorithms like First-Come, First-Serve or more complex algorithms like Weighted Round Robin or Weighted Fair Queuing in order to distinguish different priority classes and meet QoS guarantees. This will be covered in greater detail in Section 2.1.

In general, packet switch architectures may be classified into two main categories based on whether packets are buffered at the input or output of the switch. Accordingly, they are called *input-buffered* or *output-buffered* switches. Various switch architectures have been suggested with varying combinations of the components mentioned above.

2.2 Scheduling Output Buffers

2.2.1 Classifying Schedulers

Networking designers and engineers have invented and studied a myriad of packet scheduling algorithms for years, but the design of an ideal scheduler (exhibiting performance of GPS or close to it) is still an extremely challenging problem. These multitudes of packet scheduling algorithms, varying in their ease of implementation and performance, can be characterized into the following categories:

1. *Work-Conserving or Non Work-Conserving scheduler* : A scheduler is considered work-conserving if it never leaves the shared output link idle when there is a packet buffered in the system. First-come First-serve (FCFS), Round Robin (RR), Weighted Round Robin (WRR) and Weighted Fair Queuing (WFQ) schedulers are work-conserving in nature.

   Kleinrock's Delay Conservation law for work-conserving schedulers states that one flow cannot be given preferential treatment (i.e., reduced delay) without hurting the others. Consider N flows arriving at a scheduler as shown in Figure 2.2.

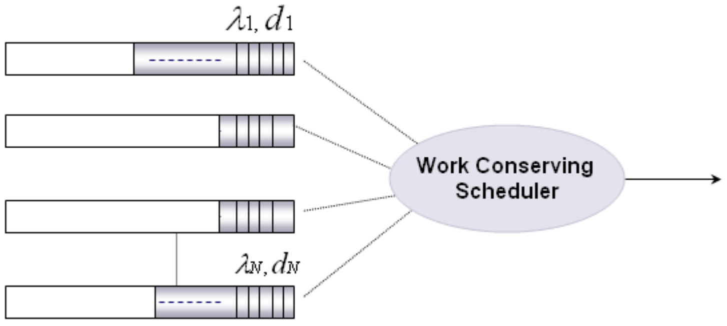


Figure 2.2: Kleinrock's Delay Conservation law for work-conserving schedulers.

According to the law,

$$\sum_{i=1}^{N} \lambda_i d_i = C = \lambda_{total} \cdot d_{FIFO} \tag{2.2}$$

where λ_i is the average utilization of flow i, d_i is the average delay of flow i due to the scheduler, C is a constant independent of the scheduling policy, λ_{total} is the total utilization and d_{FIFO} is the average delay of a single first-in first-out queue with no particular scheduling.

Further,

$$\lambda_n = r_n \mu_n \tag{2.3}$$

where r_n is the mean packet rate in packets/sec and μ_n is the mean per-packet service rate in seconds/packet. This concludes that some flows can receive lower delay only at the expense of longer delay from other flows.

On the other hand, a non-work conserving scheduler may remain idle, even when there are packets buffered in the system. Each packet buffered in the system is associated with an eligibility time and the scheduler transmits the packet from the system when the packet is eligible. This kind of scheduler can alternatively be called a regulator and helps in "smoothing" of packet flows. The output flow is controlled and makes downstream traffic more predictable.

Table 2.1: Comparison of Work Conserving and Non Work Conserving Schedulers

	Work Conserving	Non Work Conserving
Link utilization	Good	Under-utilized
End-to-end delay	Low	High
Jitter	High	Less
Complexity	Simple	Not Efficient [1]

2. Sorted Priority or Frame-based scheduler

   Sorted priority schedulers maintain a global variable referred to as a virtual time or a system potential function. Every time a packet arrives into a system or gets serviced, this variable (also called a timestamp) is updated. The virtual time function may be computed as the expected packet departure time, packet arrival time or any other definition based on the type of scheduling algorithm. Packets are usually sorted in increasing order of their timestamps and are transmitted in

[1]Not very efficient because implementation in routers may require time synchronization.

that order. Start-time Fair Queuing (SFQ), Self-clocked Fair Queuing (SCFQ), Weighted Fair Queuing (WFQ) and Worst-case Fair Weighted Queuing (WF2Q) are examples of sorted-priority schedulers.

On the other hand, frame-based schedulers neither maintain any global variable nor do they require sorting among packets. Instead, time is divided into frames of fixed or variable length. In a fixed size frame, a frame is divided into slots for different sessions. Sessions reserve the maximum amount of traffic they are allowed to transmit during a frame period. As a result, the scheduler can remain idle if sessions transmit less traffic than their reservations over the duration of a frame. Hierarchical Round Robin and Stop-and-Go queuing are constant frame sized frame-based schedulers. In contrast are the variable frame sized frame based schedulers like Weighted and Deficit Round Robin, in which the scheduler visits all the non-empty queues in a round-robin order. A new frame can start early if the traffic from a session is less than its reservation. It can be clearly seen that the former frame-based schedulers are non work-conserving while the latter frame-based schedulers are work conserving in nature.

Table 2.2: Comparison of sorted and frame-based schedulers

	SP	FB
Latency	Low	High[2]
Fairness	Good	Not Fair
Complexity	Not very efficient[3]	Efficient

2.2.2 Shapers and Schedulers

Traffic characteristics and requirements of various flows in networking applications vary widely. Architectural design of networking devices handling these traffic types needs to be well analyzed in order to support the spectrum of traffic demands. Packet

[2]A contending flow with a much higher reserved rate can lead to larger latency for lower reserved rate flows.

[3]This is due to the complexity involved in computing the virtual time function, which purely depends on the choice of the algorithm. Second is the complexity involved in sorting the timestamp values which is a minimum of O(log n) for a maximum of n sessions sharing an output link.

processing in network components like switches, routers or network processors is partitioned into a sequence of processing elements, each processing element having a specific role to play.

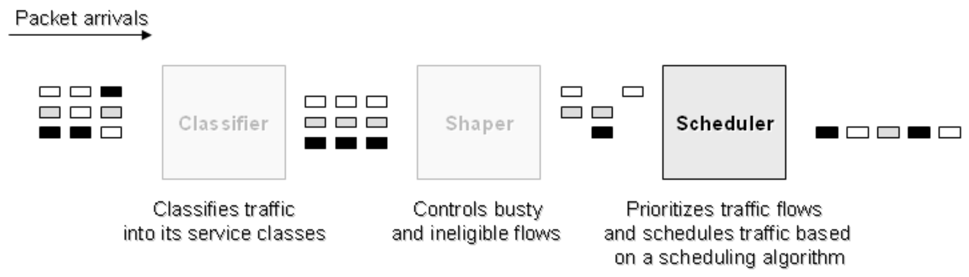

Packet arrivals
Classifier
Shaper
Scheduler
Classifies traffic into its service classes
Controls busty and ineligible flows
Prioritizes traffic flows and schedules traffic based on a scheduling algorithm

Figure 2.3: Packet Processing Elements in a network device.

Figure 2.3 highlights those processing elements in a network device which play a critical role in scheduling. In most cases, either a regulator or policer is used with a scheduler.

- Regulator:

  In most cases, a non work-conserving scheduler for traffic shaping is implemented in this block. When the traffic rate exceeds the maximum output rate, the excess packets are delayed in the regulator by buffering in a queue. The shaping scheduler gradually transmits the eligible packets to the scheduler block over time, resulting in a smooth output rate. However, buffering of packets leads to introducing delay to the traffic flow. Regulation is usually done for outbound traffic.

- Scheduler:

  A work-conserving scheduler, as discussed in section 2.1, is implemented in this block in most cases. These eligible packets are buffered into one of the queues based on the priority class or traffic type. The scheduler, then, selects from the competing eligible packets every time slot, based on the scheduling algorithm used.

Sometimes schedulers are implemented hierarchically. Similar traffic types are first grouped and scheduled by a Level 1 scheduler. Varying traffic types from Level 1 are then buffered and scheduled again by a Level 2 scheduler. Figure 2.4 shows an example of a hierarchical scheduler [4].

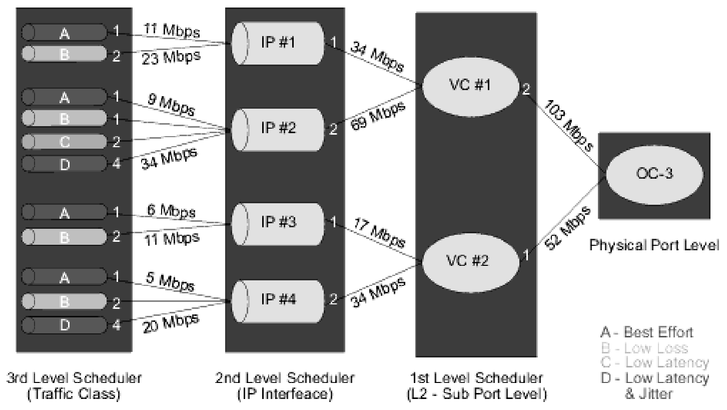


Figure 2.4: An example of a Hierarchical Scheduler

[5] is an example that holds good for those applications which only support a select set of group rates. All sessions with the same group rate are grouped together at Level 1 and are then scheduled according to Smallest Eligible Finish Time policy (SEFF) at Level 2. This grouping architecture reduces the complexity of the scheduler from one that scales with the number of sessions to the one that scales with the number of distinct rate groups.

2.3 Requirements of a Packet Scheduler

The scheduling algorithm assigns different priorities to packets from different applications (varying in service classes or rates of operation) and places these different priority packets in different queues. An arbitrator then makes the choice of the queue to serve in the order of their priority levels.

A packet scheduler must meet the following fundamental requirements:

1. **Bandwidth Utilization**

A scheduling architecture must efficiently utilize the bandwidth, being able to handle bursty sources. For example, statistical multiplexing of flows can help in achieving efficient bandwidth resource allocation [6].

2. **Isolation of Flows**

   An application flow must be isolated from the effects of other competing flows (which could be undesirable misbehaving sessions) by a fair scheduling algorithm. The algorithm must be able to meet the QoS guarantees of a flow when another flow sharing its output link is misbehaving. This characteristic of a scheduler can be applied in network security applications to defend against Distributed Denial of Service (DDoS) attacks, discussed in Section 2.5.

3. **End-to-End Delay Guarantees**

   End-to-end delay in a packet network is usually defined as the time taken for a packet to be transmitted across a network from source to destination. It is a function of the network design and scheduler architecture in its nodes. Maintaining a low end-to-end delay bound for any session is one of the key requirements of a packet scheduling algorithm. The algorithm must provide end-to-end delay guarantees for individual sessions. A work-conserving scheduler is used in most architectures to meet the delay guarantee.

4. **Ease of Implementation**

   A scheduling algorithm must have a simple implementation with minimum time to make a scheduling decision. This forces a hardware implementation of a scheduler for high-speed networks. A software implementation may be considered for packet networks operating at lower speed or handling larger packet size, but the key point to be kept in mind is the rate at which a scheduling decision is made must be kept close to the rate of arrival of the packets.

5. **Scalability**

   A scheduling algorithm implementation ideally scales to a large number of flows in a real-world application and a wide range of traffic rates for these flows. A good scheduling algorithm must perform well for large numbers of flows.

O-notation is a measure of scalability that characterizes the complexity of an algorithm with respect to execution time and implementation size. A scheduling algorithm of $O(1)$ complexity is considered ideal.

6. **Fairness**

   Fairness of a packet scheduling algorithm, in most cases, is defined in terms of bandwidth allocation among competing flows. A fair scheduler must be able to

   - Provide bandwidth guarantees to backlogged flows in a short time span, independent of past usage of the output link bandwidth by the flows.
   - Allocate the unused output link capacity to the competing flows in proportion to their "weights" (or reserved rates).

Requirements 1 to 5 are applicable to any scheduling algorithm while a packet scheduling scheme is considered *fair* if requirement 6 is met. These requirements will help define the metrics to characterize the performance of an algorithm and its suitability for QoS support in packet switched networks. The metrics are enumerated and analyzed in Section 3.4.

2.4 Complexity of a Scheduling Algorithm

In *high-speed network* environments, schedulers are usually implemented in hardware (network processor or routers) rather than in software. The implementation complexity of a scheduling discipline must be taken into consideration, in addition to the algorithmic complexity.

2.4.1 Design Complexity

1. **Time complexity**

   The time-complexity of an algorithm usually depends on the number of active or eligible flows in the scheduler. The goal of a fair scheduling algorithm must be to provide the highest degree of fairness in resource allocation and the least end-to-end delay keeping the time complexity of an algorithm low.

For example, consider any priority-based scheduling architecture with a maximum of **N** flows that shares an output link. It involves the following steps:

- *Priority Tag Computation:* This may involve computing a virtual time function, a weight based priority value, etc.
- *Insertion* into a sorted priority list.
- *Selection of highest priority flow or packet:* The priority given to a flow depends on the scheduling algorithm used. For example, in Earliest Deadline First scheduling discipline, the highest priority value may be given to the packet with the earliest deadline.

In general, the worst-case algorithmic complexity for maintaining a sorted priority queue with **N** arbitrary entries is $O(\log(N))$. A (lower time-complexity) sorted priority queue can be realized in a high-speed network by either using an efficient data structure, an efficient algorithm or by exploiting the parallelism feasible in a hardware implementation.

2. **Regulation before scheduling**

   Most implementations of a *regulator-cum-scheduler* architecture involve two separate priority queue data structure solutions. Moving packets from regulator to scheduler may become a point of concern in high-speed implementations.

3. **GPS-relative delay or Latency**

   General Processor Sharing (GPS) is the fairest packet scheduling algorithm, discussed in more detail in Section 4.1.1. However, GPS is not a realistic algorithm. The difference between the time a packet finishes servicing a scheduling algorithm and it's virtual finish time under a GPS is called *GPS-relative delay* [7]. There is a fundamental trade-off between the computational complexity of a scheduling algorithm and the end-to-end delay bound. Under slightly restrictive but reasonable computational model, the lower bound computational complexity of any scheduling algorithm that guarantees $O(1)$ GPS-relative delay bound is $\omega(log(N))$, where $\omega(f(n))$ is the loose lower boundary of function f(n).

2.4.2 Implementation Complexity

1. **Memory Access Cost**

   Appropriately partitioning memory and access requirements between on-chip and off-chip memory can directly impact the performance of a network system design. On-chip memory is expensive but provides low latency and high bandwidth. On the other hand, off-chip memory is relatively inexpensive but increases the power consumption during the activity on off-chip buses and faces higher bandwidth and latency restrictions. Data dependencies between off-chip access and storage of priority tags in off-chip memory can degrade the performance of the algorithm.

   Designing a scheme that exploits the parallelism of on-chip memories (pipelining) with limited number of off-chip memory references can help in an cost-effective, high speed implementation.

2. **Complexity of basic operations**

   It is advisable to reduce logic in *high-speed* hardware implementations and hence speed up the most basic operations like add, multiply, subtract, divide, compare, memory read and write. Computation of priority tags or virtual time1 functions involve these basic operations.

2.5 Applications of a Fair Packet Scheduler

Fair schedulers have found widespread implementation in network processors, switches and routers, residential and corporate networks. Some applications include:

1. Fair Service Scheduling to Defend Against Distributed Denial of Service (DDoS) attacks.

Fairness in scheduling algorithms is essential to protect flows from other misbehaving flows triggered by deliberate misuse or malfunctioning software on routers or end systems. Rate limiting traffic in packet schedulers can be used to defend against packet flooding and related DoS attacks that allow customers their share of utilization bandwidth even in the face of attacks.

2. Smoothing techniques in multimedia applications.

   Packet Scheduling Algorithms can be deployed for more specific applications, for example, to easily deploy multimedia applications via the Internet. Scheduling algorithms can be used as a smoothing technique to smooth bandwidth requirements for media streaming by buffering video data. Kang and Zakhor [2] propose a class of packet scheduling algorithms based on Earliest Deadline First for streaming media. Experiments with real-time video proved to outperform the conventional sequential sending scheme.

Chapter 3

Understanding QOS in Packet Switched Networks

In the previous chapter, the study of output queue schedulers and its real-world applications was discussed in detail. This chapter starts by describing packet-switched networks and their categories, common service class categories and the need for QoS. Then, QoS metrics that can be used to estimate the performance of a scheduling algorithm in supporting multiple traffic types (commonly referred to as *service classes*) are defined.

3.1 Classifying Packet Networks

Packet networks can be classified in various different ways, based on packet route, type of packet, size of packet, etc. Here we discuss two main ways of classifying packet networks, by packet size or by the way information flows from source to destination.

3.1.1 Classification by Packet Size

Packets Switched Networks can be further classified based on whether the packets used to transmit different traffic types is fixed-length or variable-length. This design choice has a major effect on QoS performance of a packet switched network.

A very common example of fixed-size packet network is Asynchronous Transfer Mode (ATM) which has a packet size of 53 bytes (48 byte data field and 5 byte header). On the other hand, Ethernet 802.3 packet has a frame structure which can range from a minimum of 64 bytes to a maximum of 1500 bytes(inclusive of 14 byte header and 4 byte CRC checksum). Gigabit Ethernet frames support upto 9000 bytes packets, referred to as Jumbo frames.

In general, packet networks handling large packets (called frames) segment the packet into smaller cells or packets at the source and transmit them over the network. These cells or packets are received, reordered and reassembled at the destination, usually handled by a higher layer protocol. For example, ATM Adaptation Layers (AAL) are standard higher layer protocols designed to transmit variable length frames in ATM cells. The applications to be supported play a crucial role in deciding whether a fixed-length or a variable-length packet network is apt, which in turn affects the design of queuing and scheduling algorithm that must be deployed to meet the QoS requirements of the applications. For example, ATM cells would be good choice to support voice traffic because the packetization delay involved is minimum. Voice is sampled at the rate of 8 kHz as 8 bit samples. An ATM cell can be completely filled in 6 milliseconds. For a longer ATM cell, there would be a longer delay in filling up the cell causing degradation in the quality of a voice call. Difference in behavior and methodology based on the size of the packets arise with respect to [8]:

1. Routing methods
2. Delay
   (a) Packetization
   (b) Routing
3. Efficiency in Bandwidth Utilization
4. Overhead
5. Format efficiency

6. Costs

7. Link efficiency

3.1.2 Classification by Packet Route

Various networking techniques use different ways of sending information from the source to the destination. There are two main approaches by which packets, along with the encapsulated information are transmitted through a network from source to destination,

- *Virtual Circuit Networks*: A connection is setup between the source and the destination, following which packets are sent through the connection. The connection is disconnected (tear-down) after the information is transmitted.

- *Datagram Networks*: No specific path is used for information flow from source to destination. Packets sent from the source take the most expedient route and arrive at the destination.

Virtual Circuit Networks are analogous to circuit-switched telephone networks. Asynchronous Transfer Mode (ATM) and X.25 are the most commonly used virtual circuit networks. Packet transfer between the source and destination nodes usually involves three stages:

- *Initial Setup Phase*: A fixed route is established between the end network nodes connecting all the intermediary nodes which are expected to be involved in this session. All packets exchanged during this session use this dedicated route only. An address table, maintained in each network node, is updated with a new entry for this connection when this route is established. Bandwidth is pre-allocated for this session at this stage.

- *Packet Transfer Phase*: When a packet is received at a intermediate network node, the address table is looked up for the entry corresponding to this packet and is routed accordingly.

- *Teardown Phase*: The route used for the packet transfer is disconnected.

Since bandwidth is pre-allocated and the route is fixed, the packets arrive at the destination in order with minimum (but variable) delay.

Datagram transmission based packet-switched networks treat each packet as a separate entity, not associated with any session type. The destination address of the packet is embedded in it and is used by the intermediate nodes to decide its next hop address, in order that the packet takes the most expedient route to its destination. The datagram approach gives the flexibility in choosing a route to the destination, due to which the system can handle congestion of traffic in the intermediate system. When an intermediate system becomes busy, overloaded with excessive traffic or breaks down, an alternate route is taken to reach the destination(as long as a route exists). However, the delivery of the packet cannot be guaranteed all the time, although this is very rarely experienced. Usually, additional error and sequence control is be used to ensure reliability.

Applications which require best effort service can be supported by using the datagram approach. Internet is a very commonly used datagram network which used the IP protocol. Examples of best effort traffic include e-mail traffic, file transfer, remote access and Internet video that can be buffered and played back.

3.2 Traffic Types in Packet Networks

Network convergence has led to the integration of multiple traffic types (for example, triple play) over a single packet network. Each of these traffic types have varying characteristics and developing a single packet networking technology that ensures high performance of all the traffic types is a challenge. If voice, video and data packet traffic were treated impartially by a network node, it would be impossible to have a good quality voice call over a packet network. Impartial treatment of packets is fine for low-bandwidth priority applications like remote access or file transfer but delay-sensitive and bandwidth-intensive traffic like voice will require timely delivery of packets at the destination. Therefore, it is important for network nodes to differentiate between the different traffic types and prioritize the traffic flows. Additionally, traffic prioritization at a network node becomes critical in very high speed networks. Common network traffic can be very broadly classified into three service classes, namely:

1. *Real Time Traffic*: Voice and Streaming video are examples of real time traffic. Real-time packet traffic is characterized by strict deadlines on the end-to-end time delay and delay jitter (refer to section 3.4) and a certain level of packet loss. It is important to use a suitable architecture (like packet encapsulation and negotiation network protocol) to transmit real-time data to ensure that the strict delay bound requirements of real-time traffic is met. Real time traffic may be transmitted either at a constant or at a variable bit rate. Common real-time applications include Voice-over-IP (VoIP), Videoconferencing and IPTV.

2. *Non-Real Time Traffic*: Non-real time traffic usually transports variable bit rate traffic traffic, which, however, attempts to achieve guaranteed bandwidth or latency. No delay bounds are usually associated with this type of traffic. Non-real time traffic is usually given less priority than real-time traffic but a higher priority than best-effort traffic.

3. *Best Effort Traffic*: Email and FTP traffic are examples of best-effort traffic. Best effort traffic does not have strict delay and jitter requirements and is usullay considered the least priority traffic. In most applications, best effort traffic is sent along with traffic sources with allocates bandwidth. Thus, under situations of congestion, the best-effort part of a traffic is usually dropped.

3.3 Achieving QoS

Quality of Service (QOS) is the ability of a network or a networking device to provide better service to selected network traffic type over various networking technologies, efficiently utilizing the available network resources [9], [10]. A packet network may support a multitude of applications ranging from delay-sensitive voice traffic to best-effort file transfer data. It is important to be aware of the available network resources, reserve the necessary resources and meet the requirements demanded by the various applications.

More formally, traffic engineers refer to Quality of Service as the capacity of a packet network in meeting a traffic contract. A traffic contract, also referred to as *SLA* or *Service Level Agreement*, is a specification of bandwidth, latency, jitter and performance guarantee that a network can provide for various traffic types and is usually mutually agreed to by the competing flows [11], [12], [13].

Why do we need QoS? In the previous section, we looked at the common network traffic types and their requirements. In addition to the nature of traffic, packets encounter a number of issues as they traverse through the packet network from the source to the destination node. Some common issues observed are:

- *Throughput*

  A networking device (for example, a switch or a network processor) may support multiple applications and may be able to provide the demanded throughput to all its applications if the resources are pre-allocated. Additionally, faulty conditions may lead to reduced throughput delivery.

- *Delay*

  When a source divides data into multiple blocks, encapsulates them in packets and sends it on a packet network, the packets may be delivered out-of-order at the destination. Additionally, each of these packets may take a different route, be enqueued in long queues in intermediate nodes and reach the destination with variable delays. The delay experienced by a packet to reach its destination is unpredictable.

- *Jitter*

  The variation of delay experienced by packets as they move from source to destination is defined as delay jitter. This jitter is a serious issue for real-time applications.

- *Packet Loss*

  A network node (switch or a router) may drop packets if:

  - Packet is corrupted.
  - The node is not a right destination.
  - Policing schemes are implemented at the node.
  - Queues at the node are full because of increased network utilization.

  Under any of these circumstances, the destination may request the source to retransmit the packet, causing additional delays.

A combination of the problems discussed may lead to inefficiency of the operation of the network system. Special strategies [14] must be deployed to ensure efficient utilization of network resources. Deploying QoS mechanisms in packet networks helps to:

- *Gain control over network resources*

  In addition to pre-allocation of available network resources, it is important for QoS-driven systems to continuously monitor the QoS parameters and dynamically re-allocate resources during run-time to handle sudden variations that may arise in a network node to ensure that a QoS contract is sustained.

- *Differentiate service levels*

  The traffic flows of multiple applications must be distinguished into multiple service levels, so that available network resources can be more carefully distributed to the competing flows. In short, it is important to prioritize the traffic flows.

- *Predictable and efficient use of network resources*

  Monitoring traffic flows with QoS parameters and controlling network resources greatly aids in predicting a traffic flow at a future time. QoS schemes involving congestion control, queue management, traffic shaping or policing ensure efficient use of network resources.

- *Network convergence*

  Integrating voice, video and data transmission on the same network is a challenging task and deploying QoS mechanisms can help in building an efficient Integrated Service Network.

Peak performance of multiple traffic types can be achieved by deploying Quality of Service either in a single network device or end-to-end between source and destination network nodes. Most QoS architectures are implemented to provide the following functions:

- *Prioritize traffic types*

  Most networking devices have a architecture as shown in Figure 2.1. At the input, the packet classifier identifies the characteristics of the traffic type embedded in the packet, prioritizes it and sends it to different queues. A higher priority may be given to higher bandwidth, controlled jitter or reduced latency or reduced loss flow (as demanded by real-time traffic). An arbitration algorithm services the queues taking the priority of the queue into consideration.

- *Control the rate of component traffic flows*

  Traffic shaping or policing is used to ensure that each traffic flow transmits packets at the desired rate. Output rate of bursty traffic is controlled by buffering. In case of heavy network utilization, low priority packets are marked and dropped at a congested node.

- *Avoid network congestion of any traffic flow*

  Scheduling schemes must ensure that prioritizing traffic does not cause a lower-priority traffic to fail. Scheduling algorithms must service queues in an order that will avoid traffic congestion problems for any flow.

- *Handle link efficiency*

  Link efficiency issues are observed in schedulers handling variable packet sizes. Low speed links are encountered with serialization delays for smaller packets in packet networks handling variable-sized packets. For example, a small voice packet may be queued behind a large data packet. This will cause the voice packet to be delayed for a long time. Fragmenting larger blocks followed by interleaving with smaller packets sometimes improves the link efficiency.

However, there are trade-offs inherent in QoS-based networks. As defined earlier, QoS is the ability of a network to provide better service to selected network traffic types over various networking technologies, *efficiently* utilizing the *available* network resources. The QoS offered to a flow can be measured in terms of its fairness, quantified by *fairness index* defined in section 3.4. Fairness is a good measure of quality of service that has been guaranteed to a flow and efficiency of network utilization. It can be seen that fairness index is higher for higher priority traffic but low for lower priority traffic. QoS schemes cannot assure 100 percent fairness for all traffic types.

Designing sophisticated QoS mechanisms increase the cost of a system. Consider a network processor as an example of a standalone network element handling QoS. A fair scheduler with complex data structure is more costly to build compared to a simple FIFO scheduler. The cost of a QoS scheme must be weighed against the improvement in fairness it can provide to all its traffic flows. Depending on the segment of the network under consideration, a decision to trade-off between cost and fairness must be made.

3.4 Metrics for QOS Guarantee

Our study in the previous section shows that a scheduling algorithm should be able to support a range of packet switched networks. It is important to characterize these networks based on some *performance metrics* defined for the scheduler to see if a particular scheduling algorithm can handle the packet switched network domain.

QoS is not measured for an entire network but for a segment of the network and for a particular connection under a certain flow condition. Differences in the quality of a flow are evident when a network is heavily loaded and can be studied by understanding these metrics for different network load conditions. In the following sections, we will define the performance metrics commonly used to study the performance of schedulers. Emphasis has been given to understanding jitter and fairness index which are the key metrics used in this study.

3.4.1 Bandwidth Sharing

Bandwidth Sharing is defined as the maximal data rate that is available for the flow. Consider the case of bandwidth sharing shown in Figure 3.1 where a scheme aims at achieving 50, 30, 50 Mbps when the optimal that can be achieved is 50, 10, 10 Mbps.

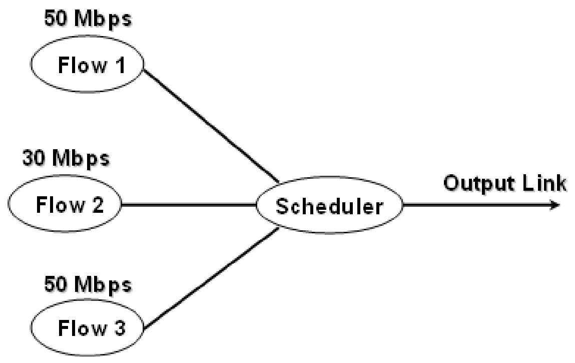


Figure 3.1: Example of bandwidth sharing.

This is a simpler case of only 3 flows accessing a output link. In real-world applications, tens of simultaneous applications on a desktop may be remotely accessing the server facilities. The flow bandwidth share depends on the specific bandwidth sharing policy used in the network, e.g., max-min sharing, proportional sharing, size-based bandwidth sharing or minimum delay policy. Deciding the appropriate scheduling algorithm to be used is a step towards deciding on how the available bandwidth is shared among the different applications. The fairness that is provided by an algorithm can be evaluated based on *Jain's fairness index* described in Section 3.5.

3.4.2 Packet Delay

Packet Delay can be defined as the elapsed time for a packet to transit the network segment or a networking device.

ITU-T defines the QoS Metrics for delay as:

- *IP packet transfer delay (IPTD)*: IP packet transfer delay is defined for all successful and error packet outcomes. If the packet is fragmented, the value corresponds to the last fragment.

- *Mean IP packet transfer delay*: Mean IPTD is the arithmetic average of IPTD for a population of interest.

On the other hand, IETF provides two definitions of packet delay:

- *One-way Delay Metric for IPPM* [15]: This is one of the basic quantitative characteristics of network delay. The metric is defined as the difference between wire-time of first bit of the Type-P packet at the transmitter and wire-time of the last bit at the receiver. The metric involves an upper bound of delay and considers that packet lost and the value of metric undefined if the last bit does not arrive within that predefined period of time. If the packet is fragmented and if, for whatever reason, reassembly does not occur, the packet will be deemed lost. Note that measuring one-way delay requires clock synchronization between the sender and receiver.

- *Round-trip Delay Metric for IPPM* [16]: At wire-time T, the first bit of the Type-P packet from source to destination is sent; after receiving the packet at destination, a Type-P packet back to source is sent immediately. The last bit of packet is received at wire-time T+dT, dT being the value of round-trip delay. An upper bound of delay is given and if the packet does not arrive inside this interval, it is considered lost and the value of metric is undefined. As time is measured only at one site, round-trip delay does not require clock synchronization between source and destination.

The delay experienced by *packet k in a queue* can be defined as :

$$d_k^N = |D_k^N - A_k^N| \tag{3.1}$$

where, A_k^N, D_k^N and d_k^N denote the arrival time, departure time and queuing delay of packet k in queue N.

3.4.3 Delay Jitter

Packets sent on a packet switched network are most often delivered irregularly to the destination due to some of the reasons discussed in Section 3.3. Particularly in the case of real-time applications, this variation in network transfer delay (at network-level) or packet departure delay from a network device (at a system level) can cause degradation of quality of service. Additionally, bursty traffic patterns increase the delay jitter of a flow.

Two kinds of jitter play a major role in network QoS, *delay jitter* and *rate jitter*. Delay jitter bounds the maximum difference in the total delay of different packets arriving at a destination, assuming that the packet source is perfectly periodic. Rate jitter bounds the difference in packet delivery rates at various times. This is measured as the difference between the minimal and maximal inter-arrival times (inter-arrival time between packets is the reciprocal of rate) [17]. Both these measures are very useful for real-time voice and video applications. For the purpose of this study, we will be focus on delay jitter performance metrics.

Several definitions for delay jitter have been defined, of which two are discussed here [18]. The first measure is called *2-point PDV (Packet Delay Variation)*, as defined by ITU-T SG13 in Rec.I.380 [19]. The 2-point packet delay variation (v_k) for an IP packet k between the source *SRC* and the destination *DST* is the difference between the absolute IP packet transfer delay(x_k) of the packet and a defined reference IP packet transfer delay $d_{1,2}$, between those same measurement points.

$$v_k = x_k - d_{1,2} \tag{3.2}$$

The reference IP packet transfer delay, $d_{1,2}$, between the source and the destination is the absolute IP packet transfer delay experienced by the first packet between those two measurement points[18]. Alternatively, the first packet delay can be replaced with average delay of the population of packets.

The second measure, called *1-point CDV* (Cell Delay Variation), was defined by IETF [20], particularly for ATM environments. According to this definition, the variation of delay is derived from the arrival time of cells, measured against an expected arrival time. Supposing a stream of cells transmitted with constant period T, the 1-point CDV of the cell k is the difference y_k between the actual arrival time a_k and a reference time c_k. The reference time (expected arrival time) is defined as follows:

$$c_0 = 0 \tag{3.3}$$

$$a_0 = 0 \tag{3.4}$$

$$If \quad c_k \geq a_k, \tag{3.5}$$

$$then \quad c_{k+1} = c_k + T \tag{3.6}$$

$$else \quad c_{k+1} = a_k + T \tag{3.7}$$

Based on the 1-point CDV, the delay jitter of a *packet k in a queue* [20] can be defined as the difference of queuing delay of this packet and the preceding packet in that queue, i.e.,

$$j_k = |d_k - d_{k-1}| \tag{3.8}$$

where d_k is the queuing delay of packet number k. Additionally, we define the aggregate jitter experienced by all packets that have been served by a queue N at a point in time τ as the sum of the jitter of each packet served by queue N from time 0 to τ, i.e.

$$\begin{aligned} j_{avg}(\tau) &= \sum_{t=0}^{\tau} |d_k^N - d_{k-1}^N| && (3.9) \\ &= \sum_{t=0}^{\tau} |(D_k^N - D_{k-1}^N) - (A_k^N - A_{k-1}^N)| && (3.10) \\ & && (3.11) \end{aligned}$$

where A_k^N, D_k^N and d_k^N denote arrival time, departure time and queuing delay of packet k in queue N respectively as shown in Figure 3.2.

There has been a lot of research focusing on estimating the maximum delay jitter bound J_{max} for different types of packets [21] and for variable packet sizes. Verma, Zhang and Ferrari [22] discuss the feasibility of bounding the delay jitter in real-time channels and control of delay jitter in real-time communication in packet switched networks. Figure 3.2 gives an idea of the bounds on delay distribution curve [23].

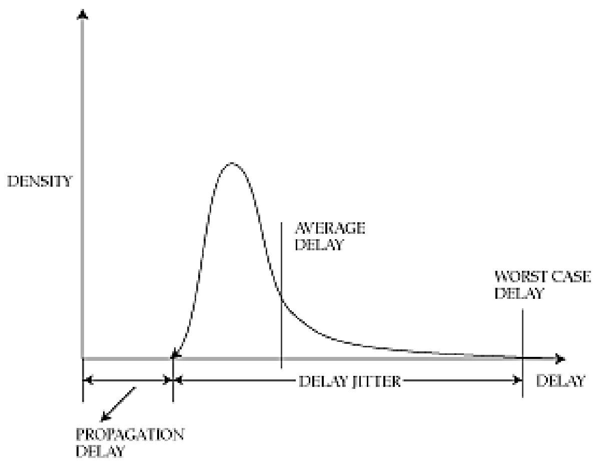


Figure 3.2: Understanding Delay and Delay Jitter.

3.4.4 Packet Loss

Packet loss is the term given to losing information packets for a flow. Packet loss may happen in a flow due to:

- High input rate leading to queue overflow.
- Corruption of packets.
- Re-ordering within the flow.

Packet loss directly affects the reliability of the connection. Excess packet loss results in a less reliable connection.

For the IP networks, the ITU-T Recommendation I.380 [19] defines three related QoS Metrics, namely:

- *IP packet loss ratio (IPLR)*: IP packet loss ratio is a ratio of total lost IP packet outcomes to total transmitted IP packet in population of interest.

$$IPLR = \frac{N_{lost}}{N_{transmitted}} \tag{3.12}$$

The applications can react on packet loss in three different ways [24].

1. *Fragile*: An application is unreliable if the packet loss exceeds certain threshold.
2. *Tolerant*: Multiple packet loss threshold levels are defined. The application can tolerate packet loss upto a particular level, but the higher the packet loss, the application is less reliable.
3. *Performance*: The application can tolerate even very high packet loss ratio but its performance can be very low in high packet loss ratio.

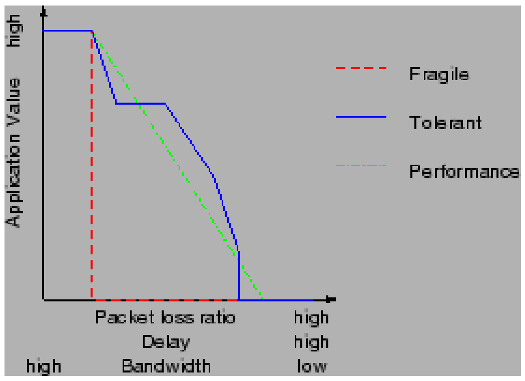


Figure 3.3: Loss graph

- *IP packet error ratio (IPER)*: IP packet error ratio is the ratio of total errored IP packet outcomes to the total of successful IP packet transfer outcomes plus errored IP packet outcomes in a population of interest. This metric is usually defined in terms of Bit Error Rate (BER) or Frame Error Rate (FER).

$$IPER = \frac{N_{erroneous}}{N_{erroneous} + N_{successful}} \tag{3.13}$$

- *Type-P One-way Packet Loss (IETF)* [25]: IETF defines a packet loss metric for IPPM, *Type-P One-way Packet Loss.* The packet is considered lost if it fails to arrive to its destination in a reasonable period of time. This time threshold is a parameter of the metric. Corrupted packets are counted as lost. The measurement methodology relies on the one-way delay.

An example of an scheduling algorithm that takes packet loss into consideration is *DWCS* or *Dynamic Window-Constrained Scheduling.* DWCS was originally designed to be network packet scheduler limiting the number of late or lost packets over a window-size of packets in loss-tolerant and/or delay-constrained heterogeneous traffic streams [26].

Service classes, discussed earlier in this chapter, represent a set of traffic types that demand specific packet delay, loss and jitter characteristics from the network on a per-hop basis. Networking applications with similar characteristics and performance requirements fall into the same service class and similar metric bounds.

We considered the quantitative metrics so far. Some of the other qualitative QoS parameters that may be considered for metrics are Cost, Compliance, and Security [27].

3.5 Fairness Index

In the previous few sections, we defined the performance metrics that can be used for the purpose of studying the behavior of various scheduling algorithms. However, our main objective is to provide fairness to all the flows in the system. Hence, it is important to measure the algorithm's fairness to individual flows and define a *fairness index* for the entire system. Depending on the application under consideration, an algorithm can be considered *fair* depending on any of the performance metric defined so far. For the purpose of this study, we will focus on the *bandwidth sharing* or *throughput* fairness characteristics of the scheduling algorithm.

3.5.1 Jain's Fairness Index

ATM Forum Traffic Management Specification version 4.0 [28] defines a fairness index, called Jain's fairness index, to evaluate the fairness of the distribution of the available bandwidth among the individual flow. Consider a scheduling algorithm allocating its output bandwidth to N flows. If x_i is the observed throughput in the $i-th$ flow (where $0 \leq i \leq N$) and r_i is the expected throughput or fair share for connection i (i.e., r_i can be defined as an equal share of the bottleneck link capacity $r_i =$ Capacity of Output Link/N), then Jain's fairness index [29] is defined as:

$$F\left(\frac{x_0}{r_0}, \cdots, \frac{x_N}{r_N}\right) = \frac{\left(\sum_{i=0}^{(} N-1)\frac{x_i}{r_i}\right)^2}{n\sum_{i=0}^{(} N-1)(\frac{x_i}{r_i})^2} \tag{3.14}$$

Jain's fairness index produces a normalized number between 0 and 1, where 0 indicates the *greatest unfairness* and 1 indicates the *greatest fairness*. Lets consider Jain's fairness index for the example shown in Figure 3.1. The scheme gives 50, 30, 50 Mbps, when the optimal is 50, 10, 10 Mbps. Let the measured throughput be t1, t2,..., tn. Use any criterion (e.g., max-min optimality) to find the fair throughput p1, p2,..., pn [30].

$$\begin{aligned}
\text{Normalized Throughput: } x_i &= \frac{t_i}{p_i} \\
\text{Fairness Index} &= \frac{(\sum(x_i))^2}{n\sum x_i^2} \\
\text{Example: 50/50, 30/10, 50/10} &= 1, 3, 5 \\
\text{Fairness Index} &= \frac{(1+3+5)^2}{3(12+32+52)} \\
&= \frac{92}{3(1+9+25)} \\
&= 0.81
\end{aligned}$$

Several fairness index definitions have been proposed by researchers [31], [32]; some examples include Gini index, Variance, Coefficient of Variation. However, Jain's fairness index is good measure of fairness because of the following characteristics:

- *Scalability*: It can be applied to any number of flows. Fairness index like co-variance is not defined for small n. Performs really well for large number of flows.

- *Scale independent*: Jain's index is independent of the scale. Additionally, Jain's Fairness Index is a generic metric that can be applied to any resource.

- *Bounded* : Jain's Fairness index always lies between 0 and 1 or 0 and 100. Variance, standard deviation, and relative distance are not bounded.

- *User perception*: Jain's index has an easier user perception. Higher value of this index implies more fairness. Other indices like variance do not have this, as higher variance means less fairness

- *Continuous*: The index is always a continuous function unlike indices like min-max.

Chapter 4

Packet Scheduling Algorithms

Various scheduling algorithms and methods have been studied extensively in the literature. In this chapter, we summarize a few of them, that we have simulated for this research. In the first section, we start with an example of an ideal scheduling algorithm, the *Generalized Processor Sharing* algorithm, before describing other algorithms [33], [34], [35]. We then discuss Weighted Jitter EDF, a modification to the Earliest Deadline First (EDF) algorithm.

4.1 Background Work

4.1.1 Generalized Processor Sharing (GPS)

Generalized Processor Sharing or GPS is an idealized fluid flow scheduling model deploying uniform network resource sharing to achieve QoS guarantees such as fair bandwidth allocation and end-to-end delay bounds in communication networks. GPS is a work-conserving scheduler in which all the participating connections are simultaneously provided with their fair service share.

Consider a GPS scheduler serving N flows with an output link rate of R. Assume that each flow i is continuously backlogged in a time interval (τ_1,τ_2] during which the traffic served by the server; let's denote the amount of service received by flow i in this time interval as $S_i(\tau_1, \tau_2)$ [36]. Let $w_1, w_2, w_3, \cdots w_N$ be the weights associated with each of the flows. A GPS scheduler is one for which,

$$\frac{S_i(\tau_1, \tau_2)}{S_j(\tau_1, \tau_2)} \geq \frac{w_i}{w_j} \tag{4.1}$$

for each backlogged flow i, j in the interval τ_1 to τ_2.

Summing over all flows j we get,

$$S_i(\tau_1, \tau_2) \sum_j w_j \geq (\tau_2 - \tau_1) R w_i \tag{4.2}$$

Hence, flow i is guaranteed a minimum fair share rate g_i equal to

$$g_i = \frac{w_i}{\sum_j w_j} R \tag{4.3}$$

In other words, the service that a flow receives in a GPS system is no worse that an equivalent dedicated link with a capacity of g_i (as shown in Eq.4.3).

Parekh and Gallager [37] show a clear example of how the flexibility of GPS multiplexing can be used effectively to control packet delay when combined with appropriate rate enforcement.

GPS is considered an ideal scheduling algorithm due to some of its unique properties.

- *Property 1*: Let us define r_i to be the average rate of flow i. Then, as long as $r_i \geq g_i$, the flow can be guaranteed a throughput of ρ_i independent of the demands of the other flows. In addition to this throughput guarantee, backlog in flow i will always be cleared at a rate greater than g_i.

- *Property 2*: The delay of an arriving flow i bit can be bounded as a function of the flow i queue length, independent of the queues and arrivals of the other flows.

- *Property 3*: By varying the w_i's, we have the flexibility of treating the flows in a variety of different ways. For example, when all w_i's are equal (say equal to 1), the system reduces to uniform processor sharing.

$$r_i = \frac{R}{N} \tag{4.4}$$

As long as the combined average rate of the sessions is less than R, any assignment of positive w_i's yields a stable system. For example, a high-bandwidth delay-insensitive flow i can be assigned g_i much less than its average rate, thus allowing for better treatment of the other flows.

- *Property 4*: Most importantly, it is possible to make worst-case network queuing delay guarantees when the sources are constrained by leaky buckets.Thus, GPS is particularly attractive for flows sending real-time traffic such as voice and video.

Although GPS has these attractive properties, it is not implementable due to two main reasons. GPS works under the assumption that the scheduler can serve multiple flows simultaneously and that the traffic is infinitely divisible according to Eq. 4.3. Secondly, GPS is an idealized scheme which does not transmit packets as entities; it rather treats them as infinitesimal quantities. Both these assumptions do not hold in practice, since only one flow can be served by a scheduler at a given time, and an entire packet has to be served before serving another packet. However, GPS serves as a good scheme to compare and evaluate other scheduling disciplines.

4.2 Weighted Jitter Deadline Scheduling

In this section, we discuss a simple scheduling algorithm called Weighted Jitter Earliest Deadline First Scheduling or *WJ-EDF*, a modification of Earliest Deadline First scheduling policy [38], designed with a focus to reduce the delay jitter experienced by packets in output buffer schedulers [39]. A work-conserving scheduler is the most appropriate for an output buffer scheduler as it significantly decreases the average end-to-end delay experienced by packets and helps in the fair distribution of available bandwidth between competing flows.

Lets us consider a simple WJ-EDF Scheduler serving m queues as shown in Figure 4.1. Packets arriving from flows with different rates are sent to different queues, i.e., packets from flows with the highest supported input rate go into Q_1 while the flows with the lowest supported input rates goes into queue Q_m. In order to give different

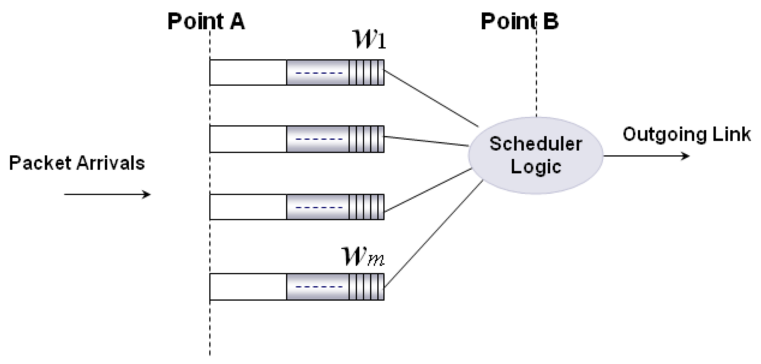


Figure 4.1: Simple WJ-EDF scheduler

priorities to the m queues, different weights are assigned to the m queues, namely $w_1, w_2, w_3, \cdots, w_m$. The weights are chosen in such a way that the queue which is the least delay and jitter-tolerance must be given the highest weight value.

It is important to timestamp packets in any deadline-based scheduling policy. *Integer timestamping* is one possibility which is used in the implementation of this algorithm, discussed in section 5.2. The advantage of using integer timestamps as opposed to floating-point timestamps is that integer timestamps have simpler hardware implementations. *Aggregate jitter* used in this algorithm is implemented according to the definition in section 3.4. The applicability of WJ-EDF has been studied for two categories of output buffer schedulers [39], namely *Simple WJ-EDF Schedulers* and *Shaped WJ-EDF Schedulers.*

4.2.1 Category 1 : Simple WJ-EDF

Figure 4.1 shows a structure of an output buffer-scheduler module where this algorithm can be applied. This algorithm is applied at two parts of the module,

- at point A, when packets enter the output buffers
- at point B, when the scheduler decides which is the next packet that must be transmitted on the output link.

On arrival of a packet or a cell k at entry point A in queue i, packets are timestamped with an entry time s_i^k. The deadline f_i^k of packet k in the queue i is calculated as

$$f_i^k = s_i^k + d_i^k \tag{4.5}$$

where d_i^k is the delay bound allowable for packet k.

According to [40], [41], and [42], EDF is known to be an optimal scheduling algorithm at a switch. Optimality is defined in terms of the schedulable region associated with the scheduling policy [41]. Consider m connections with traffic envelopes $R_i(t)$, $(i = 1, 2, \ldots, m)$, sharing an output link of rate C. Suppose that each of the m connections has an upper bound, d_i, on the scheduling delay that packets from that connection can tolerate; these bounds define a vector $d' = (d_1, d_2, \cdots, d_m)$. Then the scheduling region of a scheduling discipline ϕ is defined as the set of all vectors d' that are schedulable under ϕ. It has been proven in [43], [44] that EDF has the largest schedulable region of all scheduling disciplines under the condition

$$\sum_{i=1}^{m} R_i(t - d_i) \leq Ct \tag{4.6}$$

where traffic is assumed to be fluid and $R_i(t) = 0, \ \forall t < 0$.

In this algorithm, we will define $d_i^k = f(l_i^k, R_i, w_i)$ where R_i is the input rate for flow i with packet k of length l_i^k. More specifically,

$$d_i^k = \left\lceil \frac{l_i^k}{R_i \cdot w_i} \right\rceil \tag{4.7}$$

Using the input rate considers the peak requirement on the flow and sets a tighter bound on the delay.

```
ALGORITHM ON ARRIVAL OF A NEW PACKET $k$ IN FLOW $i$
1.  Timestamp the newly arrived packet with an entry time $s_i^k$
2.  Compute the deadline $f_i^k$ of packet k in queue i

        $f_i^k = s_i^k + \lceil (\frac{l_i^k}{R_i \cdot w_i}) \rceil$

3.  Enqueue the packet $k$ in queue $i$
```

At point B, the scheduler algorithm checks all the backlogged queues and looks for the head-of-line packet which has the earliest deadline (i.e., lowest timestamp) and starts transmitting packets in order of increasing deadline. Every queue maintains a log of the *aggregate jitter*, j_i experienced by the packets it transmitted. Hence, every queue has an estimate of the service that has provided to it. When a bias condition occurs, i.e., when two queues Q_r and Q_s have the same deadline, the priority is given to the queue with a highest aggregated jitter value , based on the "greedy-choice" property, i.e., the decision point in the algorithm is based on what seems best at the moment.

```
WJ-EDF Algorithm to select the next packet to be scheduled
1.  Extract the deadline of all backlogged queues
2.  Sequentially check all the backlogged queues whose head-of-line
packet has the smallest deadline
3.  If two flows i and j have the same deadline, select the flow
with greater value of$j_i$.
4.  Complete scanning all the backlogged queues.
5.  Retrieve the head-of-line packet from the high priority queue
obtained in Step 1-4 and transmit.
```

This algorithm works most appropriately for small packets or when large packets are fragmented into smaller blocks as the delay bound considered in Eq.4.7 does not take into consideration the time taken to enqueue or dequeue a large packet. Ideally, it would be appropriate to timestamp a large packet when its boundary just enters point A and delay bound must completely take into account the time that will be taken by a big packet to leave the queue. However, with a small packet, the enqueue-dequeue time is small and can be considered negligible.

4.2.2 Category 2: Shaped WJ-EDF

Figure 4.2 shows the structure of the shaper-scheduler combination discussed in Section 2.2.2, seen in many packet switch architectures. In this category of schedulers, the packets arriving into the system first enter the shaper. Once the packets become

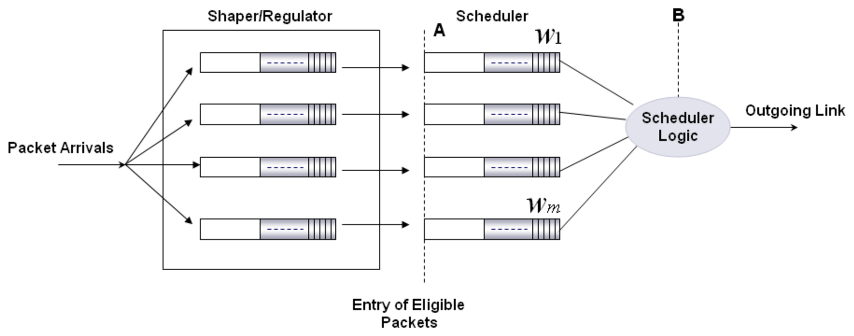


Figure 4.2: Shaped WJ-EDF scheduler structure.

eligible for transmission, the packets are sent to the scheduler. All packets waiting to become eligible for transmission wait in the shaper, which is like a waiting room, which in most implementations, is a non-work conserving scheduler. Once the packets are eligible, they are transferred to the output buffers to be scheduled to the output link. Definition of eligibility of a packet to leave the regulator or shaper to be transmitted on the output link varies is based on the implementation of the shaper being delay-jitter or rate-jitter based.

Combining shaping and scheduling changes the arrival pattern of the packets entering the scheduler. Understanding the arrival patterns of packets into the scheduler based on the traffic type supported can help optimizing the algorithm further. Adding a shaper to the scheduler does not degrade the behavior of the combined module, it only decreases the jitter and the delay experienced by the outgoing packets. There may be multiple variations in the implementation of the eligibility criteria of packets. For example, consider the example of a system receiving IP-AAL5-ATM packets. The shaper may send ATM cells that are eligible to the scheduler.The total jitter during the transmission of an IP packet is hence a function of the jitter experienced by the cells carrying this IP packet. Another variation of implementation of eligibility criteria, is the scenario where a large IP packets is fragmented into smaller fragments to aid efficient scheduling. Eligible fragments are then sent to the scheduler. Modifications can be made to the algorithm to accommodate such fragmentation.

If the fragment size, p_k, of packet k is known, then the number of equal-sized fragments $K^{"}$ in packet k is computed as

$$K^{"} = \lceil \frac{l_k^i}{p_k} \rceil \tag{4.8}$$

The algorithm on arrival of a fragment of a packet k can be modified to the following:

```
WJ-EDF Algorithm on arrival of a new packet $k$ in flow $i$
1.  Timestamp the newly arrived packet with an entry time $s_i^k$
2.  Compute the deadline $f_i^k$ of packet $k$ in queue $i$
        $f_i^k = s_i^k + \lceil (\frac{p_i^k}{R_i \cdot w_i}) \rceil$
3.  Enqueue the packet $k$ in queue $i$
```

The aggregate jitter experienced in transmission of packet k is defined as as

$$j_k^i = \sum_{n=1}^{K^{"}} j_n^i \tag{4.9}$$

Bias conditions may occur when two flows have the same deadline. Under such bias conditions, the flow with the higher value of aggregate jitter is selected in order to help the higher jitter flow to reduce its jitter.

WJ-EDF has a delay bound close to or better than EDF. The better delay bounds are obtained by appropriately scaling the weights w_1, w_2,...,w_m.

WJ-EDF incorporates the best attributes of Weighted Fair Queuing and Earliest Deadline First scheduling schemes. It is a simple algorithm which is applicable both to fixed and variable sized packets. It is extremely accurate for the case of small packets. WJ-EDF shows an $O(V)$ complexity, where V is the the number of backlogged queues. An algorithm that exhibits lower time complexity can be devised by using a better data-structure instead of a simple priority list, as discussed in section 2.4.

Chapter 5

Simulation Study

In this chapter, we discuss the implementation of *packet scheduling schemes* and the simulation study that was performed on Agere's APP650 network processor. We start our discussion with a brief overview of the APP650 architecture, followed by a description of the simulation environments, the experiments we performed to compare the scheduling algorithms and the results of the simulations.

5.1 Architecture Overview

Advanced Payload Plus, APP650, is Agere's third generation network processor and the simulations for this study were performed on its architecture . APP650 is a pipelined, multi-threaded processor that simultaneously analyzes and classifies multiple packets at the same time, monitors data traffic and schedules output data traffic, all at wire speed, operating at a clock rate of 266MHz. It includes sophisticated scheduling and shaping capabilities to support both packet or cell-based traffic. APP650 has integrated 10/100/1000 Ethernet MACs and provides flexible interface options including GMII/SMII (with integrated Gigabit Ethernet, 10/100 MACs), PMA (Gigabit Ethernet), SPI-3 and UTOPIA.

Figure 5.1 gives a high level system view of APP650's architecture. The major components of the APP650 architecture include the following:

- *Data Path Input Interface*: Consists of two 32-bit buses that can be configured

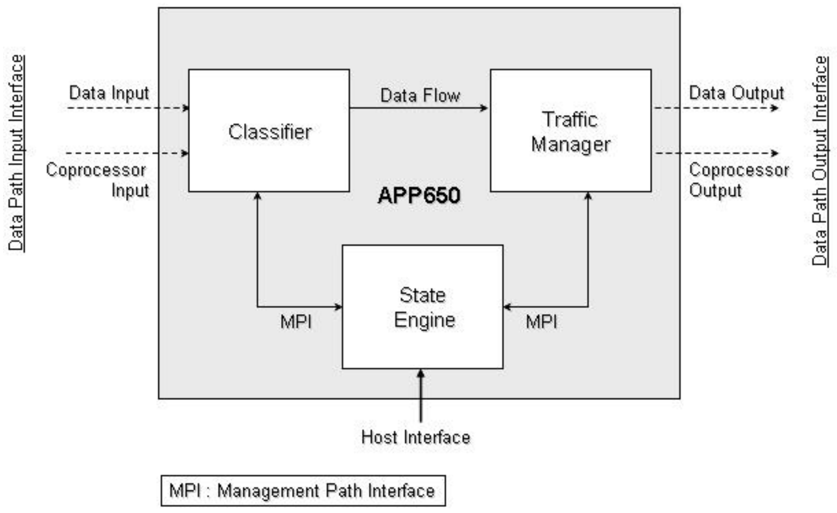


Figure 5.1: APP650 Architecture: Block Diagram.

to receive data from different sources and in different combinations. Incoming data traffic is sent to the Classifier.

- *Classifier*: A programmable, high-speed, on-chip data classification engine that simultaneously analyzes and classifies multiple incoming data packets before sending them to the traffic manager for modifications and transmission.

- *State Engine*: This compute engine works with the classifier and the traffic manager, seamlessly integrating the components with the host interface. It also supports high-speed, packet or cell-based traffic policing, thereby enabling the use of multiple policing algorithms for different protocols.

- *Traffic Manager*: Receives input from the classifier and performs traffic management operations, i.e., schedules and shapes the traffic, generates packets for network operations when necessary, and transmits the data to the network.

- *Data Path Output Interface*: Transmits data received from the traffic manager to the Stream Editor (SED), a compute engine that makes necessary modifications to the packets. Data is sent from the SED to a downstream device on a 32-bit configurable data output interface and a 32-bit configurable coprocessor interface.

- *Host Interface*: Provides interface to a host processor. The host processor is used exclusively for slow-path processing; it is not used for pattern recognition, classification, queuing or traffic management operations.

In this simulation study, we will focus on the shaper-cum-scheduler in APP650's traffic manager. The *Sched-650* acts as a scheduler and shaper in APP650 enabling multiple combinations of scheduled and rate-limiting traffic management and shaping. In each timeslot, *Sched-650* needs to decide which user (or flow) can send out a packet, so as to guarantee bandwidth of the flows and reduce the burstiness of the output traffic. Figure 5.2 gives a top level block diagram view of *Sched-650*.

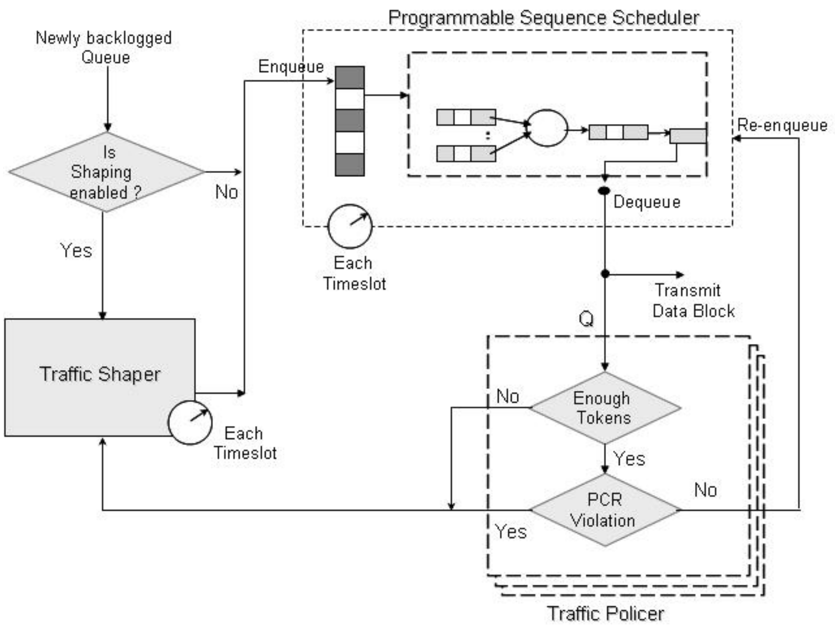


Figure 5.2: Sched-650 Architecture: Block Diagram.

The *Sched-650* has three major blocks, as shown in Figure 5.2:

- *Traffic Shaper*: This block is like a waiting room. A queue with packets is sent to the shaper when it is not eligible to be served and is guaranteed to leave the shaper when it becomes eligible. It is a non work-conserving scheduler, used to control the delay time of the queues to make sure that the queues do not get too much service. It is also used to control the service intervals of the queues to reduce the burstiness of the output traffic. Packets from the traffic shaper are sent to the corresponding programmable sequence scheduler.

- *Programmable Sequence Scheduler*: This scheduler consists of multiple *active lists* and *rate controllers*.

  1. *Active Lists*: When an active list receives eligible queues from the traffic shaper, it appends the list of queues to the tail of the active list. In each timeslot, the arbitrator selects at most one non-empty active list based on a scheduling algorithm, which is focused on as a part of this study. The head-of-line queue from the selected active list is sent to the rate controller. Each queue of packets is associated to one rate controller. The scheduler checks the rate controller identity of the selected queue and sends the queue to the corresponding scheduler list.

  2. *Rate Controllers*: Each rate controller can serve multiple queues and has a rate set for it by the user. The configured rate controls the output rate of each rate controller. Every timeslot, another arbitration logic selects the head-of-line queue from the backlogged lists and sends it to the traffic policer.

- *Traffic Policer*: This block consists of multiple policing instances. After a queue is sent out from the rate controller, the queue can transmit one data block. The queue is sent to the traffic policer to ensure that the queue meets the reserved rate but does not violate the peak rate.

5.2 Simulation Environment

Experiments for understanding the behavior of the different scheduling policies were based on two models of *Sched-650*, discussed in this section.

5.2.1 Setup 1: Behavioral C model

A behavioral C model of *Sched-650* was used to study the performance of the different scheduling algorithms. The simulator reads the queue parameters from the testcase data file every timeslot and feeds the queues to Sched-650. The simulator assumes that the queues fed to Sched-650 are always backlogged. It is important to make this assumption because all jitter and throughput analysis are studied only on backlogged queues. The results of the simulation of the various scheduler algorithms are stored in the result file. Figure 5.3 shows a top level block diagram of the C model.

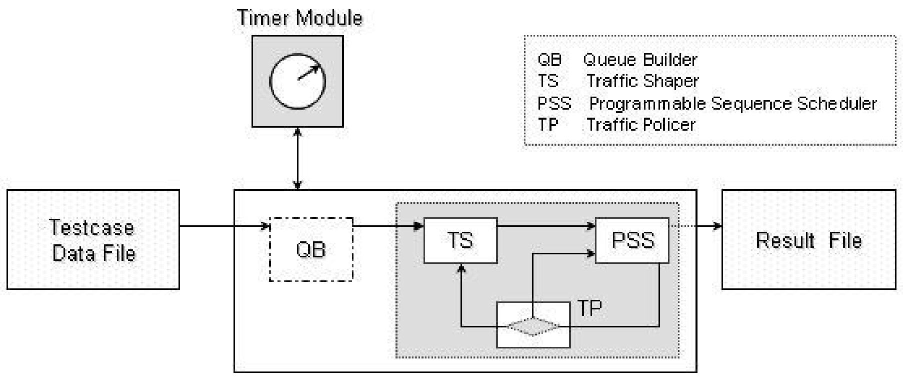
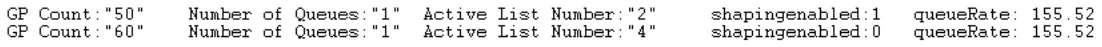


Figure 5.3: Behavioral C Model: Block Diagram

The major blocks of the C model are:

- *Testcase Data File*: This input data file contains input parameters for *Sched-650*. The structure of the input parameters are as shown in Figure 5.4.



```
GP Count:"50"    Number of Queues:"1"  Active List Number:"2"    shapingenabled:1   queueRate: 155.52
GP Count:"60"    Number of Queues:"1"  Active List Number:"4"    shapingenabled:0   queueRate: 155.52
```

Figure 5.4: Testcase Data File format.

The timer module reads the next element from the input file, when all the previous queues whose parameters have been read from the input file have been built by the queue builder and sent to *Sched-650*.

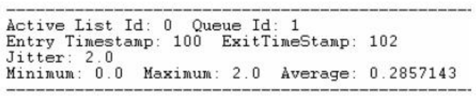
- *Timer Module*: The timer module maintains a 64-bit counter which increments every time unit. Every time unit, the module checks if all the queues that have been read from the input file have been fed to the *Sched-650* and if all the queues have already been fed, reads the next set of queues from the input file. The timer module also controls the transfer of eligible queues from the traffic shaper to the PSS and the transmission of packets from the PSS.

- *Queue Builder*: The queue builder builds queues and feeds the queues to *Sched-650* at the time unit mentioned by the input parameter. The parameters of the queue are shown in Figure 5.4. Depending on whether a queue needs to be shaped or not, the queue is sent to the traffic shaper or to an active list in the PSS.

- *Sched-650*: The three blocks of *Sched-650* discussed in Section 5.1, namely the traffic shaper, the programmable sequence scheduler and the traffic policer have been implemented in this block. The *Sched-650* receives its queues from the queue builder module and, based on whether shaping is enabled or disabled, sends the queue to the traffic shaper or to an active list in the programmable sequence scheduler. Every time a queue enters or leaves the PSS, an entry or exit timestamp is sent to the the result file with the *queue ID* and the *activelist ID*.

- *Result File*: This data file captures the timestamp values dumped by the *Sched-650* block. In addition to the entry and exit timestamp of a queue into the active list, the *queue ID* and the *activelist ID* are also sent to the result file. These values can be used to study the arrival pattern of the traffic into each active list, the jitter experienced by the queues, the scheduler behavior and hence the departure pattern of the queues from the active list.

The behavioral C model was developed to understand the functionalities of each block of *Sched-650* and to identify early issues that may occur in implementation of different scheduling policies. However, only a small testcase scenario can be studied using this behavioral C model. In order to study the effect of the various scheduling



```
------------------------------------------------
Active List Id: 0  Queue Id: 1
Entry Timestamp: 100  ExitTimeStamp: 102
Jitter: 2.0
Minimum: 0.0  Maximum: 2.0  Average: 0.2857143
------------------------------------------------
```

Figure 5.5: Result File: Jitter Measurements.

algorithms on a real sample of traffic flows, a more realistic model of *Sched-650* was required. Agere's SDE was used as a more realistic setup for the simulation study.

5.2.2 Setup 2: Agere's SDE

Agere's SDE or Software Development Environment is a software-based environment with a Java-based graphical user interface which enables development, simulation, performance analysis and optimization of data-plane functionalities on the network processor well before the devices are available. The SDE includes device configuration capabilities, compilers, a debugger, a cycle-accurate simulator, a traffic generator and analyzer with metrics window within a single framework. Figure 5.6 shows the configuration panels available in SDE's GUI.

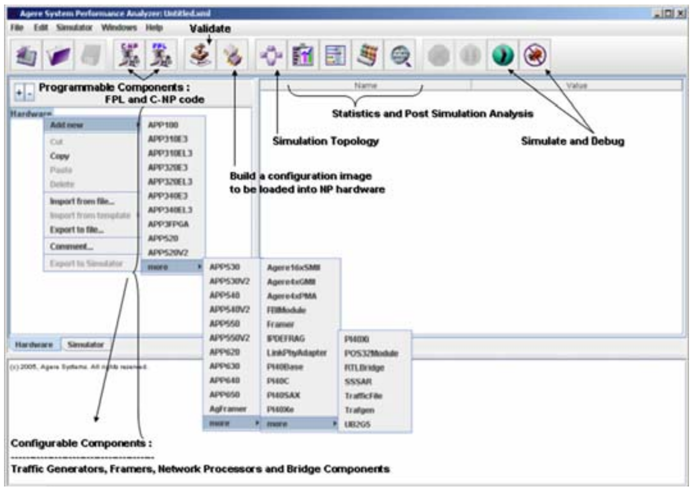

Agere System Performance Analyzer: Untitled.xml
File Edit Simulator Windows Help
Validate
Programmable Components :
FPL and C-NP code
Statistics and Post Simulation Analysis
Simulation Topology
Simulate and Debug
Build a configuration image
to be loaded into NP hardware
Configurable Components :
Traffic Generators, Framers, Network Processors and Bridge Components

Figure 5.6: Agere SDE's Java-based GUI.

For the purpose of this simulation study, modifications were made to the scheduler code in the traffic management section of the SDE. The modifications are enumerated below:

1. *Integer timestamping*: A 64-bit integer timestamp was maintained to monitor the timing behavior of the queues in *Sched-650*. Maintaining an integer timestamp as a multiple of TU (TU represents a time unit, calculated as a function of core clock speed) simplifies the analysis of the simulation results. Handling floating point timestamp values can result in increased chances of error.

   A 64-bit timestamp was chosen to maintain timing values as maintaining a smaller-sized timestamp would result in overflow (or wrapping) issues during extended simulation study periods, particularly in high packet rates. Using a 64-bit timestamp, however, increases the memory overhead and may face bandwidth constraints in hardware implementations.

2. *Jitter measurements*: In order to study the queuing delay and jitter experienced by the packets or queues in the active list, it is important to timestamp the queues at two instants:

   - *Entry*: Every time a new packet or a queue enters the active list, ready to transmit a block of data.
   - *Exit*: Every time a queue is chosen by the scheduling algorithm to leave the active list, depending on the algorithm used.

   Modifications were made to the SDE to timestamp the queues at the entry and the exit of the active list. The timestamp values are written to a result file exactly in the same format as was done in the behavioral C model, as shown in Figure 5.4.

3. *Scheduler Algorithms*: This implementation is the core part of this research. Modifications were made to the SDE to support multiple PSS scheduling algorithms based on a selection. Depending on the scheduling policy selected, the PSS works with the corresponding arbitration logic. Jitter and fairness measures were studied for multiple algorithms discussed in Section 3.3.

5.3 Metrics and Simulation Variables

In this section, we will define the metrics used in studying the performance of *Sched-650* for various scheduling policies followed by a description of all the parameters or simulation variables used in our simulation study.

- *Performance Metrics*: A number of QoS performance metrics were defined in section 3.4. In our simulation study, *fairness index* and *jitter* will be the key metrics that we will monitor to characterize a scheduling algorithm. Let us briefly recall the definitions of these metrics (discussed in detail in section 3.4).

  1. *Jain's Fairness Index* for fairness:

     Defined by ATM Forum Traffic Management Specification version 4.0, this fairness index is used to evaluate the fairness of the distribution of the available bandwidth among the individual flow. *Jain's fairness index* for N flows is defined as:

     $$F\left(\frac{x_0}{r_0}, \cdots, \frac{x_N}{r_N}\right) = \frac{\left(\sum_{i=0}^{N} \frac{x_i}{r_i}\right)^2}{n \sum_{i=0}^{N} (\frac{x_i}{r_i})^2} \tag{5.1}$$

     where x$_i$ is the observed throughput in flow *i* and r$_i$ is the expected throughput or fair share for flow *i*.

     Jain's fairness index produces a normalized number between 0 and 1, where 0 indicates the *greatest unfairness* and 1 indicates the *greatest fairness*.

  2. *1 Point CDV* for jitter:

     Defined by IETF, particularly for ATM environments, 1-Point Cell Delay Variation is derived from the arrival time of cells, measured against an expected arrival time. Supposing a stream of cells transmitted with constant period T, the 1-point CDV of the cell k is the difference y_k between the actual arrival time a_k and a reference time c_k. The reference time (expected arrival time) is defined as follows:

$$c_0 = 0 \tag{5.2}$$

$$a_0 = 0 \tag{5.3}$$

$$If \quad c_k \geq a_k, \tag{5.4}$$

$$then \quad c_{k+1} = c_k + T \tag{5.5}$$

$$else \quad c_{k+1} = a_k + T \tag{5.6}$$

3. *2-Point PDV* for jitter:

   Defined by ITU-T SG13 in Rec. I.380, 2-point Packet Delay Variation (v_k) for an IP packet k between the source SRC and the destination DST is the difference between the absolute IP packet transfer delay (x_k) of the packet and a defined reference IP packet transfer delay $d_{1,2}$, between the same measurement points.

$$v_k = x_k - d_{1,2} \tag{5.7}$$

   The reference IP packet transfer delay, $d_{1,2}$, between the source and the destination is the absolute IP packet transfer delay experienced by the first packet between the two measurement points. Alternatively, the first packet delay can be replaced with an average delay of the population of packets as a nominal delay d_{avg}, which is tabulated in Table 5.1 for our experiments.

- *Simulation Variables*: In this simulation study, we will vary the simulation parameters 1 to 4 mentioned below. The values of the simulation variables used for the simulation study are listed below :

  1. *Scheduling Algorithms*: The algorithms that we will study as a part of this work include First-Come First-Serve, Round Robin, Maximum Queue Length First, Maximum Waiting-time first, Priority Queuing, Earliest Deadline First and Weighted Jitter Earliest Deadline First. The choice

of algorithm were made to compare the performance of deadline based algorithms with other easily implementable algorithms. The first five algorithms listed above can be easily implemented on hardware.

2. *Packet sizes*: IPv4 packets over ATM-AAL5 was considered for this study as an example of cell-based traffic. In addition, IPv4 over ATM-AAL5 is a good choice to test shaped WJ-EDF scheduling. IPv4 payload sizes of 20, 64, 148 and 1000 bytes are considered in this study as an example of small and large sized packets.

3. *Number of Traffic flows*: Single-input, two-input and four-input traffic flows are used in this simulation study. Only small number of traffic flows were first considered to understand and compare the behavior of deadline based algorithms with other algorithms.

4. *Packet Rates*: All the experiments are performed with the scheduler operating at OC-3. The packet input rates for traffic chosen for this study include OC-1 (51.84 Mbps), OC-3 (155.52 Mbps) and OC-24 (1.244Gbps). These packet rates represent some of the commonly used packet rates in real-time aaplications.

5. *Traffic Types*: As a part of this research, we will only focus on Constant Bit Rate Traffic. Constant bit rate traffic is a good representation of voice traffic.

5.4 Observations and Performance Analysis

In this section, we will analyze the observations of our experiments on the simulation setups. The various scheduling algorithms were implemented on both setups. Similar results were observed in the behavior of both the setups for the different scheduling algorithms. Here is an example of the similarity of behavior seen in both setups. A testcase with two input traffic flows, one with an input rate of OC-3 and another with an input rate of OC-24 was generated by a traffic generator and fed to the APP650 with a *priority queuing* scheduler operating at OC-3 in setup 2. The pattern of queue arrivals into the traffic shaper and PSS was studied on setup 2. A testcase data file was built for setup 1 based on this observations. It was observed that the timing behavior of the packets departing from the output of *Sched-650* observed was similar on both setup 1 and setup 2, as shown in Figure 5.7.

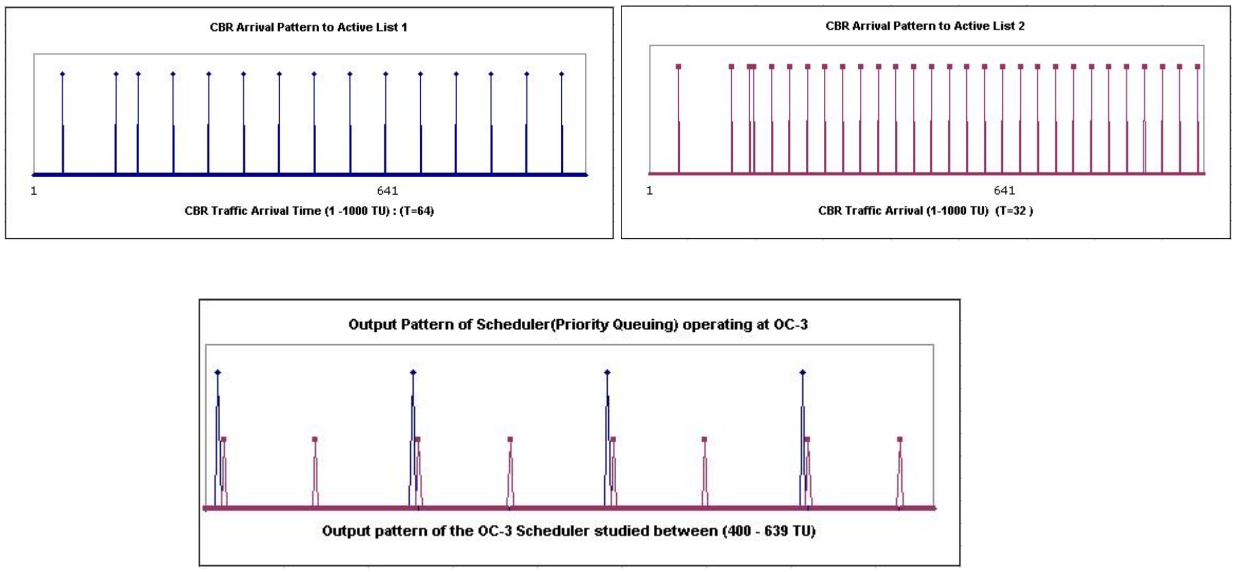


Figure 5.7: Behavior of packets from two traffic flows at the output of Sched-650.

However, only a small set of test patterns of queues could be studied on setup 1. To a have a more realistic idea of the results, most of the analysis was made with the results observed on setup 2. We will analyze these observations in the following sections.

5.4.1 Fairness Index Analysis

The *fairness* of a scheduling algorithm is defined as discussed in section 3.4.5. Experiments were performed to observe the impact of the different scheduling policies on the fairness of a scheduler to traffic flows. In addition to this, the effect of the behavior of packet size and number of flows on fairness was studied. For all the experiments in this section, tests were repeated multiple times, following which 95 percent confidence intervals [45] on the average value of the output metric were computed and have been included in the tables (Refer Appendix A).

Impact of the number of flows

Before studying the effect of a scheduling algorithm, the behavior of the scheduler was studied for multiple input flows. In this experiment, we consider three testcase scenarios of one, two and four input flows, all flows serviced by a single scheduler operating at *OC-3* speed, with *priority queuing.* Each input flow generates a constant bit rate traffic stream, with *varying input rates.* The traffic flows carry IPv4 packets over ATM-AAL5 with a IPv4 payload size of 20 bytes because this can be contained in a single ATM cell, to undeestand the simples scenario.

- *Scenario A: Single Traffic Flow*

  When *single* traffic flow is input to the scheduler, there is no issue of bandwidth sharing or competition. The only parameter that affects the throughput of the scheduler is the input rate of the traffic flow. Flows with input rates greater that the scheduler rate receive a throughput ρ less than 1. This is clearly observed in Figure 5.8 which shows the throughput of a scheduler operating at OC-3 for various input rates. Table A.1 shows the service guaranteed to the single input traffic flows of different input rates. Hence, by providing differ input rate traffic to the scheduler, / we can study the maximum service that can be guaranteed by a OC-3 scheduler.

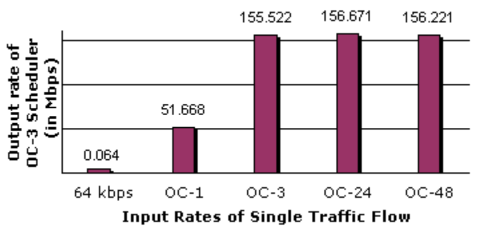


Figure 5.8: Single flow: Output rate of a scheduler operating at OC-3 for various input rates.

On the other hand, when *multiple traffic flows* are input to a scheduler (i.e., more than one traffic flow), the traffic generator simultaneously generates packets to each of the traffic flows at the specified input rates. In our experiments to study the impact of the number of flows on fairness, we will consider two scenarios of multiple flows, scenario B of two simultaneous traffic flows and a second scenario C of four simultaneous traffic flows.

- *Scenario B: Two Traffic Flows*

  Incoming packets from traffic flow 1 are recognized by their IP destination address and input rates and are sent to to their corresponding active lists. The performance of the scheduler was studied in multiple experiments performed by varying the input rates of the two traffic flows. The service that the scheduler provides to each of the input traffic flows is shown in Figure 5.9 and Table A.2.

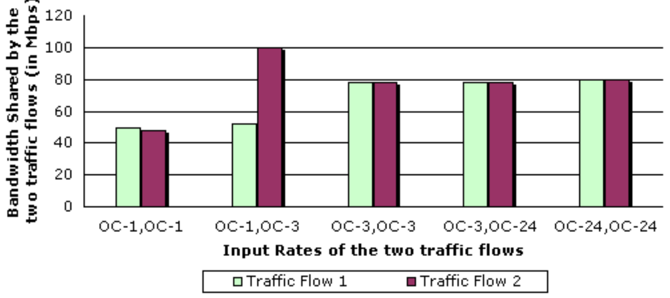


Figure 5.9: Two traffic flows: Output rate of a scheduler operating at OC-3, for various input rates.

It can be observed from Figure 5.9 that the scheduler tries to maximize the fair distribution of bandwidth between the two competing traffic flows. When the combined bandwidth requested by both the competing flows is less than the operating rate of the scheduler, the throughput of both the flows ρ_1 and ρ_2 is close to 1. In all other cases, the throughput of either of the flows is affected. A fair scheduling algorithm will distribute bandwidth to its competing flows according to its input rates. It can be observed from the case of OC-3/OC-24 input pair that the scheduler bandwidth is not fairly shared. This is also evident in *Jain's Fairness Index* tabulated in Table A.2.

- *Scenario C: Four Traffic Flows*

Incoming packets from the four flows are sent from active lists 1 to 4 respectively. As the number of traffic flows sharing a scheduler increases, the fairness that a scheduler provides to the competing flows becomes clearly evident as seen in Table A.3. The throughput achieved by the individual flows decreases as multiple flows load the scheduler.

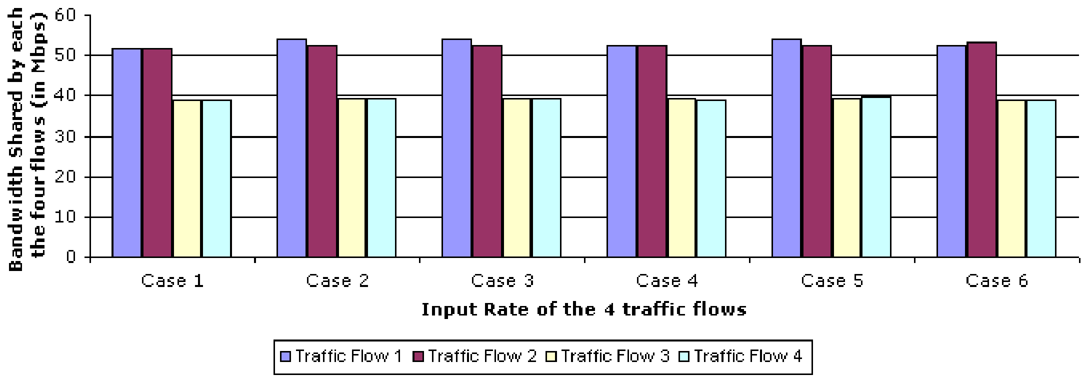


Figure 5.10: Four traffic flows: Output rate of a scheduler operating at OC-3, for various input rates.

It can be clearly seen in Figure 5.10 that priority queuing performs unfairly when one of the competing flows has a much higher input rate or bandwidth requirement compared to the other flows. We will be comparing the fairness of different scheduling algorithms later in this section.

Impact of the packet size

Experiments were conducted by fixing the number of flows and the scheduling policy and varying the IPv4 payload size. It was observed that there was negligible change of the fairness of the scheduler with change in the packet size. This was repeated for test cases with single and two input flows as can be seen in Table A.4 and Table A.5 respectively . Hence, from these observations, we can conclude that the impact of packet size is negligible on the fairness of an algorithm, when other parameters are kept constant.

Impact of the scheduling algorithm

Scenarios B and C, discussed earlier in this section, were repeated to study the behavior of different scheduling schemes in the programmable sequence scheduler of the *Sched-650*. For a clearer understanding of the effect of scheduling algorithms, experiments were conducted for the case of OC-3/OC-24 input rate pair which showed most unfairness in priority queuing. Table A.6 and Figure 5.11 show the results of experiments of scenario B, while Table A.7 and Figure 5.12 show the results of scenario C for case 5. The weights assigned to the different flows for the case of WJ-EDF are $w_1 = 4$, $w_2 = 3$, $w_3 = 2$ and $w_4 = 1$, where w_i is the weight assigned to active list i.

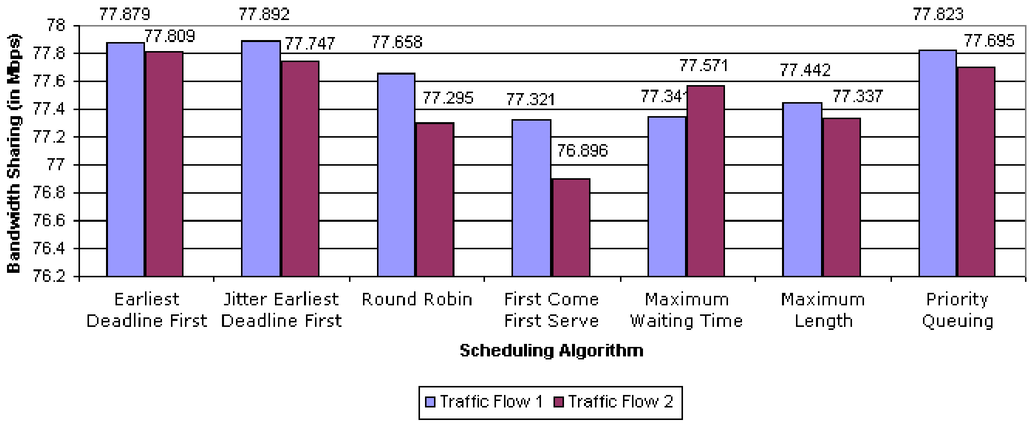


Figure 5.11: Two input traffic flows: Comparison of scheduling algorithms.

As can be seen in Table A.6, the fairness index is similar and the difference in the performance of the scheduler for the different scheduling policies is not very evident. However, the difference in the throughput of flows and the fairness index of the various scheduling scheme becomes evident in the case of scenario C. As the number of flows increases, the fairness of the scheduler gets affected.

It can be seen from Table A.7 that deadline-based scheduling policies, namely EDF and WJ-EDF provide better throughput to their flows and better fairness in comparison to the performance of other scheduling schemes. EDF and WJ-EDF define the deadlines of their packets based on the size of the packets and the input rate of the traffic. WJ-EDF, in particular, can scale the delay bounds that are set for

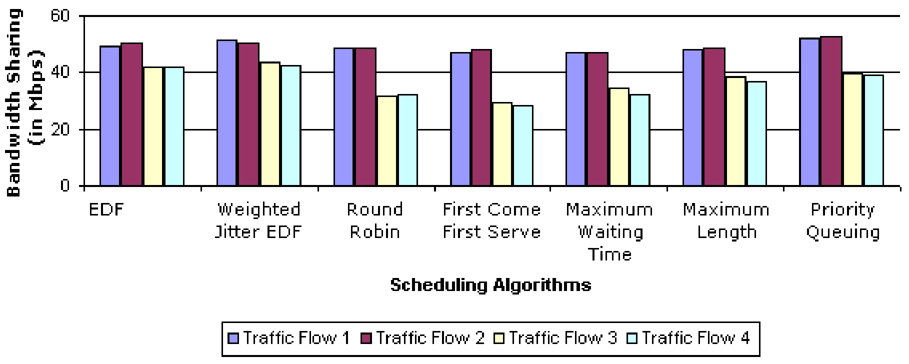


Figure 5.12: Four input traffic flows: Comparison of scheduling algorithms.

its packets by scaling them with appropriate weights.

First-Come First-Serve (FCFS) scheduling algorithm shows the poorest performance in terms of fairness and throughput. Round Robin scheduler performs better that a FCFS scheduler. It can also be observed from Table A.7 that maximum waiting time and maximum queue length first schedulers perform better than a round-robin scheduler. This is because round robin scheduling is not done based on the the basis of any priority measure and hence is observed to be a relatively unfair scheduling scheme.

5.4.2 Jitter Analysis

1-Point CDV and 2-Point PDV were calculated for the incoming packets of various sizes and incoming rates, in accordance to the definition in Section 3.4.3. The inter-arrival pattern was studied for each of these cases and the values of T and d_{avg} have been tabulated in Tables 5.1 and 5.2 respectively. The values in the bracket indicate the inter-arrival times of cells of the same packet.

It can be observed from the above tables that arrivals of the packets are seen at much closer intervals as the input rate of the incoming packets increases. Additionally, the packets arrive into the active lists from the shaper and hence, do not follow the same timing pattern of CBR packets as sent by the traffic generator. However, even packets arriving from the shaper are seen to follow a similar behavior as CBR packets

Table 5.1: T for different arrival rates and packet sizes (IETF).

Payload Size	OC-1	OC-3	OC-24
20	96	64	32
64	192 (96)	128(64)	64(32)
148	384(96)	256(64)	128(32)
980	2016(96)	1344(64)	672(32)

Table 5.2: d_{avg} for different arrival rates and packet sizes (ITU-T).

Payload Size	OC-1	OC-3	OC-24
20	96	63.352	31.82
64	190(12)	128(64.136)	63.26(31.6)
148	380(32)	254(64.198)	130.4(30.73)
980	2012(32)	1312(64.248)	670.6(31.3)

from the traffic generator but with a different inter-arrival gap.

The packet inter-arrival patterns are measured in terms on TU or *time units* which are calculated in terms of the core clock speed of 266.67 MHz.

Impact of the number of flows

Jitter measurements were also made when experiments were performed for scenario A, B and C in section 5.4.1. In addition to the study of the effect of the number of flows on the jitter for payload sizes of 20 bytes, we will study the same for payload sizes of 64, 148 and 1k bytes. Studying the jitter values for a range of packet sizes for increasing number of flows gives a better estimate of the performance of different scheduling algorithms. Tables A.11 to A.14 and Figures 5.13 to 5.20 show the 1-Point CDV and 2-Point PDV for different packet sizes.

- *Scenario A: Single Traffic Flow*

  It can be observed from tables that 1-Point CDV and 2-point PDV are negligible, particularly in the case of single traffic flow, even for small packets. This is because the input rate of the traffic is equal to the rate of operation of the scheduler. Hence, very minimal jitter is experienced in single flow operations.

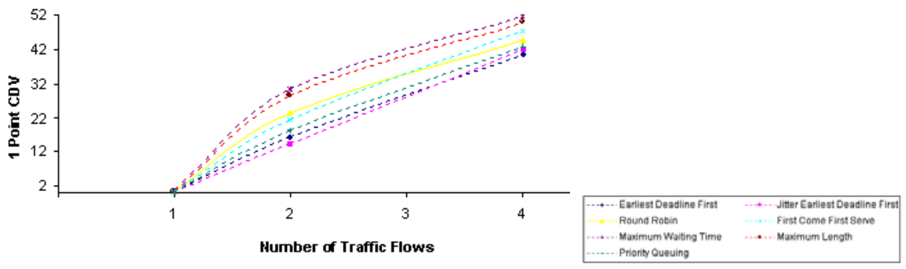


Figure 5.13: Comparison of 1-Point CDV with number of flows for Packet Size = 1 Cell

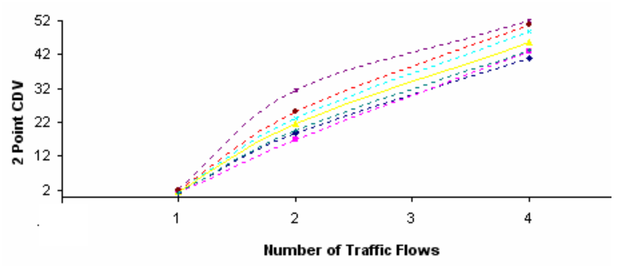


Figure 5.14: Comparison of 2-Point PDV with number of flows for Packet Size = 1 Cell

However, the jitter experienced by a flow with input rate greater than the scheduler operating rate increases the jitter experienced by the packets. Irrespective of the scheduling policy used, the jitter experienced in the case of single flow is negligible. 1-Point CDV and 2-Point PDV have shown a similar behavioral pattern for all packet sizes.

- *Scenario B and C: Multiple Traffic Flows*

  The jitter increases as the number of flows competing for the scheduler bandwidth increases. This can be observed in all cases as seen in figures. The differences in the jitter behavior of different scheduling algorithms become evident in the case of two traffic flow inputs, particularly when the sum of the input rates exceed the scheduler rate. In scheduling algorithms like priority queuing, the jitter experienced by a higher priority flow is much lesser than a lower priority flow. We will compare the jitter experienced by packets when configured in different scheduling schemes later in this section.

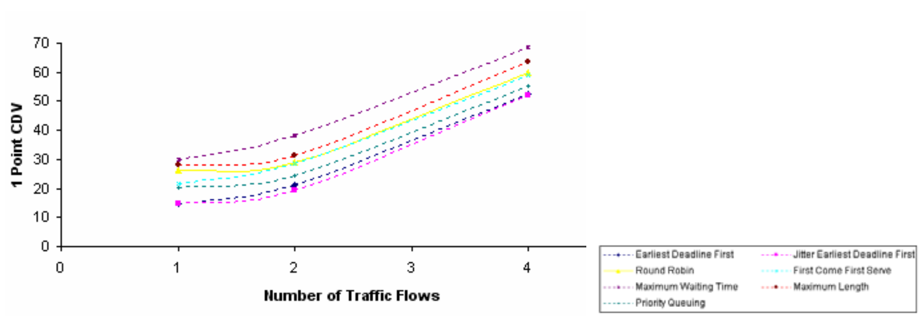


Figure 5.15: Comparison of 1-Point CDV with number of flows for Packet Size = 2 Cells

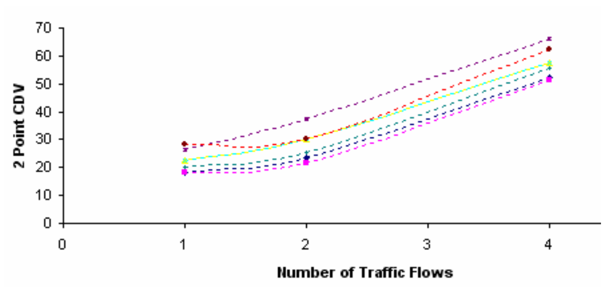


Figure 5.16: Comparison of 2-Point PDV with number of flows for Packet Size = 2 Cells

Impact of the packet size

Figure 5.21 and Figure 5.22 show the jitter behavior of single flow traffic for different packet sizes with input rate of traffic being OC-24.

Negligible or close-to-zero jitter was seen in the case of single flows with input rate less than OC-24. Details of 1-Point CDV and 2 Point PDV for single flow for different packet sizes is shown in Tables A.11 to A.14. The jitter increases as the packet size increases. This is because as the packet size increases, the average jitter experienced depends on the jitter experienced by each the component blocks and hence, increases the aggregate jitter experienced by the packet.

Similar behavior is noticed in the case of multiple flows as shown in Figures 5.21 to 5.24.

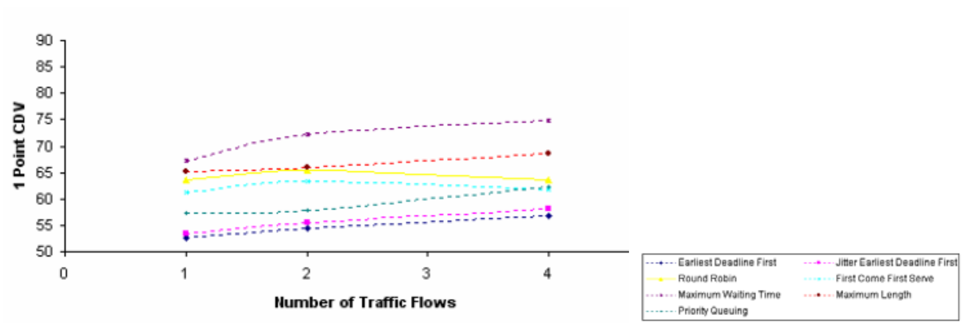


Figure 5.17: Comparison of 1-Point CDV with number of flows for Packet Size = 3 Cells

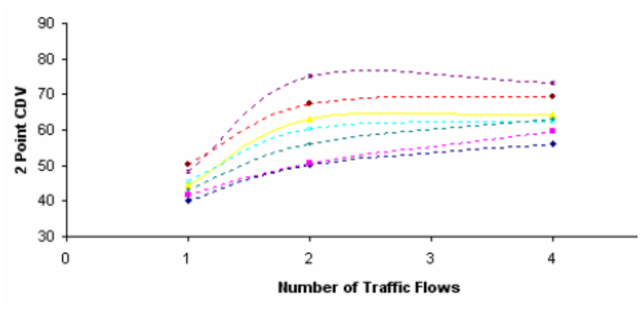


Figure 5.18: Comparison of 2-Point PDV with number of flows for Packet Size = 3 Cells

Impact of the scheduling algorithm

As can be seen in Figures 5.13- 5.24, the scheduling algorithms have a significant impact on the jitter experienced by the packets serviced by them. This difference is particularly evident in the case of multiple flows. The average jitter of all the flows was considered. The worst case jitter was experienced when the maximum waiting time scheduler was configured. The best case jitter was seen in the case of EDF algorithm, followed by WJ-EDF. This further proves that a deadline-based scheduling scheme reduces the jitter experienced. In the case of WJ-EDF, lowest jitter is seen in the traffic flow which has the highest priority.

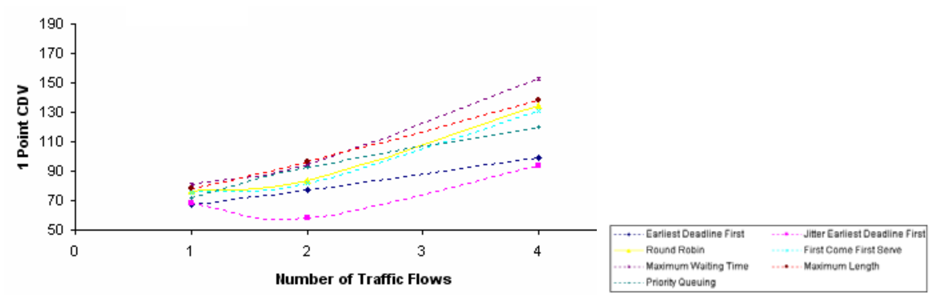


Figure 5.19: Comparison of 1-Point CDV with number of flows for for Packet Size = 8 Cells

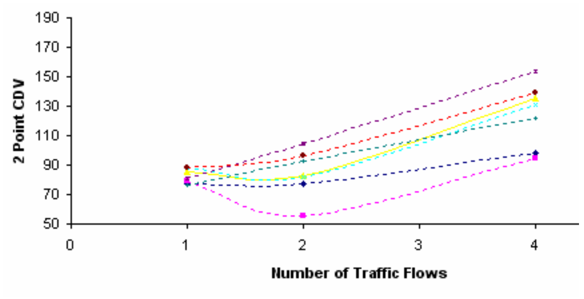


Figure 5.20: Comparison of 2-Point PDV with number of flows for Packet Size = 8 Cells.

5.5 Weighted-Jitter EDF

In this section, we will analyze the observations of experiments performed with the PSS scheduler of *Sched-650* configured for *Weighted-Jitter EDF* and operating at a rate of 155.52 Mbps.

Weighted-Jitter EDF is a version of EDF scheduling, in which the weights assigned to the different active lists help in setting tighter delay bounds or deadlines to queues or packets serviced by a active list. Weights w_1, w_2,...,w_m can be linear or exponential depending on the following decisions :

- Range of input rates of flows a queue must service.
- Packet sizes of the flows that a queue may service.

Figure 5.25 (Table A.8) and Figure 5.26 (Table A.9) shows a representation of the deadlines enforced by an EDF algorithm to packets in flows with different packet sizes

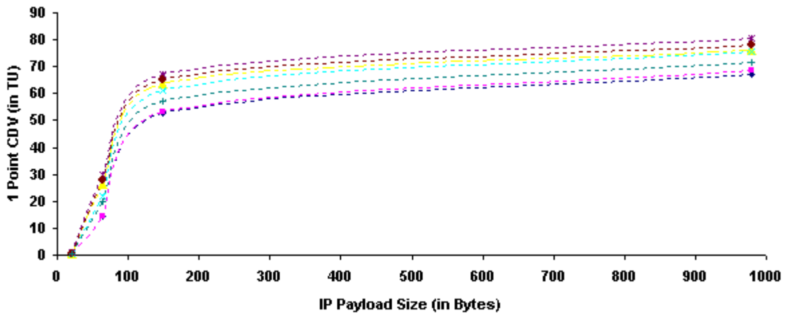


Figure 5.21: Comparison of 1-Point CDV with packet size for Single Flow.

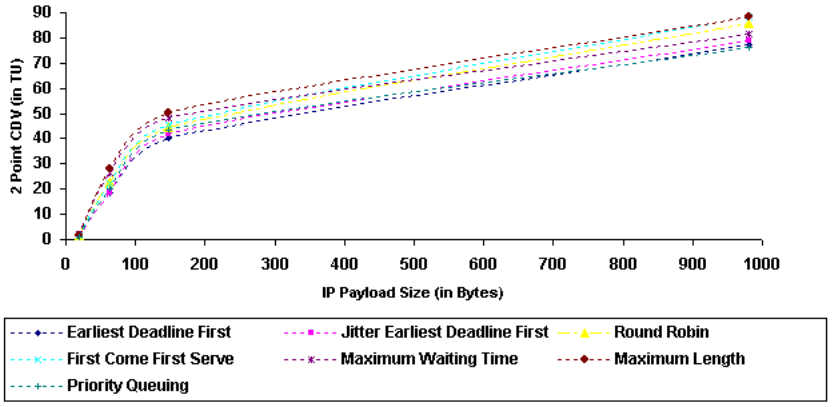


Figure 5.22: Comparison of 2-Point PDV with packet size for Single Flow.

and different input rates respectively. In addition, we can also observe in the case of WJ-EDF that scaling the delay bound by a weight factor helps in setting tighter deadlines for departure of packets. In this example, the weight w_1 or scaling factor used is 4.

In these graphs, the active list under consideration is assigned a weight of 4. The graphs clearly show that

- The weighted jitter EDF helps in setting tighter deadlines for packet departures; and,

- The priority given to a lower bit rate traffic can be raised by scaling the active

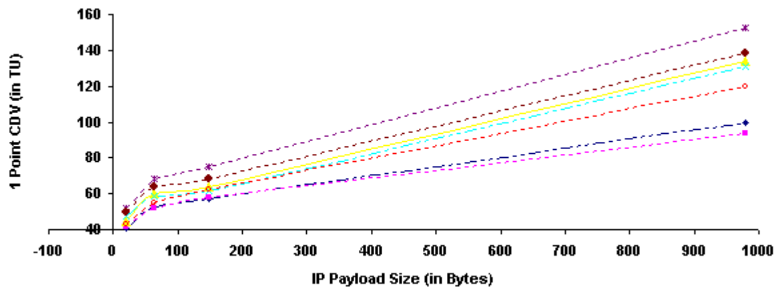


Figure 5.23: Comparison of 1-Point CDV with packet size for four Flows.

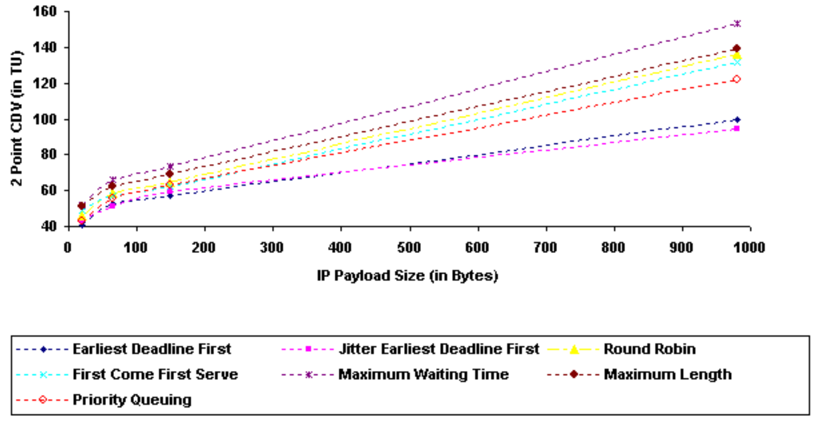


Figure 5.24: Comparison of 2-Point PDV with packet size for four Flows.

list it is directed to, if desired.

Sometimes, two or more competing flows may have the same deadline. In such a scenario, the aggregate jitter experienced by packets serviced by the different active list is compared and the active list with the highest jitter so far is given preference to help that reduce its jitter.

The programmable sequence scheduler in *Sched-650* was probed for single and four input traffic flows scenario for different scheduling algorithms, as shown in Table A.11 in page 84 to Table A.14 in page 85. It can be clearly seen that the jitter experienced by the flows is much lesser for both the EDF and the WJ-EDF scheduler.

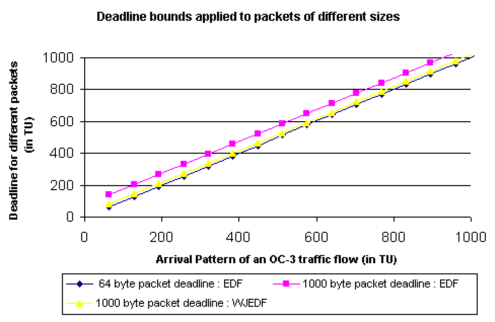


Figure 5.25: Deadlines to packet departures under EDF and WJ-EDF for different packet sizes.

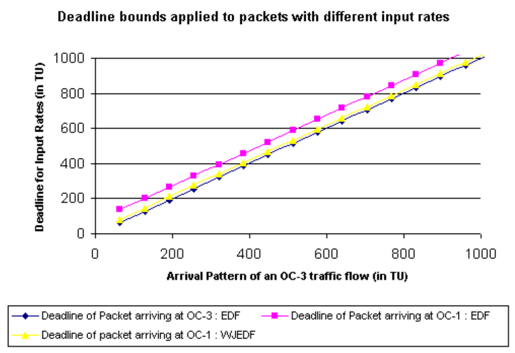


Figure 5.26: Deadlines to packet departures under EDF and WJ-EDF for different input rates.

In conclusion, the WJ-EDF tries to be fair to all its competing flows, in addition to balancing the jitter experienced by its flows.

Chapter 6

Conclusion and Future Work

The research work presented in this thesis targeted two main objectives. The first one is to devise a scheduling scheme for the programmable sequence scheduler of *Sched-650* which performs efficiently in terms of the the jitter and bandwidth shared by the flows serviced by the scheduler. Towards this goal, we defined a version of Earliest Deadline First scheduling scheme called *Weighted Jitter Earliest Deadline First* or *WJ-EDF*. Our second goal was to study the behavior of the WJ-EDF for constant bit rate traffic and compare its performance with other commonly used scheduling algorithms. In this chapter, we summarize the findings of our research and conclude the research. Future work that can be continued from this research is also discussed in this chapter.

6.1 Findings and Conclusion

Two simulation setups, a behavioral C model and Agere's SDE were used in understanding the behavior of scheduling algorithms. Since the timing behavior of the algorithms was observed to be similar on both the setups and only a small testcase could be studied on the behavioral C model, most of the experiments were performed on Agere's SDE. The bandwidth sharing property and the jitter experienced by packets serviced by different scheduling algorithms were studied.

Fairness of a scheduling algorithm was studied by varying the number of flows and packet sizes. It was observed that the throughput guaranteed by the scheduler to its flows was reduced as the number of flows increased but the fairness depends on the scheduling algorithm used. Deadline-based scheduling schemes, namely EDF and WJ-EDF were seen to provide the highest fairness followed by priority queuing. First-Come First-Serve (FCFS) scheduling showed the poorest performance in fairness.

Jitter experienced by the different flows was studied in the same set of experiments. The scheduling algorithms has a significant impact on the jitter experienced by the packets serviced by them. This difference is particularly evident in the case of multiple flows. The average jitter of all the flows was considered. The worst case jitter was experienced when the maximum waiting time scheduler was configured. The best case jitter was seen in the case of EDF algorithm, followed by WJ-EDF. In the case of WJ-EDF, lowest jitter is seen in the traffic flow which has the highest priority.

6.2 Future Plans

The performance of *Weighted-Jitter Earliest Deadline First* scheduling was analyzed for Constant Bit Rate Traffic as a part of this thesis and compared with the performance of other common scheduling schemes. Constant bit rate traffic is a good representation of voice traffic. In addition to voice, it is also important to study the performance of WJ-EDF for other applications such as video and data. Future work can concentrate on analysis of WJ-EDF for other traffic types like real-time and non-real time variable bit rate traffic and best-effort traffic types on the network processor.

Deadline-based scheduling schemes performs better in terms of jitter experienced by the packets as seen in this simulation study on setup 2. However, some of the assumptions made for this study such as implementation of a 64-bit timestamp and maintenance of aggregate jitter without wrapping need further analysis for implementation on hardware. Additionally, implementing WJ-EDF on an FPGA model of APP650 will provide us with a more realistic idea of the performance of WJ-EDF on the ASIC.

Bibliography

[1] [Online]. Available: www.webopedia.com/TERM/P/packet_switching.html

[2] J. S. Z. Zhang, J. Kurose and D.Towsley, "Smoothing, statistical multiplexing and call admission control for stored video," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 6, pp. 1148–1166, Aug 1997.

[3] M. K. Itamar Elhanany and D. Sadot, "Packet scheduling in next-generation multiterabit networks," *IEEE computer magazine*, vol. 34, no. 4, pp. 104–106, Apr 2001.

[4] D. Stephens, J. Bennett, and H. Zhang, "Implementing scheduling algorithms in high speed networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 6, pp. 1145–1158, 1999.

[5] J. C. B. Donpaul C. Stephens and H. Zhang, "Implementing scheduling algorithms in high-spped networks," Tech. Rep.

[6] R. Carter and M. Crovella, "Measuring bottleneck link speed in packet-switched networks," in *Performance Evaluation*, vol. 27, no. 8, Oct 1996.

[7] J. Xu and R. J. Lipton, "On fundamental tradeoffs between delay bound and computational complexity in packet scheduling algorithms," *IEEE/ACM Transactions on Networking*, vol. 13, no. 1, pp. 15–28, Feb 2005.

[8] A. Shaw, "Fixed-length packets versus variable-length packets in fast packet switching networks," Massachusetts Institute of Technology, Cambridge, Tech. Rep., Mar 1994.

[9] "Quality of service networking." [Online]. Available: http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/qos.htm

[10] N. Figueira and J. Pasquale, "Providing quality of service for wireless links: Wireless/wired networks," *IEEE Personal Communications*, vol. 6, no. 5, pp. 42–51, Oct 1999.

[11] B. Sabata, S. Chatterjee, M. Davis, J. J. Sydir, and T. F. Lawrence, "Addressing the contract issue, standardisation for qos," in *Workshop on Object-Oriented Real-Time Dependable Systems (WORDS).* IEEE Computer Society, 1997.

[12] I. Lock R., Dobson G. Sommerville, "Addressing the contract issue, standardisation for qos," in *Challenge2005 Conference Proceedings, Ljubljana, Slovenia*, Oct 2005, pp. 161–168.

[13] A. Francini and F. Chiussi, "Providing qos guarantees to unicast and multicast flows in multistage packet switches," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1589 – 1601, Oct 2002.

[14] "Quality of service." [Online]. Available: http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/qos.htm

[15] Z. M. Almes G., Kalidindi S., "A one-way delay metric for ippm," Internet Engineering Task Force, Tech. Rep. RFC 2679, 1999.

[16] ——, "A round-trip delay metric for ippm," Internet Engineering Task Force, Tech. Rep., Sep 1999.

[17] Y. Mansour and B. Patt-Shamir, "Jitter control in qos networks," *IEEE/ACM transactions on Networking*, vol. 9, pp. 492–502, 2001.

[18] C. Demichelis, "Packet delay variation comparison between itu-t and ietf draft definitions," IPPM, Tech. Rep., Nov 2000.

[19] "Internet protocol data communication service - ip packet transfer and availability performance parameters," ITU-T, Tech. Rep., Feb 1999.

[20] P. C. C. Demichelis, “Instantaneous packet delay variation metric for ippm,” IPPM, Tech. Rep., Oct 1999.

[21] L. S. S. G. O. Lataoui, T. Rachidi and R. H. Yan, “A qos management architecture for packet switched 3rd generation mobile systems,” in *Proc. Networld + Interop*, 2000.

[22] H. Z. Dinesh C. Verma and D. Ferrari, “Delay jitter control for real-time communication in a packet switching network,” in *Proc. IEEE TriCom*, 1991, pp. 35–43.

[23] A. Sahoo, “Scheduling.”

[24] M. Peuhkuri, “Ip quality of service.” [Online]. Available: http://www.netlab.tkk.fi/u/puhuri/htyo/Tik-110.551/iwork/iwork.html

[25] Z. M. Almes G., Kalidindi S., “A round-trip packet loss metric for ippm,” Internet Engineering Task Force, Tech. Rep., Sep 1999.

[26] R. West, “Dynamic window-constrained scheduling.” [Online]. Available: http://www.cs.bu.edu/ richwest/dwcs.html

[27] “Bandwidth trading and network commodities.” [Online]. Available: http://www.zurich.ibm.com/bandwidth/concepts_qos.html

[28] C. D. and J. R., “Analysis of the increase/decrease algorithms for congestion avoidance in computer networks,” *Journal of Computer Networks*, vol. 17, no. 1, pp. 1–14, Jun 1989.

[29] A. D. R. Jain and G. Babic, “Throughput fairness index: An explanation,” *ATM Forum / 99-0045*, Feb 1999.

[30] “Throughput fairness index.” [Online]. Available: http://www.cse.wustl.edu/ jain/atmf/ftp/af_fair.pdf

[31] T. Kihara, K. Ozawa, “A selective cell-discarding scheme using packet-size information,” *Electronic and Communications in Japan, Part 1: Communications*, vol. 81, no. 11, pp. 48–57, 1998.

[32] H. Shi, “Packet scheduling strategies for emerging service models in the internet,” Ph.D. dissertation, Drexel University, Aug 2003.

[33] H. Zhang, “Service disciplines for packet-switching integrated-services networks,” Ph.D. dissertation, UCB, 1993.

[34] S. B.S., “Packet scheduling algorithms to support qos in networks,” Master’s thesis, Indian Institute of Technology, Kanpur, Oct 1999.

[35] H. Zhang and D. Ferrari, “Rate-controlled service disciplines,” *Journal of High-Speed Networks 3, 4*, 1994.

[36] A. generalizes processor sharing approach to flow control The multiple node case, “A. k. parekh and r. g. gallager,” Lab. for Inform. and Decision Syst. M.I.T, Tech. Rep. 2076, 1991.

[37] A. generalizes processor sharing approach to flow control in Integrated Service Networks: The single-node case, “A. k. parekh and r. g. gallager,” *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, 1993.

[38] L. Georgiadis, R. Guerin, and A. K. Parekh, “Optimal multiplexing on a single link: Delay and buffer requirements,” in *INFOCOM (2)*, 1994, pp. 524–532.

[39] A. Varma and D. Stiliadis, “Hardware implementation of fair queueing algorithms for atm networks,” *IEEE Communications Magazine*, vol. 35, pp. 54–68, Dec 1997.

[40] C. Chang, “Stability, queue length and delay of deterministic and stochastic queueing networks,” *IEEE Transactions on Automatic Control*, vol. 39, no. 5, pp. 913–931, may 1994.

[41] R. Cruz, "A calculus of delay, part i: Network element in isolation," *IEEE Trans. Inform. Theory*, vol. IT-37, no. 1, pp. 114–131, jan 1991.

[42] V. Sivaraman and F. Chiussi, "End-to-end statistical delay guarantees using earliest deadline first(edf) packet scheduling," in *Proc. og GLOBECOM, Rio de Janerio, Brazil*, Dec 1999, pp. 1307–1312.

[43] R. Cruz, "A calculus of delay, part ii: Network analysus," *IEEE Trans. Inform. Theory*, vol. IT-37, no. 1, pp. 132–141, jan 1991.

[44] S. A.Demers and S.Shenker, "Analysis and simulation of a fair queueing algorithm," *Internetworking Research and Experience*, vol. 1, no. 1, 1990.

[45] C. Lawrence, "Confidence intervals," 2004.

APPENDIX

Appendix A

Tables

Table A.1: Single Traffic Flow fed to OC-3 Scheduler (Sample Size n=10)

Input Rate	Output Rate(Mbps)	Throughput	Fairness Index	95% Confidence Interval	Error in mean %
64 kbps	0.064	1	1	0.0008	1.3063
OC-1	51.668	0.9967	0.9967	2.8576	5.5307
OC-3	155.122	0.9974	0.9974	0.0343	0.0221
OC-24	156.671	0.1259	0.1259	0.0672	0.0429
OC-48	156.221	0.0628	0.0628	0.0325	0.0208

Table A.2: Two Traffic Flows fed to OC-3 Scheduler (Sample Size n=10)

Flow 1				Flow 2			
Input Rate	Output Rate(Mbps)	95% CI	Error in mean %	Input Rate	Output Rate(Mbps)	95% CI	Error in mean %
OC-1	49.743	0.0356	0.0716	OC-1	48.122	0.2345	0.4873
OC-1	51.644	0.6245	1.2091	OC-3	99.928	2.4392	2.4410
OC-3	77.991	0.0125	0.0160	OC-3	77.992	0.0021	0.0027
OC-3	78.133	0.0023	0.0030	OC-24	78.217	0.0345	0.0441
OC-24	79.973	0.0724	0.0905	OC-24	79.788	0.0327	0.0410

Input Rates	Output Rate(Mbps)	Throughput of Flow 1	Throughput of Flow 2	Fairness Index
OC-1,OC-1	49.743, 48.122	0.9595	0.9283	0.9997
OC-1,OC-3	51.644, 99.928	0.9962	0.6425	0.9555
OC-3,OC-3	77.991, 77.992	0.5015	0.5015	1.0000
OC-3,OC-24	78.133, 78.217	0.5024	0.0629	0.6232
OC-24,OC-24	79.973, 79.788	0.0643	0.0641	1.0000

Flow 1				Flow 2			
Input Rate	Output Rate(Mbps)	95% CI	Error in mean %	Input Rate	Output Rate(Mbps)	95% CI	Error in mean %
OC-1	51.832	0.2345	0.4524	OC-1	51.773	0.0961	0.1856
OC-3	54.022	0.0342	0.0633	OC-1	52.282	0.0264	0.0505
OC-24	54.11	0.0074	0.0137	OC-24	52.305	0.0475	0.0908
OC-3	52.335	0.0892	0.1705	OC-3	52.544	0.0116	0.0221
OC-24	54.176	0.0784	0.1446	OC-3	52.406	0.0567	0.1082
OC-24	52.402	0.5623	1.0731	OC-24	53.423	0.0124	0.0232
Flow 3				Flow 4			
Input Rate	Output Rate(Mbps)	95% CI	Error in mean %	Input Rate	Output Rate(Mbps)	95% CI	Error in mean %
OC-1	51.832	0.0836	0.1612	OC-1	51.832	0.0034	0.0066
OC-1	54.022	0.0786	0.1454	OC-1	54.022	0.0039	0.0073
OC-1	54.11	0.0347	0.0641	OC-1	54.11	0.0789	0.1459
OC-3	52.335	0.0287	0.0548	OC-3	52.335	0.0458	0.0876
OC-3	54.176	0.0819	0.1511	OC-3	54.176	0.2579	0.4760
OC-24	52.402	0.6169	1.1772	OC-24	52.402	0.7845	1.4971

Table A.3: Four Traffic Flows fed to OC-3 Scheduler (Sample Size n=10)

Number	Input
1	OC-1/OC-1/OC-1/OC-1
2	OC-3/OC-1/ OC-1/OC-1
3	OC-24/OC-1/OC-1/OC-1
4	OC-3/OC-3/OC-3/OC-3
5	OC-24 /OC-3/OC-3/OC-3
6	OC-24/OC-24/OC-24/OC-24

Input	Flow Output Rate				Flow Throughput				Fairness
Number	1	2	3	4	1	2	3	4	Index
1	51.832	51.773	38.956	38.961	0.9998	0.9987	0.7515	0.7516	0.9804
2	54.022	52.282	39.209	39.264	0.3474	1.0085	0.7563	0.7574	0.9016
3	54.11	52.305	39.266	39.475	0.0435	0.0420	0.7574	0.7615	0.5561
4	52.335	52.544	39.41	39.116	0.3365	0.3379	0.2534	0.2515	0.9798
5	54.176	52.406	39.347	39.567	0.0435	0.337	0.2530	0.2544	0.8072
6	52.402	53.423	39.11	39.124	0.0421	0.0429	0.0314	0.0315	0.978

Table A.4: Comparison of Scheduling Algorithms for traffic flow feeding the OC-3 Scheduler (Sample Size n=10)

	Output rate	Fairness
OC-3 (20 bytes)	155.122	0.9974
OC-3 (64 bytes)	155.0953	0.9973
OC-3 (148 bytes)	155.142	0.9976
OC-3 (980 bytes)	155.332	0.9988

Table A.5: Comparison of Scheduling Algorithms for two traffic flows feeding the OC-3 Scheduler (Sample Size n=10)

Flow		Output Rate		Throughput		Fairness
1	2	1	2	1	2	Index
OC-3 (20 bytes)	OC-3 (20 bytes)	77.991	77.992	0.5015	0.5015	0.9999
OC-3 (20 bytes)	OC-3 (64 bytes)	77.889	77.865	0.5008	0.5007	0.9999
OC-3 (20 bytes)	OC-3 (148 bytes)	78.001	77.8452	0.5015	0.5005	0.9999
OC-3 (64 bytes)	OC-3 (64 bytes)	77.8239	77.8336	0.5004	0.5005	0.9999
OC-3 (64 bytes)	OC-3 (148 bytes)	77.9837	78.1224	0.5014	0.5023	0.9999
OC-3 (20 bytes)	OC-24 (20 bytes)	78.133	78.217	0.5024	0.0629	0.6232
OC-3 (20 bytes)	OC-24 (64 bytes)	77.934	78.332	0.5011	0.063	0.6237
OC-3 (20 bytes)	OC-24 (148 bytes)	78.3342	78.3145	0.5037	0.063	0.6231
OC-3 (64 bytes)	OC-24 (64 bytes)	78.4043	78.1245	0.5041	0.0628	0.6227
OC-3 (64 bytes)	OC-24 (148 bytes)	78.2315	78.7236	0.5030	0.0633	0.6238

Table A.6: Comparison of Scheduling Algorithms for two traffic flows feeding the OC-3 Scheduler (Sample Size n=10)

Algorithm	Output Rate		Throughput		Fairness
	1	2	1	2	Index
Earliest Deadline First	77.879	77.809	0.5008	0.5003	0.9999997
Weighted Jitter Earliest Deadline First	77.892	77.747	0.5008	0.4999	0.9999991
Round Robin	77.658	77.295	0.4993	0.4970	0.9999945
First Come First Serve	77.321	76.896	0.4972	0.4944	0.9999924
Maximum Waiting Time	77.341	77.571	0.4973	0.4989	0.9999977
Maximum Length	77.442	77.337	0.498	0.4973	0.9999995
Priority Queuing	77.823	77.695	0.5004	0.4996	0.9999993

Algorithm	Flow 1		Flow 2	
	95% CI	Error in mean %	95% CI	Error in mean %
EDF	0.0312	0.0423	0.0729	0.0901
Weighted Jitter EDF	0.0348	0.0447	0.0723	0.0930
Round Robin	0.0566	0.0728	0.0872	0.1129
First Come First Serve	0.0236	0.0305	0.1261	0.1640
Maximum Waiting Time	0.0679	0.0878	0.1342	0.1730
Maximum Length	0.0495	0.0639	0.0357	0.0461
Priority Queuing	0.0295	0.0378	0.8782	1.1303

Table A.7: Comparison of Scheduling Algorithms for four traffic flows feeding the OC-3 Scheduler (Sample Size n=10)

Number	Algorithm Name
1	EDF
2	Weighted Jitter EDF
3	Round Robin
4	First Come First Serve
5	Maximum Waiting Time
6	Maximum Length
7	Priority Queuing

Number	Output Rate				Throughput				Fairness
	1	2	3	4	1	2	3	4	Index
1	50.452	49.623	42.05	41.873	0.3244	0.3191	0.2704	0.2692	0.9923
2	51.662	50.329	40.35	40.227	0.3322	0.3236	0.2595	0.2587	0.9863
3	48.859	48.547	31.428	32.006	0.3142	0.3122	0.2021	0.2058	0.9573
4	47.214	47.924	29.561	28.349	0.3036	0.3082	0.1901	0.1823	0.9404
5	47.231	46.749	34.567	32.006	0.3037	0.3006	0.2223	0.2058	0.9712
6	48.267	48.539	38.569	36.564	0.3104	0.3121	0.2480	0.2351	0.9841
7	49.847	49.821	40.546	39.361	0.3124	0.3145	0.2734	0.2388	0.9834

Number	Flow 1		Flow 2		Flow 3		Flow 4	
	95% CI	Error in mean %	95% CI	Error in mean %	95% CI	Error in mean %	95% CI	Error in mean %
1	0.0348	0.0689	0.0484	0.0975	0.832	1.9786	0.0484	0.1155
2	0.0566	0.1095	0.0786	0.1561	0.022	0.0545	0.0479	0.119
3	0.2357	0.4824	0.0035	0.0071	0.011	0.0350	0.0635	0.1983
4	0.0679	0.1438	0.0529	0.1103	0.0335	0.1133	0.0529	0.1865
5	0.0495	0.1047	0.0088	0.0188	0.1176	0.3402	0.0788	0.2462
6	0.0295	0.0610	0.6169	1.2709	0.1402	0.3635	0.0617	0.1687
7	0.0554	0.1059	0.0517	0.0983	0.3423	0.8686	0.0517	0.1320

Table A.8: Deadlines applied to packets of different sizes

Input OC-3 pattern	PACKET DEADLINE		
	64 BYTE EDF	128 BYTE EDF	128 BYTE WJ-EDF
64	64	138	83
128	128	202	130
192	192	266	194
256	256	330	258
320	320	394	322
384	384	458	386
448	448	522	450
512	512	586	514
576	576	650	578
640	640	714	642
704	704	778	706
768	768	842	770
832	832	906	834
896	896	970	898
960	960	1034	962
1024	1024	1098	1026
1088	1088	1162	1090
1152	1152	1226	1154
1216	1216	1290	1218
1280	1280	1354	1282
1344	1344	1418	1346
1408	1408	1482	1410
1472	1472	1546	1474
1536	1536	1610	1538

Table A.9: Deadlines applied to packets arriving at different input rates

Input OC-3 pattern	PACKET DEADLINE		
	OC-3 EDF	OC-1 EDF	OC-1 WJ-EDF
32	32	46	39
64	64	78	71
96	96	110	103
128	128	142	135
160	160	174	167
192	192	206	199
224	224	238	231
256	256	270	263
288	288	302	295
320	320	334	327
352	352	366	359
384	384	398	391
416	416	430	423
448	448	462	455
480	480	494	487
512	512	526	519
544	544	558	551
576	576	590	583
608	608	622	615
640	640	654	647
672	672	686	679
704	704	718	711
736	736	750	743
768	768	782	775

Table A.10: Algorithm Legend

1	Earliest Deadline First
2	Weighted Jitter Earliest Deadline First
3	Round Robin
4	First Come First Serve
5	Maximum Waiting Time
6	Maximum Length
7	Priority Queuing

Table A.11: 1-Point CDV for Single Input Flow

Algorithm	1-Point CDV	CI	% Error	1-Point CDV	CI	% Error
	20 bytes			64 bytes		
1	0.4836	0.0001	0.0265	14.6046	0.0927	0.6350
2	0.4786	0.0035	0.7292	14.5678	0.0472	0.3242
3	0.6347	0.0974	15.3510	26.2317	1.1028	4.2042
4	0.5287	0.0053	1.008	21.7893	0.9467	4.3448
5	0.7879	0.0195	2.4687	29.8128	0.8426	2.8264
6	0.6169	0.0162	2.6277	27.9235	0.2342	0.8388
7	0.5165	0.0245	4.7433	20.0560	1.3078	6.5207
	148 bytes			1k bytes		
1	52.5678	0.0239	0.0455	67.3479	0.0427	0.0634
2	53.4179	0.0867	0.1624	68.5656	0.025	0.0364
3	63.5189	0.9765	1.5373	76.2357	0.0554	0.0726
4	61.2390	0.0845	0.1380	75.6789	0.0437	0.0577
5	67.2394	0.6723	0.9999	80.4946	0.0593	0.0736
6	65.2349	0.5679	0.8705	78.2946	0.1872	0.2392
7	57.1943	0.9456	1.6533	71.4554	0.0632	0.0884

Table A.12: 2-Point PDV for Single Input Flow

Algorithm	2-Point PDV	CI	% Error	2-Point PDV	CI	% Error
	20 bytes			64 bytes		
1	0.9235	0.0073	0.7887	18.3429	0.0044	0.0239
2	0.9729	0.0047	0.4852	18.3945	0.0051	0.0276
3	1.6535	0.0066	0.3980	22.7893	0.9534	4.1835
4	1.6723	0.0045	0.2701	22.4582	0.0834	0.3714
5	2.0729	0.0327	1.578	26.2579	0.0235	0.0893
6	1.8338	0.0078	0.4278	28.2845	0.2451	0.8666
7	1.0173	0.0045	0.4444	20.0560	0.6732	3.3566
	148 bytes			1k bytes		
1	40.2857	0.0894	0.222	77.2134	0.0872	0.113
2	41.5672	0.0734	0.1766	78.6723	0.0159	0.0202
3	44.2903	0.0645	0.1456	85.8724	0.0942	0.1097
4	45.3956	0.0673	0.1483	88.3261	0.0946	0.1071
5	48.2315	0.0926	0.1921	81.3421	0.0066	0.0081
6	50.2346	0.0068	0.0135	88.2357	0.2185	0.2476
7	43.193	0.0783	0.1813	76.1678	0.0905	0.1188

Table A.13: Average 1-Point CDV for Four Input Flows

Algorithm	1-Point CDV	CI	% Error	1-Point CDV	CI	% Error
	20 bytes			64 bytes		
1	40.4836	0.0092	0.0228	52.4531	0.008	0.0152
2	41.7652	0.0082	0.0197	51.8568	0.0124	0.0239
3	44.8349	0.0042	0.0094	59.7801	1.8596	3.1107
4	47.4379	0.0091	0.0191	58.9321	0.9834	1.6687
5	51.5623	0.0035	0.0068	68.6789	0.7812	1.1375
6	49.6169	0.0979	0.1972	63.6239	0.0218	0.0342
7	42.6878	0.0894	0.2095	55.1372	0.8903	1.6148
	148 bytes			1k bytes		
1	56.837	0.0973	0.1712	99.3479	0.0189	0.0190
2	58.2346	0.0689	0.1183	93.5856	0.0838	0.0895
3	63.7236	0.8045	1.2625	134.2357	0.047	0.035
4	61.6789	0.0663	0.1076	130.6789	0.484	0.3704
5	74.7623	0.0589	0.0788	152.4946	0.044	0.0288
6	68.4523	0.0671	0.0980	138.2946	0.9084	0.6568
7	62.3227	0.8745	1.4032	120.1070	0.3481	0.2898

Table A.14: Average 2-Point PDV for Four Input Flows

Algorithm	2-Point PDV	CI	% Error	2-Point PDV	CI	% Error
	20 bytes			64 bytes		
1	40.9235	0.0042	0.0103	52.3429	0.0097	0.0185
2	42.9728	0.0032	0.0075	51.3945	0.1259	0.245
3	45.6535	0.0027	0.0060	58.7893	1.0678	1.8163
4	48.6723	0.0081	0.0167	57.4582	0.7891	1.3733
5	52.0729	0.001	0.0019	66.2579	0.2312	0.3489
6	50.8338	0.0562	0.1106	62.2845	0.0872	0.1401
7	43.2037	0.0846	0.1957	55.6068	0.6341	1.1403
	148 bytes			1k bytes		
1	56.2857	0.0812	0.1443	98.2134	0.6832	0.6956
2	59.5672	0.0229	0.0385	94.6723	0.979	1.0341
3	64.2903	0.0076	0.0118	135.8724	0.0391	0.0288
4	62.3956	0.0832	0.1333	131.3261	0.0456	0.0347
5	73.2315	1.854	2.5317	153.342age1	0.0843	0.055
6	69.2346	0.387	0.559	139.2357	0.9306	0.6684
7	63.193	0.9678	1.5315	121.8608	0.9325	0.7652