A SIMULATION STUDY OF SCHEDULING ALGORITHMS FOR PACKET SWITCHING NETWORKS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

ÖZGÜR BABUR

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

DECEMBER 2003

Approval of the Graduate School of Natural and Applied Sciences

Prof. Dr. Canan ÖZGEN
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Mübeccel DEMİREKLER
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Buyurman BAYKAL
Supervisor

Examining Committee Members

Prof. Dr. Semih BİLGEN _____

Prof. Dr. Hasan GÜRAN _____

Assoc. Prof. Dr. Buyurman BAYKAL _____

Assist. Prof. Dr. Cüneyt BAZLAMAÇCI _____

Hakan YILMAZ (M.Sc.) _____

ABSTRACT

A SIMULATION STUDY OF SCHEDULING ALGORITHMS FOR PACKET SWITCHING NETWORKS

Babur, Özgür
M.Sc., Department of Electrical and Electronics Engineering
Supervisor: Assoc. Prof. Dr. Buyurman Baykal

December 2003, 150 pages

A scheduling algorithm has the primary role in implementing the quality of service guaranteed to each flow by managing buffer space and selecting which packet to send next with a fair share of network. In this thesis, some scheduling algorithms for packet switching networks are studied. For evaluating their delay, jitter and throughput performances, a discrete event simulator has been developed. It has been seen that fair scheduling provides, fair allocation of bandwidth, lower delay for sources using less than their full share of bandwidth and protection from ill-behaved resources.

Keywords: Packet Scheduling, Flow, Packet-Switching Network, Fair Queueing, Delay, Jitter, Throughput.

ÖZ

PAKET ANAHTARLAMALI AĞLAR İÇİN ÇİZELGELEME ALGORİTMALARININ BENZETİMİ ÇALIŞMASI

Babur, Özgür
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Buyurman Baykal

Aralık 2003, 150 sayfa

Çizelgeleme algoritmaları, gönderilecek paketleri ağı adaletli paylaştırarak seçmesi ve arabellek yönetimini yapması nedeniyle, her akış için garantilenmiş hizmet niteliğinin sağlanmasında birincil göreve sahiptirler. Bu tezde, paket anahtarlamalı ağlar için çizelgeleme algoritmaları incelenmiştir. Bu algoritmaların gecikme, seğirme ve çıkan iş oran başarımlarının değerlendirilmesi için, bir ayrık olay benzetimcisi geliştirilmiştir. Adaletli çizelgeleme algoritmalarının, bant genişliğini adaletli atadığı, payına düşen bant genişliğinden daha az bant genişliği kullanan özkaynaklar için düşük gecikme sağladığı ve payına düşen bant genişliğinden daha fazla bant genişiliği isteyen özkaynaklara karşı koruma sağladığı görülmüştür.

Anahtar Kelimeler: Çizelgeleme, Akış, Paket Anahtarlamalı Ağ, Adaletli Kuyruklama, Gecikme, Seğirme, Çıkan İş Oranı.

ACKNOWLEDGMENTS

I would like to express my deep gratitude to my supervisor Assoc. Prof. Dr. Buyurman Baykal for his invaluable guidance and patience throughout my M.Sc. research.

Special thanks are due to my mother Hatice and my father İsmet who provided me with the necessary motivation and support to complete this study.

I wish to thank TUBITAK-MRC-ITRI for the facilities provided for me to complete this thesis.

TABLE OF CONTENTS

LIST OF TABLES

Table

LIST OF FIGURES

Figure

LIST OF ABBREVIATIONS AND ACRONYMS

Bps	: Bit per second
CBR	: Constant Bit Rate
CIF-Q	: Channel Condition Independent Packet Fair Queueing
CPU	: Central Processing Unit
DRR	: Deficit Round Robin
DRRWC	: Deficit Round Robin for Wireless Channels
FCFS	: Fist Come First Served
FFQ	: Fluid Fair Queueing
FPU	: Floating Point Unit
GPS	: Generalized Processor Sharing
GUI	: Graphical User Interface
IWFQ	: Idealized Wireless Fair Queueing
Kbps	: Kilo bit per second
LCFS	: Last Come First Served
LTFS	: Long Term Fairness Server
MMPP	: Markov Modulated Poisson Process
PGPS	: Packet-by-packet Generalized Processor Sharing
PQ	: Packet Queue
SBFA	: Server Based Fairness Approach
SFQ	: Start Time Fair Queueing
SQ	: Slot Queue
VBR	: Variable Bit Rate
WFI	: Worst Case Fair Index
WFQ	: Weighted Fair Queueing
WF^2Q	: Worst Case Fair Weighted Fair Queueing

WRR : Weighted Round Robin

WPS : Wireless Packet Scheduling

QoS : Quality of Service

CHAPTER 1

INTRODUCTION

Over the past few years, computing, communication and video compression technologies have advanced significantly. As new types of applications were developed (e.g. video conferencing, distance learning, news on demand services), it was seen that the bandwidth and storage space requirements of digital data must be manageable carefully because of limited resources [20].

Resource guarantees and performance assurances are not possible with the best effort services. But today's computer networks must concurrently support communication sessions from a wide range of applications. This was resulted in integration of support for communication sessions with diverse quality of service requirements. For example, the needs and characteristics of communication sessions from applications like telnet, ftp, and e-mail are considerably different from those of applications like internet phone, web browsing, and video conferencing. Some sessions require service guarantees such as bounds on delays or throughput, while others may be satisfied with a best effort guarantee [26].

At an output of a network element, many packets compete for the output link. They belong to different applications running on different hosts; they flow through different paths and, in the case of an integrated services network, request different types of service. The role of the scheduling algorithm is to define the order in which

all these packets will be transmitted according to their requirements. Besides, as at a given time there exists more packets requiring access to the output link than the number that can be transmitted, packets are queued in the queuing mechanism waiting to be transmitted. If too many packets are queued, the memory is exhausted and some packets are discarded. Choosing which packets have to be discarded is also the role of the scheduling algorithm. Thus, a scheduler is a queuing algorithm, which allocates bandwidth and buffer space.

The scheduler identifies packets as members of a class or flow (i.e. a set of packets that have to receive the same treatment). Different classes receive different treatments.

To determine the characteristics of a suitable scheduling algorithm, consider the requirements of some of the principal applications envisioned for integrated services networks [19]:

- Audio applications: To maintain adequate interactivity for such applications, scheduling algorithms must provide low average and maximum delay.

- Video applications: Variable bit rate (VBR) video sources, which are expected to impose significant requirements on network resources, have unpredictable as well as highly variable bit rate requirement at multiple time-scales. These features impose two key requirements on network resource management:

    - Due to the difficulty in predicting the bit rate requirement of VBR video sources, video channels may utilize more than the reserved bandwidth. As long as the additional bandwidth used is not at the expense of other

channels (i.e., if the channel utilizes idle bandwidth), it should not be penalized in the future by reducing its bandwidth allocation.

- Due to multiple time-scale variation in the bit rate requirement of video sources, to achieve efficient utilization of resources, a network will have to over-book available bandwidth. Since such over-booking may yield persistent congestion, a network should provide some Quality of Service (QoS) guarantees even in the presence of congestion.

Fair scheduling involves the use of internal multiple queues with the ability to sort and insert different packets into each queue. The primary difference lies in how the queues are serviced. The objective is to provide "fair" or "equivalent" service to each of the queues. Although the notion of fairness has been defined many ways, perhaps the most widely accepted is that the traffic in each queue should obtain an equal portion of the bandwidth.

Unfair scheduling algorithms penalize channels for the use of idle bandwidth and do not provide any QoS guarantee in the presence of congestion. Fair scheduling algorithms, on the other hand, guarantee that, regardless of prior usage or congestion, bandwidth would be allocated fairly. Hence, fair scheduling algorithms are desirable for video applications.

- Data applications: To support low-throughput, interactive data applications (e.g., telnet), scheduling algorithms must provide low average delay. On the other hand, to support throughput-intensive, flow-controlled applications in heterogeneous, large-scale, decentralized networks, scheduling algorithms must allocate bandwidth fairly. Due to the coexistence of VBR video sources and data

sources in integrated services networks, the bandwidth available to data applications may vary significantly over time. Consequently, the fairness property of the scheduling algorithm must hold regardless of variation in server capacity.

Hence, a suitable scheduling algorithm for integrated services networks should: (1) achieve low average as well as maximum delay for low throughput applications (e.g., interactive audio, telnet, etc.); (2) provide fairness for VBR video; and (3) provide fairness, regardless of variation in server capacity, for throughput-intensive, flow-controlled data applications [20]. Finally, to facilitate its implementation in high-speed networks, it should be computationally efficient.

In this work, our aim is to simulate and evaluate various scheduling techniques for packet switching networks. For simulating complex, state based systems; discrete event simulation fundamentals are studied. To study if the scheduling schemes provide a fair share of network, a discrete event simulation approach will be presented. As an investigation of a new approach, Deficit Round Robin (DRR) scheduling scheme is adapted to wireless networks.

This thesis consists of six chapters. In Chapter 2, the network structures are presented. Packet Switching Networks and their advantages will take precedence. Also in this chapter simulation fundamentals of Computer Communication Networks will be briefly reviewed.

Survey of various scheduling disciplines for wired and wireless networks is presented in Chapter 3. A new scheduling approach for wireless networks is introduced.

In Chapter 4, simulation methods are discussed and a discrete event simulator that has been used for evaluating the performances of scheduling disciplines is presented. Validations of simulated packet switching algorithms are also discussed.

Chapter 5 presents results of the simulation experiments and comparisons about the algorithms. Delay, jitter and throughput performance of such algorithms are evaluated.

Finally, some conclusions are drawn for the overall assessment of the study and some possible future research topics are pointed out in Chapter 6.

CHAPTER 2

NETWORK STRUCTURES AND PACKET SWITCHING

A computer network is a collection of nodes at which reside the computing resources [which themselves are connected into the network through nodal switching computers], which communicate with each other via the data communication channels. Messages in the form of commands, inquiries, file transmissions and the like travel through this network over the data transmission lines. At the nodal switching computers, the communications-oriented tasks of relaying messages (with appropriate routing, acknowledging, error and flow controlling, queueing etc.) and removing and insertion of identifiers are issued. These tasks are separated from main computing functions and dedicated to switching computers [1, 2, 22, 24, 26].

Computer–communication networks may be partitioned into two separate networks: the communication subnetwork providing the message service, and the collection of computer and terminal resources forming the "user-resource" subnetwork. Figure 2.1 presents general structural model of a computer communication network. The computing facilities, which are responsible for processing and storage of data, are connected together by means of the communication subnetwork. The communication subnetwork consists of switching computers and high – speed data links.

In this thesis our attention will be directed to the communication subnetwork forming the message service which is responsible for accepting messages from any message source such as a terminal (Local, Remote or Orphan) or a computer and routing these messages through the network, and deliver them to their destination in a rapid and reliable way.

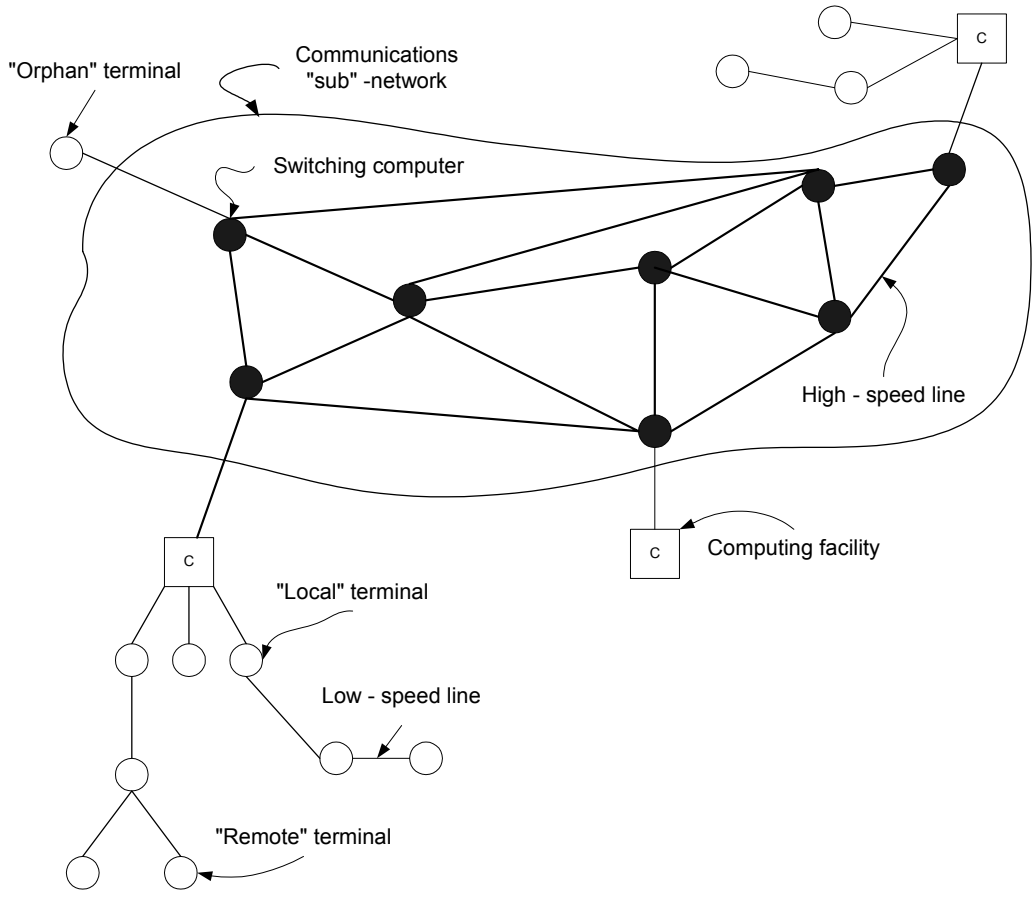


Figure 2.1 The structure of a computer - communication network.

Communication networks may be divided into three types [1, 2, 22, 24, 26]:

- Circuit or Line Switching
- Message Switching
- Packet Switching

2.1 Circuit or Line Switching Networks:

A Circuit Switching Network provides service by setting up a total path of connected lines from the origin to the destination of the demand. This complete circuit is set up by a special signaling message that thread its way through the network, seizing channels in the path as it proceeds. After the path is established, a return signal informs the source that data transmission may proceed, and all channels in the path are then used simultaneously. The entire path remains allocated to the transmission whether or not it is use, and only when the source releases circuit, all these channels will be used for other paths. Circuit switching is the common method for telephone systems [1, 2].

2.2 Message Switching Networks:

In message switching, only one channel is used at a time for a data transmission. The message first travels from its source node to next node in its path, and when the entire message is received at this node, then the next step in its path is selected; if this selected channel is busy, the message waits in a queue, and finally, when the channel becomes free, transmission begins. The message hops from node to node through the network using only one channel at a time, possibly queueing at busy channels, as it is successively stored and forwarded through the network.

2.3 Packet Switching Networks:

Packet Switching is basically same as Message Switching except that the messages are decomposed into smaller pieces called packets, each of which has a maximum length. These packets are numbered and addressed as with message switching and make their own way through the net in a packet – switched (store and forward) fashion. Thus many packet of the same message may be in transmission simultaneously, thereby giving one of the main advantages of packet switching, namely pipelining effect. As a result the transmission delay may considerably reduce over message switching.

In part (a), a network transmission path involving four nodes and three transmission lines are presented. It is assumed that no other traffic in the network interferes the transmission. The idealized sequence of events for circuit switching is shown in part (b). The connection delay at each switch, which is the major component of delay, is included in the model. It is followed by the transmission of the set up signal, which is assumed to be zero, arrives at switch B after a propagation delay. This cycle repeats for the other nodes of the path and when it reaches to the last node of the path, a return signal is sent by the destination node to the source node. As seen from part (b) only one data transmission is required.

As seen from part (c) for message switching networks first a small switch processing delay (for selecting routes) and messaging from node to node proceeds. More than circuit switching a message header for identifying and routing is added. This is because the path is not set up as in circuit switching.

In part (d), the sequence of events for Packet Switching is presented. The message is divided into three parts, each of which requires its own header. The sequence of packets is seen to be pipelining down to chain. Because of header overload, the number of bits transmitted is least for circuit switching, next larger for message switching, and largest for packet switching. In the figure showing the sequence of events of Packet Switching, the delay caused by the control signals is omitted.

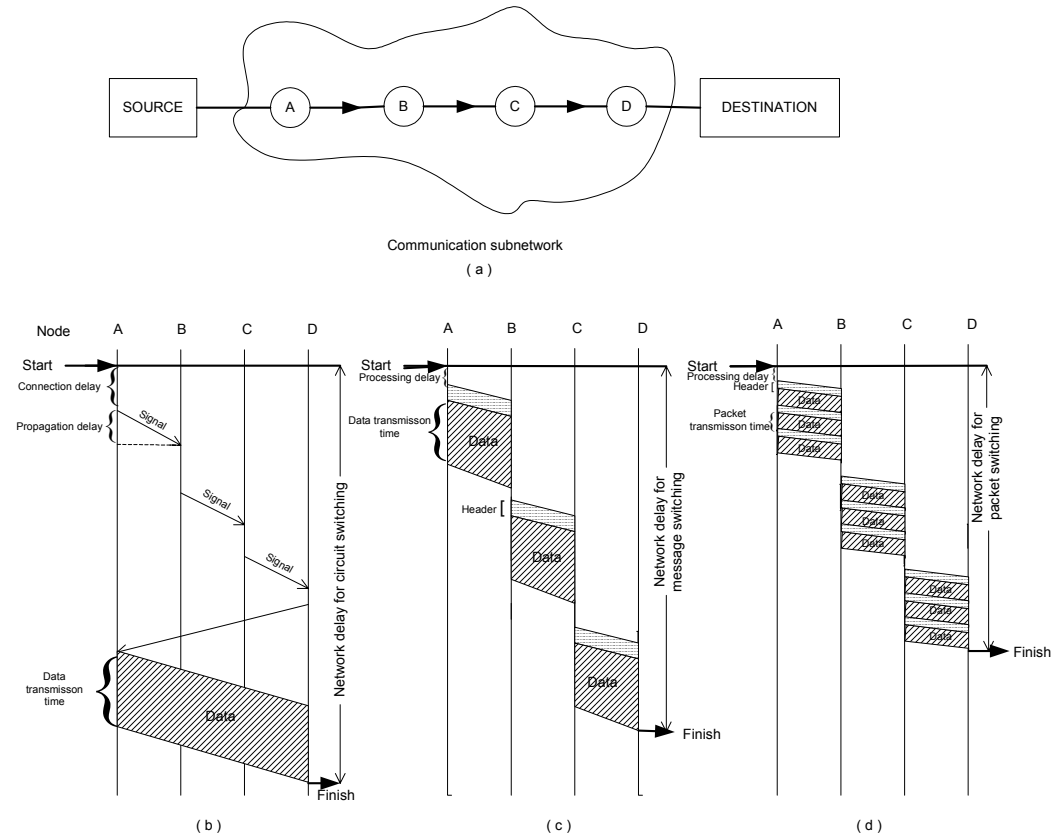


Figure 2.2 Comparison of network delay for circuit, message, and packet switching. (a) The transmission path. (b) Circuit Switching (c) Message Switching (d) Packet Switching

Circuit-switched connection is ideal for transmission of long, continuous stream of data. For bursty data transmission, Packet Switching makes good sense.

There are many key features of packet switching network such as ability to select good paths for data transmission paths as a function of the network congestion. In a heavily loaded Circuit Switched Network, a set-up signal may not find a complete path of available channels from source to destination and a busy signal may be sent back to the source. This results in a blocked network. But in Packet Switching Network, only the next channel needs to be available. This results in rarely blocking of network.

With packet switching, more than a one message is allowed to be transmitted across the network at the same time. This is in addition to the packet pipelining for a single message [1, 2, 26]. This message multiplexing is possible due both to pipelining along a given path and to alternate routing along many paths.

Another key feature of Packet Switching is rapidly handling small messages in spite of the presence of long messages that may be sent at the same. This is a result of decomposing of long messages into packets. And this decomposition causes storage requirements of the nodes to be reduced.

In this work, the following network measures will be focused on:

- Delay
- Throughput
- Cost

Delay and throughput measures are closely related and they are basic the performance criteria for various kinds of traffic. Interactive traffic such as internet phone, web browsing, and video conferencing must be delivered quickly and throughput is not the main goal. On the other hand along file transfer like telnet, ftp, and e-mail is concerned with throughput and delay is not the central issue.

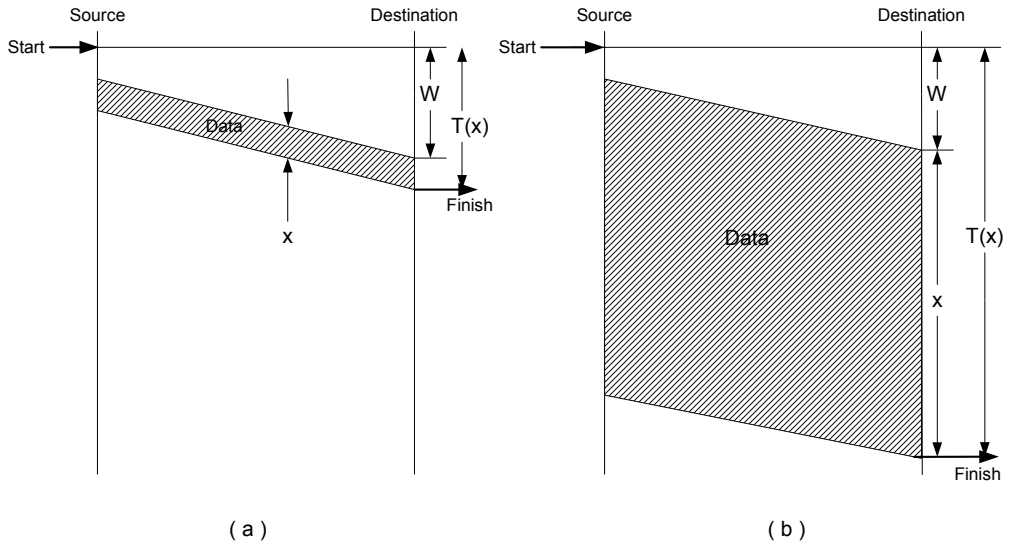


Figure 2.3 Network delay and throughput for (a) Short and interactive traffic (b) Long message

In Figure 2.3 the network delay and throughput for two cases explained before is shown. In the figure the measure of the response of the net, denoted by "W" is the average time from when the first bit is presented to the network until the first bit is delivered and service time "x", is the time from when the message begins delivery until the delivery is complete. The average network delay T(x), is the main

performance measure. For a short interactive message the average network delay is almost affected by the response of the net, which is shown in part (a). In part (b) of Figure 2.3 it is seen that in cases where x>>W, the service time dominates T (x). The network throughput can be defined as γ_{jk} msg/sec between source j and destination k which is inverse of service time x.

As long as handling both interactive messaging and long file transfers, a network may be required to support real-time traffic, which requires low delay and high throughput at the same time. This kind of traffic demands different kinds of control procedures.

As a summary, the properties of packet switching network is reviewed, as one that pipelines addressed messages along a single path as well as among alternate paths, decomposes messages into packets placing headers on each packet, and sends them through network in a store and forward fashion. Since these packets may be influenced by unexpected impacts such as errors, blocked storage, time-outs; they may arrive at the destination out of order or duplicated or more unluckily may get lost. It is expected to handle these kinds of events in an acceptable way. As stated in [1] the properties of Packet Switching Networks can be summarized as:

- Random Delay
- Random Throughput
- Out of order packets
- Lost and duplicate packets
- Nodal storage
- Speed matching between net and attached systems

Responding to these properties, the network must provide the following functions:

- Packetizing
- Buffering
- Pipelining
- Routing
- Sequencing and Numbering
- Error Control
- Storage Allocation
- Flow Control

2.4 Design Problems of Computer Communication Networks:

The design of a store-and-forward network is extremely complex task because of the complexity of network flow theory and queueing disciplines. As discussed before, a communication network is composed of:

1. Switching computers and the communication channels making up the physical network

2. Messages (described by their origin, destination, origination time, length and priority class) that forms flow, moving through the network in a store and forward fashion

3. Operating rules for handling the flow of traffic

The message routing procedure, the flow control procedure, the channel capacity assignment, the priority queueing discipline, and the topological configuration are the basic design issues of computer communication networks. Some of these topics will be covered in the following sections.

2.4.1 The Message Routing Procedure:

Message routing procedure is the rule determining the next node that a message will visit on its way through the net. The algorithm may use such parameters as source and destination, priority, availability of certain channels, and congestion of certain nodes and channels. There can be alternate routing paths, when more than one path is allowed. The routing paths may be determined either in a deterministic or random fashion. If the decisions are based on some measure of the observed traffic flow and/or the breakdown of nodes and channels, then the routing algorithm said to be dynamic adaptive routing procedure.

2.4.2 The Flow Control Procedure:

The flow control procedure is responsible for controlling how much traffic is permitted to enter the network. The traffic entering from the traffic form user-resource network into communications sub-network is regulated by this procedure. By doing so, the congestion is prevented.

2.4.3 The Priority Queueing Discipline:

As discussed before, when the communication channel assigned to a packet to be transmitted over is in use, the message must be queued. Transmission time for a

given packet can be found by dividing the length of the packet by the capacity of the channel. When the channel is available the packet is transmitted to the next node in its way and the channel is released for the use of other messages. Again, after reaching the next node, if the new channel is busy, the packet must enter the queue until it has the given link, and so on. Message or network delay is the total time spent in the network until the packet has reached its destination. The queueing discipline manages the order of service for the various channel queues.

Each packet of an admitted connection is conveyed through the network along the path established for that connection. At a switching node, the packet is multiplexed onto the next link, along with packets of other connections using the same link. Figure 2.4 present a simple model of the processing performed at each output link of a node. The steps of processing are:

- Packet demultiplexing, which inserts a packet into one of a set of a queues, corresponding to different QoS guarantees

- Queue insertion, which is either FCFS or priority based

- Queue multiplexing, which selects the next queue to service, and how many packets to remove and transmit from that queue

A scheduling policy can be either work-conserving or non-work-conserving. A method is work-conserving if an output link will never be idle as long as there are packets waiting. Work conservation might seem attractive, since it promises lower

average and end-to-end delay for packets. However, methods, which minimize jitter, are always non-conserving [13].

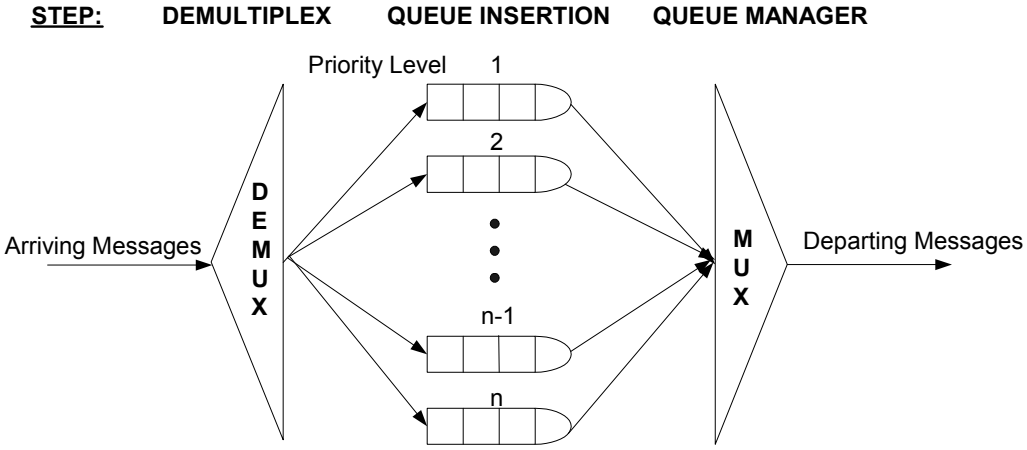


Figure 2.4 Node processing per output link

Queueing mechanisms for wired and wireless networks are the subject of this study and they will be discussed in Chapter 3; and in Chapter 5 simulation results of such algorithms will be presented.

2.4.4 Topological Configuration:

The topological configuration of the communication net strongly affects its behavior as regards its reliability, message delay, routing, and the like [1]. Topological constraints such as reliability may complicate topological design. And the form of the cost function that is included in the design, affects the structure. After designing the topology, capacity assignment to each channel must be made and queueing discipline must be decided.

2.5 Performance Evaluation of Computer Communication Networks:

As computer communication networks are rapidly becoming more complex in order to provide efficient service to requirements, performance evaluation of these networks have become non-trivial tasks [6]. In this section, a step-by-step procedure for developing a simulation tool for computer communication networks will be studied.

Generally, for evaluating the performance of computer communication networks, the following approaches are used [3, 5, 6, 14]:

- Analytical Techniques
- Real time measurements
- Simulation Techniques

For evaluating the performance of computer communication networks, extensive help is needed in mathematical models. These models are quick, economical, and easy to work with [3]. In order to obtain an approximate solution, analytical models must be mathematically tractable. However, a tractable analytical model often restricts the range of system characteristics that can be explicitly considered in a performance model. This is because; analytical models require a high degree of abstraction. So a modeler may end up with a correct solution, but to the wrong problem [5].

Since real time measurements need an operational system, it provides the most direct means of network performance evaluation. But it is also the most expensive in

the sense that a network must exits. Experiments running in testbeds or labs automatically capture important details that might be missed in a simulation, however building testbeds and labs is expensive, reconfiguring and sharing them is difficult, and they are relatively inflexible. Further, reproducing some networking phenomena, such as wireless radio interference, can be difficult, complicating efforts to compare or evaluate protocol design [30].

The objectives of making measurements on a computer network is to gather statistics about events, interpret them in terms of network performance and tune the network parameters to achieve the most optimal performance possible. Some important considerations of this technique are, how often measurements are taken, how long it takes to gather meaningful statistics and how to interpret them. The main problems associated with this technique are; *i*) Time required to gather enough statistics is a question. The statistics may be captured when the network is operating in a sub-optimal state. *ii*) How much to increase or decrease a parameter to achieve performance goals. A wrong adjustment may push the network into a region of worst performance [3].

In the simulation approach the network is simulated to any level of detail. Although simulation procedures are normally time consuming, they are more accurate, since they are not based on as may assumptions as analytical procedures [3]. In this technique less abstraction is required and the process of model formulation is a more straightforward task, although it remains advantageous from a solution standpoint to abstract out as many secondary system details as possible [5]. Such models can contain tremendous detail, especially for large networks. However, the execution of detailed models may require prohibitive amounts of computational resources. It is not uncommon for network simulations to require days of processing

time on a fast workstation. The analyst must aware of the trade-off between model detail and simulation execution time [14].

Simulation consists of a computer program that "behaves" like the system under study. Various components of the actual system are represented within a computer program. Unlike analytical models, which often require many assumptions and too restrictive for most real-world systems, simulation modeling replaces few restrictions on the classes of systems. For communication networks, developing a simulation program requires [14]:

- Modeling random user demands for network resources.
- Characterizing network resources needed for processing those demands.
- Estimating system performance based on output data generated by simulation.

2.5.1 Developent of a Simulation Model:

Before a useful simulation can be developed a detailed description and understanding of the system is needed. The plan that must be worked out before a simulation study, should contain the following information to address all issues related to the system [6]:

- Level of detail of a system is to be simulated.
- The various input parameters and the range of their values.
- Language to be used for the simulation model.
- Method of confidence intervals to be used.
- Criteria to be used for terminating the simulation.

- Validation method of the model.

Some topics listed above, will be discussed in the following sections.

2.5.1.1 Simulation Detail:

Since a detailed model requires more processing time, one has to decide on the basis of tradeoff between processing time and level of detail of a simulation model. The right level of a simulation model depends on the purpose of the performance evaluation task, the degree of understanding of the system being modeled and the output statistics required [6]. Since the interrelationship and interdependence among various events and components of the system is usually so complex, the right level of detail is not a trivial task.

2.5.1.2 Input Parameters:

In most cases the simulators are capable of generating the sequence of arrivals as an input to the system. Such systems called as self-driven models [6]. But trace-driven models needs external input to determine timing of arrival events. Since arrival processes occur in a random fashion, it is not easy to predict arrivals and departures of the messages. This is because executing a simulation is analogous to conducting an experiment involving randomness; simulation outputs must be treated as random observations [5, 14]. However, this randomness can be very closely represented by certain probability distributions. If the representations of arrival times of messages are justified by a random variable, then random number generators may be used to generate a sequence of arrival events.

In some situations, models need a better representation of the arrival events, in which measurements are made on the actual system. Trace-driven simulation technique is not very flexible, since for a different set-up one does not a freedom to gather statistics [6].

2.5.1.2.1 Generation of Random Numbers:

Since self-driven models cannot do without generation of random variables, in this section generation of random numbers will be discussed.

Random numbers might be used, for example, to generate the length of messages, inter-arrival times of messages at a given node, time between failures of a link, or probability of a transmission error. Most of the random number generators accept seed values as inputs and produce a random number as an output.

The output of a random number generator is usually uniformly distributed between 0 and 1. However, for using these numbers in a simulation model we have to convert these numbers so that they follow a distribution function of our choice. The two common methods for converting uniformly distributed numbers to a desired probability distribution are: inverse transform method and rejection method [6].

Inverse Transform Method: For probability distributions whose inverse transform can be easily found (such as exponential distribution), inverse transform method may be helpful. Let U be a random variable uniformly distributed between 0 and 1. Let Y be the required random variable to be generated according to a given probability distribution function F(y). If F(Y) is set to be equal to U, then it can be shown that

the variable Y defined by $Y= F^{-1}(U)$ follows the cumulative distribution F(y) [6]. The inverse mapping $F^{-1}(.)$ can be performed analytically.

Acceptance-Rejection Method: If f(u) is defined to be the probability density function and if it is bounded by B and have a finite support, say x<u<y; U which is the required random number to be generated according to f(u) can be computed as follows [6]:

1. Generate a pair of uniformly distributed numbers (V_1, V_2) between 0 and 1.
2. Calculate a random number U_1, such that, U_1=x + (y-1) V_1.
3. Accept U_1, if $B V_2 < f(U_1)$ and reject otherwise.
4. Stop if U_1 is accepted and otherwise try again with a new pair of random numbers.

2.5.1.3 Convergence of Results:

The accuracy of the results produced by a simulation model depend upon several factors such as accuracy of the model, simulation time, starting state of the model, stream of random variables etc. If the simulation had been run for either more or less time, or a different stream of random numbers had been used, different values would have been obtained for these performance measures. Usually we are interested in the steady state results of the system [6].

If the network modeler is interested in the average delay performance of the first 200 messages through a node, the results may depend on quite strong on the initial conditions of the simulation (the number of messages initially in the node).

Performance measures depending on the initial conditions of the simulation are referred to as *transient* measures [5].

If the simulation starts from an empty state, it will of course take some time before it reaches to the steady state. If we deal with the steady state or long-term results of the system, we must ensure that we have passed the transient portion. But determining enough simulation time, for convergence of the results to a reasonable level is a hard task, since there is no well-specified point in time at the transient phase ends [5]. To make sure that, running the simulation for an extremely long period, so that the effects of the initial state of the simulation on the performance measures of interest are negligible; is a crude method. Rather, the effects of the initial configuration become less important as the length of the simulation increases. A relatively better approach is to run the simulation for longer duration of time when the rate of occurrence of events is high or vice-versa [6].

Convergence of simulation results is usually determined with the help of the confidence intervals. A simulation is run until a desired level of confidence is achieved and then can be terminated [6].

Since the performance results produced by one simulation run depend on the particular random number stream, the results of a performance model will typically vary from one run to another. If the simulation had run for five times, and obtain five different values, all within 0.1 percent of each other, our confidence in that value would be very high. But if the five values obtained vary greatly, our confidence in any one value would be small. So confidence intervals also are used to quantify such confidence in a performance estimate [5]. If some interval (x, y) is a t percent confidence interval for the performance measure η, then if the simulation were to be

independently repeated ten times, the estimated value for η obtained from the simulation would fall in the interval (*x, y*) in approximately *t* percent of these runs.

There are ways to speed-up the process of achieving a desired level of confidence. The rule is that faster the steady state is reached, faster the desired level of confidence can be obtained. Usual ways of reaching the steady state of the simulation are [6]:

- Start the simulation from a random state instead of a empty state
- Discard a certain percentage of events in the transient portion of the simulation
- Start collecting statistics after the simulation clock has reached a certain value.

Several techniques have been devised for generating confidence intervals [5]:

Independent Replications: In this type of method the simulation is run for *m* independent times and *m* estimates are obtained for each performance measure of interest. For *m* independent samples standard statistical techniques are applied.

Batch Means: This technique divides a single run into *M* equal-length periods of time. The values of the performance measure during each of these *M* periods are taken as *M* independent samples. Determination of *M* is an important and difficult problem, because if *M* is too small, the samples will be correlated; if it is large an excessive amount of simulation time will be required.

Regenerative Method: This technique based on partitioning a single simulation run into independent sub-runs. In this approach, the simulation run is partitioned on the basis of a modeler-defined *regeneration state*, a state such that the future evolution of the simulation is statistically identical following each entry into this state. In a queuing network with Poisson arrivals, the regeneration state might be the state no messages in the system. The regeneration state thus serves to divide the simulation into independent and identically distributed partitions of time of random length. There are two drawbacks of this technique: *i*) Regeneration state may be hard to identify; *ii*) Large models may require an excessive amount of simulation time to pass through enough regeneration points to produce a valid confidence intervals.

Interval Technique: This method is also known as spectral technique. This is a single run method and it takes into account the correlation between data captured by the simulation.

2.5.1.3.1 Calculation of Confidence Intervals:

The normal procedure to calculate the confidence intervals is as follows. Let $X_1, X_2, \ldots, X_M$ be the simulation results of the same experiment but produced by M different runs. The upper and lower limits of a confidence interval regarding the simulation results are then defined by:

$$\text{Upper limit} = \overline{M} + \frac{S_Y}{\sqrt{M}} t \tag{2.1}$$

$$\text{Lower limit} = \overline{M} - \frac{S_Y}{\sqrt{M}} t \tag{2.2}$$

where $\overline{M}$ is the average value and S_Y is the standard deviation of the results. t depends on the degrees of freedom and the level of confidence, which is obtained from a t-table. Degrees of freedom (df) is defined to be df=M-1.

2.5.1.4 Validation of Simulation Results:

Validation of a simulation model is a process of making sure that the model logically does what it is supposed to do or equivalently it is the process through which the modeler satisfy himself that the simulation model is in fact a realistic and satisfactory representation of the network operating under actual traffic conditions [5, 6]. For a system, comparing the simulation results with those obtained through analytical procedures may be lead to the right direction but in simple and normal situations, one may decide about the validity through simple comparison of results, intuition and confidence intervals [6].

CHAPTER 3

SCHEDULING ALGORITHMS FOR PACKET SWITCHING NETWORKS

This chapter gives the key features of scheduling algorithms for packet switching networks. Since nearly all scheduling algorithms for wireless channels are derived from wireline scheduling algorithms, the most important of them are presented.

Before the key features of scheduling algorithms are presented, let us formalize the notion of fair allocation. Let w_i be the weight of flow *i*, and $W_i(t_1,t_2)$ be the total share in terms of bytes or bits sent in interval $[t_1,t_2]$ for that flow. Then a bandwidth allocation is considered to be fair if, for all intervals $[t_1,t_2]$ in which two flows i and j are available for transmission, the normalized work (by weight) received by them is identical (i.e. $\frac{W_i(t_1,t_2)}{w_i}-\frac{W_j(t_1,t_2)}{w_j}=0$) [20, 28, 32]. Clearly, this is an idealized definition of fairness as it assumes that packets can be broken into atomic units. Since the flows are scheduled for a quantum at a time, there will be some unfairness. The objective of a fair scheduling algorithm is to minimize the resultant unfairness (i.e., ensure that $\left|\frac{W_i(t_1,t_2)}{w_i}-\frac{W_j(t_1,t_2)}{w_j}\right|$ is as close to 0 as possible).

3.1 Scheduling Algorithms for Wireline Environment:

Since the link for wired environment is constant and always available for transmission and transmission errors are negligible, scheduling mechanisms for wired environment is relatively easy. Therefore the purpose of these algorithms is distributing this fixed bandwidth to different traffic in a fair way – often taking account the different QoS requirements agreed on at admission time of the flow.

3.1.1 First Come First Served:

The simplest of all scheduling algorithms called First Come First Served (FCFS) is used in most of the QoS aware equipment today. All the incoming packets are enqueued in a single queue, and the packet at the head of the queue is served whenever the channel is ready for transmission. If the queue is filled completely, the incoming packets are rejected and this property is called as tail dropping.

Since the main advantage of packet switching networks is to share the bandwidth on a packet-by-packet basis resulting in a statistical gain with isolation; using FCFS scheduling, no isolation is provided. Moreover FCFS does not take any specific requirements of QoS guarantees such as delay or throughput, into account. A high-throughput data can starve a low-throughput connection.

There are some queueing disciplines derived from FCFS like prioritized-FCFS, Last Come First Served (LCFS) and Strict Priority. LCFS serves the last incoming packet first, as a reverse order of FCFS. Prioritized FCFS provides one-sided protection, as packets with higher priority are isolated from packets from lower priority. Packets belonging to the “upper class” can not be isolated from each other

as there is only one “upper class”. Strict Priority can provide isolation because for each priority there exists a single queue, and the discipline selects the packets to transmit in FCFS fashion from the first non-empty queue of the highest priority.

3.1.2 Generalized Processor Sharing:

With uniform processor sharing, at a given instant, there is a set of N non-empty FIFO queues waiting to be served. During any time interval the server serves all N packets at the head of these queues simultaneously, each at a rate of 1/Nth of the link speed. Generalized Processor Sharing (GPS) is a generalization of uniform processor sharing, which allows different connections to have different service shares. Fluid Fair Queueing (FFQ) is an alternative name for GPS. To understand how FFQ works one can image the link as a pipe in which different flows are assigned specific fractions of the pipe's cross sectional area. The total cross sectional area is the link's bandwidth *r*. The fraction of area reserved for a flow i depends on the weight, Φ_i, assigned to the flow. The fraction of the area changes dynamically as the set of flows currently in transit change. The area is shared among the flows currently in transit in proportion to their assigned weights.

GPS is the ideal scheduling scheme which most algorithms try to approximate as far as possible since GPS itself is not suitable for implementation in a packet network [12]. This is caused by the fact that GPS assumes work can be done in infinitely small steps which is not the case in a packet-based network. Here the minimal amount of work is processing a packet of minimal size.

A session in transmit is said to be backlogged at time *t* if it has packets queued for service. Let $S_i(\tau,t)$ be the session *i* traffic served in the interval (τ, *t*]. Under the GPS scheme we have

$$\frac{S_i(\tau,t)}{S_j(\tau,t)} \geq \frac{\Phi_i}{\Phi_j}, j = 1,2,....,N \tag{3.1}$$

for any session that is continuously backlogged in the interval $(\tau,t]$.

Summing over all sessions *j*:

$$S_i(\tau,t)\sum_j \Phi_j \geq (t-\tau)r\Phi_i \tag{3.2}$$

Hence a session *i* is guaranteed a rate of

$$g_i = \frac{\Phi_i}{\sum_j \Phi_j} r \tag{3.3}$$

If r_i is defined to be the average rate of session *i*, then as long as $r_i \leq g_i$, the session can be guaranteed a throughput g_i. Session *i*'s backlog will be cleared at a rate $\geq g_i$. The delay of a session *i* bit arriving at a time *t* can be bounded as a function of session *i*'s queue length. These throughput and delay bound guarantees are independent of the queues and arrivals of other sessions. Assigning real numbers as needed in choosing the values of Φ_i gives good flexibility of resource allocation.

We can summarize the properties of a GPS server as follows:

- It is work conserving: No capacity is “lost” because of scheduling.

- The flows are isolated: A flow is guaranteed a service share independent of the amount of traffic of other flows.
- Flexible: Assigning a small weight to an unimportant traffic, will not affect the higher priority traffic heavily.
- The maximum queue size and the packet length bound the maximum delay a flow will experience.

Since GPS is a fluid model that can not be implemented, various approximation algorithms are designed to provide services that are almost identical to that of GPS [25, 34].

3.1.3 Weighted Fair Queueing:

Demers, Shenker and Keshav first proposed an approximation of GPS called Weighted Fair Queueing (WFQ), which is practically implementable. Parekh and Gallager [12, 15] further studied the same scheme under the name of Packet-by-packet GPS (PGPS), in the context of integrated services networks. Their main contribution was in combining the mechanism with Leaky Bucket admission control in order to provide performance guarantees in terms of both throughput and delay. The WFQ scheme is flexible in the range of throughput and delay guarantees it can order to flows while maintaining the work conserving nature of the GPS scheme.

The WFQ scheme is based on the time the packets finish service under the GPS scheme. Let F_p be the time when a packet finishes service under the GPS scheme. The WFQ is an approximation of the GPS scheme that services packets in the order of increasing F_p. However, it is possible that by the time the server is free to pick the next packet for service, the packet that would have the next smallest F_p under the

GPS scheme may not have arrived. Consequently, the scheme cannot be work conserving. According to the WFQ scheme, the server picks the next packet to finish service under GPS if no packets were to arrive after it.

Despite this shortcoming, the WFQ approximation [12] has been shown to be very close to the GPS scheme in behavior. The following points have been proved.

1. For all packets waiting to be served at any time τ, the order in which the packets will finish service under the GPS and the WFQ scheme will be the same.

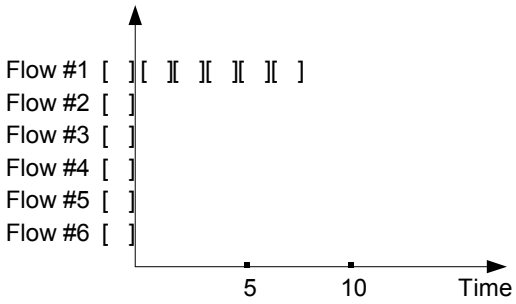
2. Let $\hat{F}_p$ be the time at which packet finishes service under WFQ. Then, $\hat{F}_p - F_p \leq \frac{L_{\max}}{r}$, where $L_{\max}$ is the maximum packet length. The service completion time for a packet under the WFQ scheme never lags behind that in case of the GPS scheme by a value more than the time it takes to service a maximum size packet.

3. Let $\hat{S}_i(t,\tau)$ be the amount of session *i* traffic in bits served under WFQ. For all times τ and all sessions *i*, $S_i(0,\tau) - \hat{S}_i(0,\tau) \leq L_{\max}$. For the amount of traffic served under the two schemes, the PGPS scheme never lags behind the GPS scheme by a value more than the maximum size packet. Consequently, the backlog in case of WFQ will never exceed the backlog in case of GPS by a value more than the maximum size packet.

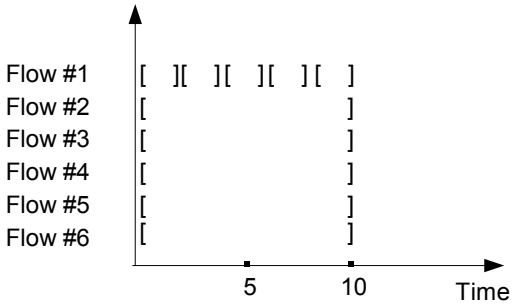
For a better understanding of how WFQ works, let us have a look at the example shown in Figure 3.1 where there are 6 flows sharing the same link. The horizontal line shows the time line and vertical axis shows the sample path of each flow. For simplicity, assume all packets have the same size of one (1) and the link speed is also one (1). Also let the weight for Flow #1 be 0.5, and the weight for each of the other 5 flows be 0.1.

In the example Flow #1 sends 6 back-to-back packets starting at time 0 while each of other 5 flows sends only one packet at time 0. If the server is GPS, it will take 2 time units to service a packet from Flow #1, and 10 time units to service a packet from another flow. This is illustrated in Figure 1 (b). If the server is WFQ, at time 0, all 6 flows have packets backlogged. Since the first packet of Flow #1 finishes at time 2 while all other packets finish at time 10 at GPS system,

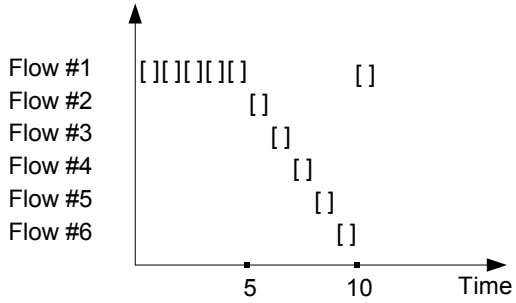
WFQ will service the fist packet of Flow #1 first. In fact, the first 5 packets of Flow #1 all have finishing times smaller than packets belonging to any other flow, which means that 5 packets of Flow #1 will be serviced back to back before packets of other flows can be transmitted. This is shown in Figure 1 (c). After the burst the next packet of Flow #1, will have a larger finishing time in the GPS system than the 5 packets at the head of other flows' queues, therefore, it will not be serviced until all the other 5 packets are transmitted, at which time, another 5 packets from Flow #1 will be serviced back to back. This cycle of bursting 5 packets and going silent for 5 packets can go indefinitely. With more flows, the length of the period between bursting and silence can be larger [18].



(a) PacketArrivals



(b) GPS Service Order



(c) WFQ Service Order

Figure 3.1 An example showing how WFQ works. (a) Packet Arrivals (b) GPS Service Order (c) WFQ Service Order

WFQ will service the fist packet of Flow #1 first. In fact, the first 5 packets of Flow #1 all have finishing times smaller than packets belonging to any other flow, which means that 5 packets of Flow #1 will be serviced back to back before packets of other flows can be transmitted. This is shown in Figure 1 (c). After the burst the next packet of Flow #1, will have a larger finishing time in the GPS system than the 5 packets at the head of other flows' queues, therefore, it will not be serviced until all the other 5 packets are transmitted, at which time, another 5 packets from Flow #1 will be serviced back to back. This cycle of bursting 5 packets and going silent for 5 packets can go indefinitely. With more flows, the length of the period between bursting and silence can be larger [18].

3.1.4 Worst Case Fair Weighted Fair Queueing:

GPS based schemes have been used in the context of feedback based congestion control. A source constantly samples feedback from receiver in order to check for symptoms of network congestion. The source controls the rate, which it admits packets into the network by reacting appropriately to these symptoms. Misbehaving sources might try to take advantage of network resources by sending packets dis-regarding, or otherwise simply ignoring symptoms of congestion. Fair allocation of bandwidth at queueing points would ensure that such misbehaving sources do not hog up the network resources. The GPS discipline offers fair allocation of bandwidth and protection from misbehaving sources. Robust congestion control algorithms can be built based on the more accurate measurement and protection provided by a GPS like servicing discipline.

It has been shown that delay of any packet in the WFQ scheme as compared to its delay in the GPS discipline is no greater than the transmission time of one packet.

In terms of service rate, the WFQ discipline does not fall behind the corresponding GPS discipline by more than one max packet size. WFQ was considered as the best way of approximating the GPS scheme. It was found later by Bennet and Zhang [18], contrary to popular belief, that large discrepancies could occur between behavior of the GPS scheme and the WFQ scheme. They found that while no flow can lag behind too much, but particular flows can be significantly ahead.

Bennet and Zhang [18] proposed a new approximation to the GPS service discipline called Worst-Case Fair Weighted Fair Queueing (WF^2Q). This service discipline shares both the bounded delay and the worst-case fairness properties of the GPS discipline [31]. They noted that the problem in WFQ is due to the fact that service of a packet can start earlier than its start time in the GPS [19]. While WFQ selects the next packet to service among all available packets, the WF^2Q scheme selects the next packets to service among a subset of the available packets. When picking the next packet for service, WF^2Q considers only the set of packets whose service would have started in the corresponding GPS scheme. Among the packets in this set it chooses that packet which will finish service first under GPS as the next packet to be serviced. The service order for the sessions with arrival pattern as shown in Figure 3.1 (a) will be as shown in Figure 3.2.

Many useful properties of WFQ are retained by WF^2Q. Like in the case of WFQ, the worst-case delay bounds for packets in fluid GPS and WF^2Q system differ by no more than the time to service a single packet of maximum size. For any session, the service rate in terms of the bits served by WF^2Q does not lag behind the fluid GPS system by any value greater than the maximum packet size. Consequently, the backlog of any session will not exceed its backlog in GPS by a value greater the

maximum packet size. In addition to these properties, while WFQ can be quite ahead of the GPS system, WF^2Q cannot go ahead of GPS by more than a fraction of the maximum packet size. Since the service provided can be neither far behind nor too far ahead, WF^2Q provides a service almost identical with GPS system.

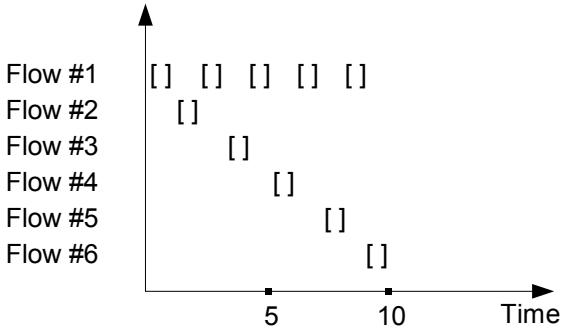


Figure 3.2 WF^2Q Service Order

3.1.5 Worst Case Fair Weighted Fair Queueing +:

Maintaining the relative GPS finish order for packets in the WFQ system by using a priority queue mechanism is based on the notion of a system virtual time function $V(\tau)$, which is the normalized fair amount of service that all backlogged sessions should receive by time τ in the GPS system. Each packet p_i^k (k^{th} packet on session *i*) has a virtual start and finish time S_i^k and F_i^k, where $V^{-1}(S_i^k)$ and $V^{-1}(F_i^k)$ are the times packet p_i^k starts and finishes services in the GPS system respectively. Another way of interpreting F_i^k is that it represents the amount of service, normalized with respect to its service share, session *i* has received right after packet p_i^k is served. In the GPS system, all backlogged sessions should receive the same normalized amount of service. Since both the system virtual time and the per

packet virtual start and finish times represent the normalized amount of service, they are measured in unit of bits. In the special case of a fixed rate server, the elapsed time of a backlogged period is also a measure of the service provided by the server; therefore, virtual times can also be measured in unit of seconds. The exact algorithm for computing virtual times are shown in Table 3.1, where $B_{GPS(\tau)}$ is the set of backlogged queues at time τ, t_0 is the beginning of the system backlogged period that includes t, $r(\tau)$ is the server rate at time, and a_i^k and L_i^k are the arrival time and the length of packet p_i^k respectively. Notice that the definition of virtual times in unit of bits is more general, and it is applicable to both fixed-rate and variable-rate servers [23].

Table 3.1 Calculation of Virtual Times [23].

	Measured by Seconds	Measured by Bits
F_i^k	$S_i^k + \frac{L_i^k}{r_i}$	$S_i^k + \frac{L_i^k}{\Phi_i}$
S_i^k	$\max\{F_i^{k-1}, V(a_i^k)\}$	$\max\{F_i^{k-1}, V(a_i^k)\}$
$V(\tau)$	$\int_{t_0}^{t} \frac{1}{\sum_{i \in B_{GPS}(\tau)} \Phi_i} d\tau$	$\int_{t_0}^{t} \frac{r(t)}{\sum_{i \in B_{GPS}(\tau)} \Phi_i} d\tau$

Since there can be N backlogged or unbacklogged sessions during an arbitrary small interval, the worst-case complexity of computing $V_{GPS}(.)$ is O (N) [16].

After then Bennet and Zhang [16] proposed a new scheduling technique named Worst Case Weighted Fair Queueing+ (WF^{2}Q+) that has the same delay bounds as WF^{2}Q. This discipline has a lower complexity since it does not need to compute

$V_{GPS}(.)$. The new time function they have proposed has an overall complexity of O (log N). This new approach eliminates the need to simulate the corresponding GPS system.

The service discipline named Worst Case Fair Index (WFI), introduced in [18] for characterization of fair queueing mechanisms is provided to be as same as WF^2Q by this new scheme. Also same delay bound guarantees is provided. WF^2Q defines its virtual time function as:

$$V_{WF^2Q+}(t+\tau) = \max(V_{WF^2Q+}(t) + W(t,t+\tau), \min_{i \in \hat{B}}(S_i^{h_i(t)})) \qquad (3.4)$$

where $W(t,t+\tau)$ is the total amount of service provided by the server during the period [t , t+τ], $\hat{B}(t)$ is the set of sessions backlogged in the WF^2Q + system at time *t*, $h_i(t)$ is the sequence number of the packet at the head of the session *i*'s queue, and $S_i^{h_i(t)}$ is the virtual start time of the packet. Other than updating virtual start and finish times on arrival/transmission of packets, these variables are recalculated, when a packet p_i^k reaches its head of queue according to the following formula:

$$S_i = \begin{cases} F_i & if\ Q_i(a_i^k -) \neq 0 \\ \max(F_i, V(a_i^k)) & if\ Q_i(a_i^k -) = 0 \end{cases} \qquad (3.5)$$

$$F_i = S_i + \frac{L_i^k}{r_i} \qquad (3.6)$$

where $Q_i(a_i^k -)$ is the queue size of session *i* just before time a_i^k .

Since at least one packet has a virtual start time lower or equal to the system's virtual time, the algorithm is work conserving by selecting the packet with the lowest eligible start-time for transmission like in WF^2Q.

3.1.6 Weighted Round Robin:

Weighted Round Robin (WRR) is based on an idea first proposed by Nagle [4]. To introduce fairness, Nagle proposed to store packets in a different queue for each source and to serve one packet from each queue in a round-robin manner. With this discipline, the strategy for one queue is no more to transmit all packets, but to use its share of bandwidth. Sending more than one packet results in delays to increase and possibly loss of packets from other queues. Note that this mechanism also provides isolation between flows.

Nagle has defined "fairness" in terms of number of packets, so that a source using large packets will receive larger "fair share" (in terms of bandwidth) than a source using small packets. To overcome this issue, Demersi Keshav and Shenker [7] proposed an algorithm emulating a bit by bit round-robin that incoming packets are labeled with the sending time of its last bit in a bit by bit round robin, and served in the increasing order of these labels. This scheme is illustrated in Figure 3.3.

Dealing with the problem of possible deservation of bigger share, every flow can be given weights, so that all of them will be allowed to send a given number of bits at each round. So no flow receives more than its request.

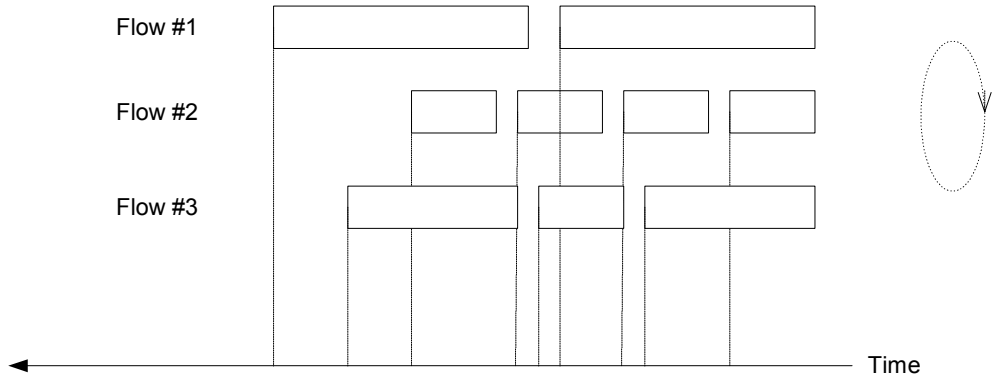


Figure 3.3 Bit-by-bit Round Robin Emulation

WRR assumes that an average packet size is known. Every flow has a weight w_i corresponding to the service share it is supposed to get. A non-empty session will serve the packet at the head of its queue at a rate of $\frac{w_i}{\sum_j w_j} r$, where r is the total rate. Despite its ease of implementation and reduced computational complexity (require only O (1) work to process a packet), missing a packet in a heavily loaded system will result in a significant delay. Assumption of knowing average packet size is another disadvantage for systems where it is not known or highly varying. Figure 3.4 presents how WRR can be implemented to reduce the delays for packets belonging to the flows with small weights.

In the example shown in Figure 3.4, Flow #1, Flow #2 and Flow #4 receive twice more frequent service than Flow #3. Figure 3.4 (a) shows the classic round robin. Figure 3.4 (b) illustrates a first approach to WRR scheduling. In part (c) of the figure, the frequencies of visits are maintained the same, but the visits to the "frequent" clients are spread more evenly in time [9]. By doing so, the delay bound for flows with lower weights may be smaller.

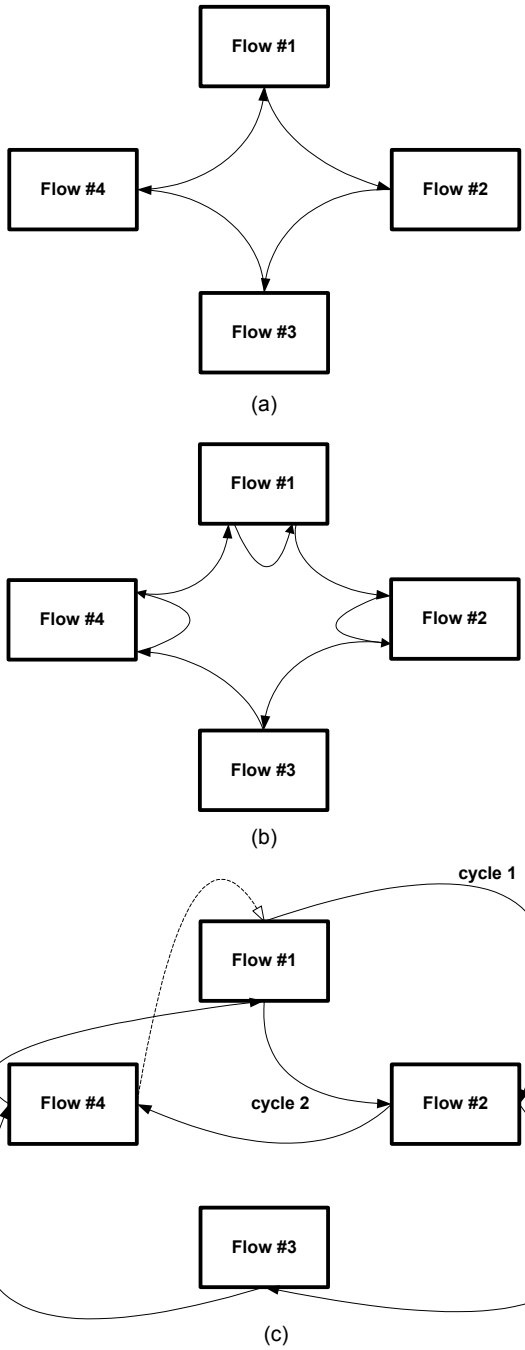


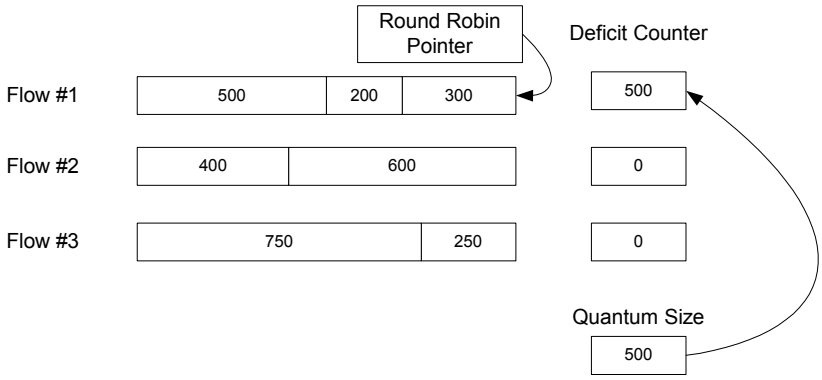
Figure 3.4 (a) Round-robin (b) WRR (c) Visits

3.1.7 Deficit Round Robin:

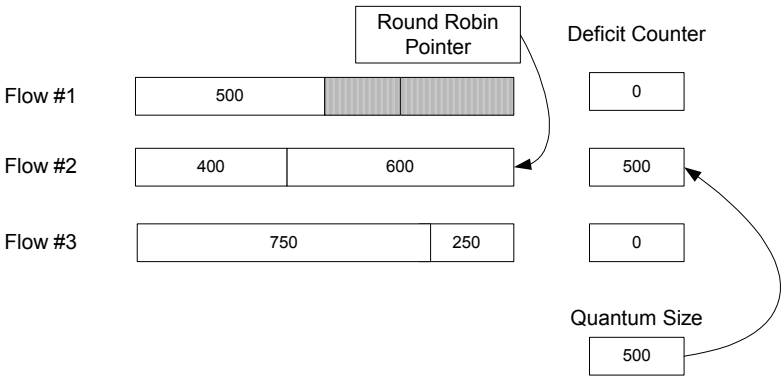
Deficit Round Robin was suggested to overcome the flaw of WRR as it is assumed that WRR can be fair if the average packet size is known. Besides the ease of implementation and achieving nearly perfect fairness in terms of throughput; this new scheme proposed by Shreedhar and Varghese requires only O (1) work to process a packet [17]. The basic idea is to keep track of the deficits a flow experiences during a round and to compensate them in the next round. When the packet at the head of queue of a flow can not be sent because of its size in a round, the missing amount of service is added in the next round.

Let the number of bytes sent out for queue *i* in round *k* be $bytes_{i,k}$. Each queue *i* is allowed to send out packets in the first round subject to the restriction that $bytes_{i,1} \leq Quantum_i$. If there is no more packets in queue *i* after the queue has been serviced, a state variable called $DeficitCounter_i$ is reset to 0. Otherwise, the remaining amount $\left(Quantum_i - bytes_{i,k}\right)$ is stored in the state variable $DeficitCounter_i$. In subsequent rounds, the amount of bandwidth usable by this flow is the sum of $DeficitCounter_i$ of the previous round added to $Quantum_i$. Avoiding examining the empty queues, an auxiliary list named *ActiveList*, is kept to point out the queues that have at least one packet. If a packet arrives to an empty queue, it is added to the end of *ActiveList*.

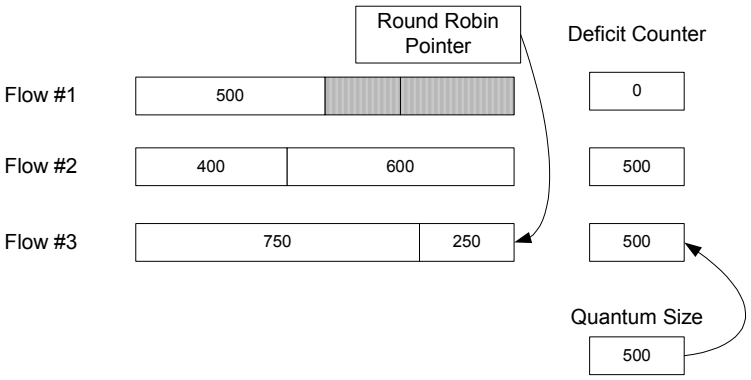
The algorithm services up to $Quantum_i + DeficitCounter_i$ worth of bytes from queue *i*. If the queue has still packets to send but $DeficitCounter_i$ is not sufficient for transmission, the round robin pointer moved to the next queue in the *ActiveList*.



(a)



(b)



(c)

Figure 3.5 An example showing how DRR works **(**a) Step 1 (b) Step 2(c) Step3

To get a better understanding of how DRR works, let us have a look at the example shown in Figure 3.5. At start all *DeficitCounter* variables initialized to zero. The round robin pointer points to the first element of *ActiveList*. In the example, when the queue of Flow #1 is serviced the variable *QuantumSize* is added to the *DeficitCounter* of Flow #1. The remainder after serving the packet size of 200 is left in the *DeficitCounter* variable. But the size of next packet is equal to the *DeficitCounter*, so it is served too. *DeficitCounter* of Flow #1 is now equal to 0. Since there is no share for Flow #1, round robin pointer points to the next non-empty queue in *ActiveList,* which has a packet size of 600, bigger than the *QuantumSize* of it. So pointer moves to the next element in *ActiveList,* but *QuantumSize* is retained. The scenario for few steps can be seen in Figure 3.5 (a), 3.5 (b) and 3.5 (c).

In the simplest case $Quantum_i = Quantum_j$ for all flows *i; j*. Exactly as in Weighted Fair Queuing [7], each flow *i* can ask for a larger relative bandwidth allocation and the system manager can convert it into an equivalent value of $Quantum_i$. Clearly if $Quantum_i = 2xQuantum_j$, the manager intends that flow *i* get twice the bandwidth of flow *j* when both *i* and *j* are active [17].

3.1.8 Start Time Fair Queueing:

To achieve fairness, a slightly different approach is proposed by Goyal, Vin and Cheng [20] named Start-time Fair Queueing (SFQ). SFQ assigns a start tag to each packet and schedules them in the increasing order of start tags. To define the start tag, let the packets be scheduled for variable length quantum at a time. Also, let q_f^i and l_f^i denote j^{th} the quantum of packet *f* and its length (measured in bytes or

bits), respectively. Let $A(q_f^j)$ denote the time at which the j^{th} quantum is requested. If the flow is making a transition from a non-backlogged mode to backlogged mode, then $A(q_f^j)$ is the time at which the transition is made; otherwise it is the time at which its previous quantum finishes. Then SFQ algorithm is defined as follows:

1. When quantum q_f^i is requested by packet *f*, it is stamped with start tag S_f computed as:

$$S_f = \max(v(A(q_f^j)), F_f) \tag{3.7}$$

where *v(t)* is the virtual time at time *t* and F_f is the finish tag of packet *f*. F_f is initially 0, and when j^{th} quantum finishes execution it is incremented as:

$$F_f = S_f + \frac{l_f^j}{r_f} \tag{3.8}$$

where r_f is the weight of the flow that packet *f* belongs to.

2. Initially the virtual time is 0. When the channel is busy, the virtual time at time *t*, *v (t)*, is defined to be equal to the start tag of the packet in service at time *t*. On the other hand, when the channel is idle, *v (t)* is set to the maximum of finish tag assigned to any packet.
3. Packets are serviced in the increasing order of the start tags; ties are broken arbitrarily.

The example shown in Figure 3.6 illustrates the computation of the virtual time, as well as the start and the finish tags (and hence, the process of determining the execution sequence) in SFQ. Consider two flows Flow #1 and Flow #2 with weights 1 and 2, respectively, which become backlogged at time t = 0. Let the $l_f = 10$ for all packets. Let each packet consume the full length of the quantum each time it is

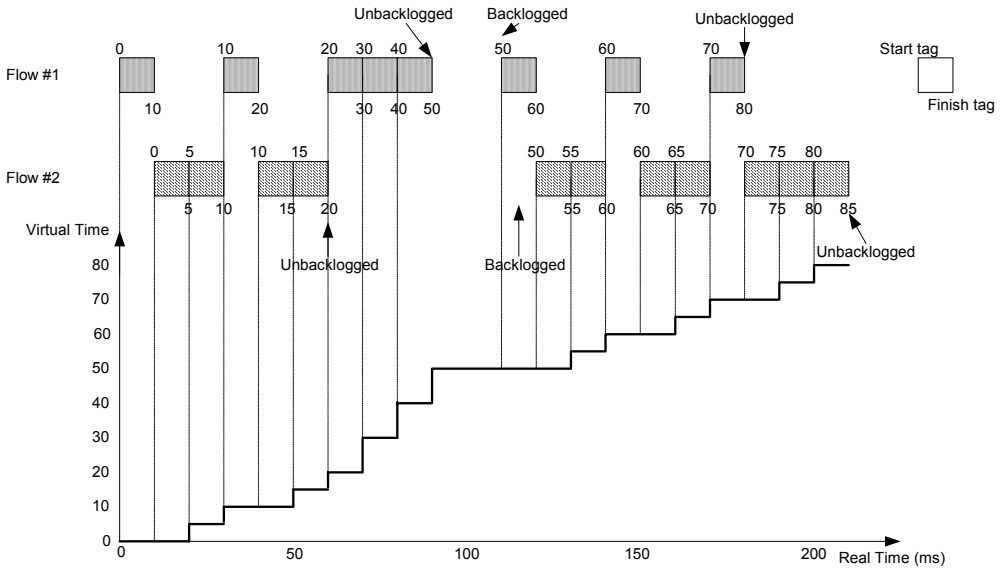


Figure 3.6 An example showing how SFQ computes virtual time, start tag and finish tag

scheduled. Initially, the virtual time *v (t)* = 0. Similarly, the start tags of Flow #1 and Flow #2, denoted by S_1 and S_2, respectively, are zero (i.e., $S_1 = S_2 = 0$). Since ties are broken arbitrarily, let us assume, without loss of generality, that the first packet at the head of Flow #1 is scheduled first. Since *v (t)* is defined to be equal to the start tag of the packet in service, for $0 < t \leq 10$: *v (t)* = $S_1 = 0$. At the end of that quantum, the finish tag of the first packet of Flow #1 is computed as $F_1 = 0 + \frac{10}{1} = 10$. Moreover, assuming that the flow remains backlogged at the end of the quantum, it is stamped with $S_1 = \max(v(10), F_1) = 10$. At this time, since $S_2 < S_1$, the first packet of Flow #2 is scheduled. Note that since $S_1 = 0$, the value of *v (t)*; $10 < t \leq 20$ continues to be equal to 0. At the end of this quantum, the finish tag for B is set to

$F_2 = 0 + \frac{10}{2} = 5$. Moreover, assuming that Flow #2 remains runnable at the end of the quantum, we get $S_2 = \max(v(20), F_2)$=5. Carrying through this process illustrates that, before Flow #2 become unbacklogged at time *t*= 60, Flow #1 and Flow #2 are scheduled for 20ms and 40ms, respectively, which is in proportion to their weights. When Flow #2 is unbacklogged, the entire channel is available to Flow #1, and the value of *v (t)* changes at the beginning of each quantum of Flow #1. Now, when Flow #1 becomes unbacklogged at time *t* = 90 the system contains no backlogged flows. During this idle period, *v(t)* is set to $\max(F_1, F_2) = \max(50,20) = 50$. When Flow #1 becomes backlogged at time *t* = 110, *v(t)* = 50. Hence, Flow #1 is stamped with $S_1 = \max\{50, F_1) = 50$, and is immediately scheduled for transmission. On the other hand, when Flow #2 becomes backlogged at time *t* = 115, *v(t)* = S_2 = 50. Hence, it is stamped with $S_2 = \max(50, F_2) = \max(50,20) = 50$. From this point, the ratio of bandwidth allocation goes back to 1:2. Finally, when Flow #1 becomes unbacklogged at *t* = 170, the entire bandwidth becomes available to Flow #2, until it becomes unbacklogged at *t* = 210.

3.2 Fair Scheduling Algorithms for Wireless Networks:

Since a session belonging to a wireless network may receive significantly less service than it is supposed to, and another may get more; it is difficult to provide both delay-guarantees and fairness simultaneously. This may happen because of exhibiting high, variable error rates. Many applications and end-to-end transport protocols may perform very poorly when packets are lost due to link errors. This results in large discrepancies between sessions' virtual times. So scheduling disciplines for wireline environment can not be implemented directly to wireless networks [25].

Wireless channels have the following unique characteristics, which wireline fair-scheduling algorithms do not address [28]:

1. Bursty channel errors,
2. Location-dependent channel capacity and errors.

These two properties implies that at any time, it may happen that some flows can transmit but others can not due to channel errors, resulting in only a subset of backlogged sessions to be scheduled at any time instant; also this subset is dynamically changing as time evolves. As a result, a session with an error-free channel may receive more normalized amount of service than that by a session with an error channel. To achieve long time fairness more service may be given to a previously error session [25, 28].

These characteristics make it difficult to satisfy requirements of time-sensitive applications such as high quality audio and video. Also host mobility makes it difficult to satisfy QoS requirements since routing and admission control in the mobile communications is dynamical and therefore complex.

Many resource management algorithms for wireless mobile networking environments like Idealized Wireless Fair Queueing, Wireless Fair Service, Channel State Independent Wireless Fair Queueing and Wireless Multiclass Priority Fair Queueing are proposed to suit for capacity-constrained and highly dynamic networks in order to support communication intensive applications with QoS requirements. The main design goals of these disciplines are delay and throughput guarantees for error-free sessions, long-term fairness for error sessions, short-term fairness for

error-free sessions and graceful degradation for sessions that have received excess service [25].

The following sections will give information about Wireless Packet Scheduling (WPS), Server Based Fairness Approach (SBFA) and Channel Independent Fair Queueing (CIF-Q) in detail. This chapter will also introduce adaptation of DRR for wireless channels.

3.2.1 Wireless Packet Scheduling:

WPS tries to approximate IWFQ, since for a base station performing the task of scheduling IWFQ does not predict channel state perfectly and has limited knowledge on uplink flows [27, 28]. For implementation, either WFQ or WF^2Q can be chosen as an error-free service; but WRR can also be chosen because of its ease of implementation. Also it has been stated that WRR and fair scheduling will approximately result in same performance [28].

The key features of WPS are: *Spreading*, which generates frames based on WFQ or WF^2Q when all flows are backlogged; *Swapping within Frame*, which interchanges the slots of a frame when a flow can not transmit in its slot because of channel error; *Credit/Debit Adjustment*, which slots' credit and debit are adjusted when a slot does not transmit in its frame and can not swap with another slot in the same frame, but there exits another backlogged flow that can transmit at the same time. When a new frame needs to be generated, these credit and debits are taken into account in terms of effective weights; *One-step Prediction*, which predicts the state of the physical channel using the previous channel state. In Figure 3.7 A WPS example is given. Consider three backlogged flows Flow #1, Flow #2 and Flow #3,

which have weights 3, 2, 1 respectively. The first line of Figure 3.7 shows how spreading generates frames using WF^2Q. The second line presents how the swapping works. At the time when slot *c* is scheduled, the one-step prediction algorithm predicts that both Flows #1 and Flow #3 have channel errors but Flow #2 perceives a clean channel. Since Flow #3 can not transmit in slot due to channel errors, and Flow #1 cannot either, Flow #2 is transmitted in slot *c*; therefore, Flow #2 and Flow #3 swapped slots *c* and *e*. It should be noted that at the time when slot *c* is scheduled, no predictions will be made on whether slot is e good for Flow #3 or not. As shown in the third line of Figure 3.7, if slot *e* is bad for Flow #3 according to the one-step prediction made, when slot is *e* scheduled, Flow #3 has to be scheduled in the next frame. In this case, Flow #3 still cannot be transmitted due to predicted error state for slot *e*, because there does not exist any slot for Flow #3 to swap within Frame #1, we simply maintain the credits for Flow #3 as 1, and the debits for Flow #2 as 1. Therefore, at the beginning of Frame #2, the effective weights for Flow #1 is still 3; the effective weights for Flow #2 is given by its weights minus its debits, that is 2-1=1; the effective weights for Flow #3 is the sum of its weights and its credits, that is, 1+1=2. The slot allocation for Frame #2 will be spreaded using WF^2Q based on the effective weights, and the results are shown in Figure 3.7.

It should be noted that the number of credits must be bounded in order not to allow a flow capturing the entire channel for along time. IWFQ also has a similar effect, which can be solved by bounding the compensation.

Since WPS tries to approximate IWFQ in the average case, there is a difference in the worst-case delay since it compensates by swapping rather than by giving precedence in channel access to longest lagging flows. By swapping the next attempt

of flow to access the channel is delayed to a later slot. This means that precedence history is maintained by IWFQ.

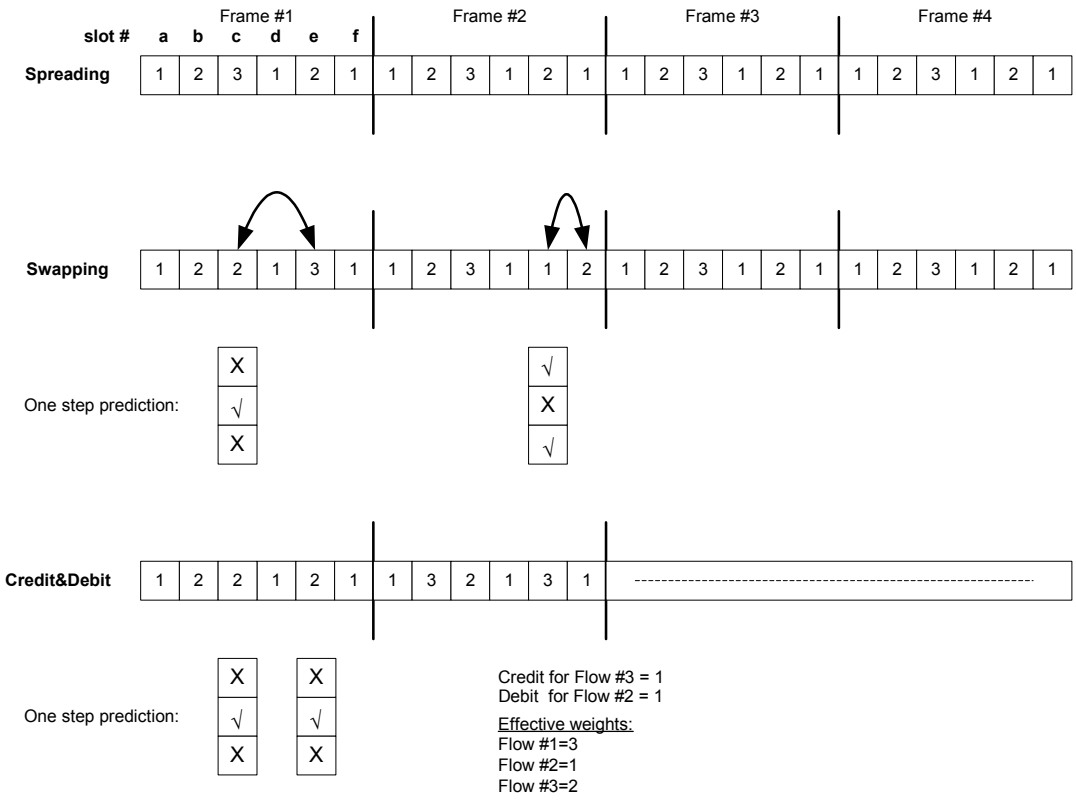


Figure 3.7 A WPS example

Another problem in WPS is, choosing a flow in a future frame instead of the current backlogged flow that has to be transmitted but cannot be given service due to a channel error. Remember that the coming frame is generated by the information based on the weights for all known backlogged flows.

3.2.2 Server Based Fairness Approach:

Another discipline proposed by Ramathan and Agrawal [27] as SBFA introduces the concept of long-term fairness server (LTFS) which shares the bandwidth with the other flows and is responsible for providing additional capacity to flows to maintain the long-term fairness guarantees. SBFA maintains two different queues for each flow, Packet Queue (PQ) and Slot Queue (SQ). When a new packet arrives for flow *i*, it is inserted into PQ_i, at the same time a *slot* is inserted with a tag *i* (identifying the slot as belonging *i*) into SQ_i. The scheduler operates on slot queues.

If a flow *i* can not transmit because of an erroneous channel, the packet is inserted in the LTFS queue. This means that SBFA distributes an extra quantum of bandwidth to lagging flows for compensation without degration of other flows. In general more than one LTFS may be created. For example one LTFS could be used for real-time traffic and a different one for interactive traffic. But all flows may be assigned to the same LTFS too.

To understand SBFA more, let us have a look at the example shown in Figure 3.8. In this example, there are two flows and a LTFS sharing a wireless link. As seen from Figure 3.8 (a) at time=0 there are two packets waiting for transmission for each flow. It is assumed that all packets are unit length and WRR scheduling policy is applied for all flows and LTFS. Suppose that the scheduler has selected Flow #1, and the wireless link is not suitable for transmission. So the packet at the head of the packet queue is deferred and the next active flow, Flow #2 is selected. Be aware that Flow #1 did not get its share of the bandwidth at time 0, but a slot with tag 1 is

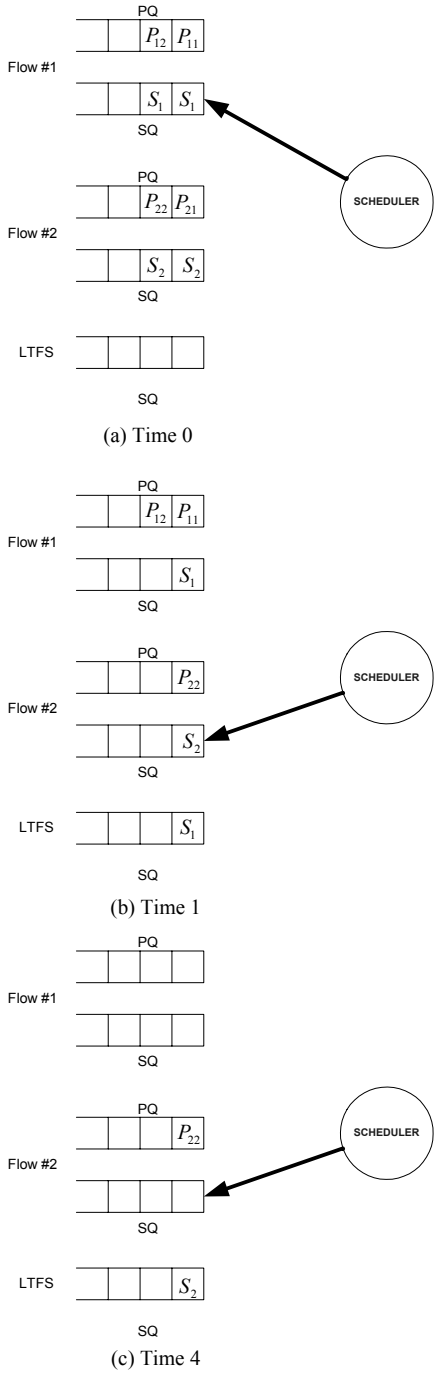


Figure 3.8 SBFA compensation in case of a deferment

inserted into LTFS. Figure 3.8 (b) presents the states of all queues assuming that the transmission of the packet at the head of packet queue of Flow #2 is successful.

At time 2, when LTFS is selected for transmission the slot's tag refers to Flow #1, so the packet at the head of packet queue of Flow #1 is transmitted assuming that the channel is good. At time 3, scheduler selects Flow #1 for transmission and the packet from Flow #1 is transmitted. Assuming that all transmissions are successful the states of queues at time 4 is shown in Figure 3.8 (c). We have seen that how SBFA compensates when a packet is deferred in case of receiving a bad channel.

3.2.3 Channel Independent Packet Fair Queueing:

CIF-Q uses SFQ as the error-free service system, in order to keep implementation low [25]. But it should be noted that any other scheduling discipline could be chosen as a reference model.

CIF-Q schedules the packet with minimum start tag for transmission, which would be served next in SFQ. When the selected flow can not send, the service is distributed to another flow. But the service is charged to the candidate flow. To do this CIF-Q introduces a lag parameter to keep the track of the amount of traffic needs to be compensated for.

The parameter lag_i represents the difference between the services that flow *i* should receive in a reference error-free model and the service it has received in real system. An active flow *i* is said to be *lagging* if its lag_i is positive and *leading* vice-versa. It is called *satisified* if lag_i is zero. It is obvious that for an error-free system

all flows will be *satisified*. The algorithm maintains at all time the following invariant:

$$\sum_{i \in A} lag_i = 0 \tag{3.9}$$

where A is the set of the active flows. So the system is work conserving. The simple version of CIF-Q is given in Appendix A.

The key features of CIF-Q scheduling scheme is given below:

- In CIF-Q, a session's virtual time does not keep track of the normalized service received by that session in the real system *S*, but in the *reference* error-free system S^r_{SFQ}.
- The additional parameter *lag* is used for keeping the track of the difference a session should get in an error-free system *S* and it get in real system S^r_{SFQ}. To provide perfect fairness all lags of sessions must be zero.
- If no session can transmit, a session is forced to receive service and we charge for that forced session. This is done even if it can not send any packet.

CIF-Q is easier to implement than IWFQ, since it is self-clocked and it does not need to emulate a fluid system.

3.2.3 DRR for Wireless Channels:

As an investigation of a new scheduling approach, DRR for wireless channels can be proposed. In this technique the quantum of the flow will be added to the

deficit counter when it receives an error. It is obvious that the scheduling scheme is not work conserving since it does not try to compensate the excess share.

CHAPTER 4

EVENT DRIVEN SIMULATION OF SCHEDULING ALGORITHMS

For evaluating the delay, jitter and throughput performances of such scheduling algorithms, a discrete event simulator has been developed to see how they perform under various loading conditions. Average delay associated with a packet transmission and throughput statistics are kept, in order to graph versus various channel loads. By capturing these, information about the behavior of fair queueing disciplines is obtained. The language to be used for the simulation model is C++.

4.1 Simulation Methods:

Before one attempts to produce a prototype, it is desirable to simulate the expected behavior of the design. Simulation of the behavior can demonstrate how successful will the product be when it is fabricated. Simulation enables to focus on a variety of conditions to see how the design will be behaving in complex systems.

In a simulation model, generally a simulated clock controls timing of events. Timing control is necessary to execute events in an appropriate sequential order. In a computer communication network events are limited to arrivals and departures of data or control messages, which a simulation model must be able to synchronize [6].

The two famous methods used for controlling timing of events are[1] [6]:

- Unit advance (synchronous timing) or Time-Driven Simulation
- Event advance (asynchronous timing) or Event-Driven Simulation

As events occur in a simulation model, the simulation clock is advanced so as to move the system forward in time. In unit-advance case or equivalently in time-driven simulation, the simulation clock is advanced by a fixed quantity where as in event advance case; the simulation clock is advanced to the time of the next event.

The choice of the method for advancing the simulation clock has a significant impact on processing or the execution time of a simulation program. In time-driven simulation, the clock is advanced by a fixed duration of time and then the system is checked as to whether or not any events has taken place. If the event has taken place, the model variables are updated according to the type of event and the process is repeated again. If no event has taken place, the clock is advanced again without any change in the model variables and the process is repeated.

In unit advance case, the duration, the simulation clock will be increased is a critical decision. If a very small duration is chosen, it will take the model unnecessarily slow. On the other hand if one chooses a relatively long duration, the results may not be accurate.

In the event advance case, the duration used for advancing the simulation clock depends on the actual events. Thus the clock is updated dynamically. The model

---

[1] http://www.winslam.com./laramee/sim/, "Event Driven Simulation Tutorial", Tom Laramee, December 1995.

finds out when the next event is going to take place and the simulation clock is advanced to that time. If the events are occurred after long intervals, the clock will also be advanced by a longer duration and vice versa.

For illustrating the difference between two methods of simulation let us have a look at the following example:

One way to simulate the floating-point unit (FPU) of an Intel® Pentium® central processing unit (CPU) is to examine every clock cycle to see what is happening in the FPU. The "events" that causes the system to change the system to change its state are:

- Beginning of Division Operation
- Completion of Division Operation
- Beginning of Multiplication Operation
- Completion of Division Operation

A sample trace of time-driven simulation is given in Table 4.1. In the example division operation completes in 4 cycles and multiplication operation completes in 5 cycles.

Table 4.1 Time-Driven simulation of FPU

CLOCK CYCLE	EVENT
1	Beginning of Multiplication Operation
2	None
3	None
4	None

CLOCK CYCLE	EVENT
5	None
6	Completion of Multiplication Operation
7	None
8	Beginning of Division Operation
9	None
10	None
11	None
12	Completion of Division Operation
13	None
14	None

An event-driven simulation trace is given in Table 4.2.

Table 4.2 Event-Driven simulation of FPU.

CLOCK CYCLE	EVENT
1	Beginning of Multiplication Operation
6	Completion of Multiplication Operation
8	Beginning of Division Operation
12	Completion of Division Operation

As seen from the table, event-driven simulation only examines the events that cause state changes. Time intervals, where the state of a system does not change, are skipped.

If a queueing network is simulated in a time-driven manner, every distinct moment of time must be traced. This results in a large amount of processing overhead. For example, if there is a possibility that system loading is such that there could be 100 events in a second, then "time" must be quantified into pieces each 0.01 seconds long, and each of these 100 units of time must be examined during the simulation to see if an event has occurred.

If only 70 of these events actually occur, it is still necessary to examine all 100 "units of time" in the time-driven simulation. But in the event-driven simulation, you only examine the 70 changes of state. This saves a big amount of processing time to complete the simulation[1].

In this thesis, since event-driven simulation technique believed to be both efficient as well as more accurate [6], scheduling algorithms for packet switching networks will be simulated as in event advance case.

4.2 A Scheduling System Simulation Skeleton:

This part introduces the basic event handling mechanism of the discrete event simulator that has been developed.

The simulator developed, has a graphical user interface (GUI) in order to seed simulator with different operating parameters. With this GUI user can input simulation time (in seconds), link capacity (in megabits per second), the number of flows that exist in the system, traffic models and weights of these flows and the scheduling algorithm that is to be simulated.

The simulator can support maximum of five flows and the weights of these flows can alter between 1 and 5. Hierarchical models are not applied in the basic structure.

For comparing performance of queueing algorithms, an accurate traffic model has to be found. This is sometimes difficult because nature of the source or the encoding method may affect the statistics of the traffic. The source of data is practically a sensor, which samples a physical quantity to produce a signal. The following source models are applied, as shown in Figure 4.1:

- Constant Bit Rate (CBR): Fixed-size packets arrive at deterministic intervals as shown in Figure 4.1 (a). The sources generate data, which has few redundancies. The data must not be compressed in a lossy way because the content is important.

- Variable Bit Rate (VBR):
    - On/Off Sources: The source alternates between a period in which fixed-size packets arrive with deterministic spacing and an idle period as shown in Figure 4.1 (b) where *T* is period. Voice traffic is a good example.

    - Periodic with Variable Packets Sizes: The source submits a variable-size packet to the network at deterministic intervals as shown in Figure 4.1 (c) where *T* is period. Compressed video may be a good example.

- Aperiodic with Variable Packet Sizes: Variable-size packets are generated at randomly distributed intervals as shown in Figure 4.1 (d).

- Aperiodic with Constant Packet Sizes: Fixed-size packets are generated at randomly distributed intervals as shown in Figure 4.1 (e).

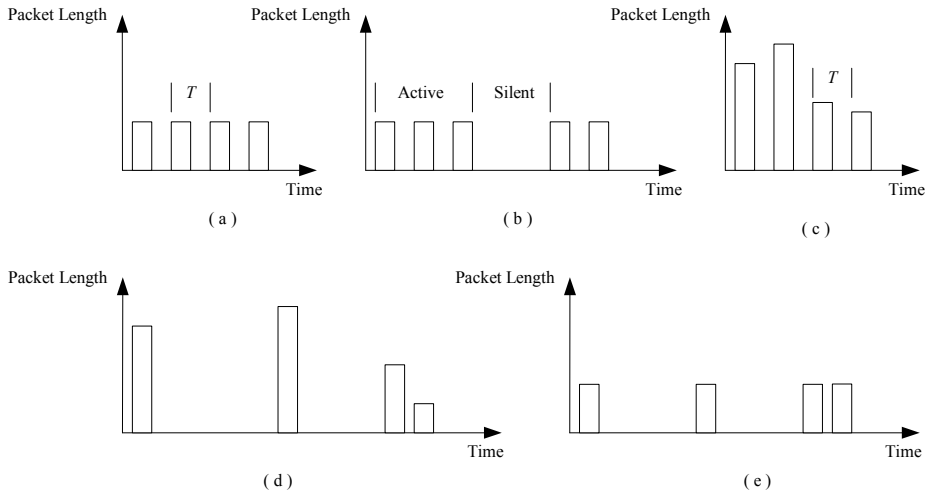


Figure 4.1 Examples of different types of traffic sources

User can choose one of four traffic models for flows from the GUI. These may be CBR or VBR, which generates variable size packets at exponentionally, distributed interarrival times. User can also choose On-Off and Poisson, which is another kind of VBR that generates fixed-size packets at exponentionally distributed interarrival times. User can input the period (in seconds), mean arrival rate (in milliseconds), packet length (in bytes) and mean packet length (in bytes) for various traffic models. Also user can assign the weights of the flows that exist in the system.

For CBR type of traffic user can input the period and packet length, for VBR user can input mean packet length and mean interarrival time. The mean interarrival time and mean packet length are Poisson distributed. For On-Off type user can input mean "on" time. The "off" period is the multiple of that "on" period. The packets are generated with a constant period within the "on" time. The packet size and the period of generation in the "on" time are declared as macros. For Poisson distributed type of traffic user inputs the mean interarrival time and the packet length is fixed (the value that user entered does not change until the end of simulation).

For generating Poisson distributed arrivals, inverse transform method described in section 2.5.1.2.1 is used. If the arrival process is Poisson, the interarrival time distribution is exponential with rate parameter λ [14]. Let:

$$P_i(t) = \Pr[i \text{ arrivals in a period of length } t] \tag{4.1}$$

and let X be the random variable describing the interarrival times for a Poisson process:

$$\Pr[X > t] = P_0(t) = e^{-\lambda t} \tag{4.2}$$

$$F_x(t) = \Pr[t \geq X] = 1 - e^{-\lambda t} \tag{4.3}$$

where $F_x(t)$ is the cumulative distribution function for random variable X. Define:

$$u = 1 - F_x(t) = e^{-\lambda t} \tag{4.4}$$

The inverse of u is:

$$t = F_x^{-1}(u) = -\frac{\ln(1-u)}{\lambda} \tag{4.5}$$

If u is uniform in (0,1) then 1-u is also uniform in (0,1). Hence:

$$t = -\frac{\ln(u)}{\lambda} \tag{4.6}$$

As statistics, the delay associated with each packet is recorded. For the simulator, delay is the difference between the first bit of a packet started to transmit and the last bit of the same packet is received.

Jitter is defined as the standard deviation of the delay. In the literature the term delay jitter is defined as the maximum deviation between the delays experienced by packets in a single connection [13]. For example, in a connection if the minimum end-to-end delay seen by a packet is 3 milliseconds and the maximum is 7 milliseconds, the delay jitter is 4 milliseconds.

Interactive applications require a bound on both delay and jitter, but some certain applications such as non-interactive television and audio broadcasting, require bounds on jitter but not delay [13].

Another performance measure recorded is throughput. Throughput is typically measured in bits/second. In Section 5, before the presentation of the simulation results, definition of fairness measures will be introduced. Comparing this quantity for a scheduling algorithm with another one will be helpful in fairness analysis.

The process flow diagram of the discrete event simulator is given in Figure 4.2.

The simulator stores the simulated state of the system in a set of system state variables. Event routines cause state variables to be modified. An event list is used to control the execution sequence of these event routines. This event list consists of events in increasing chronological order. Event routines can add or delete items from the event list. The random number generators in these event routines provide randomness for modifying and scheduling of future events. In our case the event list is implemented as a linked list. Running a simulation is a repeated execution of a loop, where at each iteration the event with earliest scheduled time is executed.

In the model the incoming packets are represented by data stuctures. These entities contain simulation-specific information as well as length indication and flow identificaiton. For example packet-creation time stamp is used for statistics colllection. Such encapsulation of data structures is a common feature in communication networks and is also supported by object-oriented programming [30]. The termination of the simulation depens on the simulation time that is determined by the user.

For clarity in showing the effects of channel errors and for ease of interpretation, errors are modeled as simple periodic bursts. Error patterns represent a periodic burst of predefined period, with predefined period of intermediate error-free time.

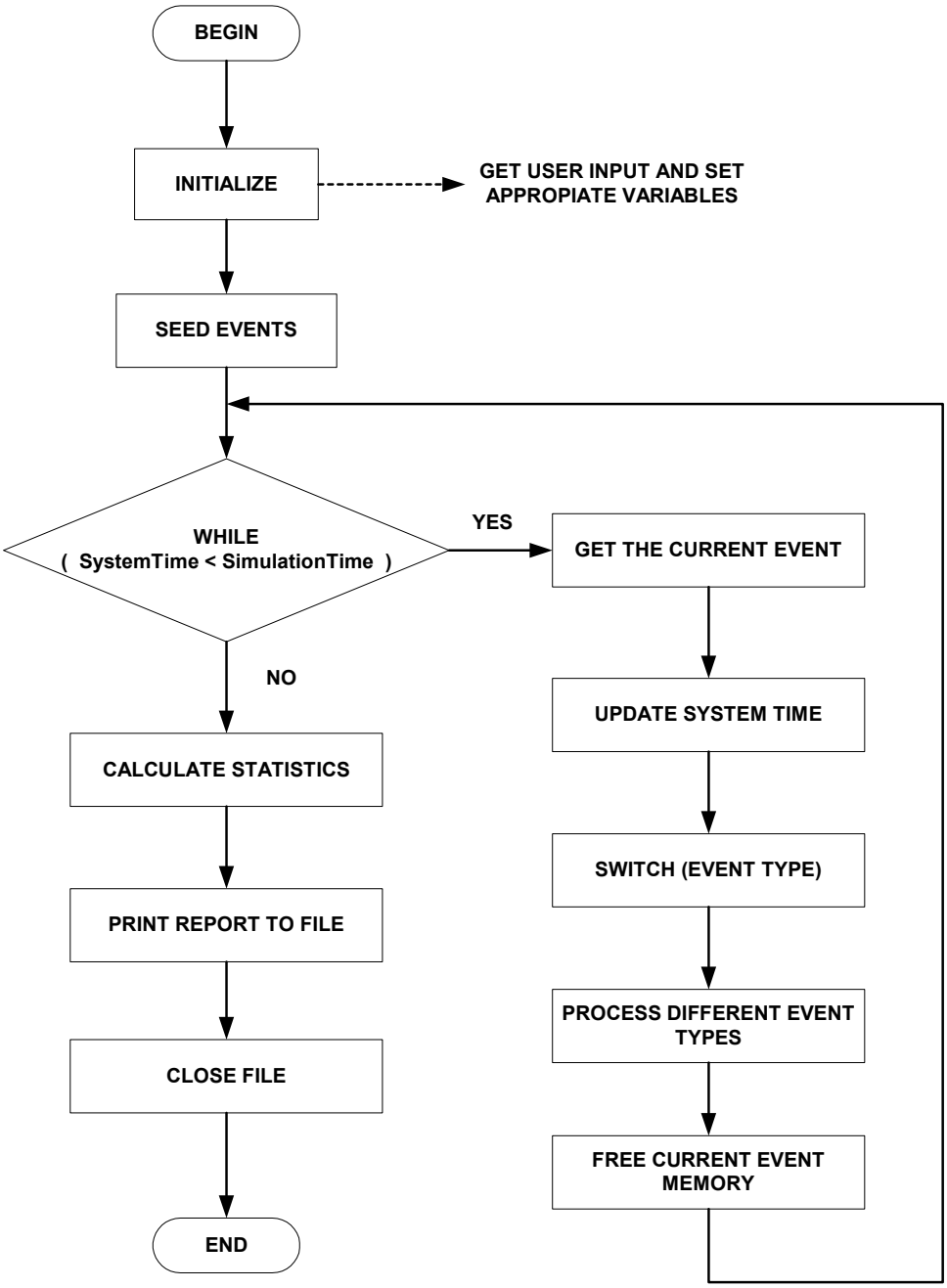


Figure 4.2 Process Flow Diagram for Event Handling Mechanism

Since single-node fairness analysis is the scope of this thesis, the simulator does not deal with routing, acknowledging, and flow controlling or removing and insertion of identifiers. Those may be considered in the end-to-end fairness analysis.

4.3 Validation of Implementations:

In this section, the simulation results obtained from the simulator we have designed will be compared to the analytical and simulation results taken from literature, to make it sure that the model logically does what it is supposed to do. In the cases where no simulation results for the models exist, comparison with the analytical results will be performed for the validation process.

4.3.1 Validation of Weighted Fair Queueing Simulation:

The validation strategy is based on the observation that the WFQ algorithm approximates the behavior of a GPS server except for an error term that is bounded by $L_{\max}$ [12, 18]:

$$W_{i,GPS}(0,\tau) - W_{i,WFQ}(0,\tau) \le L_{\max} \forall i, \tau \tag{4.7}$$

where $W_{i,GPS}(0,\tau)$ and $W_{i,WFQ}(0,\tau)$ are the total amounts of service received by session i, (the number bits transmitted) by time τ under WFQ and GPS respectively, and $L_{\max}$ is the maximum packet length.

The main goal of the following simulations is to verify the difference between services received by WFQ and GPS never violates the maximum theoretical limit. In fact, with the virtual time implementation of the WFQ algorithm, the server

keeps the track of the progress of the corresponding GPS server. However, this is not a good validation strategy, since both the output WFQ system and the corresponding GPS system could be wrong. A good validation strategy should take as reference the behavior of a system external to the WFQ implementation.

Remember that equation 3.3 holds when all sessions in the system are continuously backlogged. In these time intervals the behavior of the GPS server (from the standpoint of a backlogged session) is exactly the same as the behavior of a FIFO scheduler scheduler with rate g_j. Therefore, in order to monitor the difference between the services received by GPS and WFQ scheme, it is possible to compare the output of the WFQ system with the one of a FIFO system with the same input traffic pattern[2]. However, in order to make things work, other sessions' backlogs should never be zero. In the simulation scenario, the behavior of the WFQ implementation is compared with the behavior of a reference FIFO system.

In the simulation study, a WFQ server with a rate 150000 bit per second (bps) is shared by two sessions producing Poisson distributed traffic with fixed packet lengths is considered. As a reference, the behavior of the first session is taken and a FIFO server with rate 100000 bps with the same input pattern is considered.

Poisson distributed traffic is chosen to avoid periodical phenomena. In fact, if CBR sources had been used, the discrepancy between the amounts of service share would have shown a replicated behavior.

---

[2] http://netgroup-serv.polito.it/netgroup/hp/qos/validation.ps, "Validation of the WFQ code"

In the following table the sessions' parameters are given.

Table 4.3 Session parameters for WFQ validation.

	Mean inter-arrival time (millisecond)	Packet Size (bytes)	Average Rate (bps)	Weight
Session 1	20	400	160000	2
Session 2	40	360	72000	1

Be aware that, to respect the condition of non-zero backlogs, the total link required by the sessions is almost the twice of actual link capacity. The simulation is run for 200 seconds and the behavior of session 1 is monitored both in FIFO and WFQ simulations. We expect that the difference between the service shares both in FCFS and WFQ simulations for session 1 to be less than $L_{\max}$=3200. For 10 runs the results are given in Table 4.4.

Table 4.4 Differences of service shares in terms of bits for FCFS and WFQ simulations.

Simulation number	$W_{1,WFQ}(0,200)$	$W_{1,FCFS}(0,200)$	$\lvert Difference \rvert$
1	20000000	20000000	0
2	20000000	20000000	0
3	20000000	20000000	0
4	19996800	20000000	3200
5	20000000	20000000	0
6	20000000	20000000	0
7	19996800	20000000	3200
8	19996800	19996800	0

Simulation number	$W_{1,WFQ}(0,200)$	$W_{1,FCFS}(0,200)$	$\lvert Difference \rvert$
9	20000000	20000000	0
10	19996800	20000000	3200

From the table it is seen that, for all runs the difference between the amounts of network share does not exceed the maximum bound 3200. The lower and upper limits of the %90 confidence interval for the quantity $W_{1,WFQ}(0,200)$ are 19996000 and 20001000 respectively. The same confidence limits for the quantity $W_{1,FCFS}(0,200)$ are 19998000 and 20001000 respectively.

These results show that the behavior of the WFQ implementation is correct.

4.3.2 Validation of Worst Case Fair Weighted Fair Queueing+ and Start Time Fair Queueing Simulations:

The scheduling algorithm at node *j* is said to provide a fairness guarantee, if in any time interval $[t_1,\ t_2]$ during which two flows *f* and *m* are continuously backlogged, the number of bits of flows *f* and *m* transmitted by the server, $W_{f,j}(t_1,t_2)$ and $W_{m,j}(t_1,t_2)$ respectively satisfy:

$$\left|\frac{W_{f,j}(t_{1,},t_2)}{r_f}-\frac{W_{m,j}(t_1,t_2)}{r_m}\right|\le U_{j,\{f,m\}} \tag{4.8}$$

where r_f and r_m are the rates reserved for flows *f* and *m* respectively and $U_{j,\{f,m\}}$ is the unfairness measure – a constant that depends on the scheduling algorithm and traffic characteristics at server j [20, 28, 32]. Table 4.5 lists the $U_{j,\{f,m\}}$ values for WF^2Q and SFQ algorithms. Since WF^2Q+ provides the same WFI as WF^2Q [23], the unfairness measure for WF^2Q+ is the same as WF^2Q.

Table 4.5 Unfairness measures for SFQ and WF^2Q [32].

Algorithm	$U_{j,\{f,m\}}$
SFQ	$\frac{l_f^{\max}}{r_f} + \frac{l_m^{\max}}{r_m}$
WF^2Q	$l^{\max}\left(\frac{1}{r_f} + \frac{1}{r_m} - \frac{1}{C}\right)$

To validate SFQ and WF^2Q+ simulations we have used the analytical results above. For all the results presented here, a link of 448000 bps bandwidth is shared by following two flows: Flow 1 is an Poisson type of traffic in which 320 byte packets are generated with a mean inter-arrival time of 5 millisecond, corresponding to an average bandwidth need of 512000 bps. Flow 2 is also a Poisson traffic, which generates 800 byte packets with the mean inter-arrival time of 10 milliseconds. The bandwidth requirement of Flow 2 is 640000 bps. Notice that, in order to respect the condition of non-zero backlogs, the total link requested by the sessions is almost the twice of actual link capacity.

The weights of flows are 2 and 5 respectively. The unfairness measure for WF^2Q+ according to the given parameters is 4479.986. The simulation had been run

for 10 times. Table 4.6 presents the measured differences between the normalized service shares. The simulation time is 200 seconds for all runs.

Table 4.6 Differences of normalized service shares in terms of bits for WF^{2}Q+ simulation.

Simulation number	$W_{1,j}(0,200)/2$	$W_{2,j}(0,200)/5$	$\lvert Difference \rvert$
1	12802560	12798720	3840
2	12800000	12800000	0
3	12800000	12800000	0
4	12800000	12800000	0
5	12798720	12798720	0
6	12797440	12798720	1280
7	12800000	12800000	0
8	12800000	12800000	0
9	12798720	12800000	1280
10	12798720	12800000	1280

As seen, for all runs the difference between the normalized amount of network share does not exceed the unfairness measure 4479.986. The lower and upper limits of the %90 confidence interval for the quantity $W_{1,j}(0,200)$ are 12798000 and 12801000 respectively. The same confidence limits for the quantity $W_{2,j}(0,200)$ are 12799000 and 12800000 respectively.

With the same flow parameters, the unfairness measure for SFQ is 2560. Table 4.7 presents the results of 10 runs for SFQ.

Table 4.7 Differences of normalized service shares in terms of bits for SFQ simulation.

Simulation number	$W_{1,j}(0,200)/2$	$W_{2,j}(0,200)/5$	$\lvert Difference \rvert$
1	12800000	12800000	0
2	12800000	12800000	0
3	12800000	12800000	0
4	12801280	12800000	1280
5	12801280	12800000	1280
6	12800000	12800000	0
7	12800000	12800000	0
8	12800000	12800000	0
9	12801280	12800000	1280
10	12800000	12800000	0

From the table it is seen that the difference between service shares had never exceeded the unfairness measure. The lower and upper limit of the 90% confidence interval for the measured quantity $W_{1,j}(0,200)$ is 12800000 and 12801000 respectively. Since standard deviation for $W_{2,j}(0,200)$ is 0, there is no need to compute confidence interval for it.

These results show that the behavior of the WF^2Q+ and SFQ implementations are correct.

4.3.3 Validation of First Come First Served and Deficit Round Robin Simulations:

To answer the question about the performances of DRR, Shreedhar and Varghese [17] experimentally confirms that DRR provides isolation superior to FCFS as the theory indicates for a single router case. They have also proved the fairness provided by DRR still good when the flows arrive at different rates and different distributions.

They have measured the throughput in terms of delivered bits in a simulation interval, typically 2000 seconds. In the single router case (Figure 4.3) there are one or more hosts. Each host has twenty flows, each of which generates packets at a Poisson average of 10 packets/second. The packet sizes are randomly selected between 0 and *Max* packet size (4500 bits). Ill-behaved flows send packets at a Poisson average of 30 packets/second. Each host is configured to have one ill-behaved flow.

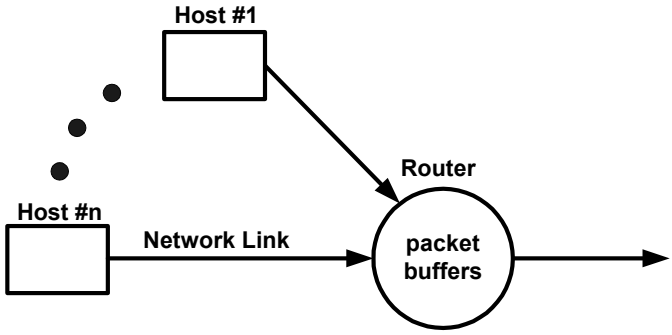


Figure 4.3 Single router configuration

In order to show how DRR performs with respect to FCFS, a single router configuration and a host with twenty flows sending packets at the default rate

through the router is used. The only exception is that Flow 10 is a misbehaving flow. The outgoing link was set to 10 Kbps. Therefore if there are 20 input flows each sending at rates higher than 0.5 Kbps, there is contention for the outgoing link bandwidth.

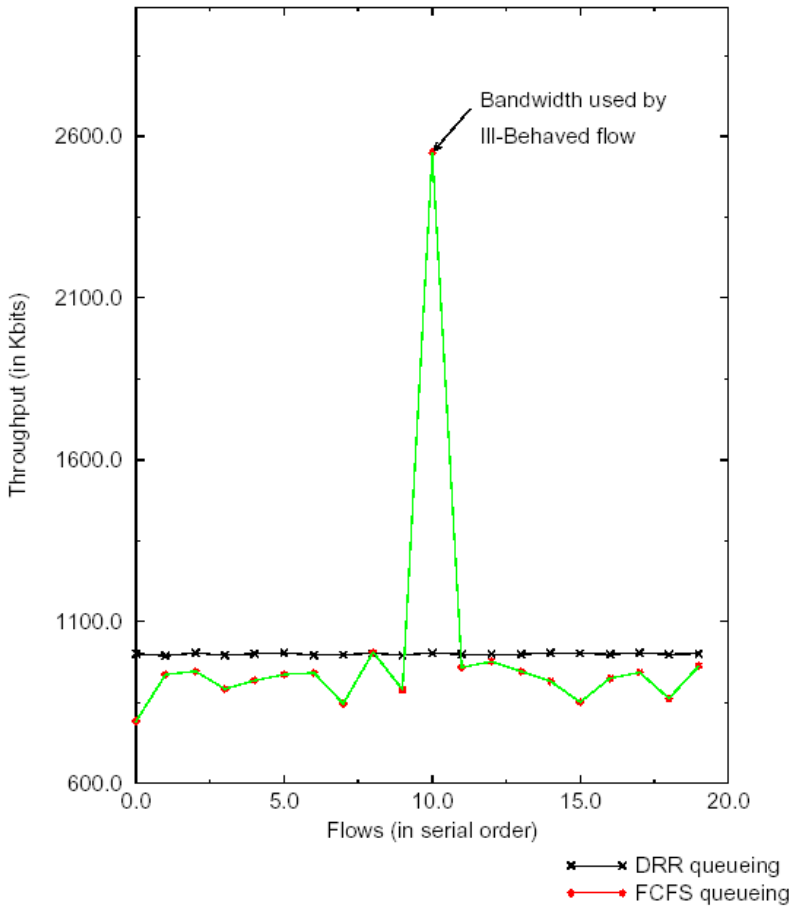


Figure 4.4 Plot showing the results of the experiment done by Shreedhar and Varghese [17], which interprets the bandwidth distribution among the flows.

Figure 4.4 shows the plot of the bandwidth offered to flows using FCFS queueing and DRR. The plot is directly taken from the paper written by Shreedhar and Varghese [17]. In FCFS the ill-behaved flow (Flow 10) obtains an arbitrary share of the bandwidth. The plot clearly illustrates the isolation property of DRR.

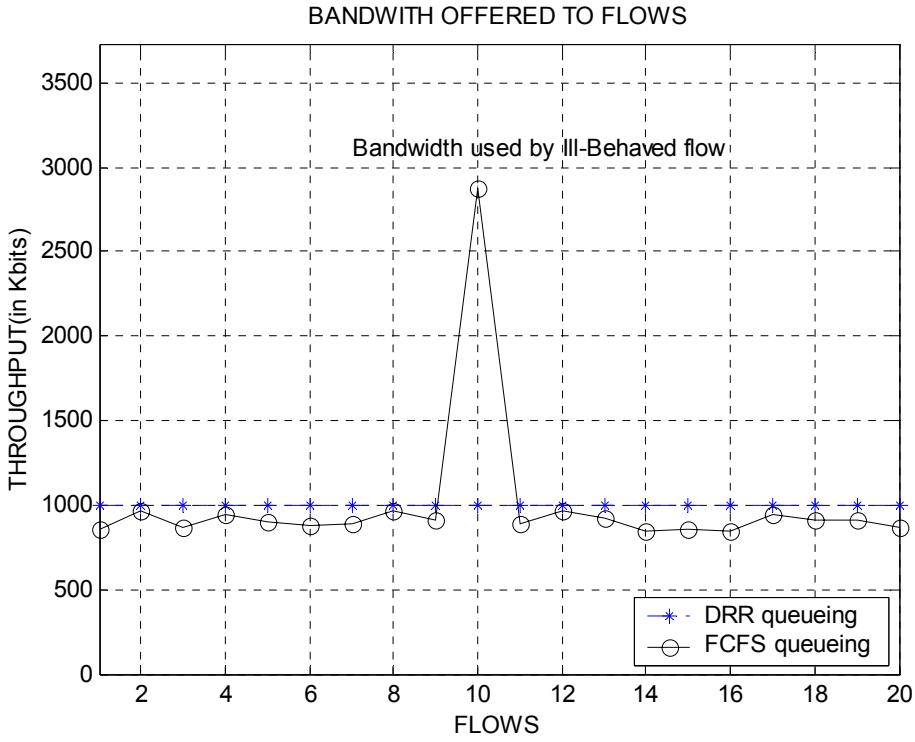


Figure 4.5 The results of DRR implementation.

We have simulated DRR and FCFS with the same operating parameters to validate the behavior of the implementations. The result can be seen in Figure 4.5. As in the experiment performed by the authors, the ill-behaved flow, namely Flow 10,

grabbed an arbitrary share of the bandwidth, while in DRR there is nearly perfect fairness. In their experiment Shreedhar and Varghese measured the maximum deviation from the ideal bandwidth share (in terms of Kilobits) turned out to be 0.32%. This value is measured as 0.3002% in our case.

Table 4.8 lists the upper and lower limits of the 90% confidence intervals for the throughput distributed among the 20 flows by FCFS and DRR implementations. The values are in kilobits.

Table 4.8 The upper and lower limits of the 90% confidence intervals of throughputs distributed among the 20 flows.

Flow number	FCFS implementation		DRR implementation	
	Upper limit	Lower Limit	Upper limit	Lower Limit
1	824.2	964	998.7	1002.4
2	824.2	942	999.2	1002.2
3	845	978.6	997.4	1001.9
4	863.8	950	998	1001.8
5	793.5	977.7	998	1002.2
6	890.4	952.5	997.1	1002.5
7	853	950.6	998.1	1003
8	869.4	996.8	998.2	1002
9	865.7	988.3	998.2	1002.9
10	2651.6	2837.6	998.6	1001.8
11	839.6	948.2	997.8	1001.7
12	860.8	961.5	997.7	1001.4
13	844.2	989.6	997	1003.3
14	831.7	981.3	997.6	1001.8

Flow number	FCFS implementation		DRR implementation	
	Upper limit	Lower Limit	Upper limit	Lower Limit
15	894.8	951.3	998.6	1002
16	863.8	968.2	997.6	1002.1
17	846.8	956.5	997.6	1002.2
18	831	966.1	997.9	1002.8
19	861.4	946.9	998.5	1002.1
20	856.5	981.8	998.1	1001.3

These results show that the behaviors of the FCFS and DRR implementations are correct.

4.3.4 Validation of Wireless Packet Scheduling Simulation:

For the simple scenario demonstrating the effectiveness of WPS, Lu, Bharghavan and Srikant [28] have considered an example with three loss-sensitive sources with WFQ weights $r_1 = 20$, $r_2 = 10$ and $r_3 = 1$.

It is assumed that the channels for sources evolve errors according to a two-state discrete Markov Chain. Knowing that p_g is the probability that the next time slot is good, given that the current slot is in error, and p_e is the probability that the next time slot is in error, given that the current slot is good, then the steady-state probabilities P_G and P_E of being in the good and bad states, respectively, are given by:

$$P_G = \frac{p_g}{p_g + p_e} \tag{4.9}$$

$$P_E = \frac{p_e}{p_g + p_e} \tag{4.10}$$

The two-state discrete Markov Chain used to model one-step prediction algorithm is shown in Figure 4.6. Let the bad state be 0 and good state be 1.

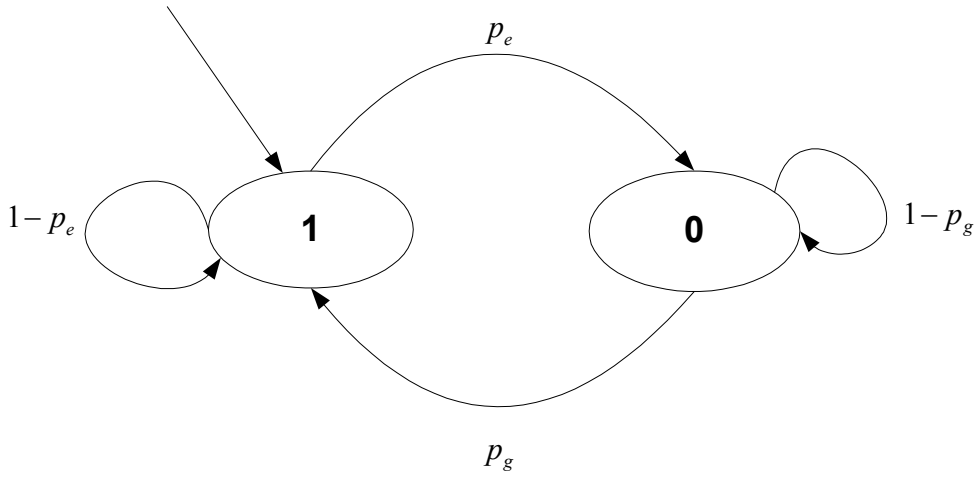


Figure 4.6 Two-state discrete Markov Chain used to model errors evolved by channels.

The arrival processes are assumed to be as follows: Source 1 is a Markov-modulated Poisson process (MMPP) where the modulated process is a continuous-time Markov chain which is in one of two states *ON* and *OFF* [28, 33]. The

transition rate from *ON* to *OFF* is 9 and *OFF* to *ON* is 1. When the Markov chain is *ON* state, arrivals occur according to a Poisson process. Source 2's arrivals occur according to a constant inter-arrival time and Source'3 arrivals are Poisson. The source parameters are given in Table 4.3. For each packet, the maximum number of retransmissions is limited to 2, i.e. a packet is dropped if it is not successfully transmitted after three attempts. Also the number of credits and number of debits are limited to 20. The values of p_g and p_e parameters can be seen in Table 4.9.

Table 4.9 Source and channel parameters for WPS simulation.

Source	λ_i	p_g	p_e	r_i
1	4.4	0.08	0.02	20
2	0.25	0.095	0.005	10
3	0.025	0.09	0.01	1

The simulation had been run for 100 000 time units and the results of the simulation experiment done by Lu, Bharghavan and Srikant [28] are presented in Table 4.10. The performance measures compared for the WPS algorithm for Source *i*, i = 1,2,3 are:

W_i the aggregate number of packets that have been successfully transmitted for flow *i*;

l_i loss Probability, i.e., fraction of packets that are dropped after three transmission attempts;

Table 4.10 Results of WPS simulations.

	W_1	l_1	W_2	l_2	W_3	l_3
Original Results	50000	0	25000	0	2496	0

The results of the WPS implementation for 10 runs are given in Table 4.11. The 90% confidence intervals for the parameters W_1, l_1, W_2, l_2, W_3 and l_3 are also given in Table 4.12.

Table 4.11 Simulation results for WPS implementation.

Simulation number	W_1	l_1	W_2	l_2	W_3	l_3
1	49960	0	24995	0	2551	0
2	49904	0	24999	0	2439	0
3	49926	0	24984	0	2428	0
4	49976	0	24997	0	2444	0
5	49990	0	24995	0	2449	0
6	50152	0	24996	0	2517	0
7	49914	0	24993	0	2551	0
8	49954	1	24997	0	2546	0
9	49917	0	24990	0	2509	0
10	49950	0	24996	0	2488	0

Table 4.12 Confidence intervals for WPS implementation.

Confidence limits	W_1	l_1	W_2	l_2	W_3	l_3
Lower	49165	-0.3373	24988	0	2424	0
Upper	49363	0.5373	250000	0	2560	0

4.3.5 Validation of Channel Condition Independent Packet Fair Queuing Simulation:

To demonstrate the fairness properties of CIF-Q, Eugene Ng, Stoica and Zhang [25], considered an example with seven sessions (a real-time audio session, a real-time video session, four FTP sessions, and a cross traffic session) with the parameters shown in Table 4.13. The audio and video sessions are CBR sources such that their packets are evenly spaced at 50 milliseconds apart and their throughputs are 160 Kbps 1.25 Mbps respectively. The four 2 Mbps FTP sessions are all continuously backlogged. The cross traffic session is a Poisson source with an average rate of 10 Mbps.

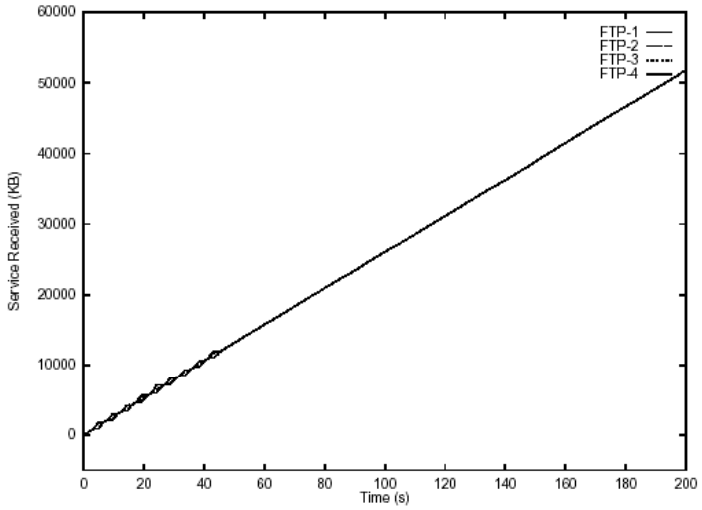
Table 4.13 Properties of the 7 sessions used in the CIF-Q simulations.

	Packet Size	Guaranteed Rate	Source Model	Error
Audio	1KB	160 Kbps	CBR	None
Video	8KB	1.25 Mbps	CBR	None
FTP-1	3KB	2 Mbps	Greedy	None
FTP-2	3KB	2 Mbps	Greedy	Pattern 1

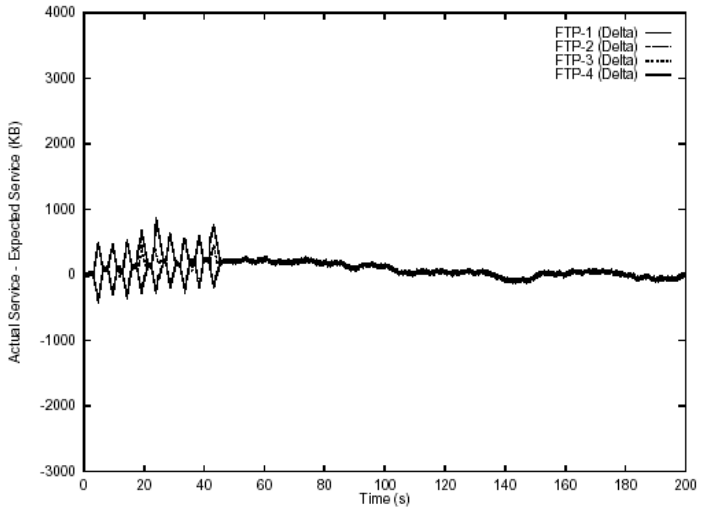
	Packet Size	Guaranteed Rate	Source Model	Error
Audio	1KB	160 Kbps	CBR	None
FTP-3	8KB	2 Mbps	Greedy	Pattern 2
FTP-4	8KB	2 Mbps	Greedy	Pattern 1
Cross	4KB	10 Mbps	Poisson	None

For showing the effects of channel errors and for ease of interpretation, the errors are modeled as simple periodic bursts. During the 200 second periods of simulation experiments, the channel errors occur during the first 45 seconds, leaving enough error-free time to demonstrate the long term fairness property of the algorithm. Error pattern 1 represents a periodic error burst of 1.6 second with 3.2 seconds of intermediate error-free time. Error pattern 2, a less severe error pattern, represents a periodic error burst of 0.5 seconds with 5.5 seconds of intermediate error-free time. Be aware that FTP-2 and FTP-4 have experience identical error pattern but have different packet sizes, while FTP-1 experiences no error at all. The results of simulation presented by the authors, using $\alpha=0$ can be seen in Figure 4.7.

The results of the CIF-Q implementation can be seen in 4.8. As seen the service received by all four FTP sessions, regardless of the amount of errors they have experienced, converges very rapidly when the system becomes error free. Figure 4.8-b demonstrates the changes and lags more easily. The long-term and short-term fairness guarantees provided by the algorithm holds.

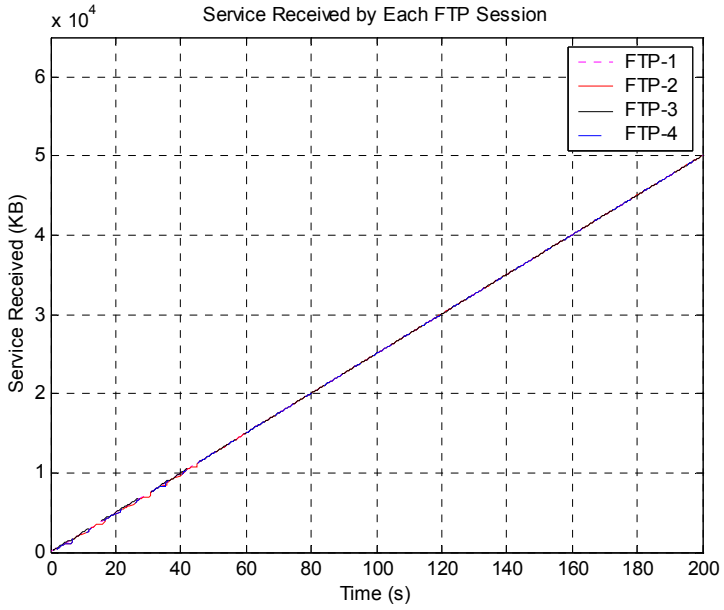


(a)

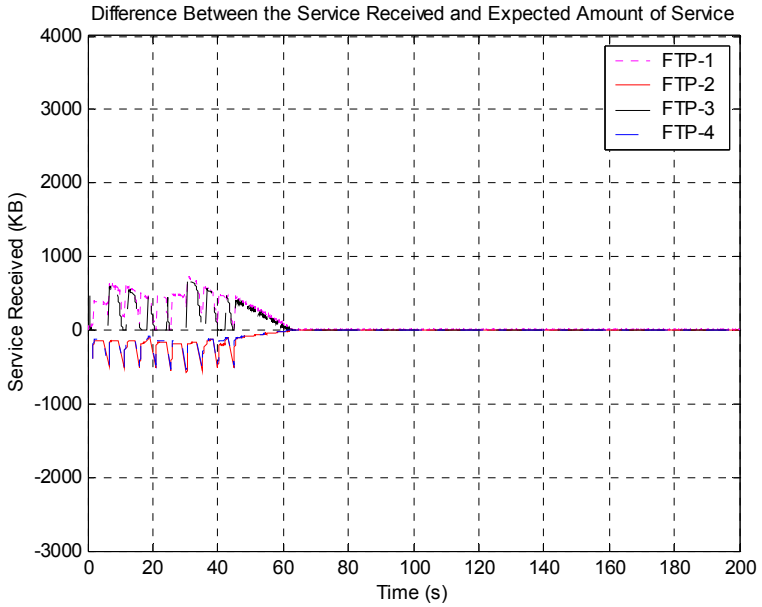


(b)

Figure 4.7 Behavior of the FTP sessions when α=0. (a) Service received by each FTP sessions. (b) Difference between the actual service received by the FTP sessions and the corresponding amount of service.



(a)



(b)

Figure 4.8 Results of CIF-Q implementation. (a) Service received by each FTP sessions. (b) Difference between the actual service received by the FTP sessions and the corresponding amount of service

The upper and lower limits of the 90% confidence interval for the total bandwidth distributed to FTP sessions can be seen in Table 4.14. The values are in kilobytes.

Table 4.14 The upper and lower limits of the 90% confidence intervals of throughputs distributed among the FTP sessions.

Session	Upper limit	Lower limit
FTP1	50000	50000
FTP2	49990	50000
FTP3	49990	49990
FTP4	49990	50000

These results show that the behavior of the CIF-Q implementation is correct.

CHAPTER 5

SIMULATION RESULTS

In this chapter, a set of simulation studies for comparing the performance of several flow-based scheduling algorithms, namely FCFS, WFQ, WF^2Q+, SFQ, DRR for wired environment and WPS and CIF-Q for wireless networks will be presented. Adaptation of DRR for wireless media will also be evaluated. Delay, jitter and throughput performances of the above algorithms are evaluated.

5.1 Simulation Study:

In the first simulation scenario 4 flows (1, 2, 3 and 4) as well as a server with a capacity of 280 Kbps were considered. The server assigns 70 Kbps to each flow, since the weights of all flows are equal to 1.

The traffic models used for each flow are given in Table 5.1.

Table 5.1 Traffic models for each flow for the first simulation scenario

Flow	Type	Weight	Packet Length / Mean Packet Length (Bytes)	Period / Mean Interarrival Time (milliseconds)
1	CBR	1	450	50
2	Poisson	1	450	50
3	CBR	1	900	50

Flow	Type	Weight	Packet Length / Mean Packet Length (Bytes)	Period / Mean Interarrival Time (milliseconds)
4	VBR	1	500	25

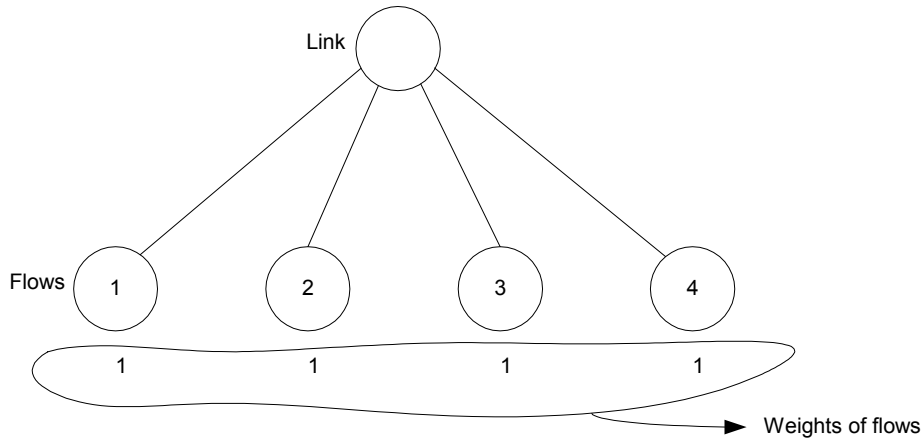


Figure 5.1 The first simulation scenario

Flow 1 and flow 2 corresponds to an average bandwidth need of 70.3125 Kbps. Flow 3 is also a CBR type of traffic and need an average bandwidth of 140.625 Kbps. Flow 4, which is a VBR type of traffic, corresponds to an average bandwidth need of 156.25 Kbps. It is seen that total bandwidth required for four flows corresponds to an average bandwidth need of 437.5 Kbps but the link provided is only 280 Kbps. The results were examined for 200 seconds. It is thought that 200 seconds will be enough for the simulations to be stable. The simulations have run 10 times for the same traffic set and the results are averaged.

The fairness measure for a scheduling algorithm can be defined as:

$$\sum_{\forall i} \left| \overline{R_i} - R_i \right| \tag{5.1}$$

where $\overline{R_i}$ is the measured rate (Kbps, Mbps etc.) that is provided by the algorithm to the corresponding flow and R_i is the assigned rate, which is proportional to the weight of the flow.

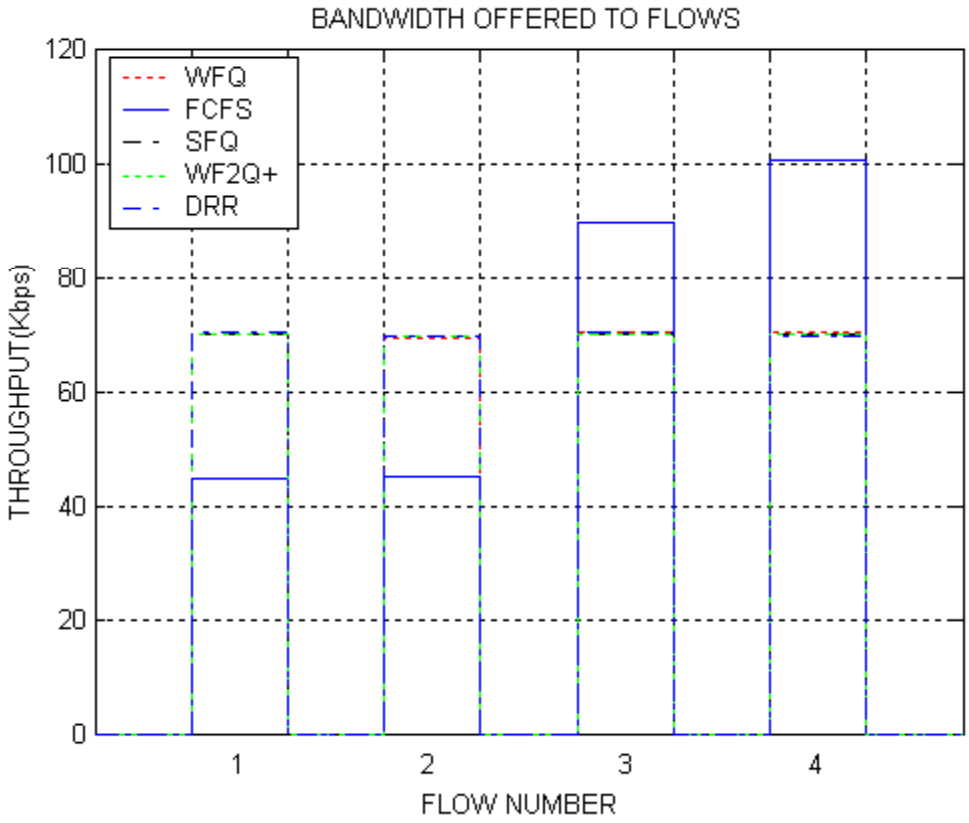


Figure 5.2 Bandwidth distributions among the four flows for the first simulation scenario – wireline scheduling algorithms.

Figure 5.2 presents the bandwidth distribution among four flows for FCFS, WFQ, W^2FQ+, SFQ and DRR. As seen from the figure the total throughput 280 Kbps is shared fairly among the flows except for the FCFS. At the average FCFS algorithm allocates a bandwidth of 44.475 Kbps to Flow 1, 45.775 Kbps to Flow 2, 89.557 Kbps to Flow 3 and 100.48 Kbps to Flow 4. As expected these results show that FCFS does not allocate the total bandwidth fairly among the flows.

It can be said that the other servers using WFQ, W^2FQ+, SFQ or DRR type of scheduling mechanisms allocate the bandwidth fairly for the given traffic set by looking at the fairness measures given in Table 5.2.

Table 5.2 Fairness measures of wired scheduling algorithms for the first simulation scenario.

Algorithm	Fairness Measure
FCFS	100.12
DRR	1.29
WFQ	1.3
W^2FQ+	0.42
SFQ	0.4

As seen the best performing scheduling algorithms for the first traffic set are W^2FQ+ and STFQ. The fairness provided by DRR and WFQ are nearly the same.

The upper and lower limits of the 90% confidence interval for the rates of flows are given in Table 5.3.

Table 5.3 90% confidence intervals of the rates provided by the scheduling mechanisms for the first scenario.

	Flow1	Flow2	Flow3	Flow4
FCFS	$44.2 \leq \leq 45.3$	$44.2 \leq \leq 46.1$	$88.4 \leq \leq 90.7$	$99 \leq \leq 102$
DRR	$70.1 \leq \leq 70.2$	$69.1 \leq \leq 70.4$	$70.1 \leq \leq 70.7$	$69.3 \leq \leq 70$
WFQ	$69.9 \leq \leq 70.2$	$68.7 \leq \leq 69.9$	$69.9 \leq \leq 70.6$	$69.9 \leq \leq 71$
W^2FQ+	$69.8 \leq \leq 70.2$	$68.7 \leq \leq 70.8$	$69.6 \leq \leq 70.6$	$69.6 \leq \leq 70$
SFQ	$69.9 \leq \leq 70.2$	$69.3 \leq \leq 70.3$	$69.9 \leq \leq 70.2$	$69.9 \leq \leq 70$

The mean and maximum delay performances of wired scheduling algorithms for the first scenario are presented in Figure 5.3 and 5.4 respectively. It is seen that, since FCFS does not deal with the weights of any flows, the delays experienced by the flows are equal. This is because FCFS does not compensate the discrepancies between the bandwidths that a flow supposed to have and provided by the server. The average number of packets transmitted for the algorithms is given in Table 5.4.

It is seen that, since DRR, WFQ, W^2FQ+ and SFQ tries to prevent unfairness, flow 3 and 4 experiences more delay than the other two flows. Remember that Flow 3 and Flow 4 require more bandwidth than provided by the server.

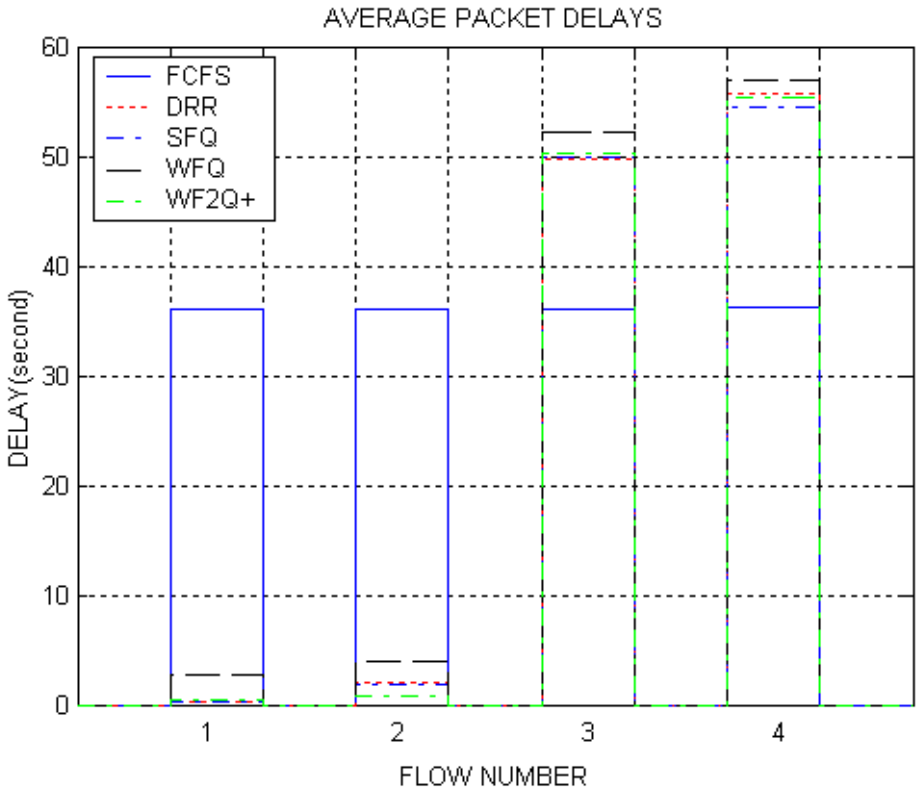


Figure 5.3 Mean delays experienced by the flows for the first simulation scenario – wireline scheduling algorithms.

Table 5.4 Average number of packets, transmitted by the scheduling algorithms for the first scenario.

	Flow1	Flow2	Flow3	Flow4
FCFS	2552.4	2555.7	2552.4	5107.9
DRR	3993.3	3967.4	2003.8	3534
WFQ	3987.1	3942.6	1998.7	3600.6
W^2FQ+	3982.7	3969.8	1993.9	3551.6
SFQ	3984.7	3970.6	1993.1	3616.5

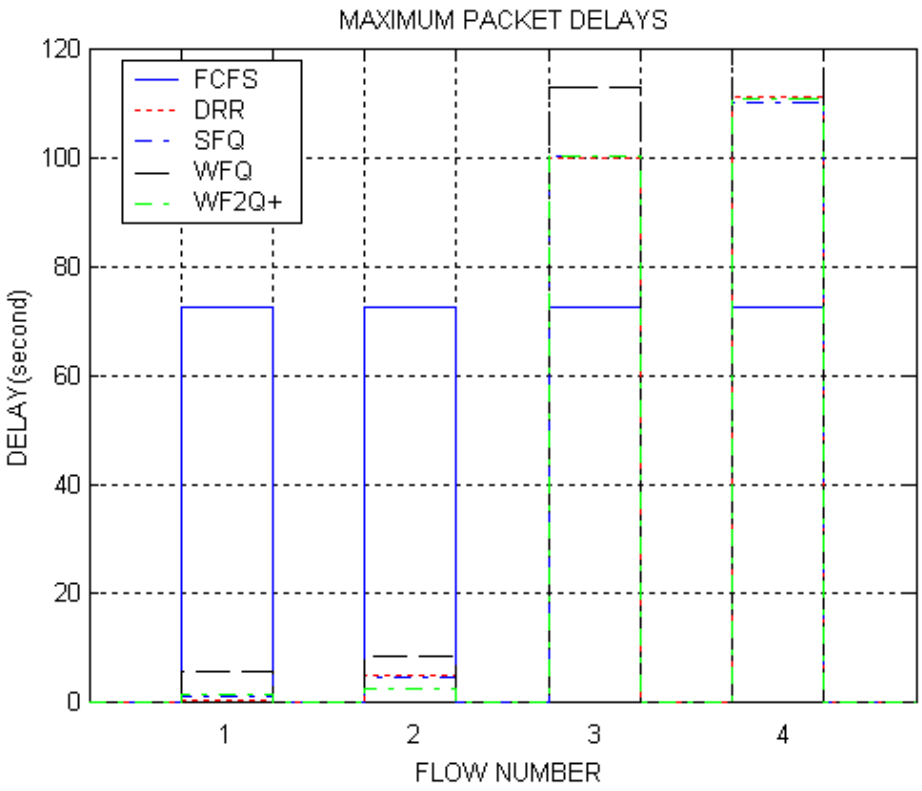


Figure 5.4 Maximum delays experienced by the flows for the first simulation scenario – wireline scheduling algorithms.

The upper and lower limits of the 90% confidence interval for the mean and maximum delays experienced by the flows are given in Table 5.5 and Table 5.6 respectively. The values are in seconds.

Table 5.5 90% confidence intervals of the mean delays experienced by the flows for the first scenario.

	Flow1	Flow2	Flow3	Flow4
FCFS	35.1 ≤ ≤ 36.9	35 ≤ ≤ 36.9	35.1 ≤ ≤ 36.9	35.4 ≤ ≤ 36.9

	Flow1	Flow2	Flow3	Flow4
DRR	0.12 ≤ ≤ 0.25	1.23 ≤ ≤ 2.89	49.3 ≤ ≤ 50.1	53.8 ≤ ≤ 57.7
WFQ	2.53 ≤ ≤ 2.8	3.35 ≤ ≤ 4.69	51.9 ≤ ≤ 52.5	534 ≤ ≤ 59.9
W^2FQ+	0.09 ≤ ≤ 0.9	-0.4 ≤ ≤ 2.07	49.8 ≤ ≤ 50.7	53. ≤ ≤ 57.6
SFQ	0.05 ≤ ≤ 0.6	-0.5 ≤ ≤ 4.1	49.4 ≤ ≤ 50.5	52.1 ≤ ≤ 56.8

Table 5.6 90% confidence intervals of the maximum delays experienced by the flows for the first scenario.

	Flow1	Flow2	Flow3	Flow4
FCFS	71.6 ≤ ≤ 73.1	71.5 ≤ ≤ 73.1	71.6 ≤ ≤ 73.1	71.6 ≤ ≤ 73.1
DRR	0.27 ≤ ≤ 0.4	3.7 ≤ ≤ 5.8	99.4 ≤ ≤ 100.1	110 ≤ ≤ 112.3
WFQ	5.19 ≤ ≤ 5.6	7.1 ≤ ≤ 9.6	95.8 ≤ ≤ 130	107.8 ≤ ≤ 132
W^2FQ+	0.78 ≤ ≤ 1.5	0.39 ≤ ≤ 4	99.8 ≤ ≤ 100.8	109 ≤ ≤ 112.8
SFQ	0.55 ≤ ≤ 1.08	1.14 ≤ ≤ 7.3	100.1 ≤ ≤ 100.5	108 ≤ ≤ 111.7

The jitter performances of the flows for the wired scheduling algorithms are presented in Figure 5.5. The jitters experienced by the flows scheduled by FCFS type of mechanism are the same since FCFS does not venture the weights of flows. But as expected the flows which uses equal to or less then their bandwidth gets less delay and jitter when a fair scheduling algorithm schedules them.

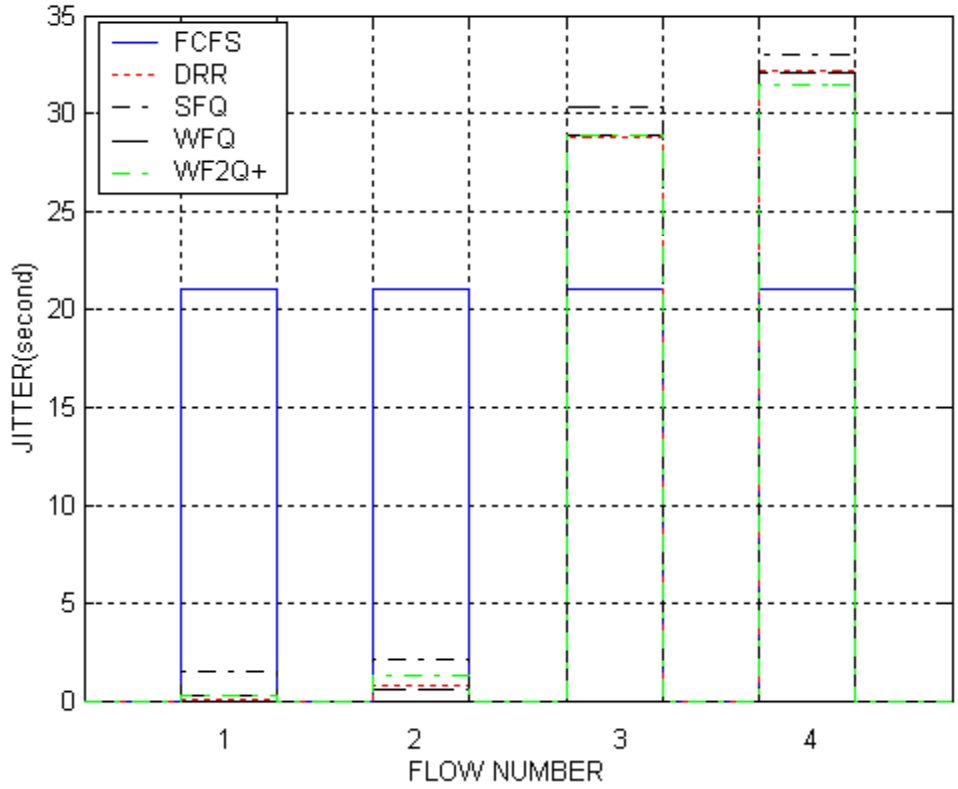


Figure 5.5 Jitter performances of wireline scheduling algorithms for the first simulation scenario.

Table 5.7 90% confidence intervals of the jitters experienced by the flows for the first simulation scenario.

	Flow1	Flow2	Flow3	Flow4
FCFS	$20.6 \le \le 21.5$	$20.6 \le \le 21.5$	$20.6 \le \le 21.4$	$20.5 \le \le 21.4$
DRR	$0.03 \le \le 0.09$	$0.32 \le \le 1.17$	$28.6 \le \le 29$	$31.1 \le \le 33.2$
WFQ	$1.41 \le \le 1.65$	$1.44 \le \le 2.78$	$30.3 \le \le 30.4$	$31.6 \le \le 34.4$
W^2FQ+	$0.19 \le \le 0.4$	$-0.16 \le \le 1.3$	$28.77 \le \le 29.1$	$31.6 \le \le 32.58$
SFQ	$0.07 \le \le 0.36$	$-0.18 \le \le 2.65$	$28.7 \le \le 29.2$	$30.6 \le \le 32.2$

The first simulation scenario, where its traffic characteristics are introduced at Table 5.1 is simulated for the wireless packet scheduling schemes. Remember that for clarity in showing the effects of channel errors and ease of implementation, the errors are modeled as simple periodic bursts. During the simulation experiments, channel errors occur during the first 100 seconds to demonstrate long time fairness. It is also possible to make the channel errors occur only during the whole simulation to demonstrate short time fairness.

During the WPS simulations, the maximum number of credits and debits are both chosen as 20, and the maximum retransmissions are taken to be 2. That is a packet is dropped if it is not successfully transmitted after three attempts.

For CIF-Q, the α as chosen to be 0, so that a leading session *i* will no service as long as there exists a lagging error-free session in the system. This will ensure the short time fairness.

The channels of Flow 1 and Flow 2 experience the same error pattern and require approximately same bandwidths. The error pattern experienced by Flow 1 and Flow 2 is a periodic error burst of 1.8 second with 3 seconds of intermediate error free time. The error pattern experienced by Flow 3 is a less severe error-pattern, represents a periodic burst of 0.3 seconds with 5.7 seconds of intermediate error-free time. Flow 4 experiences no error at all. The following figure presents the bandwidth distribution among four flows for WPS, CIF-Q and DRR for wireless channels (DRRWC). Remember that the simulations had been run for 10 times and the results are averaged. To ensure if the simulated schemes provide long time fairness simulation time is selected as 1200 seconds.

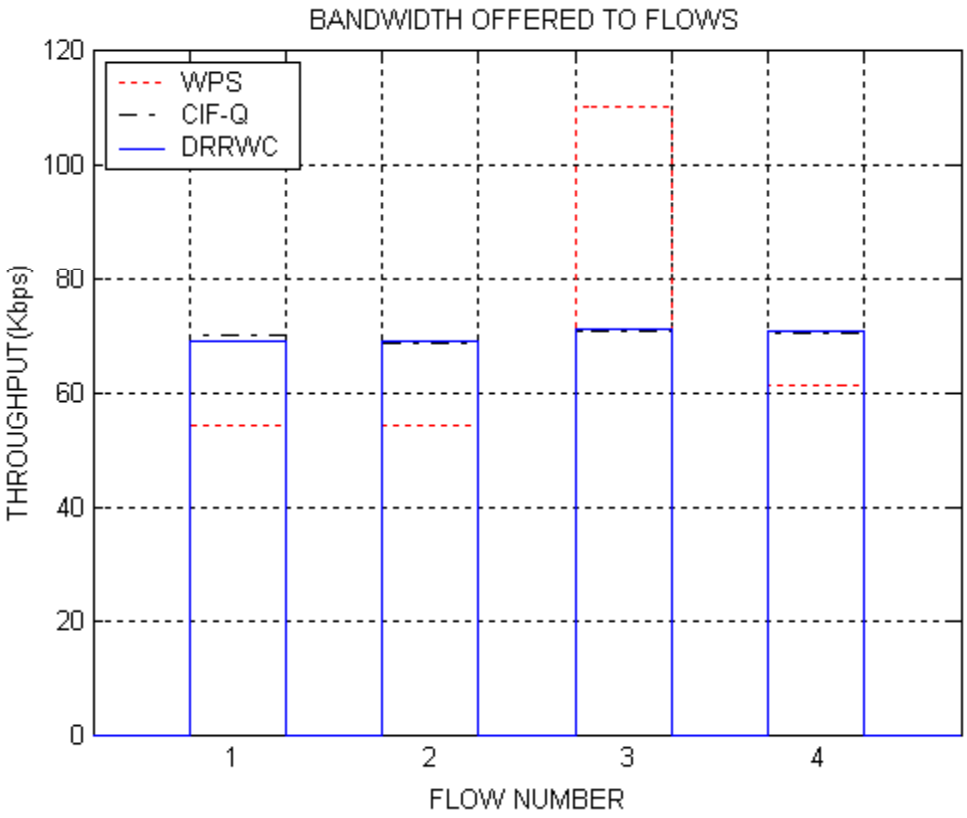


Figure 5.6 Bandwidth distributed by the wireless scheduling algorithms among the four flows for the first simulation scenario.

As seen from Figure 5.6 the total throughput 280 Kbps was shared fairly among the flows except WPS. For the given traffic set CIF-Q is the best fair scheduler among the others. The fairness performance of DRRWC is also good as compared to WPS. Table 5.8 lists the average bandwidth distributed by WPS, CIF-Q and DRRWC. The results are in Kbps. The fairness measures are also presented in Table 5.9.

Table 5.8 Average bandwidth, distributed by the wireless scheduling algorithms for the first simulation scenario.

	Flow1	Flow2	Flow3	Flow4
WPS	54.06	54.07	110.11	61.32
CIF-Q	70.03	68.7	70.74	70.51
DRRWC	69.1	69.1	71.17	70.62

Table 5.9 Fairness measures of wireless scheduling algorithms for the first simulation scenario.

Algorithm	Fairness Measure
WPS	80.65
CIF-Q	2.58
DRRWC	3.61

According to the results given in Table 5.8 for the first simulation scenario WPS was not successful as CIF-Q or DRRWC. Both CIF-Q and DRRWC did not allow Flow 3 to get a higher share, but WPS did. Remember that channel for Flow 3 evolves a less severe error pattern and requires a bandwidth more then provided. For WPS, only Flow 4 got a relatively close to the ideal share of network among the other flows.

The upper and lower limits of the 90% confidence interval for the rates provided to flows are given in Table 5.8.

Table 5.10 90% confidence intervals of the rates provided by the wireless scheduling mechanisms for the first scenario.

	Flow1	Flow2	Flow3	Flow4
WPS	53.96 ≤ ≤ 54.15	53.98 ≤ ≤ 54.17	109.9 ≤ ≤ 110.3	60.9 ≤ ≤ 61.7
CIF-Q	70.01 ≤ ≤ 70.03	68.5 ≤ ≤ 68.8	70.6 ≤ ≤ 70.87	70.48 ≤ ≤ 70.53
DRRWC	68.08 ≤ ≤ 69.1	69.08 ≤ ≤ 69.09	71.15 ≤ ≤ 71.18	70.59 ≤ ≤ 70.65

The mean delay performances of the simulated wireless scheduling algorithms for the first scenario are presented in Figure 5.7.

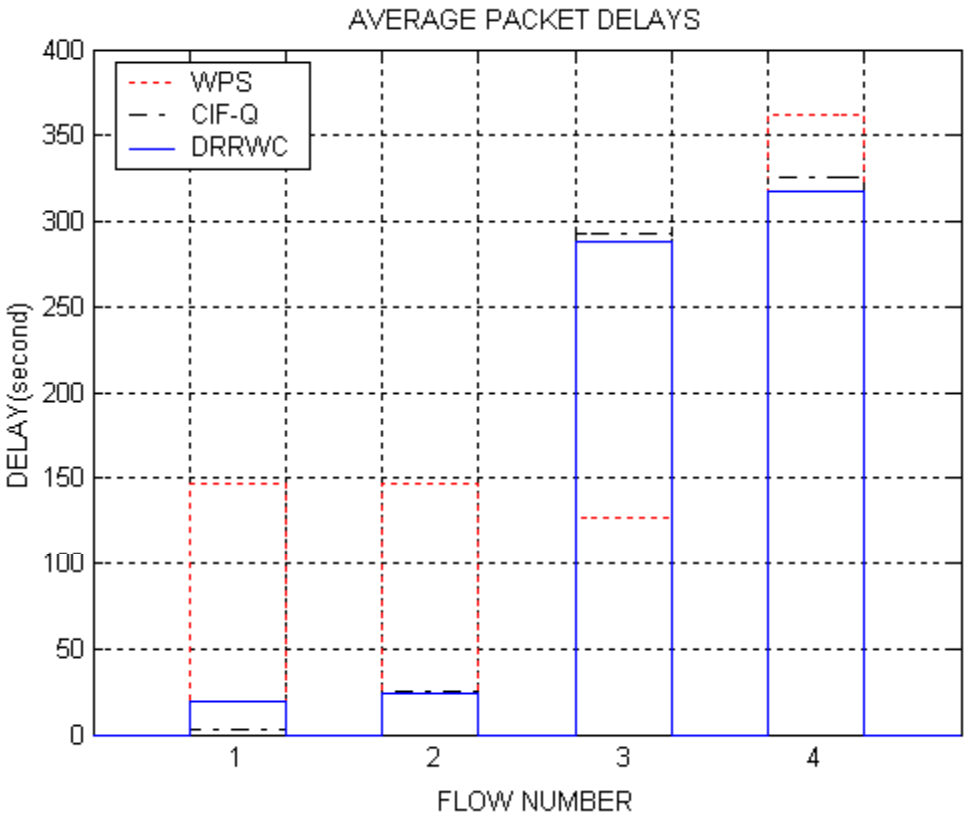


Figure 5.7 Mean delays experienced by the flows for the first simulation scenario – wireless scheduling algorithms.

Investigating the delay performance of the wireless scheduling algorithms, it is seen that for CIF-Q, while exiting from error mode, always Flow1 take precedence to give its lag. This is the effect of α (Remember that we have chosen α=0). So the mean delay experienced by Flow 1 is less than it receives when the packets are scheduled by DRRWC.

The upper and lower limits of the 90% confidence interval for the mean delays experienced by the flows are given in Table 5.9. The values are in seconds.

Table 5.11 90% confidence intervals of the mean delays experienced by the flows for the first scenario – wireless scheduling algorithms.

	Flow1	Flow2	Flow3	Flow4
WPS	$145.6 \leq \leq 147.2$	$141.7 \leq \leq 151.2$	$126.6 \leq \leq 128.2$	$359 \leq \leq 363.5$
CIF-Q	$2.24 \leq \leq 2.49$	$18.1 \leq \leq 31.7$	$290.98 \leq \leq 294$	$320.3 \leq \leq 330.$
DRRWC	$19.6 \leq \leq 19.9$	$20 \leq \leq 27.9$	$288.5 \leq \leq 288.8$	$314.1 \leq \leq 322$

The average number of packets transmitted by the flows is listed in Table 5.10.

Table 5.12 Average number of packets, transmitted by the wireless scheduling algorithms for the first scenario.

	Flow1	Flow2	Flow3	Flow4
WPS	18453	18457	18792	18814
CIF-Q	23902	23448	12073	21590
DRRWC	23584	23581	12145	21700

Figure 5.8 presents the maximum delays experienced by the flows for the first simulation scenario.

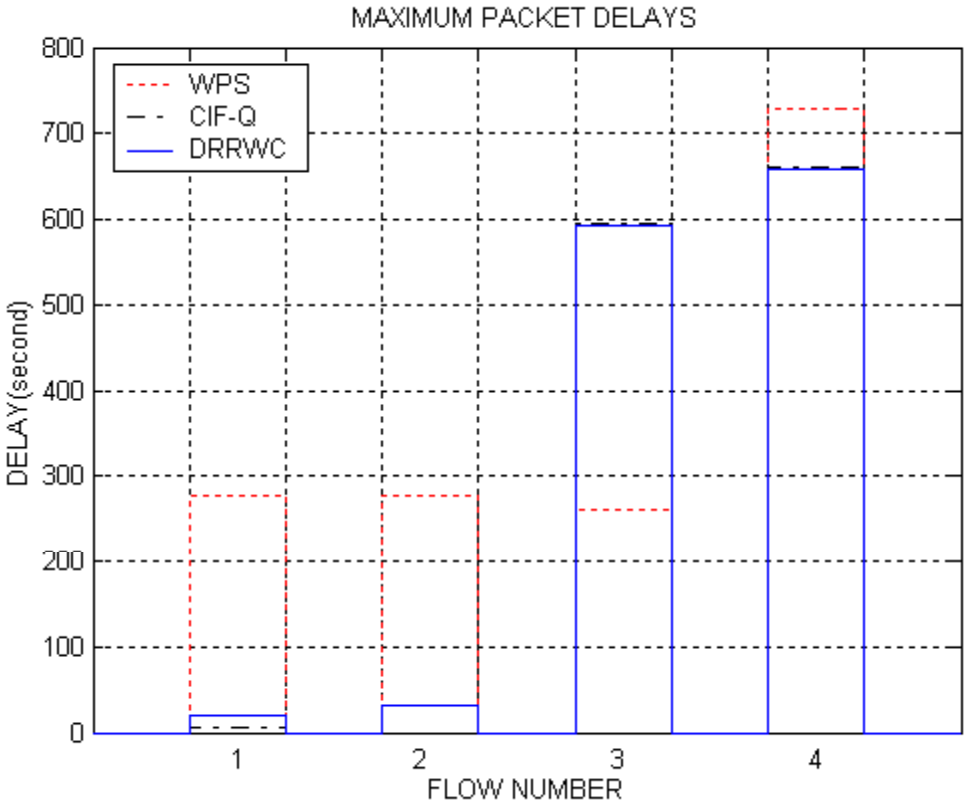


Figure 5.8 Maximum delays experienced by the flows for the first simulation scenario – wireless scheduling algorithms.

The 90% confidence limits for the maximum delays experienced by the flows are given in Table 5.13. The results are in seconds.

The jitter performances of wireless scheduling algorithms can be seen in Figure 5.9.

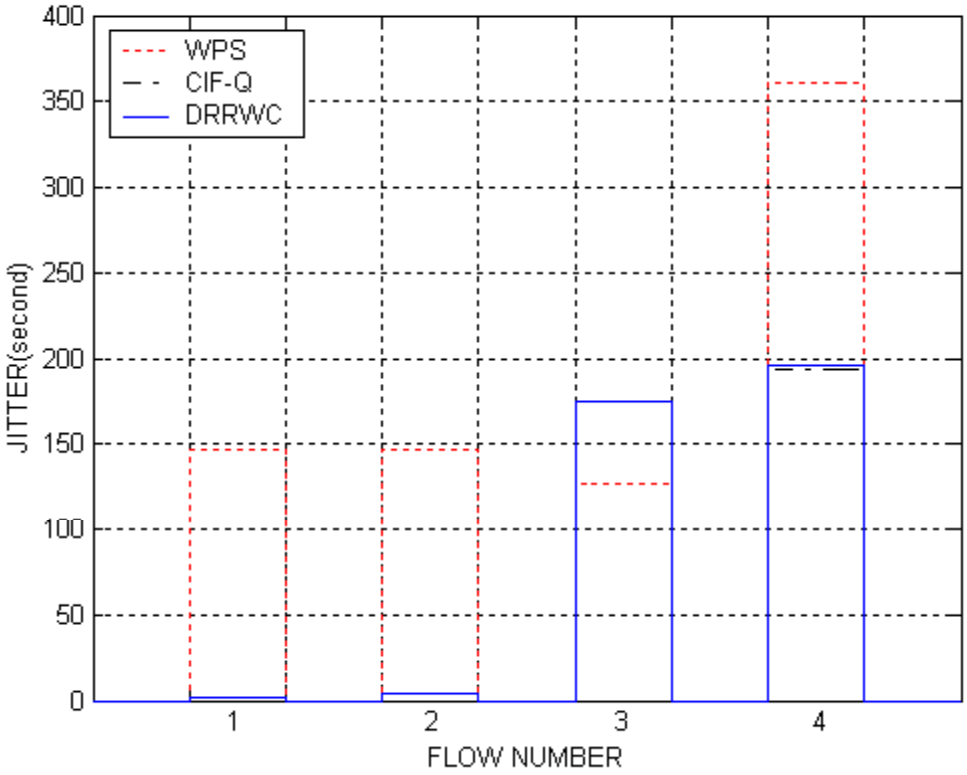


Figure 5.9 Jitter performances of wireless scheduling algorithms for the first simulation scenario.

The upper and lower limits of the 90% confidence limits for the jitters experienced by the flows are given in Table 5.14. The results are in seconds.

Jitter performances of CIF-Q and DRRWC are close, but the jitters experienced by the flows are more in WPS.

Table 5.13 90% confidence intervals of the maximum delays experienced by the flows for the first scenario – wireless scheduling algorithms.

	Flow1	Flow2	Flow3	Flow4

	Flow1	Flow2	Flow3	Flow4
WPS	276.1 ≤ ≤ 278.5	273.3 ≤ ≤ 282.9	259.1 ≤ ≤ 261.6	727.5 ≤ ≤ 732
CIF-Q	5.44 ≤ ≤ 5.66	24.4 ≤ ≤ 40.8	595.5 ≤ ≤ 597.2	655 ≤ ≤ 665.3
DRRWC	20.7 ≤ ≤ 20.9	26.5 ≤ ≤ 34.5	592.6 ≤ ≤ 592.7	651.7 ≤ ≤ 664

Table 5.14 90% confidence intervals of the jitters experienced by the flows for the first simulation scenario – wireless scheduling algorithms.

	Flow1	Flow2	Flow3	Flow4
WPS	145.5 ≤ ≤ 147	141.5 ≤ ≤ 151	126.5 ≤ ≤ 128.1	359 ≤ ≤ 363.1
CIF-Q	1.43 ≤ ≤ 1.44	1.93 ≤ ≤ 7	174.7 ≤ ≤ 174.8	191.5 ≤ ≤ 194
DRRWC	1.87 ≤ ≤ 1.90	2.38 ≤ ≤ 6.17	175 ≤ ≤ 175.1	191.5 ≤ ≤ 199

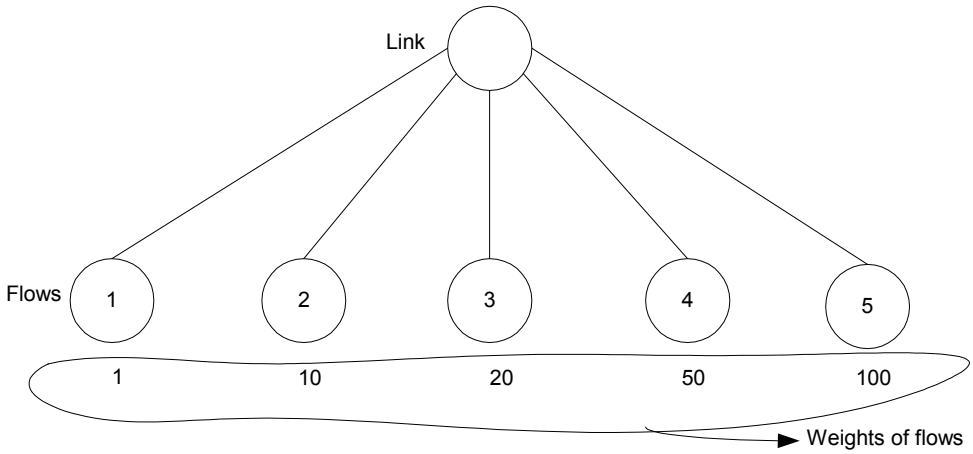


Figure 5.10 The second simulation scenario

For the second simulation scenario 5 flows (1, 2, 3, 4 and 5) were considered, as well as a server with a capacity of 9050 Kbps. In this experiment the fairness performances of the scheduling algorithms with significant differences in weights will be evaluated. The traffic models used for each flow are given in Table 5.15.

Table 5.15 Traffic models for each flow for the second simulation scenario

Flow	Type	Weight	Packet Length / Mean Packet Length (Bytes)	Period / Mean Interarrival Time (milliseconds)
1	CBR	1	450	25
2	VBR	10	2000	25
3	Poisson	20	800	6
4	VBR	50	1700	5
5	CBR	100	4200	5

Flow 1 and Flow 5 are CBR type of traffic, which require 140.625 and 6562.5 Kbps bandwidths. Flow 2 is a VBR source corresponding an average bandwidth of 625 Kbps. Flow 3 is a Poisson process, in which 800 byte packets are generated at a mean inter-arrival time of 6 milliseconds. An average bandwidth of 1041.17 Kbps is needed for Flow 3. Flow 4 is also a VBR source, which corresponds to an average bandwidth of 2656.25 Kbps. Be aware that all flows require more bandwidth than provided.

Figure 5.11 presents the bandwidth distribution among five flows for FCFS, WFQ, W^2FQ+, SFQ and DRR for the second scenario. According to the fairness measures given in Table 5.16 it is seen that W^2FQ+ and SFQ performs the best among the simulated algorithms. WFQ performs better than DRR, but as expected FCFS is the worst of all for the given simulation scenario. Be aware that according to the weights Flow 1 must get 50 Kbps, Flow 2 must get 500 Kbps, Flow 3 must get 1000 Kbps, Flow 4 must get 2500 Kbps and Flow 5 must get 5000 Kbps.

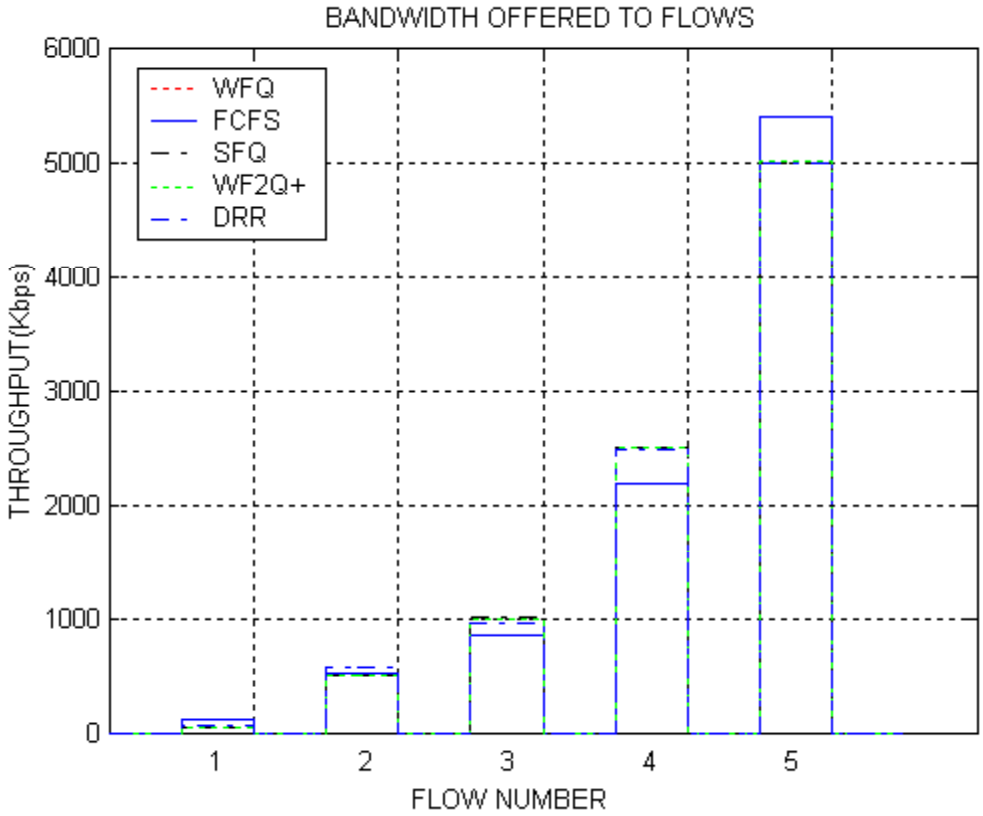


Figure 5.11 Bandwidth distributions among the five flows for the second simulation scenario – wired scheduling algorithms.

Table 5.16 Fairness measures of wireline scheduling algorithms for the second simulation scenario.

Algorithm	Fairness Measure
FCFS	932.5877
DRR	167.2243
WFQ	3.5007
W^2FQ+	0.3184
SFQ	0.3163

The upper and lower limits of the 90% confidence intervals for the rates distributed to flows are given in Table 5.17.

Figure5.12 presents the mean delay performance of the wireline scheduling algorithms for the second scenario. As stated before, for FCFS since the server does not deal with the weights of any flows, the mean delays experienced by the flows are equal. Since the other simulated schedulers try to compensate the discrepancies of bandwidth share, the flows that require more bandwidth then required delays more than the others. Table 5.18 lists the upper and lower limits of the 90% confidence intervals for the mean delays experienced by the flows.

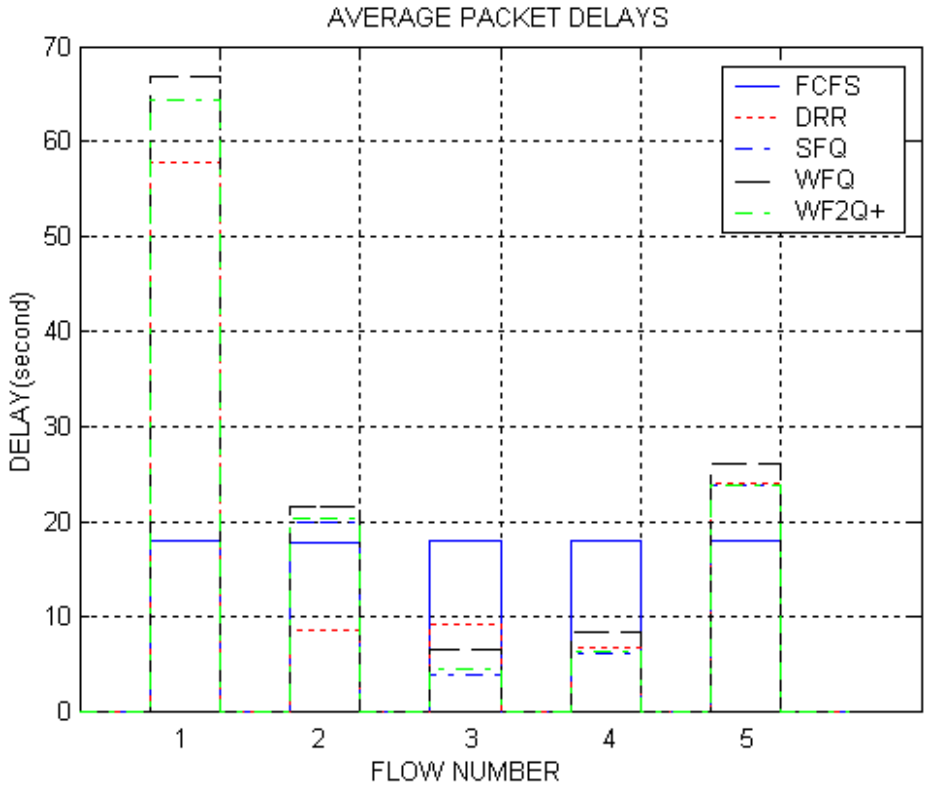


Figure 5.12 Mean delays experienced by the flows for the second simulation scenario.

Table 5.17 90% confidence intervals of the rates provided by the wired scheduling mechanisms for the second scenario.

	Flow1	Flow2	Flow3	Flow4	Flow5
FCFS	115.27≤ ≤115.71	500.52≤ ≤520.48	848.48≤ ≤861.27	2166≤ ≤2190.8	53800≤ ≤5400.5
DRR	59.30≤ ≤59.36	573.33≤ ≤575.15	948.55≤ ≤949.81	2479≤ ≤2484.5	49830≤ ≤49878
WFQ	49.97≤ ≤50.03	499.07≤ ≤499.94	998.90≤ ≤1000.5	2496.7≤ ≤2501	49998≤ ≤50033
W²FQ+	49.98≤ ≤50.01	499.79≤ ≤500.19	999.79≤ ≤1000.1	2499≤ ≤2500.4	49994≤ ≤50007
SFQ	49.99≤ ≤50.02	499.91≤ ≤500.27	999.83≤ ≤1000.2	2499≤ ≤2500.2	49995≤ ≤50003

Table 5.18 90% confidence interval of the mean delays experienced by the flows for the second scenario – wireline scheduling algorithms.

	Flow1	Flow2	Flow3	Flow4	Flow5
FCFS	17.542≤ ≤18.171	17.48≤ ≤18.1245	17.5848≤ ≤18.133	17.604≤ ≤18.15	17.54≤ ≤18.167
DRR	57.7≤ ≤57.82	5.9868≤ ≤11.104	8.3203≤ ≤9.7755	6.0321≤ ≤7.234	23.95≤ ≤24.053
WFQ	66.748≤ ≤66.866	19.8287≤ ≤23.451	5.5672≤ ≤7.4841	7.4869≤ ≤9.2	26.08≤ ≤26.204
W²FQ+	64.378≤ ≤64.473	17.9810≤ ≤22.857	3.3953≤ ≤5.4128	5.2171≤ ≤7.253	23.78≤ ≤23.83
SFQ	64.382≤ ≤64.44	17.7131≤ ≤22.257	3.2813≤ ≤4.3166	5.1414≤ ≤6.828	23.79≤ ≤23.82

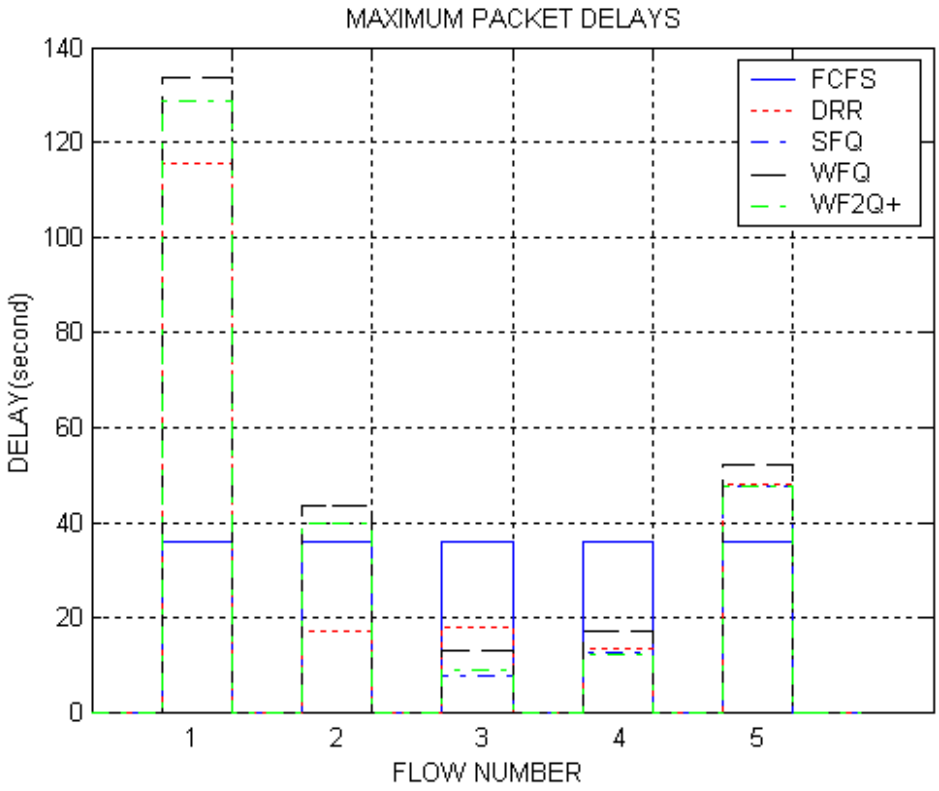


Figure 5.13 Maximum delays experienced by the flows for the second simulation scenario – wireline scheduling algorithms

The maximum delay performance is presented in Figure 5.13. As seen since DRR, WFQ, W^2FQ+ and SFQ tries to prevent unfairness, Flow 1, 2 and 5 experiences more mean and maximum delay than the other two flows. This is not true for FCFS, since the mechanism does not care about weights of the flows.

Table 5.19 lists the average number of packets transmitted by the wireline scheduling algorithms.

Table 5.19 Average number of packets, transmitted by the wireline scheduling algorithms for the second scenario.

	Flow1	Flow2	Flow3	Flow4	Flow 5
FCFS	6571	6571	27356	32870	32855
DRR	3376	7352	30374	37432	30387
WFQ	2845	6425	31990	37621	30486
W^2FQ+	2844	6442	31998	37567	30477
SFQ	2845	6404	32000	37635	30476

90% confidence intervals of maximum delays are given in Table 5.20.

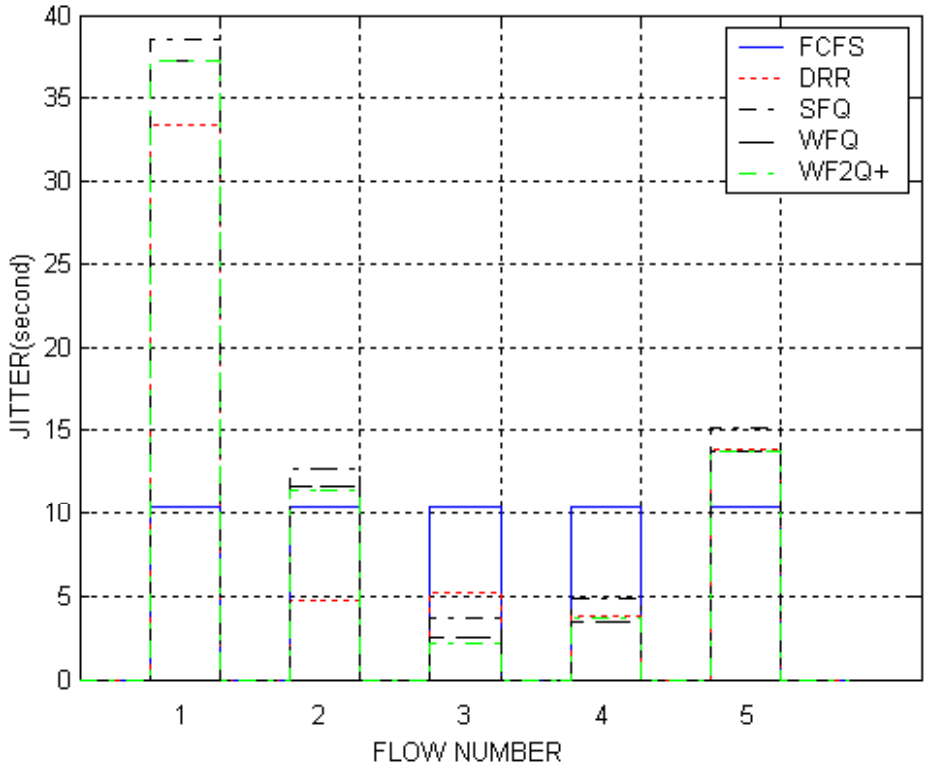


Figure 5.14 Jitter performances of wired scheduling algorithms for the second simulation scenario.

Table 5.20 90% confidence interval of the maximum delays experienced by the flows for the second scenario – wireline scheduling algorithms

	Flow1	Flow2	Flow3	Flow4	Flow5
FCFS	35.4970≤ ≤35.952	35.4906≤ ≤35.943	35.4976≤ ≤35.952	35.49≤ ≤35.953	35.49≤ ≤35.951
DRR	115.53≤ ≤115.598	13.9920≤ ≤20.391	17.1289≤ ≤18.885	12.33≤ ≤14.97	48.009≤ ≤48.11
WFQ	133.616≤ ≤133.65	41.5276≤ ≤45.960	11.495≤ ≤14.2999	15.8≤ ≤18.4893	52.31≤ ≤52.399
W²FQ+	128.819≤ ≤128.87	37.596≤ ≤42.6309	7.5625≤ ≤10.0622	11.03≤ ≤13.139	47.59≤ ≤47.631
SFQ	128.82≤ ≤128.862	36.6596≤ ≤43.513	6.996≤ ≤8.2065	11.63≤ ≤13.143	47.61≤ ≤47.62

Table 5.21 90% confidence interval of the jitter experienced by the flows for the first simulation scenario.

	Flow1	Flow2	Flow3	Flow4	Flow5
FCFS	10.2177≤ ≤10.484	10.1799≤ ≤10.5	10.2263≤ ≤10.466	10.234≤ ≤10.45	10.21≤ ≤10.484
DRR	33.3683≤ ≤33.389	3.4008 ≤ ≤6.1644	4.7505 ≤ ≤5.6979	3.4097≤ ≤4.196	13.8745≤ ≤13.9
WFQ	38.5771≤ ≤38.615	11.8527≤ ≤13.503	3.2115 ≤ ≤4.1552	4.2925≤ ≤5.434	15.13≤ ≤15.138
W²FQ+	37.1947≤ ≤37.218	10.1733≤ ≤13.167	1.9389≤ ≤3.0672	2.9855≤ ≤3.846	13.75≤ ≤13.748
SFQ	37.2057≤ ≤37.226	10.1438≤ ≤12.551	1.8199≤ ≤2.5352	3.3796≤ ≤4.032	13.742≤ ≤13.75

Figure 5.14 shows the jitter performance of the wireline scheduling algorithms for the second simulation scenario. Remember that jitter is defined to be the standard deviation of the delays experienced by the transmitted packets. The 90% confidence intervals of the jitters for the wireline scheduling algorithms are listed in Table 5.21

The traffic set whose characteristics are given at Table 5.15 is also simulated for wireless packet scheduling schemes. As in the first simulation for clarity in showing the effects of channel errors and ease of implementation, the errors are modeled as simple periodic bursts. The channel errors occur during the first 50 seconds to demonstrate long time fairness. The channels of Flow 1 and Flow 2 experience the same error pattern. The error pattern experienced by Flow 1 and Flow 2 is a periodic error burst of 1.8 second with 3 seconds of intermediate error free time. The error pattern experienced by Flow 3 is a less severe error-pattern, represents a periodic burst of 0.3 seconds with 5.7 seconds of intermediate error-free time. Flow 4 and 5 experiences no error at all.

As in the first simulation, the maximum number of credits and debits for WPS are both chosen as 20, and the maximum retransmissions are taken to be 2.

For CIF-Q, the α as chosen to be 0, so that a leading session *i* will no service as long as there exists a lagging error-free session in the system.

Figure 5.15 presents the bandwidth distribution among five flows for WPS, CIF-Q and DRRWC. Remember that the simulation has run for 10 times and the results are averaged. As in the first scenario the simulation time is 1200 seconds.

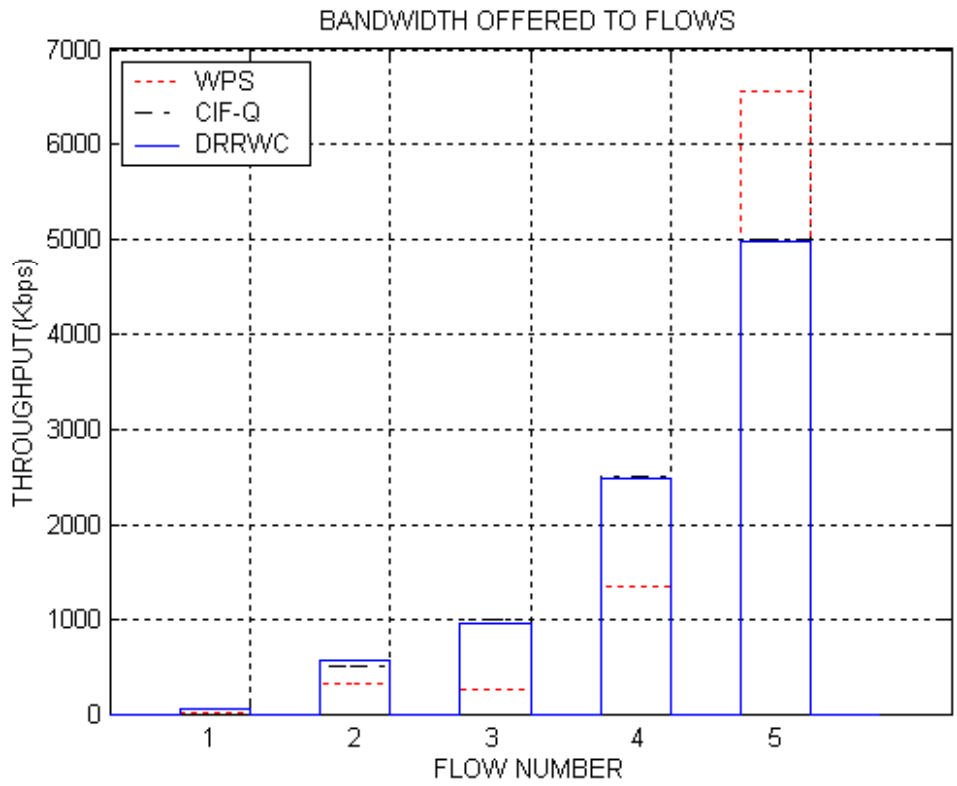


Figure 5.15 Bandwidth distributed by the wireless scheduling algorithms among the five flows for the second simulation scenario.

Table 5.22 lists the average bandwidth distributed by WPS, CIF-Q and DRRWC. The results are in Kbps.

Table 5.22 Average bandwidth, distributed by the wireless scheduling algorithms for the second simulation scenario.

	Flow1	Flow2	Flow3	Flow4	Flow5
WPS	7.01	311.8	250.6	1333.2	6561.8
CIF-Q	50.1	491.8	998.4	2502.3	5005.8
DRRWC	58.4	565.9	948.3	2485.6	4991.7

The fairness measure for the second simulation set is presented in Table 5.23.

Table 5.23 Fairness measures of wireless scheduling algorithms for the second simulation scenario.

Algorithm	Fairness Measure
WPS	3709.2
CIF-Q	17.95
DRRWC	148.7

The results given in Table 5.22 and 5.23, confirms that WPS was not successful as CIF-Q or DRRWC for the second traffic set. The best performing scheme for the second traffic set is CIF-Q. The long time fairness performance of DRRWC is better enough. The fairness performances of CIF-Q and DRRWC are closer to the ideal case more than WPS. As seen from Table 5.22 WPS allows the error free channels to get big share than channels evolving errors. But the other two algorithms compensate the discrepancies between the shares of error free channels and channels evolving errors.

The upper and lower limits of the 90% confidence interval for the rates provided to five flows are given in Table 5.24. The results are in Kbps.

Table 5.24 90% confidence intervals of the rates provided by the wireless scheduling mechanisms for the second scenario.

	WPS	CIF-Q	DRRWC
Flow1	$7.01 \leq \leq 7.02$	$50.05 \leq \leq 50.06$	$58.37 \leq \leq 58.384$

	WPS	CIF-Q	DRRWC
Flow2	307.7 ≤ ≤ 315.8	491.7 ≤ ≤ 491.8	565.74 ≤ ≤ 566.08
Flow3	250.4 ≤ ≤ 250.8	998.2 ≤ ≤ 998.5	948.23 ≤ ≤ 948.4
Flow4	1328 ≤ ≤ 1338	2502 ≤ ≤ 2502.7	2484.9 ≤ ≤ 2486.2
Flow5	6561.9 ≤ ≤ 6562.3	5004.9 ≤ ≤ 5006.6	4991.2 ≤ ≤ 4992.2

Figure 5.16 demonstrates the mean delay performances of the simulated wireless scheduling algorithms for the first scenario. The average number of packets transmitted by the wireless scheduling algorithms can be seen in Table 5.25.

Table 5.25 Average number of packets, transmitted by the wireless scheduling algorithms for the second scenario.

	WPS	CIF-Q	DRRWC
Flow1	2396.4	17086.7	19926.1
Flow2	23926.7	37721.2	43415.5
Flow3	48121.2	191694	182075.8
Flow4	120367.4	225872.6	224629.9
Flow5	239974.2	183068	182554.9

It is seen that, the mean delays experienced by the flows are associated with the bandwidths required by the sessions and the bandwidths provided to them. As seen from Figure 5.16 and Table 5.26, for CIF-Q Flow 3 experiences the minimum mean delay, because the bandwidth requirement is the closer to the provided compared to other flows.

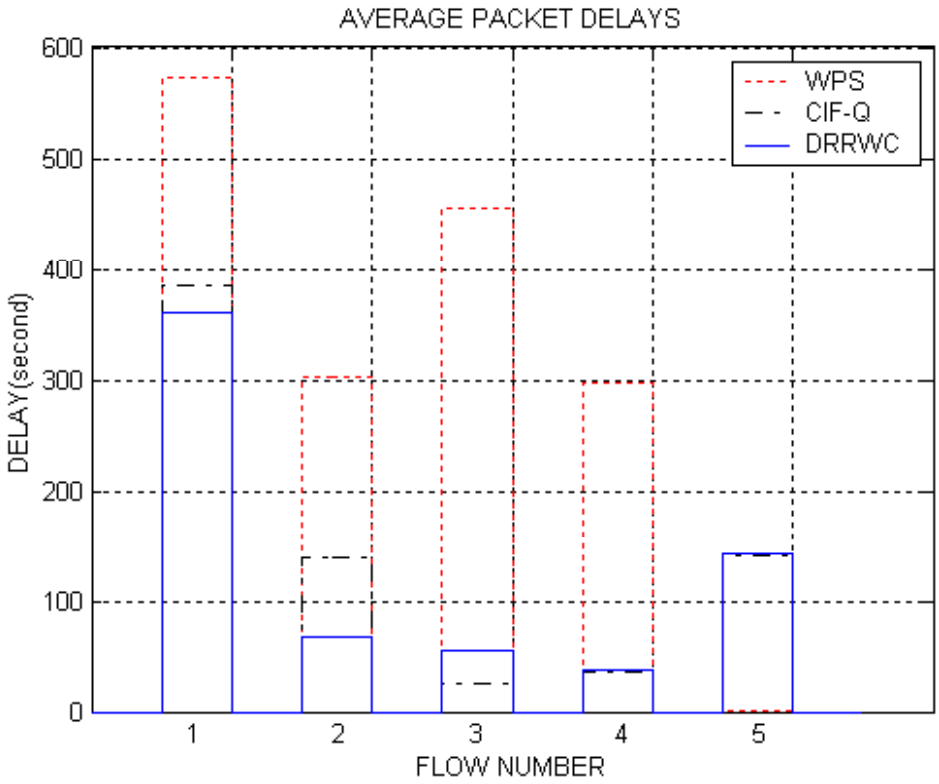


Figure 5.16 Mean delays experienced by the flows for the second simulation scenario – wireless scheduling algorithms.

The upper and lower limits of the 90% confidence interval for the mean delays experienced by the five flows are given in Table 5.26. The values are in seconds.

Table 5.26 90% confidence intervals of the mean delays experienced by the flows for the second scenario – wireless scheduling algorithms.

	WPS	CIF-Q	DRRWC
Flow1	571.9 ≤ ≤ 572.6	385.5 ≤ ≤ 385.9	360.24 ≤ ≤ 360.4
Flow2	300.9 ≤ ≤ 306.1	135.2 ≤ ≤ 144	63.8 ≤ ≤ 71.5

	WPS	CIF-Q	DRRWC
Flow3	454.1 ≤ ≤ 456.6	24 ≤ ≤ 27.1	53.7 ≤ ≤ 55.4
Flow4	296.1 ≤ ≤ 299.9	33.8 ≤ ≤ 37.6	35.6 ≤ ≤ 40.2
Flow5	0.102 ≤ ≤ 0.103	141.5 ≤ ≤ 141.8	142.6 ≤ ≤ 142.8

Figure 5.17 presents the maximum delays experienced by the flows for the second simulation scenario.

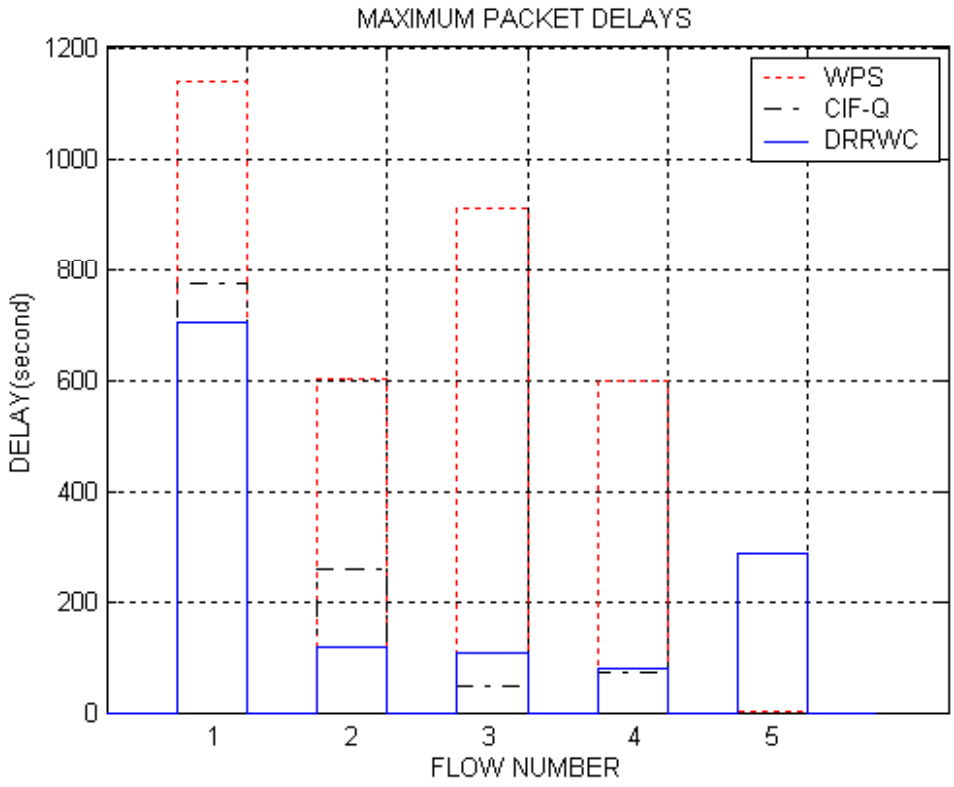


Figure 5.17 Maximum delays experienced by the flows for the second simulation scenario – wireless scheduling algorithms.

The 90% confidence limits for the maximum delays experienced by the flows are given in Table 5.27. The results are in seconds.

Table 5.27 90% confidence intervals of the maximum delays experienced by the flows for the second scenario – wireless scheduling algorithms.

	WPS	CIF-Q	DRRWC
Flow1	1139.7≤ ≤1139.9	772.6≤ ≤772.9	701.78≤ ≤701.85
Flow2	597.9≤ ≤605.6	253.3≤ ≤262.3	110.8≤ ≤121.7
Flow3	910.1≤ ≤913.46	46.3≤ ≤51.2	105.75≤ ≤108.56
Flow4	595.86≤ ≤599.91	68.8≤ ≤74.1	75.4≤ ≤79.7
Flow5	0.26≤ ≤0.448	284.5≤ ≤284.8	287.15≤ ≤287.3

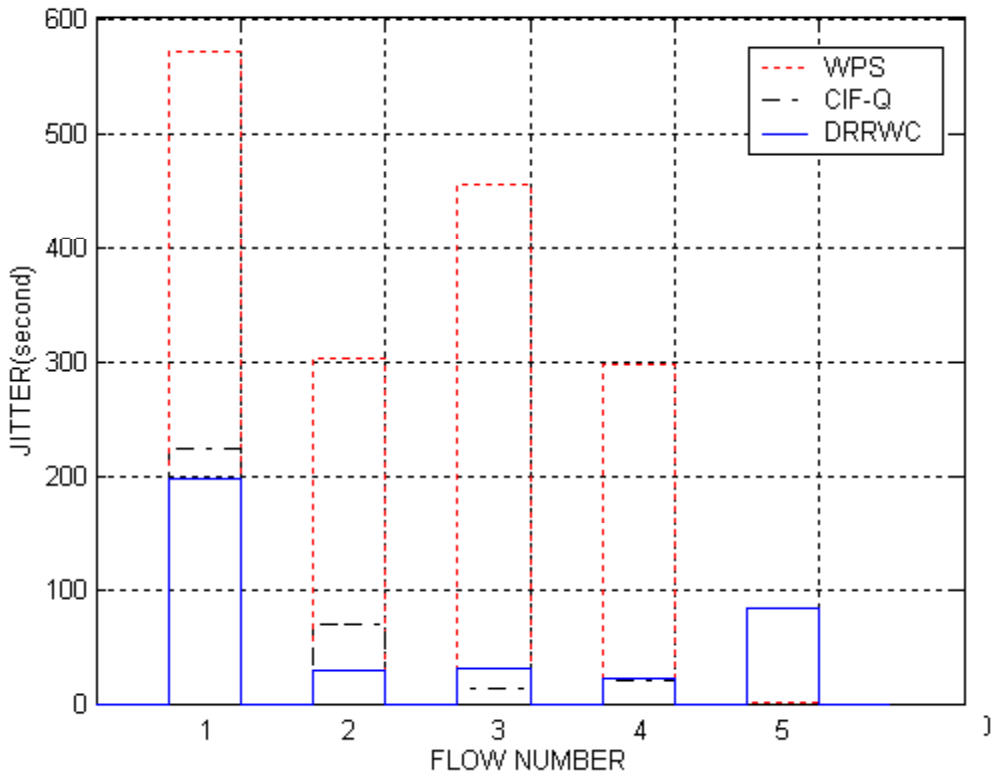


Figure 5.18 Jitter performances of wireless scheduling algorithms for the second simulation scenario.

The jitter performances of wireless scheduling algorithms can be seen in Figure 5.18. As stated before since the bandwidth requirement of Flow 3 is the more closer to the provided, it experiences the minimum jitter among the other flows when the flows are scheduled by CIF-Q or DRRWC server.

The upper and lower limits of the 90% confidence limits for the jitters experienced by the flows are given in Table 5.14. The results are in seconds.

Table 5.28 90% confidence intervals of the jitters experienced by the flows for the second simulation scenario – wireless scheduling algorithms.

	WPS	CIF-Q	DRRWC
Flow1	$571.3 \leq \leq 572$	$223.2 \leq \leq 223.37$	$197.4 \leq \leq 197.5$
Flow2	$300.6 \leq \leq 305.8$	$66.6 \leq \leq 71.4$	$26.6 \leq \leq 30.9$
Flow3	$453.7 \leq \leq 456.1$	$12.6 \leq \leq 14.7$	$30.2 \leq \leq 31.3$
Flow4	$295.8 \leq \leq 299.6$	$20 \leq \leq 21.7$	$21.66 \leq \leq 23.8$
Flow5	$0.102 \leq \leq 0.103$	$82.5 \leq \leq 82.53$	$83.3 \leq \leq 83.43$

CHAPTER 6

CONCLUSIONS

In this thesis, simulation of various scheduling algorithms for packet switching networks was studied. For evaluating the performance measures of some selected scheduling disciplines, the issues related to the development of computer aided modeling and design of computer communication networks were addressed. Methods to eliminate transient periods in a simulation process, generation of random variables, computation of confidence intervals and validation of simulation models were described.

To provide a common platform for comparing the performances of selected scheduling disciplines in serving different traffic types, a discrete event simulator has been developed. It was seen that, event-driven simulation of scheduling algorithms is both efficient as well as more accurate.

The common trend in the literature is to assign bandwidth fairly in which users with moderate bandwidth requirements are not penalized because of the excessive demands of others. As a result of equalizing the bandwidth, these schemes typically provide satisfactory QoS for sessions whose bandwidth requirements are less than their fair share.

Among the existing packet scheduling algorithms for wireline environment FCFS is the poorest one since it does not deal with the notion of flows or weights. GPS is the ideal algorithm but it is not practically implementable. So the other scheduling schemes tries to approximate it. WFQ or W^2FQ schedules the packets from different queues by referencing GPS scheme. W^2FQ+ approximates GPS ideally by using a virtual time function and as WFQ and W^2FQ it guarantees fair share of bandwidth. SFQ is another scheme that uses virtual time function and it schedules the packets in the increasing order of their start tags. WRR is an extension of Round Robin scheduling scheme that tries to provide fairness. DRR is designed to support variable size packet scheduling but it does not support guaranteed bandwidth.

Wireless packet schedulers are built up from the wireline schedulers and they try to provide fairness in the long term by introducing terms lag, lead, credit or debit. WPS uses WRR as its error-free reference algorithm and it uses the credit/debit adjustment for providing fair share of bandwidth. SBFA maintains an additional flow named LTFS to provide fairness. LTFS keeps tracks of the packets that are not transmitted due to channel errors.

From the simulation studies it was seen that, fair scheduling provides, fair allocation of bandwidth, lower delay for sources using less than their full share of bandwidth and protection from ill-behaved resources. For wired networks, by looking at the fairness measures, the fairness performances of WFQ, WF^2Q+ and SFQ are closer to the ideal case. For wireless media CIF-Q has performed the best in terms of fairness. The fairness performance of proposed scheduling scheme was better than WPS but worse then CIF-Q. It was also seen that, although FCFS does

not guarantee a fair allocation of bandwidth, it is easy to implement. GPS itself is not suitable for implementation. Since WFQ and WF^2Q approximate the GPS scheme, the implementation efforts are higher than the WF^2Q+. DRR and DRRWC are easy to implement.

Implementing more scheduling schemes, like hierarchical resource management models and algorithms that support both link sharing and guaranteed real-time services with priority is also desirable.

REFERENCES

1. Kleinrock, L., Queueing Systems Volume II: Computer Applications, Wiley-Interscience Publications, pp. 292-304, 1976.

2. W. Chou, Computer-Communications-Volume I Principles, Prentice-Hall, pp. 347-358, 1983.

3. M. Ilyas, and H.T. Mouftah, ”Performance Evaluation of Computer Communication Networks”, IEEE Communications Magazine, Vol. 23, No 4, April 1985, pp. 18-29

4. John B. Nagle, “On Packet Switches with Infinite Storage”, IEEE Transactions on Communications, COM-35 (4), pp. 435-438, April 1987.

5. Kurose, J.F. and Mouftah, H. T., ”Computer-Aided Modeling, Analysis and Design of Communication Networks”, IEEE Journal on Selected Areas in Communications, Vol. 6, No.1 January 1988, pp. 130-145

6. M. Ilyas, and H.T. Mouftah, ”Simulation Tools for Computer Communication Networks”, Global Telecommunications Conference, 1988, and Exhibition. 'Communications for the Information Age.' Conference Record, GLOBECOM '88., IEEE , 28 Nov.-1 Dec. 1988, pp. 1702 -1706 Vol.3

7. Alan Demers, Srinivasan Keshav, and Scott Shenker, “Analysis and Simulation of a Fair Queueing Algorithm”, Proceedings of SIGCOMM’89, Vol. 19, Austin, Texas, September 1989.

8. Verma, D. C., Zhang, H., and Ferrari, D., “Delay Jitter Control for Real-Time Communication in Packet Switching Networks”, In Proceedings of Tricomm’91, pp. 35-46, North Carolina, April 1991.

9. Manolis Katevenis, Stefanos Sidiropoulos, and Costas Courcoubetis, “Weighted Round-Robin Cell Multiplexing in a Genaral-Purpose ATM Switch Chip”, IEEE Journal on Selected Areas in Communications, Vol. 9, No. 8, pp. 1265-1279, October 1991.

10. Paul E. McKenny, “Stochastic Fairness Queueing”, Journal of Internerworking Research and Experience, Vol 2, pp. 113-131, 1991.

11. David D. Clark, Scott Shenker and Lixia Zhang, “Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism”, In SIGCOMM Symposium on Communications Architectures and Protocols, pp. 14-26, Baltimore, Maryland, August 1992.

12. Parekh, A. K., and Gallager, R. G., “A Generalized Processor Sharing Approach to Flow Control in Integrated Services Network: The Single Node-Case”, IEEE/ACM Transactions on Networking 1 (3), pp. 344-357, Jun 1993.

13. Çağan M. Aras, James F. Kurose, Douglas S. Reeves and Henning Schulzrinne, “Real-Time Communication in Packet Switched Networks”, Proceedings of the IEEE, 82(1):122-139, January 1994.

14. Victor S. Frost and Benjamin Melamed, “Traffic Modeling for Telecommunication Networks”, IEEE Communications Magazine, March 1994, pp. 70-81

15. Parekh, A. K., and Gallager, R. G., “A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple Node Case”, IEEE/ACM Transactions on Networking 2 (2), pp. 137-150, April 1994.

16. S. Golestani, “A self-clocked fair queueing scheme for broad-band applications”, In Proceedings of IEEE INFOCOM'94, pp. 636-646, Toronto, CA, June 1994.

17. M. Shreedhar and George Varghese, “Efficient Fair Queueing using Deficit Round Robin”, Proceedings of SIGCOMM’95, pp. 231-242, 1995.

18. Jon C. R. Bennett and Hui Zhang, H., “ WF^2Q : Worst-case Fair Weighted Fair Queueing”, In Proceedings of IEEE INFOCOM (1996), San Francisco, CA, Mar. 1996.

19. Jon C. R. Bennett, Hui Zhang, “Why WFQ Is Not Good Enough For Integrated Services Networks”, In Proceedings of the 6th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), Shonan Village International Conference Center, Zushi, Japan, April 1996.

20. P. Goyal, H. M. Vin., and H. Cheng, “Start-time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks”, In Proceedings of ACM SIGCOMM’96, pp. 157-168, August 1996.

21. P. Goyal, X. Guo and H. Vin. ``A Hierarchical CPU Scheduler for Multimedia Operating Systems", in Proc. OSDI'96, USENIX, pp. 107-121,Oct. 1996.

22. A. S. Tanenbaum, Computer Networks, Prentice-Hall Inc., pp. 130-134, 1996.

23. Jon C. R. Bennett and Hui Zhang, “Hierarchical packet fair queueing algorithms”, IEEE/ACM Transactions on Networking, 5 (5), pp. 675-689, 1997.

24. S. Keshav, An Engineering Approach to Computer Networking, Addison Wesley Professional Computing series, pp.163-184, 1997.

25. T. S. Euene Ng, Ion Stoica, Hui Zhang, “Packet Fair Queueing Algorithms to Wireless Networks with Location-Dependent Errors”, Proceedings of IEEE INFOCOM’98, 1998.

26. B. Forouzan, Introduction to Data Communications and Networking, McGraw Hill International Editions, pp. 358-370, 1998.

27. P. Ramanathan, and P. Agrawal, “Adapting Packet Fair Queueing Algorithms to Wireless Networks”, Proceedings of MOBICOM, 1998.

28. Songwu Lu, V. Bharghavan, and R. Srikant, “Fair Scheduling in Wireless Networks”, IEEE/ACM Transactions on Networking, Vol. 7, pp 473-489, August 1999.

29. C. Deleuze, “Scheduling”, COST 237, Final Report, 1999.

30. Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, John Heidemann, Ahmed Helmy, Polly Huang, Steven McCanne, Kannan Varadhan, Ya Xu, and Haobo Yu, “Advances in Network Simulation”, IEEE Computer, 33 (5), pp. 59-67, May, 2000.

31. C. Santiago, and R. Valadas, “A Simulation Study of Flow-Based Scheduling Algorithms”, In the Proceedings of ConfTele 2001, 2001.

32. Jasleen Kaur and Harrick M. Vin, “End-to-End Fairness Analysis of Fair Queueing Networks”, Real-Time Systems Symposium, 2002. RTSS 2002. 23rd IEEE , 3-5 Dec. 2002, pp. 49 –58

33. Xiaohong Yuan and Mohammad Ilyas, “Modeling of Traffic Sources in ATM Networks”, Proccedings IEEE SoutheastCon 2002, pp. 82-87.

34. Assoc. Prof. Dr. Buyurman Baykal, “Communication Network Analysis and Teletraffic Engineering”, March 2003, Class Notes.

APPENDIX A

CIF-Q ALGORITHM

In this appendix, we provide the simple version of CIF-Q algorithm.

A.1 Simple Version:

on session *i* **receiving** packet *p*:

enqueue($queue_i$; p)

if (i $\notin$ A)

$v_i = \max(v_i \ ; \ \min_{k \in A} \{v_k\})$;

$lag_i = 0$;

$A = A \cup \{i\}$; /* mark session active*/

on sending current packet: /* get next packet to send */

$i = \min_{V_i} \{i \in A\}$; /* select session with min. virtual time */

if ($lag_i = 0$ **and** (i can send)) /* session i non-leading, can send */

p =**dequeue**($queue_i$);

$v_i = v_i + p.length / r_i$;

else

$j = \max_{lag_k / r_k} \{k \in A | \ k \text{ can send } \}$;

if (j **exists**)

p =**dequeue**($queue_j$);

$v_i = v_i + p.length / r_i$; /* charge session i */

$lag_i = lag_i + p.length$;

$lag_j = lag_j - p.length$;

if ($i \neq j$ **and empty**($queue_j$) **and** $lag_j \geq 0$)

leave(j);

else /* there is no active session ready to send */

$v_i = v_i + \delta / ri$;

if ($lag_i < 0$ and empty($queue_i$))

/* *i is leading, unbacklogged* */

$j = \max_{lag_k / r_k} \{k \in A\}$;

$lag_i = lag_i + \delta$;

$lag_j = lag_j - \delta$; /* forced compensation */

set_time_out(**on sending** , δ / R);

if (**empty**($queue_i$) **and** $lag_i \geq 0$)

leave(i); /* session i leaves*/

$A = A \setminus \{ i \}$;

for ($j \in A$) /* update lags of all active sessions*/

$lag_j = lag_j + lag_i \times r_j \Big/ \sum_{k \in A} r_k$;

if ($\exists j \in s.t.$ **empty** ($queue_j$) Λ $lag_j \geq 0$)

leave(j);