Design of a Scheduling Mechanism for an ATM Switch

Nicholas Oliver Harvey

Design of a Scheduling Mechanism for an ATM Switch

Abstract

The quest for a unified voice, video and data network led to the design of Asynchronous Transfer Mode (ATM), a packet switching architecture. One of the promises of ATM is the provision of guaranteed Quality of Service (QoS). A major factor in the provision of QoS in ATM switches is the capability of a scheduling mechanism to control access to the switching fabric. Historically the use of queuing and scheduling at the switch fabric outputs has been the favoured approach used in the design of ATM switches. However, input queued switch architectures have recently been receiving considerable interest in the research community. Input queued switch architectures for various reasons have become an attractive solution for the design of high performance packet switches. In particular input queued switches scale better to larger sizes. They are also cheaper to construct in terms of the speed of the switch fabric and the memory size and bandwidth required. However, there are still considerable problems associated with the implementation of input queued scheduling algorithms.

The dissertation to follow is devoted to design and related theory of a centralized scheduling algorithm to maximize the throughput and provide fair bandwidth allocation in an input queued ATM switch. In particular, the use of Virtual Output Queueing (VOQ) at the input ports to avoid head of line blocking is considered. The problem faced by an input queued scheduling algorithm utilizing VOQ is formalized as finding the maximum weighted matching on a bipartite graph in which nodes represent input and output ports and edges represent cells to be switched.

In this dissertation, the candidate proposes the use of a ratio to multiply the weights used in the matching algorithm to control the delay that individual connections encounter. We demonstrate the improved characteristics of a switch using a ratio presenting results from simulations.

The candidate also proposes a novel scheduling mechanism for an input queued ATM switch. In order to evaluate the performance of the scheduling mechanism in terms of throughput and fairness, the use of various metrics, initially proposed in the literature to evaluate output buffered switches are evaluated, adjusted and applied to input scheduling. In particular the Worst-case Fairness Index (WFI) which measures the maximum delay a connection will encounter is derived for use in input queued switches

The evaluation of the scheduling algorithm is conducted with the use of a specifically developed software cell based simulator. The simulator is designed to evaluate the performance of input scheduling algorithms under various traffic conditions and different switch configurations. In particular the use of algorithms which accurately mimic real network traffic allows the evaluation of the scheduling mechanism under realistic conditions. By using the simulator to compare the performance of the proposed scheduling algorithm with those in the literature, the candidate will demonstrate the scheduler's advantages. The dissertation is concluded with a study of the practical considerations of implementing the scheduler in hardware, and with some thoughts on simulating scheduling algorithms. Some ideas for further research are also proposed.

Table of Contents

List of Figures

List of Tables

Chapter 1 : Introduction

1.1 Quality of Service in Packet Switched Networks

Rapid advances in computer and telecommunications network technologies have meant that in the near future we will experience as part of every day life, applications and services that a decade or two ago were in the realms of science fiction. Examples of these applications are videophones, video on demand and multi person virtual reality.

Traditionally there was one network for each type of traffic, for example a business enterprise may have had a private Time Division Multiplex (TDM) based voice network, an Internet Protocol (IP) network to the Internet, and an ISDN video conferencing network as well as a multi-protocol based Local Area Network (LAN). However today network traffic is becoming more varied and in the future there will be requirements to support an even more diverse set of applications. It is evident that it would be ideal to support all of these applications on one network.

When the different applications had separate dedicated networks, controlling the levels of service provided was less of a problem as all the traffic in each network was similar and the networks could be engineered and fine tuned to provide the required service levels. When different types of traffic, each with different requirements are transported over a single network, the different traffic types often react unfavourably together. For example a voice application expects to experience no packet loss and a minimum but fixed amount of packet delay. The voice application receives this service level when it operates over a TDM network. A best effort IP network has varying amounts of packet loss and variable delay. The result is that the voice application runs very badly on the IP network, however the alternative of using circuit switching to transmit data, results in ineffective use of the resources of the network.

1.1.1 QoS and the Converged Network

What is required in a multi-service network is a means of preventing different applications from affecting the performance of one another. The term for this concept is "Quality of Service" (QoS). A best effort IP network effectively provides no QoS. Other networks

technologies such as Asynchronous Transfer Mode (ATM) have been engineered to provide QoS.

From the network user's point of view, QoS is a term used to describe the complete experience an application will receive from a network. A more network centric definition would be that QoS refers to the classification of packets for the purpose of treating certain classes or flows of packets in a particular way compared to other packets.

The aim of this chapter is to provide an introduction to QoS and to show why it is necessary and how it is incorporated in the modern network and what mechanisms are required to provide this.

Today most networks are converging to an IP based transport layer. Data applications are already all IP-based, while traditionally Time Division Multiplexing (TDM) based voice networks are also transitioning to being IP-based. Video conferencing is also moving towards IP however at a slower pace.

Traditionally IP has been routed over ATM using IP over ATM virtual circuits or multi protocol over ATM (MPOA). These forwarding methods have proven to be complicated and cumbersome. However packet switched IP does have numerous advantages over traditional IP routers, the most important of which is the ability to provide QoS. Along with this move to IP transport networks, the Multi Protocol Label Switching (MPLS) protocol is being widely deployed in the core network as a means to bring the speed advantages of packet switching to the IP realm. Multi protocol Label switching allows IP flows to be routed through a packet switched network and so allows IP to acquire some of the QoS mechanisms of ATM and packet switched networks. The intention of this chapter is not to give an introduction to MPLS or any particular technology, but merely to show how each technology provides QoS.

1.1.2 QoS Parameters

How is QoS quantified? A number of QoS parameters can be measured and monitored to determine whether a service level offered or received is being achieved. Not all are equally important to every application. Perhaps the most basic is network availability. Network availability can have a significant affect on QoS. Network availability is the sum of the availability of many items that are used to create a network, such as fibre, network interface cards, power sources etc. Network operators can determine their networks

availability by among other things the amount of redundancy they incorporate into their network.

When the term QoS is discussed with regard to packet switched networks what comes to mind is most commonly associated with a guaranteed bandwidth. When more than one user shares the use of a network they compete for the available bandwidth. They get more or less bandwidth depending on various factors. If QoS is to be implemented in a network some mechanism is required to divide up the bandwidth and share it fairly among users. Since the point of congestion in most modern computer networks is the switch or router, dividing up the bandwidth amounts to implementing a scheduling mechanism which allocates access to the switch fabric. This topic is the focal point of this text.

Network delay is the transmit time a user experiences from the ingress point to the egress point of the network. Delay can cause significant QoS issues with applications such as voice and video. The delay can be fixed, for example due to propagation delay, or it can be variable, for example due to contention in switches.

Jitter is the measure of delay variation between consecutive packets for a given traffic flow. Jitter has a pronounced effect on real time delay sensitive applications such as voice and video. Losses can occur due to various reasons. Two of the main causes of packets loss are errors introduced by the physical transmission medium and congested network nodes. Because congestion has a direct impact on packet loss, congestion avoidance mechanisms are often deployed. In the ideal converged network, those connections which are within their allowed traffic agreement will not experience any of the congestion.

1.2 A Quality of Service Framework

Providing QoS in a network has become an active area of work in the last few years, and many approaches have been investigated. Generally implementing QoS in a network requires a number of technologies to work together. These QoS technologies operate at different layers in the protocol stack and provide different services. Some of these are listed in Table 1.The purpose of this section is not to delve in depth into these technologies but to provide an understanding of how they are used to support QoS.

Table 1. – QoS Technologies

QoS Monitoring	QoS Aware Application	RSVP
	IP QoS	IP Differentiated Services
	Network-Signalled QoS	ATM PNNI, MPLS RSVP-TE or MPLS CR-LDP
	Traffic Engineered Paths	ATM Virtual Circuits, MPLS Label Switched Paths(LSP)
	Link Layer QoS	ATM, MPLS, PPP, Frame Relay
	Physically Layer QoS	Wavelengths, Ports, Virtual Circuits

Physical technologies allow for the separation of traffic. The separation may take the form of wavelengths, Virtual Circuits, ports on a device, or frequencies over the air. This is the simplest form of QoS whereby different levels of QoS are provided through traffic separation at the physical layer. For example different wavelengths may provide different services. This type of QoS works well when the transmission media is inexpensive or abundant, but when the resources are limited it becomes inefficient and expensive, for example the Wireless frequency spectrum. ATM currently has the most comprehensive QoS support. The ATM forum has created ATM service categories, each with a different QoS traffic management parameters and performance levels. A more comprehensive introduction to a QoS framework is provided in Appendix A.

1.3 Packet Switching

Both MPLS and ATM have packet switching as the core of the technology. This could be said to form the basis of the ability of these technologies to support QoS. Packet switching means the forwarding of the packet through the switch fabric from the input port to the output port. Scheduling consists of choosing which packet to transmit or forward across the switch fabric when there is more than one packet destined for the same output port. The switch scheduling mechanism thus determines the order in which traffic is forwarded as it traverses a network node. Traffic with higher priority is typically forwarded ahead of traffic with lower priority. The scheduling mechanism may be designed to classify traffic as belonging to different classes or each connection may be assigned a specific bandwidth

share. Thus the scheduling priority determines the amount of latency introduced to the traffic by the switching node. In its simplest form a scheduling mechanism is a simple priority scheme where higher priority connections are always given priority ahead of lower priority traffic. This is usually accomplished using some form of priority scheduling. The problem with this approach is that lower priority traffic may never get serviced, if there is always higher priority traffic entering the switch. More elaborate scheduling schemes provide a weighted approach to the transmission of traffic to improve fairness. In this way lower priority connections will not always yield to higher priority traffic. The ideal scheduling situation is when every connection gets its exact share of the bandwidth at every moment. This is called Generalised Processor Sharing, and is a fluid system, and is not possible to implement exactly in Packet Switched Networks.

It has been shown that the element that has the most effect on the QoS provided by packet switched networks is the mechanism that determines which packets will be transmitted next on the output link. We will refer to this mechanism as the traffic scheduling algorithm. Although this traffic scheduling algorithm is the basis for the provision of QoS, as has been shown the successful provision of QoS is also dependant on various other technologies at different layers in the protocol stack.

1.4 Scheduling in a Packet Switch

One of the major barriers to building high performance networks is the difficulty of high speed switching. With the introduction of fibre optics the performance bottleneck has moved from the transmission media to the switching node. Many ATM switches have been proposed in the literature and this is still an area of active research.

The aim of this dissertation is to focus on scheduling strategies for packet switches. Although it is impossible to separate the scheduling mechanism from the switching fabric, we make certain assumptions on the architecture of the switching fabric. The switch fabrics considered all fall into the space division, internally non-blocking category. These fabrics provide multiple concurrent paths between the input and output ports and allows multiple cells to be simultaneously switched across the fabric, assuming that all cells are destined for distinct outputs. This dissertation investigates the issues involved in the design of scheduling algorithms.

Scheduling mechanism that packet switches use can be generally classified into two categories depending on whether cells are buffered at the inputs or the outputs of the

switch. Scheduling algorithms for input buffered and output buffered switches are investigated in this thesis, although the main focus is input buffered switches.

Output buffered switches are examined first. Traditionally ATM switches have employed output queuing. Switches with output queues are conceptually simpler and were more popular before the recent interest in input queued switches. In output queuing, when packets arrive at an input port they are transferred by a high speed switching fabric to the correct output port. At the output port a scheduling mechanism then allocates the output bandwidth to the different connections according to a specific policy. The function of the scheduling mechanism is thus to select the particular packet to be transmitted next on the output link.

Scheduling algorithms for output buffered switches are currently able to offer guaranteed bandwidth protection to individual connections, however for various reasons, mainly due to speed limitations, output buffered switches are not practical for all but the smallest switches. These factors are investigated more completely in the following chapters.

To overcome the many problems associated with output queuing, input queuing is being widely investigated. In an input buffered switch cells are buffered at the input ports of the switch. A set of paths must be established in the switch fabric for transmitting packets from the input ports to the output ports of the switch. Input Buffered have their own set of problems, the most serious is that they are currently not able to offer any meaningful delay guarantees to connections.

1.5 The Evaluation of Scheduling Mechanisms

The evaluation of scheduling algorithms is a non trivial exercise. There are many factors to consider and often trade offs need to be made. Most of the considerations apply to both output and input queued switches; however there are some which apply exclusively to each. We now briefly discuss some of the issues involved in the design of a scheduling mechanism.

Perhaps the primary requirement for a scheduling algorithm is that it should be able to isolate a connection from the effects of other possibly misbehaving connections. This means that the scheduling algorithm must provide per connection isolation. There are certain QoS frameworks which provide different classes of service to different types of

applications, for example guaranteed delay traffic would always have the highest priority. However providing per connection QoS means that a scheduler should also be able to isolate traffic belonging to the same class. Note that the ability to isolate traffic is necessary even if policing mechanism are used to shape traffic entering the network, since traffic can become bursty while in the network.

A scheduling algorithm should also be able to effectively utilize the available bandwidth. In other words the algorithm should be able to achieve the benefits of statistical multiplexing, while also being able to handle bursty sources. It should be able to do this in real-time as well. For example if a low delay, high priority connection is not using its bandwidth, it should be instantly available to low priority traffic, but as soon as the high priority application requires its bandwidth capacity, it should be available.

A scheduling algorithm should preferably minimize the delay that cells encounter as they are routed across the switch. This delay should preferably be independent of the behaviour of other connections. If a switch is not able to provide proper traffic isolation, behaving connections end up enduring unwanted delay. Algorithms which provide proper traffic isolation may still differ considerably in their ability to provide low delay.

Another requirement is that the scheduling mechanism implementation should also be fair. This is not exactly the same as providing traffic isolation. The algorithm should attempt to allocate each connection its fair share of bandwidth every instant, and should also distribute excess bandwidth fairly among connections including those with no minimum bandwidth reservations. In reality it is helpful to consider the ability of a scheduling mechanism to offer fair bandwidth distribution over the long term and over short time intervals. While two schedulers may distribute their bandwidth equally fairly over a long time interval, one may for instance penalize a connection for bandwidth received in excess of its reservation at an earlier time. Thus a backlogged connection would be starved until all the other connections are given the same amount of service. This situation would lead to short-term unfairness. Thus two scheduling algorithms having otherwise similar properties may exhibit vastly different fairness properties over the short term. This factor is very important for delay sensitive traffic, such as packet based voice applications.

Scheduling algorithms usually need to be implemented in hardware in high speed networks.
Some of the scheduling algorithms presented in the next chapters perform very well theoretically but are immensely complex. It is important that the scheduling algorithm be

Scheduling algorithms usually need to be implemented in hardware in high speed networks.

Some of the scheduling algorithms presented in the next chapters perform very well theoretically but are immensely complex. It is important that the scheduling algorithm be simple enough to implement in hardware. In ATM the available time for scheduling cells is very short. At ATM speeds less than 3 μs are available for scheduling. The result of this is that only hardware implementations are feasible. This places severe restrictions on the complexity of algorithms that can be implemented. For example some of the factors that can increase the implementation complexity of a scheduling algorithm are the need to maintain per cell or per connection timestamps and to generate random numbers. However the scheduling algorithms which don't use timestamps generally provide worse delay bounds and fairness. The scheduling algorithm designer usually has to trade off complexity of implementation with the properties of low delay and good short-term fairness. In practice often the only way to determine whether a scheduling algorithm is feasible and can support the desired speed is to implement it in a prototype.

The scheduling algorithm should also scale well to larger sizes and maintain its performance characteristics. It should also perform equally well at different link speeds. It is quite a difficult task to simultaneously provide good delay bounds to a 150mb/s connection and a 56kb/s connection. Again often the only way to determine if a scheduling algorithm is feasible to implement at a specific switch size is to prototype it.

An important criterion for scheduling algorithms is that they perform well under real network conditions. Real network traffic is correlated and in practice packets arrive in bursts. Understanding the nature of this sort of traffic and evaluating its impact are important for the design of network elements. Many ways of modelling traffic bursts in network traffic have been proposed. Studies have shown that burstiness has more influence on the performance of output queued switches than input queued switches. In other words burstiness tends to concentrate the conflicts on outputs rather than inputs. Generally burstiness has been shown to increase the queuing delay of a particular connection; however some scheduling algorithms may be more susceptible to decreased performance under bursty traffic.

to mimic a real world switch as closely as possible. For example a typical configuration would be a 16 by 16 switch with up to a few hundred connections on each port.

To model a core network switch the connections would be randomly distributed across all the ports and the destinations would also be more or less randomly distributed. To model a network edge switch all the input ports connections destination would be focused all on one or two of the output ports, to model a switch connecting several DSL access multiplexers (DSLAMS) to the network.

An important part of the simulation is the choice of traffic model. As is widely known real network traffic is not uniform, and has been shown to be bursty over many time scales. Ideally the traffic model should mimic real network traffic as closely as possible. The modelling of real network traffic is a complex topic. Generating a bit stream that closely approximates real traffic is processor intensive, while using a trace from real network traffic requires large disk space and is slow. While a life-like traffic model is useful to determine the performance of a scheduling algorithm under real traffic, using a simple uniform traffic model is useful to gain a more intuitive understanding of the algorithm. It also provides a method of comparing an algorithm's performance with others.

Another consideration is the modelling of the cell queues. A real switch has a finite number of buffers and perhaps only a small number of buffers dedicated to each connection. Other factors which may be considered are the use of early packet discard mechanisms which are usually implemented in real switches. The behaviour of traffic is also not immune to the condition of a switch; a connection experiencing very long switching delays may lower its transmission rate or find an alternative route through the network.

The simulation also needs to be carefully designed so that it is practical to implement in the time and hardware constraints available. A supercomputer would be the ideal simulation platform, but on smaller platforms approximations may have to be made in order to decrease the complexity of the simulation.

1.7 Conclusion

This chapter has provided a review of the current state of QoS in packet switched networks and given a very brief introduction to output buffered switch scheduling and input buffered switch scheduling. In a switch the quality of the service that is offered to the packets of a particular connection depends on several factors including the arrival pattern of the connections traffic, the number of buffers in the switch and the congestion information from downstream nodes. However the most important factor which affects the quality of service is the scheduling mechanism which determines which packet to transmit next on the output link.

In chapter 2, the architecture of ATM switches is investigated in further detail. This chapter provides the necessary background for scheduling in an ATM switch. In chapter 3 scheduling an output buffered switch is presented. This chapter provides an introduction to scheduling in output buffered switches it also lays some of the foundation for material presented in the next chapter. An introduction to Generalised Processor Sharing is presented. The topic of fairness is addressed and in particular the worst-case fairness and service fairness indices are introduced. Some simpler mechanisms are described, followed by some more complicated algorithms which attempt to make provision for per connection QoS. In chapter 4 the topic of scheduling in input queued ATM switches is presented. The development of scheduling algorithms of output buffered ATM switches is covered, and hence this chapter serves as a review of the current state of research into scheduling algorithms for input buffered ATM switches. The chapter starts with an investigation into the benefits of input buffered switches. Algorithms such as maximum size matching and maximum weighted matching are studied. Some of the input scheduling algorithms which have been presented in the literature very recently are reviewed. The methods used to evaluate the performance of input scheduling algorithms are derived from similar methods used in output buffered ATM switches.

In chapter 5, we present the idea of using a multiplying ratio to control the delay that a cell experiences in an input buffered switch. We also present a new algorithm for input buffered switches called *Worst-case Iterative Matching* (WIM). We highlight some of its predicted advantages, and we present a design for implementing it in hardware. In chapter 6, the results of some experiments to determine the performance of WIM scheduling algorithm for input buffered switches compared to other scheduling algorithms for input buffered switches. The chapter is concluded with an investigation of the effect of the switch size on the performance of scheduling algorithms for input buffered switches. In

chapter 7 we give a summary of the work which has been conducted, together with some conclusion. We also present some directions for future research.

Chapter 2 : Overview of ATM switch Architectures

2.1 Overview of ATM switch Architectures

The aim of this section is to provide an introduction to ATM switching. The aim is not to provide an extensive discussion on various switch architectures, but rather to describe general switching fundamentals and to provide a description of the various architectural options. The design of a switch fabric is a complex topic. For a more detailed introduction to the topic of ATM switch architectures see the paper by Turner [1]. Ahmadi and Denzel [2] provide an extensive survey of ATM switching, perhaps a little outdated but still largely relevant. A good review of commercial ATM switches is presented By Chao in [3].

The switching function is performed by a switching fabric. This fabric switches between N links usually of the same speed. A physical port is bi-directional and consists of an input and an output port on the same physical port. There are thus N input ports and N output ports. The input port is usually responsible for VPI/VCI translation. This operation is performed using a look-up table, possibly implemented using Content Addressable Memory (CAM). Other information which may be included in the look-up table is the output port, class of service and priority. The switch fabric routes from the input port to the desired output port and in the case of multicast two or more output ports. In all switches time can be divided into switching slots or the amount of time it takes to switch one cell. The design of the switch fabric has the most impact on the performance of an ATM switch.

There are many ways to categorize ATM switches, but the broadest is according to their structure.
Switch fabrics have been classified into 2 general categories in terms of their structure, namely space division and time division. A time division switch consists of a single dual ported memory block or a common high speed medium such as a parallel bus which is accessed by all input links and output links sequentially. The memory is usually partitioned per output link. A space division switch fabric uses many concurrent spatial paths from each input link to each output link`

ATM switches can also be classified according to their size. Smaller switches (4-16 ports) are usually made up of one switching stage, while larger switches have multiple switching stages. Most ATM switches today use one of several single stage switching techniques. Single stage switches are relatively simple, but of course are limited in the number of ports and also their total throughput. This section reviews the major categories of single and multi stage switches and discusses the key design issues.

2.2 Single stage ATM Switches

Most small ATM switches are of the shared memory or shared medium variety. The shared memory switch uses a common memory to transfer cells from the input ports to the output ports. Shared medium switches use a common medium like a bus or a ring.

2.2.1 Shared memory switches

In a shared memory switch cells from the input ports are multiplexed into a single stream and written sequentially into logically partitioned memory. The cells are then written to a particular section of memory depending on their destination. There are several issues affecting the performance of shared memory switches the main one being the structure of the shared memory, for example it can be a common memory for all ports which needs less memory, or it can be partitioned into separate sections for each port. It could also be a combination of both.

2.2.2 Shared medium (Bus)

Shared medium switches use a shard medium like a bus or a ring to sequentially send cells to the output ports. Buses are the most common form of single stage switches today. A simple single stage switch is illustrated in figure 2.1. See [1]. The input port processor handles the input processing functions at the ATM layer; this includes synchronizing arriving data to the internal timing of the switch, VPI/VCI translation, checking ATM header errors. To transfer data to the outputs, input port processes contend for access to the shared bus using one of a variety of bus contention techniques. Once an input port has possession of the bus it then transfers a cell or cells onto the bus in parallel form. The input port processor also transmits the output port number on the bus. The output port processors compare the number of the output port on the bus to their own address and buffer matching cells in a queue prior to transmission. The output port may implement a

simple FIFO queue, or several more sophisticated per virtual circuit queues. It may also implement one or more congestion control techniques to improve queuing performance in the presence of bursty data traffic.

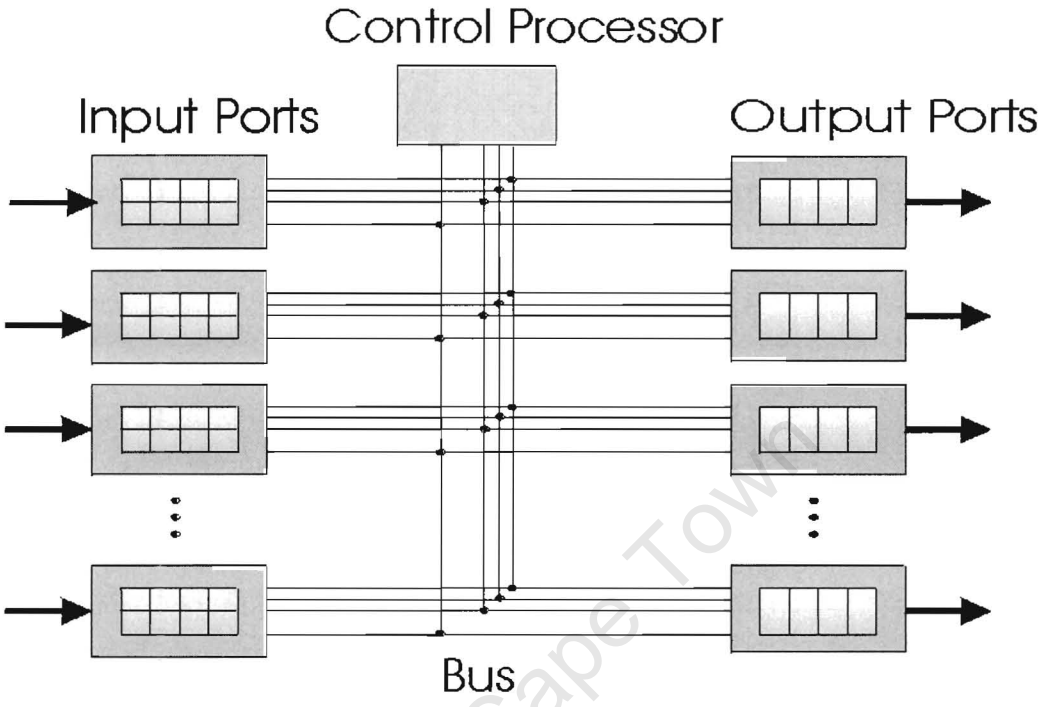

Figure 2.1 Bus Based Switch

In addition to the data path components described above, a switch must also have a control processor, which configures the input port routing tables in response to user requests. The control processor must also have control of the queuing system in the output port. The control processor is usually implemented as a general purpose processor with software to perform terminal-to-switch and switch-to-switch signalling, in addition to switch control and maintenance functions. In some systems the control processor is directly connected to the bus and often also has a separate bus connecting it to the memory implementation in the routing tables. This enables the control processor to modify the routing tables. The control processor may be connected to the switch through one of its external ports and communicate control information through the use of ATM cells carried on a specific VPI/VCI. This of course requires that the input and output ports have the ability to interpret and react to control cells. This solution becomes more attractive as the switch size increases as multiple processors may be required to maintain the switch. In order to provide non-blocking transfer medium, a bus supporting n ports, operating at a rate of R bits per second, must provide a bandwidth of at least Rn bits per second. If the clock

frequency used for the bus is r Hz, the bus width w must be at least $\frac{Rn}{r}$. Notice that as the number of ports in a system increases, both the number of ports connecting to the bus and the width of the bus must increase. This yields quadratic growth factors, making bus based switches unfeasible for larger systems. Another problem with bus based systems is the capacitive loading effect, as the number of ports connecting to a bus increases the capacitive loading of the signal lines increase, thus reducing the maximum frequency that can be used. The result of this is that the bus bandwidth must increase faster than the port count to maintain sufficient bandwidth.

Shared medium switches have the advantage that the technologies that are used to construct them are well developed and commercially available. For example Gigabit Ethernet is one technology that is well suited to building a shared medium switch.

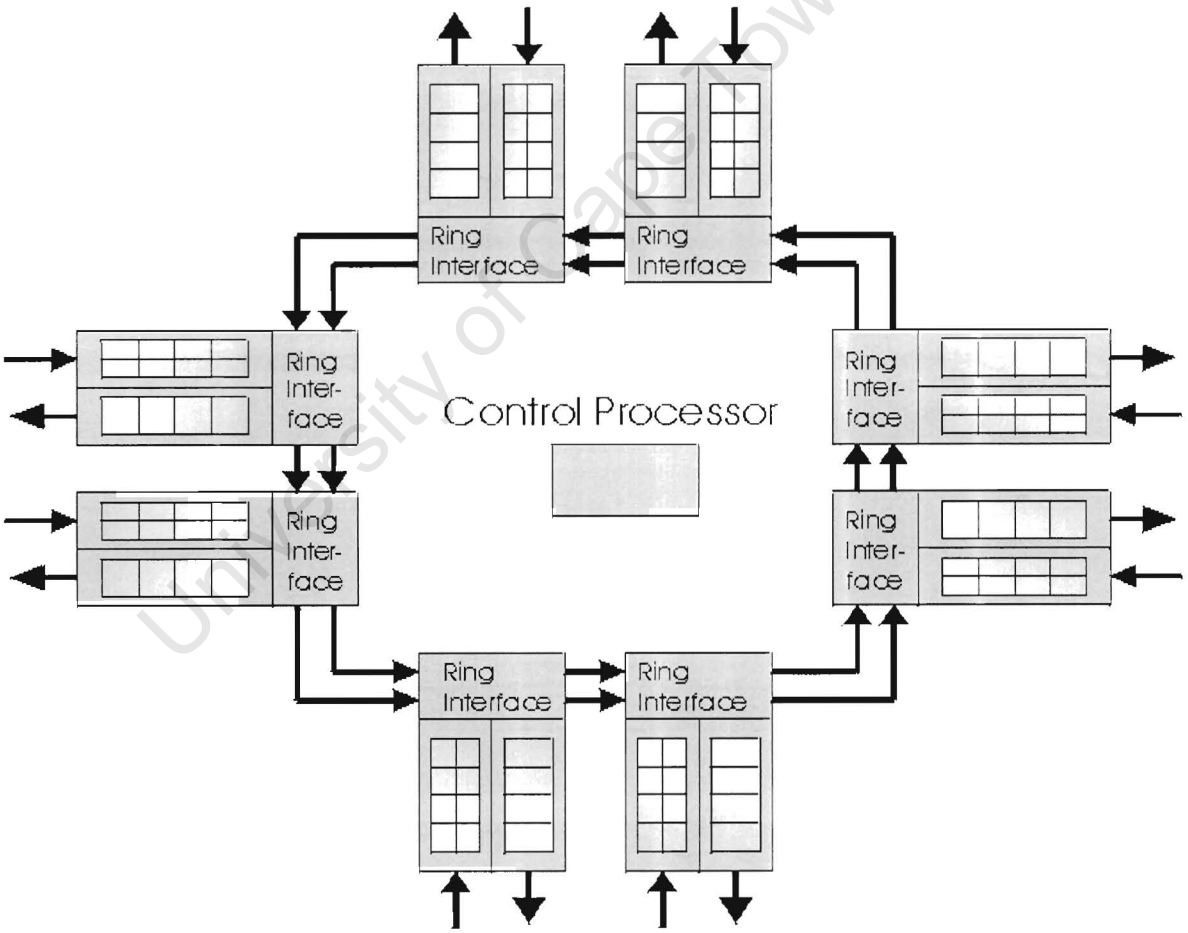

Figure 2.2 Ring Based Switch

2.2.3 Rings

This switch architecture replaces the bus in the previous section with a ring interconnect system. Each input port or output port interfaces with the ring, which is where cells are inserted or removed from the ring. The simplest ring protocol uses a time slotted approach where cells are sent synchronously during specified time slots, and a busy/idle bit is used to indicate the availability or unavailability of a time slot. An input port which has a cell to send waits for the start of a timeslot in which the busy/idle bit is clear and then changes the value of the bit and then transmits the cell on the timeslot. The ring rotates and the output ports compare the output number of the transmitted cell to their own address and copy the matching cells of the ring. They then change the value of the busy idle bit to indicate that the slot is again available for transmission.

Rings, like buses have quadratic scaling properties; however they are not hampered by capacitive loading effects. Rings do introduce some additional latency, relative to buses, but for switching applications these latencies are not that significant. If carefully designed they also may allow for more than one cell to be switched in each time slot.

2.3 Multistage Switches

Single stage switching techniques are inherently limited by their inability to scale to larger sizes. While the use of higher speed cores to handle lower speed ports provides some relief, for any given technology there comes a point where greater amounts of parallelism are needed to obtain higher throughput. For systems implemented using modern integrated circuits, multistage networks using identical small switching nodes are among the most attractive. An example of a multistage network is shown in Figure 2.2. Each of the switch elements is capable of simultaneously switching all of its inputs to all of its outputs (possibly using speedup). Each element may also have a small cell buffer.

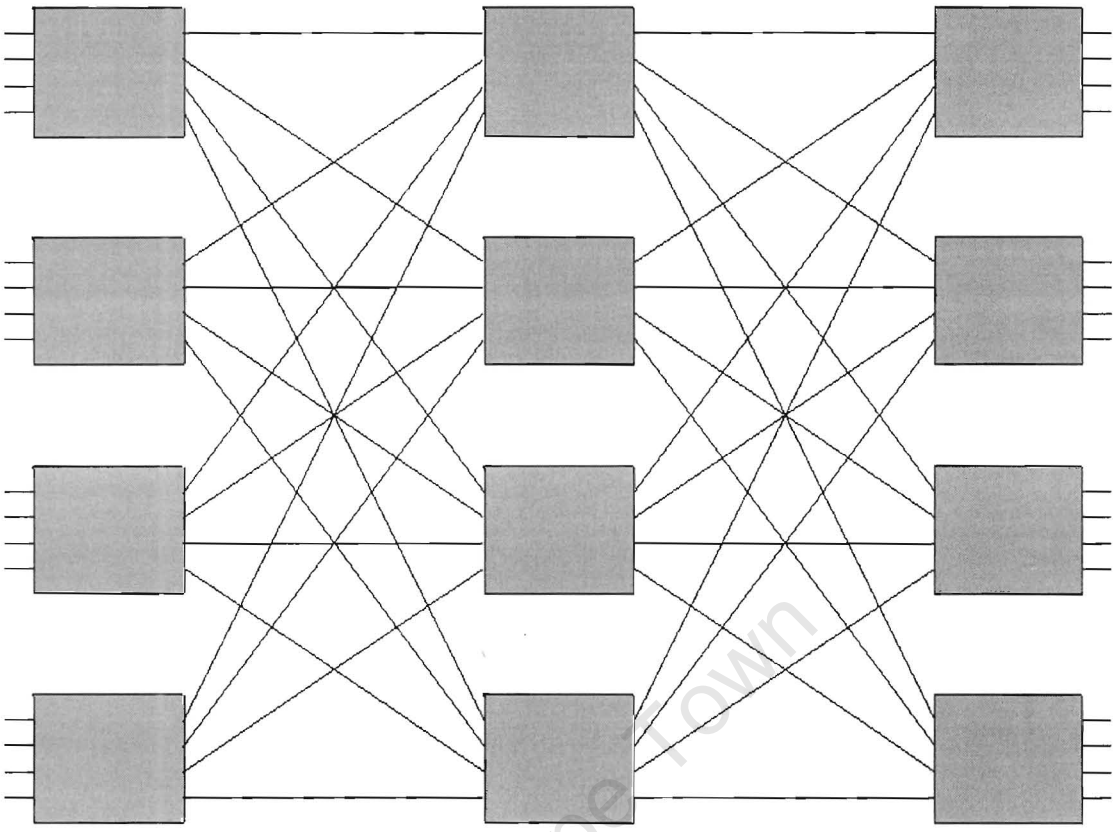

Figure 2.3 A Multistage Switch

The most convenient method of classifying space division (multistage) switches is based on whether there is more than one path between any two pairs of input and output ports. If only one path exists it is a single path switch, and if more than one path exists it is a multipath switch. The switching network illustrated in figure 2.3 is multipath.

In order for cells to get from the input ports to the correct output ports there needs to be some method of routing cells through the switch fabric. There are two main classifications of switches in terms of routing, those that use dynamic routing and those that use static routing. In systems that use dynamic routing, each cell is routed independently, spreading traffic as evenly as possible among all different paths between the pair of input and output ports. In static routing systems all cells in a given virtual circuit follow the same path. Static routing networks maintain cell ordering directly, but require explicit path selection and are subject to virtual circuit blocking. Virtual circuit blocking occurs when one cell can't be routed from its input port to its output port, because the resources are in use by another connection. Dynamic routing networks require the use of some other mechanism to restore cell order, after cells emerge from the interconnection network. This usually involves some form of re-sequencing buffer.

2.3.1 Performance of Multistage Switches

After routing, queuing in multistage interconnection networks is the second key performance issue for this class of systems. The design parameters that have the most influence on performance are the speed advantage, the dimensions of the switch elements, their queue organization and queue size, and the absence or presence of inter-stage flow control. In general, large switch elements with shared queues give the best performance for a given total amount of buffering. [2]

In dynamic routed multistage ATM switches only limited studies have been performed for time dependant traffic. For switches that use static routing, the queuing performance in the presence of time dependent traffic is somewhat more problematic. In these systems congestion can occur on any inter stage switch link in the interconnection network. The most effective way to solve this problem [1] is to use separate queues for each virtual circuit that passes through a switch element. In this approach a switch element with a congested output link signals to its upstream neighbours, blocking new cell transmissions for all virtual circuits using the congested link. These signals would be propagated backwards through the network to the input port where typically large buffers are located for the storage of bursts. Note that scheduling algorithms similar to those required for single stage switches are required in each stage of a static routed buffered multistage switch.

The switches considered in this text are static routed switches. The switch is capable of making as many connections as there are ports. For example in a 16 by 16 switch there can be 16 simultaneous connections all originating and destined for separate ports. The exact details of how the cells are switched are not relevant to the operation of the scheduling mechanism.

2.4 Queuing in ATM Switches

This section compares the different queueing options that a switch designer might use. Basically cells can be queued at the input or the output of a switch or both or not at all.

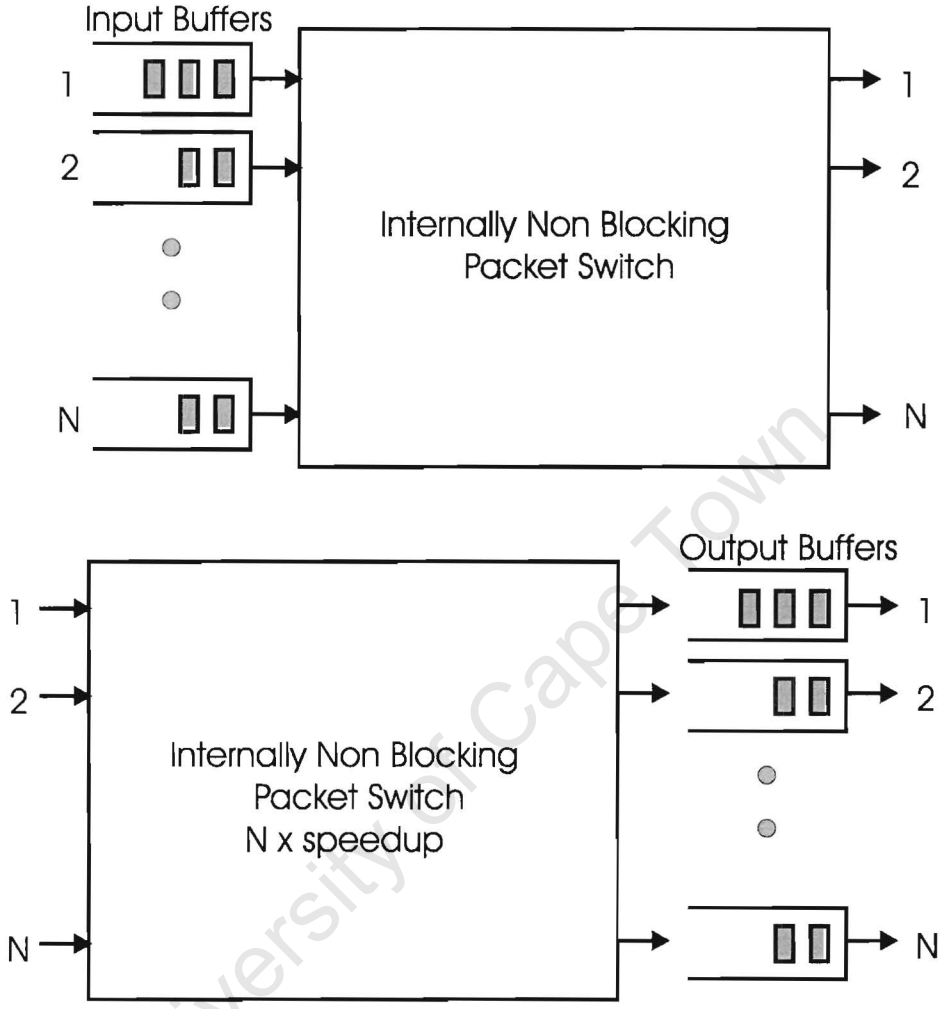

Figure 2.4 Input and Output Queueing

2.4.1 Output Queuing

In output queuing, when packets arrive at an input port they are transferred by a high speed switching fabric to the correct output port. The cells are then transferred into memory to await transmission on the output link. At the output port a scheduling mechanism then allocates the output bandwidth to the different connections according to a specific policy. If there are no input buffers the switch fabric has to be capable of transmitting as many cells as there are ports to any output in any switching interval. This absence of input buffers makes two very demanding requirements on the switch.

2.4.2 Input Queuing

In a switch with only input buffers, cells contend for access to the output port before they enter the switch fabric. Cells that are not selected are stored in the input buffers with cells that arrive in the next time slot. Again the switch fabric transfers at most one cell per unit time slot from each input port. Output buffers are obviously not needed in this case.

When a first in first out (FIFO) service discipline is used for the input buffers the maximum switch throughput is 0.586 of the total switch bandwidth. The maximum throughput of a non blocking switch with FIFO input buffers is thus lower than that of a switch fabric with no buffering. Note that the cell loss performance is still much better for an input buffered switch.

An input buffered switches using FIFO service discipline suffers from head of Line blocking. This means that cells at the head of the queue block which are blocked stop cells which could otherwise be routed to different output ports. A number of methods are available to increase the switching capacity of input buffered switches. One approach is to increase the speed of the switching fabric. Another is to use a non – FIFO input buffered service discipline. This topic is dealt with in more detail in subsequent chapters.

2.4.3 Combined Input and Output Queuing

The option of using both input and output buffering has recently received a lot of attention. This combines both the advantages and disadvantages of input and output queueing. The switch fabric is required to be faster than that required for input queueing, but does not need the speed required of a typical output buffered switch. This solution is usually referred to as Combined Input Output Queueing (CIOQ). However in larger switches even a small speed-up is not very practical.

For a switch fabric without any buffers, when output contention occurs, only one cell gets transmitted. The other cells are dropped or recycled into another input link. Assuming the cell switches at most one cell per time slot from each input, this sort of switch generally has a higher probability of dropping cells than the other solutions discussed here.

2.5 Buffer Management

The size and the way in which the cell buffers in an ATM switch are allocated in an output buffered switch have a large impact on the performance of the switch. Dedicated per VC queues are required to guarantee buffer space for non congested connections and traffic classes. Large buffers (of the order of several megabytes) are required to support bursty self similar traffic. Sharing a large central memory allows a switch to provide sufficient buffers for each port to handle large traffic bursts with a smaller total amount of memory. The alternative of providing several hundred megabytes on each port can be prohibitively expensive.

Usually the first step in providing buffer management on an ATM switch is to provide traffic isolation. Non-congested ports and traffic classes should always have available buffer space. The simplest way to achieve this is to assign dedicated queues to the five ATM forum management version 4.0 traffic classes (i.e. CBR, rt-VBR, nrt-VBR, ABR, and UBR). A Typical buffer allocation scheme is shown in figure 2.5.

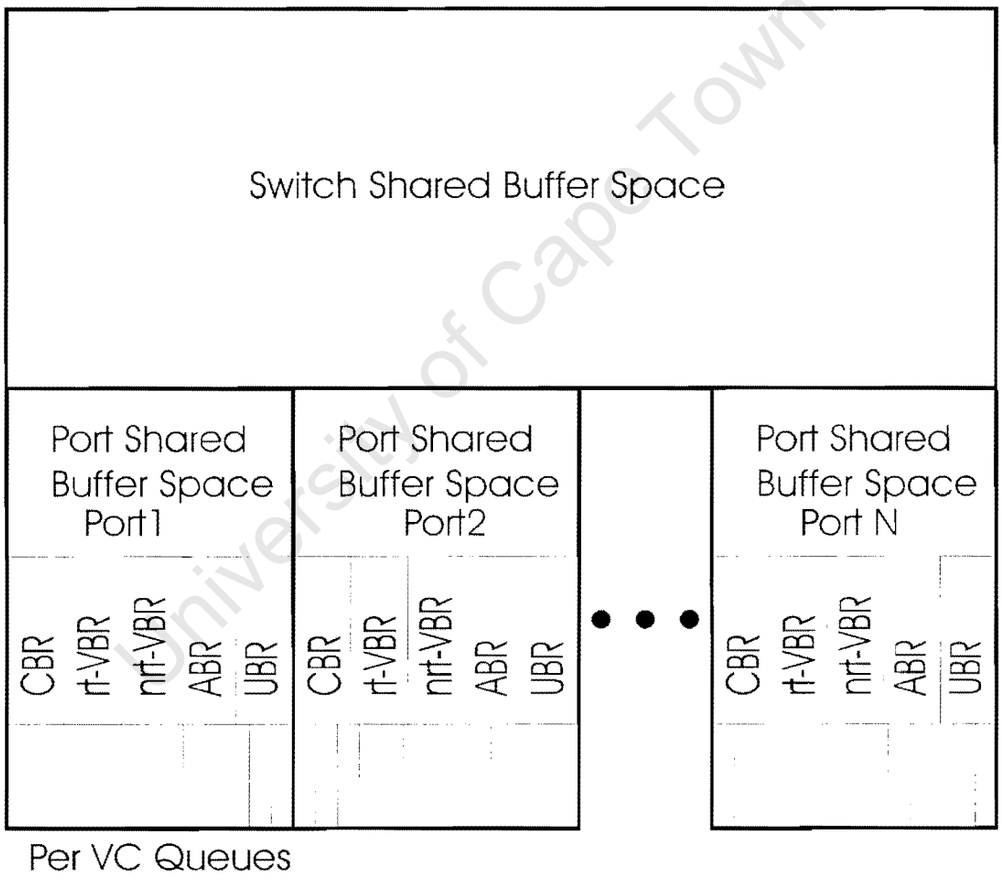

Figure 2.5 Buffer Allocation in an ATM switch

2.5 Buffer Management

The size and the way in which the cell buffers in an ATM switch are allocated in an output buffered switch have a large impact on the performance of the switch. Dedicated per VC queues are required to guarantee buffer space for non congested connections and traffic classes. Large buffers (of the order of several megabytes) are required to support bursty self similar traffic. Sharing a large central memory allows a switch to provide sufficient buffers for each port to handle large traffic bursts with a smaller total amount of memory. The alternative of providing several hundred megabytes on each port can be prohibitively expensive.

Usually the first step in providing buffer management on an ATM switch is to provide traffic isolation. Non-congested ports and traffic classes should always have available buffer space. The simplest way to achieve this is to assign dedicated queues to the five ATM forum management version 4.0 traffic classes (i.e. CBR, rt-VBR, nrt-VBR, ABR, and UBR). A Typical buffer allocation scheme is shown in figure 2.5.

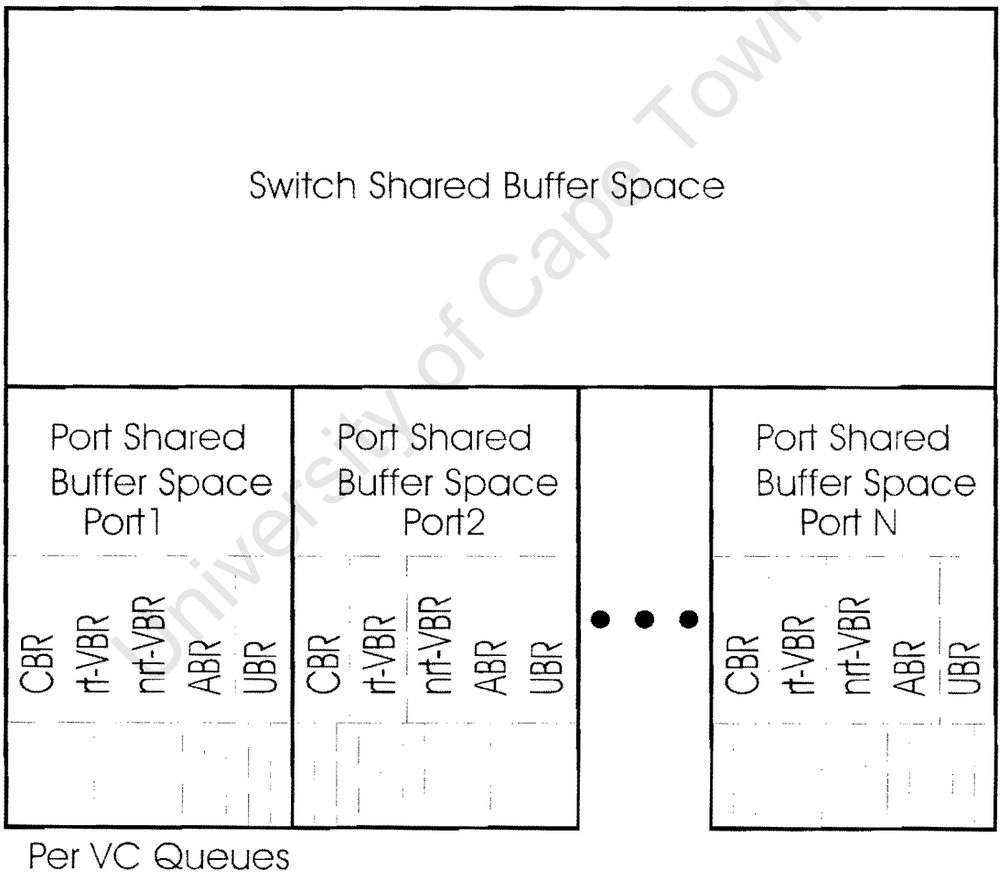

Figure 2.5 Buffer Allocation in an ATM switch

Switched cells are placed in an appropriate queue based on origin, destination and traffic class. If the per connection becomes congested the traffic will back up into the traffic class queue. If the dedicated traffic class queue fills up then the traffic backs up into the port shared buffer space. If the per port shared buffer fills up the traffic backs up into the shared switch memory.

2.6 Summary

As stated the aim of this dissertation is to focus on scheduling strategies for ATM switches. Although it is impossible to separate the scheduling mechanism from the switching fabric, we make certain assumptions on the architecture of the switching fabric. The switch fabrics considered all fall into the space division, internally non-blocking and internally blocking category. These fabrics provide a limited number of concurrent paths between the input and output ports and thus allow multiple cells to be simultaneously switched across the fabric.

In the case of output buffered switches it is assumed that all required cells can be switched, while in the case of input buffered switches only cells that are destined for separate outputs can be switched. This dissertation investigates some of the issues involved in the design of scheduling algorithms for this sort of switch.

Chapter 3 : Scheduling in Output Buffered Switches

3.1 Introduction

As scheduling for output buffed switches is generally a simpler topic than scheduling for input buffered switches we discuss this topic first. Some of the material presented here is necessary for the following chapter. Traditionally ATM switches have employed output queuing. They are conceptually simpler and provide a benchmark for the performance of input buffered switches which are examined later. In output queuing, when packets arrive at an input port they are transferred by a high speed switching fabric to the correct output port. At the output port a scheduling mechanism then allocates the output bandwidth to the different connections according to a specific policy. See figure 3.1. The function of the scheduling mechanism is thus to select the particular packet to be transmitted next on the output link.

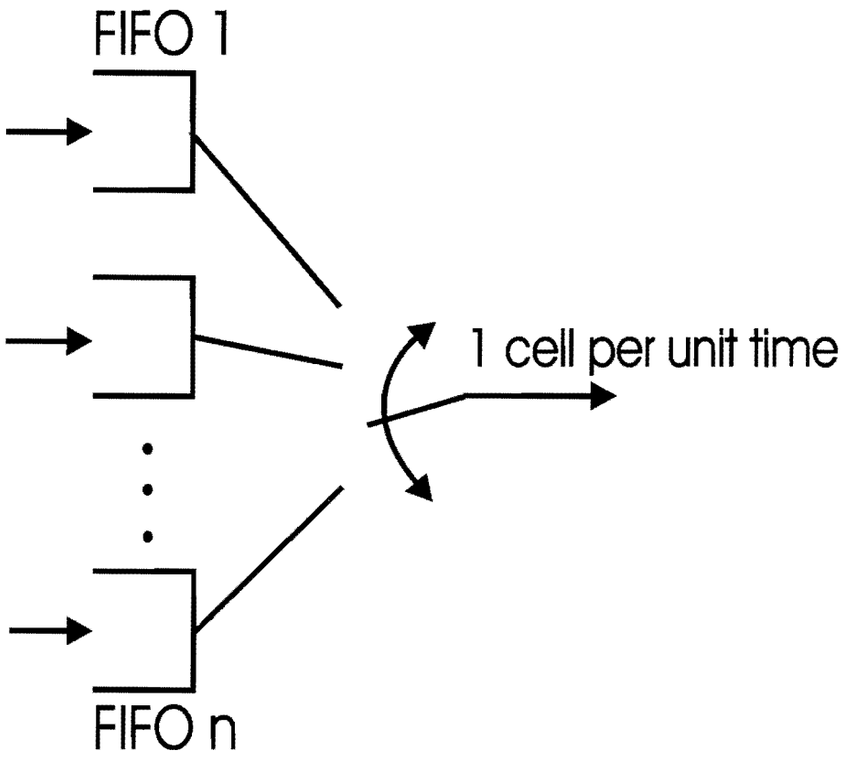

Figure 3.1 Scheduling in an Output Buffered ATM Switch

The main function of this output scheduling mechanism function is to provide the desired service to each connection. Each ATM connection has a set of ATM forum traffic management parameters, and these should be supported by the ATM switch.

3.2 Output Scheduling Algorithms

Many scheduling mechanisms for output buffered switches have been proposed in the literature in recent years, in the general context of connection orientated packet network architectures with explicit resource allocation and admission control policies. One of the first to studies to investigate fair scheduling was by Demers Keshav and Shenker in [4]. Many have since followed. This section aims to give a review of some of the algorithms proposed in the literature. The area of investigation will be restricted to rate based scheduling algorithms. A rate based service discipline is one that attempts to provide connections with a service rate, independent of the traffic characteristics of other clients [5]. Typically such a discipline manages the following resources at an output port: bandwidth, service priority, and buffer spaces. In conjunction with appropriate admission

policies, such disciplines allow clients to get performance guarantees in terms of throughput, delay, jitter and loss rate.

3.3 Generalized Processor Sharing

Generalised Processor Sharing (GPS) is the ideal theoretical scheduling algorithm, first proposed for use in broadband networks in [6]. Generalised Processor Sharing is defined with respect to a fluid model. In GPS each connection gets its fair share of the total available bandwidth at every instant. It is thus a fluid system and can not be exactly implemented in practice.

A GPS server serving N sessions is characterised by N positive real numbers, $\phi_1, \phi_2, ..., \phi_N$ which represent the bandwidth that each of the N connections will receive. The server operates at a fixed rate r and is work conserving (If there are packets available to switch it will always switch them) If $W_i(t_1, t_2)$ is the amount of traffic from session i served in the interval (t_1, t_2) then a GPS server is defined as one for which

$$j = 1,2,\Lambda\ ,N$$

$$\frac{W_i(t_1, t_2)}{W_j(t_1, t_2)} \geq \frac{\phi_i}{\phi_j} \tag{1}$$

holds for any session i that is backlogged throughout the interval $[t_1, t_2]$. From the definition, it follows that $B_{GPS}(\tau)$ the set of backlogged sessions at time τ remains unchanged during any time interval $[t_1, t_2]$. the service rate of session i during the interval will be exactly

$$r_i^*(t_1, t_2) = \frac{\phi_i}{\sum_{j \in GPS(t_1)} \phi_j} r \tag{2}$$

Where r is the output link speed. Since $B_{GPS}(t_1)$ is a subset of all the sessions on the server, it is easy to see that $r_i^*(t_1, t_2) \geq r_i$ holds where:

$$r_i = \frac{\phi_i}{\sum_{j=1} \phi_j} r \tag{3}$$

Therefore session i is guaranteed a minimum service rate of r_i during any interval when it is backlogged.

Of course realise that GPS is an idealised server that does not transmit packets as entities or even as bits. It assumes that the server can serve all backlogged sessions simultaneously and that the traffic is infinitely divisible. In packet system, only one session can receive service per timeslot.

In order to implement a GPS-like scheduling policy in practice many packet approximations have been proposed. These approximations generally involve the computation of a virtual finishing time for each packet and the computation of a system virtual time. The packets are then scheduled in increasing order of their virtual finishing times. These algorithms are usually referred to as Packet Fair Queuing (PFQ) algorithms.

3.4 Fairness in Output Buffered Switches

The service provided by a scheduling mechanism to a group of connections can differ significantly over the short term. The generally accepted definition of fairness is that we would like to serve connections in proportion to their reservations and distribute unused bandwidth among connections which are able to use it. (Not all connections can use excess unused bandwidth). However allowances also have to be made for connections which have no reserved bandwidth which only use excess bandwidth (UBR connections). Fairness can also be defined in terms of average delay jitter, some connections require this parameter to be very low, and it would be unfair to give one of two identical connections a low average delay jitter and the other one a much larger average jitter. Two indices for determining the performance of an ATM scheduling mechanism have been proposed in the Literature. These are the *Service Fairness Index* and the *Worst-case Fairness Index*.

3.4.1 The Service Fairness Index

The *Service Fairness Index(SFI)*, originally introduced by Golestani in [7] is the maximum difference of normalised service that two sessions i, j can receive during any time interval when they are continuously backlogged:

$$F_{i,j} = \max_{\substack{n_1<n_2 \\ i,j\in B(n_1,n_2)}} \left| \frac{W_i(n_1,n_2)}{r_i} - \frac{W_j(n_1,n_2)}{r_j} \right| \quad (4)$$

where $W(n_1,n_2)$ is the number of cells session transmitted by the server between timeslot n_1 and n_2, and $B(n_1,n_2)$ is the set of sessions that are continuously backlogged between n_1 and n_2. The SFI captures the difference in fairness of the scheduler from the ideal fairness of GPS in distributing service to connections that are continuously backlogged.

3.4.2 The Worst-case Fairness Index

The Worst-case Fairness Index, defined by Parekh in [8] and also used by Bennett and Zhang in [9], measures the maximum amount of time that a backlogged connection may have to wait between two consecutive services. In the case of ATM., the WFI can only be measured in cells and a Cell Worst Case Fairness Index (C-WFI) is thus defined. A scheduler is called worst-case-fair if the C-WFI for any session i is minimal and independent of the total number of allocated sessions. The distribution of service to competing connections in a scheduler with small C-WFI is closer to GPS than in a scheduler with identical SFI but larger C-WFI.

Parekh's definition of the Worst-case Fairness Index for output buffered switches is presented here for completeness and for comparison with the WFI for input buffered switches which is presented in the next chapter.

Definition 1 *A service discipline s is called worst case fair for session i, if for any time* τ, *the delay of a packet arriving at* τ *is bounded by* $\frac{1}{r_i}Q_{i,s}(\tau) + C_{i,s}$, *i.e.,*

$$d_{i,s}^k < a_i^k + \frac{Q_{i,s}(a_i^k)}{r_i} + C_{i,s} \quad (5)$$

where r_i *is the throughput guarantee to session i,* $Q_{i,s}(a_i^k)$ *is the queue size of session i at time* a_i^k *and* $C_{i,s}$ *is a constant independent of the other sessions sharing the multiplexer.* $d_{i,s}^k$ *is the departure time of the packet. A service discipline is called worst-case fair if it is worst case fair for all sessions.*

To perform a comparison between connections with different bandwidth reservations Bennett and Zhang define the Normalised Worst-case Fair Index for session i at server s to be:

$$c_{i,s} = \frac{r_i C_{i,s}}{r} \tag{6}$$

For a server that is worst-case fair, its Normalised Worst-case Index is defined to be:

$$c_{s,p} = \max_i \{c_{i,s}\} \tag{7}$$

GPS is worst-case fair with $c_{GPS} = 0$. Thus we can use the Worst-case Index to quantify the service discrepancy between GPS and a another scheduling algorithm, or indeed we can use it to compare any two scheduling algorithms

3.5 Practical Output Scheduling Algorithms

Although generalized processor sharing is the optimal scheduler, it is extremely complex if not impossible to implement in practice. Hence various approximations have been suggested and the following section reviews theses schedulers, which vary in their performance, complexity

3.5.1 Weighted Round Robin Scheduling

In round robin schedulers the time axis is split in frames of a certain maximum size. In each frame packets from different channels are serviced in a round robin manner. A credit counter is associated with each connection, and this counter is decreased each time a packet from a connection receives service. Credit counters are reset at the beginning of a frame to the maximum amount of traffic that the connection may transmit during a frame, and a connection is not eligible for service if its credit counter is zero. The simplicity of the algorithm allows a high speed hardware implementation.

The maximum delay a packet will encounter is proportional to the maximum frame size and the requirement for a fine bandwidth allocation results in a large frame size and hence a high end-to-end delay bound. Thus WRR makes a good scheduling algorithm for separating classes of traffic but it is not as good at scheduling individual connections.

3.5.2 Packet-by-packet Generalised Processor sharing Policy (PGPS)

PGPS was proposed in [4] by Demers, Keshav and is also called Weighted Fair Queueing (WFQ) and was the first proposed policy to approximate GPS for packet transmission. In this policy a GPS server is emulated to transmit packets: each packet receives a virtual service starting timestamp and a virtual service finishing timestamp. These represent the times that the packet would start and complete transmission through the server in a GPS system. Then the packets are scheduled in ascending order of their virtual finishing timestamps. A virtual time function (also called the system potential) is defined as

$$P^{PGPS}(t) = P(t_0) + \frac{r(t - t_0)}{\sum_{i \in B(t_0, t)} \phi_i} \tag{8}$$

Where $B(t_0, t)$ is the set of backlogged session in interval *[t0, t]*

The virtual starting and finishing timestamps (S_i^j, F_i^j) of the *i*th session's *j*th packet are computed as

$$S_i^j = \max(F_i^{j-1}, P(a_i^j)) \tag{9}$$

$$F_i^j = S_i^j + \frac{l_i^j \cdot r}{\phi_i} \tag{10}$$

where l_i^j is the size of the *j*th packet of session *i* and a_i^j is its arrival time. Notice that the rate at which the virtual time is updated every time the set $B(t_0, t)$ changes due to the ending or beginning of a session's transmission. PGPS requires a computation of the system potential which requires a real time simulation of GPS. This is the major drawback of the PGPS policy because the complexity of virtual time updating is in *O((N))* in an ATM network, where *N* is the number of connection. ATM networks would generally have a great number of sessions in parallel and the traffic is likely to be very bursty, so that the set of backlogged sessions is changing constantly. Therefore the total complexity of PGPS is linear with the number of sessions. It is not very difficult to see that PGPS is thus not a very practical scheduling algorithm to implement in practice. However PGPS does provide the best emulation of GPS. In [6] Parekh demonstrated that for any packet the delay between the real finishing time under PGPS, $(F^{PGPS}(p))$ and the virtual finishing time $F^{GPS}(p))$ of a packet of maximum size l^{max} is given by:

$$F^{PGPS}(p) - F^{GPS}(p) \leq \frac{l^{\max}}{r} \tag{11}$$

In a similar manner the inequality (11) provides an upper bound on the advance of the service under GPS compared to the service under PGPS, the PGPS server may be far ahead of the corresponding GPS server. This advance in packet based system is not bounded and may cause discrepancy in the output flows of some sessions, which results in a large delay jitter. An example illustrating this problem was presented by Bennett and Zhang in [9] where they also proposed a solution. A summary of their work (Worst-case Fair Queueing) is presented in a following section.

3.5.3 Self-Clocked Fair Queueing Scheduling (SCFQ)

SCFQ was proposed by Golestani in [7] in order to improve the virtual time computing complexity of the PGPS policy. A SCFQ server schedules the packet having the minimum virtual finishing time in the same way as PGPS. The virtual timestamps are computed according to the same formulae but the virtual time is computed differently. In SCFQ the virtual time is determined only using information from the real system (hence the name SCFQ), without referring to a corresponding virtual time obtained from a simulation.

$P^{SCFQ}(t)$ = *the virtual finishing timestamp of the packet in service at time t*

Hence the complexity of implementing this algorithm consists of finding the minimal virtual finishing timestamp, which is done in *O(logN).* Obviously this simplification affects the quality of the service of the scheduler; it introduces some unfairness, which does however remain bounded. It has also been shown that the latency of a session depends on the number of sessions backlogged in the system. Furthermore this policy also suffers from a poor WFI in the same way as PGPS does.

3.5.4 Start Time Fair Queuing Policy (SFQ)

The SFQ policy was proposed by Goyal et al [10] in order to offer a scheduling mechanism better adapted to low bandwidth flows. They modified the definition of the virtual time and the packet selection criterion. The virtual time of SFQ is given by:

PSFQ(t) = *the virtual start time of packet* ***in service*** *at time t.*

The packets are scheduled according to their Virtual starting timestamps instead of their virtual finishing timestamps, which is the main reason why the SFQ policy favours low bandwidth sessions. Such low bandwidth sessions have smaller bandwidth parameters, so their virtual finishing times are greater. Thus under scheduling algorithms using virtual finishing times, packets of low bandwidth connections must wait for the transmission of packets belonging to sessions with larger bandwidth reservations (i.e. smaller virtual finishing timestamps), even if they started transmission previously. This property is desirable for low bandwidth CBR connections which would normally carry delay sensitive voice. Intuitively giving the priority to the low bandwidth sessions will not significantly damage the fairness of the scheduler since these sessions are supposed to emit a smaller number of packets.

3.5.5 Discrete Rate Schedulers

The discrete rate approach was first introduced by Bennett et al in [11]. The discrete rate scheduler is based on the assumption that it is required to support a relatively small number of guaranteed service rates at any time. This assumption is certainly realistic in most if not all ATM switches. In the discrete rate scheduler, backlogged sessions with the same service rate are grouped together in the same rate FIFO queue. When a connection becomes backlogged after being idle, it is assigned a new timestamp and appended at the tail of the corresponding rate FIFO queue. At any time scheduling is only performed among sessions at the head of rate FIFO queues at that time. Since only the timestamps at the head of the queues need to be searched to find the minimum virtual finishing time, the number of timestamps to be searched is greatly reduced (equal to the number of rate FIFO queues). The session selected by the scheduler is extracted from the head of its rate FIFO queue at that time and if still backlogged assigned a new timestamp and queued back at the tail of the rate FIFO queue.

For a discrete rate scheduler to have delay bounds and fairness indices which are independent of the number of connections or the source behaviour, the discrete rate implementation of a scheduler must satisfy the two conditions originally defined in [11], the *globally bounded timestamp* and *locally bounded timestamp* properties

For any pair of connections having i,j having the same allocated service rate ($r_i = r_j$), the following condition holds:

$$\left|F_i - F_j\right| \leq \frac{1}{r_i} \tag{12}$$

The locally bounded timestamp (LBT) property proves that the difference between two consecutive timestamps of a continuously backlogged connection *i* is always equal to $\frac{1}{r_i}$, and therefore that the contribution of the source model to the delay bounds is independent of the scheduler considered. However the LBT property does not enforce a fixed value of latency, which still depends on the scheduler and on its specific implementation

The globally bounded timestamp property is defined as follows:
At any timeslot *m* the following relation holds between the timestamp F_i of any backlogged connection *i* and the system potential *P*:

$$P(m) \leq F_i \leq P(m) + \frac{C}{r_i} \tag{13}$$

Where *C* is a constant value shared by all the connections.

For a discrete rate scheduler the combination of the GBT and LBT properties with the minimum slope property determines the following bounds for the Service Fairness Index [11]:

$$Fi, j \leq \max\left(\frac{C}{r_i} + \frac{1}{r_j}, \frac{C}{r_j} + \frac{1}{r_i}\right) \tag{14}$$

The Cell Worst-case Fairness Index is [11]:

$$CWFI = C - 1 \tag{15}$$

The discrete rate scheduler with per connection timestamps represents an important improvement in reducing the complexity of GPS related schedulers while maintaining near optimal delay bounds and SFI, as well as maintaining a minimal CWFI.

3.5.6 Worst Case Fair Weighted Fair Queuing (WF2Q)

In order to attain as fair a scheduling mechanism as possible, Fair queuing algorithms generally try to approximate what is happening in GPS as closely as possible. The main difference between the fluid GPS system and a packet system is that at any time there can be multiple packets being serviced simultaneously in GPS while there can only be one

packet serviced in the packet system. In fact in a fluid system every connection has at least one packet being serviced. While the service time of a packet with L bits in a packet system is $\frac{L}{r}$ it can be much longer in the GPS system depending on the guaranteed fraction of bandwidth for the connection. Therefore even though a packet may start transmission later in a packet scheduler, it will still finish transmission before it would in GPS. Then if a second packet starts transmission in the packet scheduler before the first packet finishes transmission in GPS we can reach a situation where the difference between the fluid and the packet system in the short term can be quite large. In the long run the service that each session gets is the same amount of service as in GPS. To avoid this discrepancy, in the WF^2Q scheduler the next packet to transmit is selected only from those packets that would have started transmission in the corresponding GPS scheduler. It has been shown that WF^2Q differs by no more than one packet in providing identical service to GPS and is thus considered the optimal output scheduling algorithm.

3.5.7 Single Bit Discrete Rate Schedulers

This scheduler was proposed by Chiussi in [12] and represents an important step in that it is realistically practical to implement the single-bit-timestamp scheduler is very similar to the discrete rate scheduler with per connection timestamp. There are a few simplifications. The scheduler only supports only a fixed number of rates at any given time. Connections with the same service rate are queued in the same rate FIFO. When a connection becomes backlogged, it is queued at the tail of the corresponding rate queue. The single bit timestamp differs from the discrete rate scheduler with per connection timestamps in that it does not use per connection timestamps, but only maintains one timestamp and one bit per rate queue, and a single bit per connection. A set of rules govern the relations ship between per connection and per rate bits. This scheduler effectively reduces the complexity of implementation of worst case fair GPS – related schedulers while maintaining near optimal delay and fairness properties.

3.6 Other Work

A large amount of work has been done in the area of scheduling algorithms for output buffered switches. Perhaps one of the most popular scheduler compared to GPS- related schedulers are those called Earliest Deadline First (EDF) schedulers, first proposed by Ferrari and Varma in [13][14]. EDF associates a per-hop deadline with each packet, and

schedules packets in order of deadlines. Using traffic shapers at every node in the network in conjunction with a variant of EDF called rate controlled EDF or RC-EDF, it has been shown that exact delay bounds can be guaranteed. There are some problems in that RC-EDF does not necessarily fully utilize the bandwidth available [15].

3.7 Summary

This chapter has examined the development of scheduling algorithms for output buffered ATM switches. This chapter has presented algorithms for output buffered ATM Switches which have the ability to control the delay of a cell through the switch accurately. It has also presented algorithms which are in fact practical to implement and which still maintain per connection delay guarantees.

These are desirable properties for a switch to have. However output buffered switches still have the significant drawback that the internal switching fabric and the output queues must operate at N times the line speed, where N is the number of ports. This limits the use of output queueing to smaller switches, and led to further interest in scheduling algorithms for input queued switches which are investigated in the next chapter. However many of the ideas presented in this chapter are again applied to input queueing in the form of virtual output queueing which is described in the next chapter.

Chapter 4 : Scheduling in Input Buffered ATM Switches

4.1 Traffic Scheduling in Input Buffered Switches

Although scheduling algorithms for output buffered switches are currently able to offer guaranteed bandwidth protection to individual connections, output buffered switches are not practical for all but the smallest switches. The switch fabric and the buffers at each output port are required to operate at N (where N is the number of ports) times the line transmission speed. In the case of output buffered switches there needs to be some sort of provision for this "speedup" By speedup we refer to the ratio of the fabric speed to the input/output speed. Achieving this speedup is usually very costly in all but the smallest switches, and is simply impossible in larger switches.

To overcome the scalability problems associated with output queuing, input queuing is being widely investigated. One of the earlier studies of input queuing was presented in [16], where it is compared with output buffering. In the input queuing paradigm incoming cells are stored at the input before being transferred across the switching fabric to the output side where they are transmitted immediately or perhaps queued for resource management. The decision of which cells are transmitted across the switch fabric is made by an input scheduling algorithm. One of the main advantages of input queuing is that no speedup is required. This makes scaling the switch to larger sizes feasible, in terms of current technologies and cost.

In an input buffered switch cells are buffered at the input ports of the switch. A set of paths must be established in the switch fabric for transmitting packets from the input ports to the output ports of the switch. The scheduling problem can be defined as a two stage problem. A path through the switch fabric must first be established to transmit packets from the input port to the required output port. Another form of scheduling that has to take place in input buffered switches is to select which packet to select among those waiting at the input port for transmission. Note that output buffers are not needed; as soon as a cell is switched it is transmitted.

In a non-blocking switch architecture, connection requests must still contend with other requests, if two separate packets arrive at separate input ports, but are both destined for the same output port, then still only one of them can be transmitted on the output link at the next timeslot while one will have to be buffered. This is commonly known as output port contention and is one of the reasons why output buffered switches require large output buffers. In input buffering packets are buffered at the input side until they are selected for switching to the output port. This can create a problem with this sort of buffering in that it suffers from what is known as Head of line Blocking. In Head of Line Blocking a packet can not be transmitted even if its destination output port is capable of transmitting it because of output contention at the head of the input queue. Head of line blocking has been shown to have very restrictive effect on the performance of input buffered switches, limiting the throughput of an input buffered switch to 58% of its capacity when the traffic is uniformly distributed.

The performance of an input buffered switch can be improved by negating this head of line blocking effect by allowing the switch to select among more than just the first cell in the queue. The number of cells in an input queue among which a cell can be transmitted is called the window. Even a small window has been shown to improve the performance of an input buffered switch compared to FIFO queuing. When more than one cell is accessible in each input buffer, it becomes possible to maximize the throughput by selecting the packets in the input ports for transmission to output ports such that the largest possible number of packets is scheduled for transmission in every cycle. Head of Line Blocking (HOLB) can be eliminated entirely using a queuing technique known as virtual output queuing (VOQ) in which each input maintains a separate queue for each output. It has been shown that with a suitable centralized scheduling algorithm the throughput can increased from 58.6% to 100% [17]

Overall there has been much interest recently in input queueing for packet switches. A good introduction and review of the subject is given by Marsan et al in [18] and [19]. Other studies worthy of mention are by Gupta[20] and Anderson [21]

There has also been quite an interest in combined input output queuing (CIOQ). various studies have been done such as in [22], [23], [24], [25]. [26] and [27] however the focus is restricted to input queueing in this chapter.

4.2 Virtual Output Queuing

The idea of Virtual Output Queuing was developed by Mckeown et al in [28]. This scheme effectively circumvents the effects of head of line blocking by maintaining per connection queues for every output port at each input. See figure 4.1 has proved very successful and a number of studies have built on this the scheme. A practical method of implementing virtual output queues in hardware is proposed in [12].

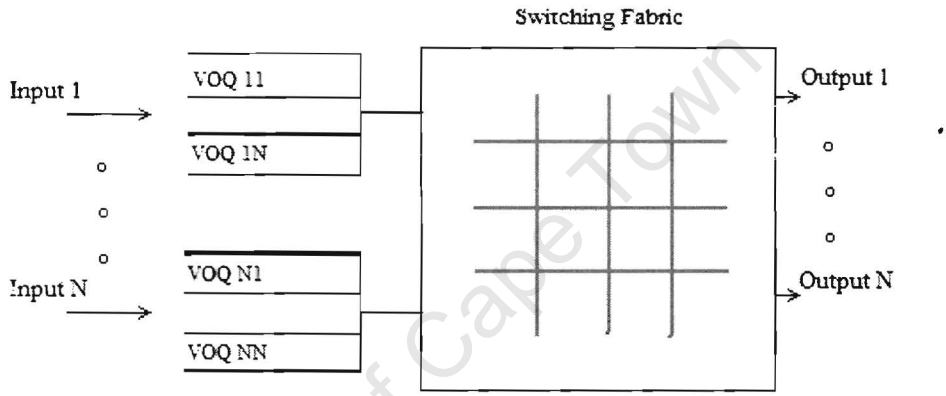

Figure 4.1 Virtual Output Queueing

4.3 The Matching Problem

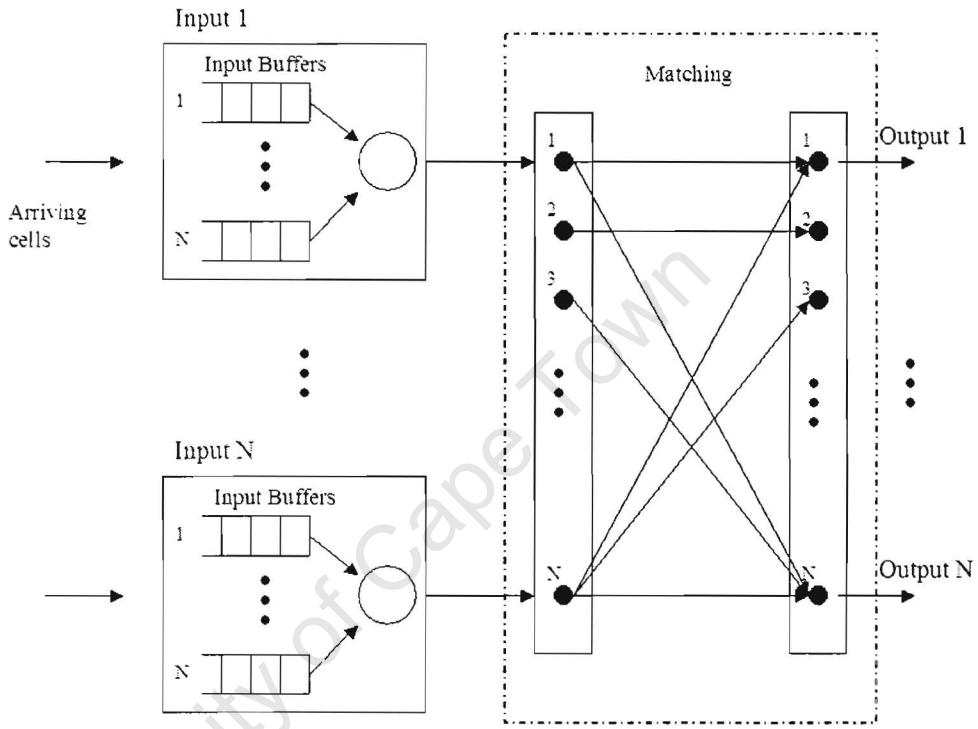

Figure 4.2 The Matching Problem

The problem of switching the maximum number of packets in a crossbar with windowed or other access method queues is reduced to a matching problem. See figure 4.2. Every input port is represented by a circle in the left hand column of the matching section and every output port by a circle in the right hand column. Each available packet is represented by a line between the packets input port and its destination output port. Not all the input and output ports are shown. The matching problem then amounts to finding the set of packets for transmission that maximize the number of lines from input port to output ports, remembering that there can only be one line from every input port or to every output port.

Various techniques have been described to optimally solve this matching problem and these will be described in some detail in the following sections.

The problem of switching the maximum number of packets in an input based switch can be reduced to the matching problem in a bipartite graph. The bipartite graph is constructed by representing each input port with a vertex in a first set and every output port as a vertex in the second set. Each backlogged connection is represented by an edge between the originating input port and the destination output port. The bipartite matching problem tries to maximize the number of vertices in the first set connected to the second group by selecting a set of edges such that no two edges have a common vertex. This constraint of no more than one edge connecting to a vertex is due to the packet nature of ATM (only one cell can be transmitted in each cell interval).

In abstract terms a switch is usually represented as a bipartite graph G= (U, V, E) where U are the input nodes, V are the output ports and the edges E represent possible transmissions. The constraints of the ATM switch specify that a set of transmissions can occur at the same time only if it corresponds to a matching - a subset of edges such that each edge has at most one connecting edge. Some scheduling algorithms associate a weight w (e) to each edge.

Input queuing is attractive for very high bandwidth ATM switches because no portion of the internal data path has to run faster than the line rate. The longstanding was that that input queued switches was impractical due to the head of line blocking problem, and the standard approach was to use output queuing. Today however input queuing is the more popular option. By increasing the internal bandwidth of the switch fabric multiple cells can be forwarded in the same timeslot to an output port, and queued there for transmission on the output link.

One problem with input scheduling algorithms is that of providing each connection with its fair share of bandwidth. Most input scheduling algorithms are not very good at doing this. We will examine several solutions in some detail, excluding FIFO queuing which has already been dealt with sufficient detail.

4.4 Maximum Size Matching (MSM).

Several algorithms have been proposed for achieving a maximum size match. By this we mean that the algorithm attempts to maximize the total number of connections in each cell time. Maximum size matching algorithms have been shown to perform well when the arriving traffic is uniformly distributed over all the switch outputs.

If a new packet or connection cannot be added to a set of matches without altering the current assignments, then a *maximal* match has been found. In this case all packets are either scheduled or blocked. If the number of packets scheduled is the maximum possible among all possible matching sets, then the matching is a *maximum* matching. A *maximum* matching is always *maximal*, but a *maximal* matching is not necessarily *maximum*.

Unfortunately maximum size matching performs poorly for non-uniform traffic in a number of ways. When traffic is non-uniform, the occupancies of the various input queues can differ greatly, this coupled with the inability of MSM to consider queue length when allocating bandwidth can cause queues with heavy traffic to overflow. MSM also has a reduced total throughput under non-uniform traffic compared to uniform traffic see [29] for an explanation of this. MSM also suffers from the starvation problem. The result of this is that a connection can have a large maximum delay. Referring to Figure 4.3., we see that even a simple traffic pattern can lead to traffic starvation for certain inputs when using a maximum size matching algorithm.

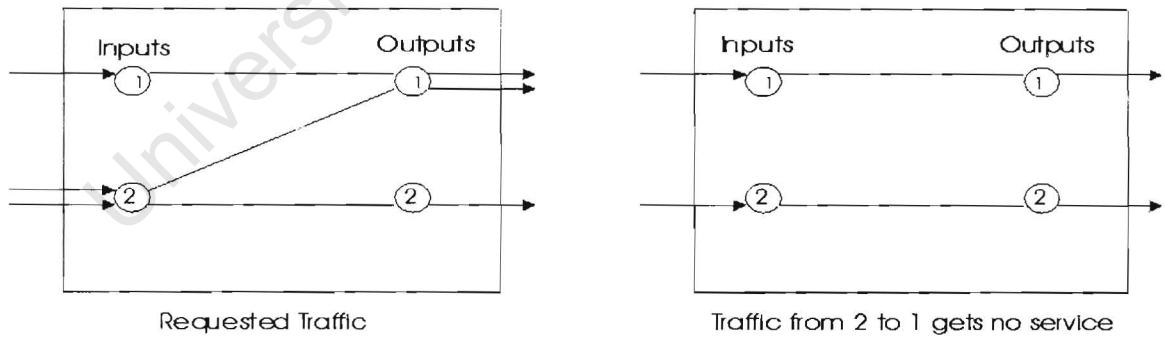

Figure 4.3 Service denial using a Maximum Size Matching Algorithm

4.5 Maximum Weighted Matching (MWM).

Maximum weighted matching attempts to perform a match but instead of just finding the maximum size match, a weight is assigned to each individual packet and the algorithm

4.5 Maximum Weighted Matching (MWM).

Maximum weighted matching attempts to perform a match but instead of just finding the maximum size match, a weight is assigned to each individual packet and the algorithm attempts to maximize the total weight. Thus it can be seen that this can provide for dynamic priority scheduling. Maximum weighted matching operates in a similar method to Maximum Size Matching except that edges are assigned weights. The maximum weighted matching for a bipartite graph is one that maximizes the sum of the edge weights and is found by solving an equivalent network flow problem.

As in MSM, if a new packet or connection cannot be added to a set of matches without altering the current assignments, then a *maximal* weighted match has been found. In this case all packets are either scheduled or blocked. If the summed weighted number of packets scheduled is the maximum possible among all possible matching sets, then the matching is a *maximum* weighted matching. Again a *maximum* matching is always *maximal*, but a *maximal* matching is not necessarily a *maximum* matching.

The complexity of the best maximum weight matching algorithms is $O(N^3)$, thus this sort of scheduling gets very complex for even small switches. Edmonds and Karp [31] 'algorithm for maximum matching on graphs' is the basis for most other algorithms solving maximum weighted matching. Gabow's "Efficient Implementation of Edmonds Algorithm for Maximum Matching on Graphs" [47] was one of the first implementations and has a running time of O (N^3). Various approximations to Edmonds Algorithm have since been proposed.

The NxN switch has connections which are represented in an NxN matrix whose elements are the edge metrics in graph $G=[V,E]$, called the weight matrix W. The elements are denoted w_{ij}, and w_{ij} is assumed to be zero when no cells at input i are destined to output j.

The same scheduling algorithms used in output buffered switches are used to provide the weights used in maximum weighted matching. MWM was tested with weights according to the current queue length, called largest queue first and to the current packet delay, oldest packet first in [29]. It can be easily seen that weights used in maximum weighted matching can be calculated in an arbitrary way.

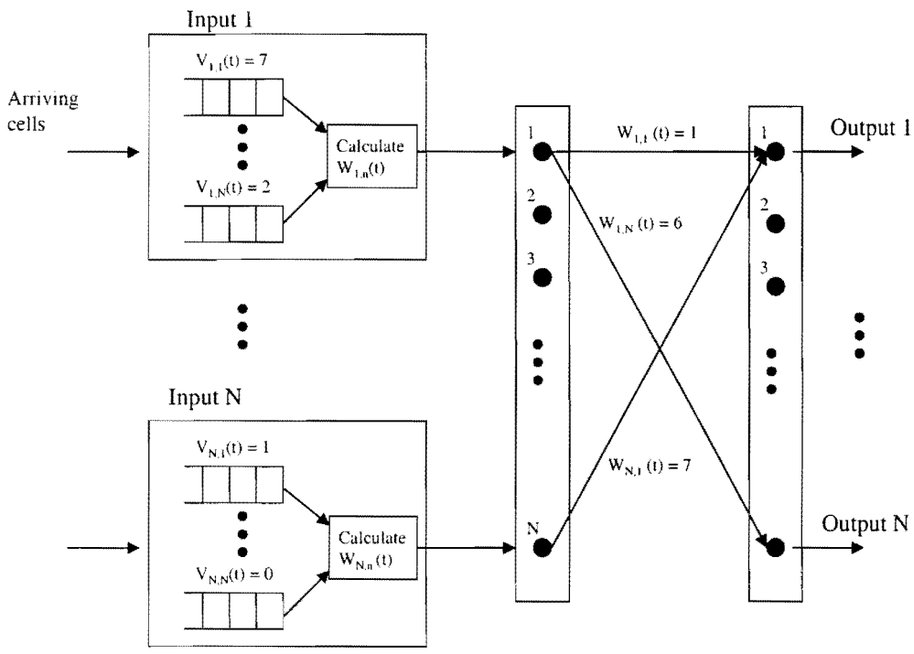

Figure 4.4 Example of Weights for Maximum Weighted Matching Algorithm.

There are variations on what to use as the edge weights. One choice proposed in [29] is to use the queue length L(n), to thus give preference to queues with greater occupancy. This is called Longest Queue First (LQF) scheduling. It has been proved that LQF provides a maximum throughput of 100% for independent and either uniform or non-uniform arrivals. [29]. In **Error! Reference source not found.**4 we show the operation of MWM using the age of the timestamp at the head of the queue. One problem with MWM is that the algorithms for solving it are too expensive in terms of running time. Faster algorithms are still required. Various approximation algorithms have been investigated, a good review of some of these is given in [30].

As noted the algorithms which attempt to mimic MWM are too complex to implement in hardware and are therefore unsuitable for switches operating at the speeds that ATM requires they operate at. Instead most switches use a much simpler scheduling algorithm to configure the switch fabric.

4.6 Iterative Scheduling Algorithms

The algorithms that are discussed in this section differ from those in the previous section in that they are more or less feasible to implement in hardware and in some cases have actually been implemented in prototypes or even commercial products [45], [32], [33] and [34]. These algorithms generally try to find some sort of maximal weighted match, but with less complexity than is required for a maximum weighted match. In general the algorithms don't achieve the optimal match that MWM achieves, but that is to be expected. Most of these algorithms are iterative in their operation. These algorithms generally only find a maximal match, which means that once they make a pairing between an input and an output they will not break that pairing.

4.6.1 Parallel Iterative Matching

Parallel Iterative Matching (PIM) was one of the first algorithms to use an iterative approach to find a maximal match and was proposed by Anderson, Owicki, Saxe and Thacker in [21]. PIM attempts to quickly converge on a conflict-free match using an iterative method involving three steps. All inputs and outputs are initially unmatched and only those inputs and outputs not matched at the end of each iteration are eligible for matching in the next loop. The three steps of each loop operate in parallel on each input and output and are as follows:

1. Each unmatched input sends a request to every output for which it has a queued cell.
2. If an unmatched output receives any requests, it grants to one by randomly selecting a request uniformly over all requests.
3. When an input receives a grant it accepts one by selecting an output among on of those that granted

This algorithm has the following properties: As the authors show in [21] this algorithm converges in O(log(N) iterations. It also ensure that all requests will eventually get service. This algorithm has the following problems. Firstly it uses randomness which is difficult and expensive to implement at high speed in hardware, and secondly it suffers from being unfair as shown in [20].

4.6.2 Weighted Probabilistic Iterative Matching

Weighted Probabilistic Iterative matching which was proposed by Stiliadis in [35] was the first input scheduling algorithm which attempted to provide some sort of bandwidth protection. It does this by breaking the scheduling problem into two levels:

1. Connection Level Scheduling: Each output port of the switch selects the input port to service in accordance with the connection-level bandwidth reservations.
2. Flow-level scheduling: Each input port selects a packet from one of the incoming flows sharing the same input-output connection based on its total share of bandwidth.

In Weighted Probabilistic Iterative Matching we assume that the switch maintains a separate queue for packets destined to each output of the switch. Within each queue, the packets are ordered by a flow level scheduling algorithm used to allocate bandwidth to connections sharing the same output destination. Every output port must also maintain a count of the packets serviced during the current frame from each input. Each repetition of the WPIM algorithm consists of four stages:

1. Every input port of the switch that has not yet been matched with one of the output ports sends a request to every output corresponding to destinations for packets in its queues
2. On receiving the requests from the input ports, each output port creates a mask consisting of one bit per request as follows: For those inputs that have transmitted at least as many packets as their credit to the output port in the current frame the mask bit is set to one. For others the mask is set to 0. Among the requests received by the output port, only those originating at unmasked input ports are used in the matching process and the rest are ignored.
3. From the requests that remain from the masking phase, the output selects one randomly with uniform probability and sends a grant signal to its originating input port.
4. Every unmatched input port that receives one or more grants selects one with equal probability and notifies the corresponding output port. The input and output ports are now matched and can be removed from subsequent iterations.

The above cycle is repeated for a fixed number of times or until no more pending requests remain.

4.6.3 Iterative Round Robin Matching with Slip

Iterative Round Robin Matching with Slip or iSLIP, like PIM is an iterative algorithm. It was described by Mckeown in [29]. The three steps which are iterated in iSLIP as described in [29] are presented below. In iSLIP each input and output port maintains a set of registers with a pointer to the current port which should be given highest priority.

1. *Request:* Each unmatched input sends a request to every output for which it has a queued cell.
2. *Grant* If an unmatched output receives any requests it chooses the one that appears next in a fixed round robin schedule starting from the highest priority element. The output notifies each input whether or not its request was granted. The pointer to the highest priority element of the round robin scheduler is incremented (modulo N) to one location beyond the granted input if and only if the grant is accepted in step 3 of the first iteration. The pointer is not incremented in subsequent iterations
3. *Accept* If an unmatched input receives one or more grants it accepts the one that appears next in a fixed, round robin schedule starting from the highest priority element. The pointer to the highest priority element of the round robin schedule is incremented (modulo *N*) to one location beyond the accepted output only if this input was matched in the first iteration.

So each output arbiter decides among the set of ordered, competing requests using a rotating priority. When a requesting input is granted and the input accepts that grant, the input will have the lowest priority at that output the next cell interval.

4.6.4 Iterative Fair Scheduling

*i*FS was proposed by Ni and Bhuyan in [36], although it belongs to a broad class of iterative input scheduling algorithms. The iFS scheme can be formalized in the following way:

Initially, all inputs and outputs are unmatched.

In each iteration

1. Each unmatched output selects a flow with the smallest virtual time and marks the cell as a candidate for the corresponding input
2. Each unmatched input examines its candidate set as processed by the outputs, and selects a winner according to smallest virtual time. The specific input and output are then considered matched.

3. The candidate set is re-initialised

Ni and Bhuyan show in [36] that iFS gives fairer bandwidth sharing than WPIM or iSLIP, However it is shown in the next section that iFS still performs poorly under some circumstances, the evaluation is incomplete.

4.7 Other Work

Recently Schoenen [37],[38] has proposed a good although fairly simple scheduling algorithm for input buffered switches called SIMP which stands for Successive Incremental Matching over multiple Ports. It approximates MWM using a sequential search for the largest weights. Starting with a sequentially rotated output port, the maximum input port weight is chosen. For the next output port there is then one less input port to consider. They also evaluate the algorithms performance under bursty traffic.

Another scheduler worthy of mention is the MUCS scheduler presented by Lockwood et al in [33], [34]. They present an interesting method of maximizing the throughput of a switch. By giving the ports with the least amount of contention priority they are able to maximize the number of matches and hence increase the throughput of the switch. Their implementation is also interesting, using mixed analogue and digital core. They do little however to address the maximum delay that a cell might experience, indeed this could be said to be a weakness intrinsic in their design.

Reservation with Pre-emption and Acknowledgment or RPA is another scheduler proposed recently [39]. Our impressions of it are that while it attempts to provide bandwidth protection it suffers from being unnecessarily complex.

There are various other scheduling algorithms which have been proposed in the literature, SFA by Tamir and Chi [40] is one worthy of mention. Very recently (2002) Shah and Kopikar [41] have done a study of latency of MWM approximate scheduling algorithms.

4.8 Evaluation of scheduling algorithms for input buffered switches.

There are a number of factors that would lead a switch designer to select one scheduling algorithm over another one. The evaluation of scheduling algorithms for input buffered switches is a complex topic. Firstly it must be feasible to implement. This is the over-riding consideration. Even a module as simple as a random number generator can prove complicated to implement in hardware. Secondly the algorithm should provide low latency, high throughput and be fair in allocating bandwidth. *i*SLIP is one algorithm which has proved that iterative algorithms are not too complex to implement in practice [29], while also providing good performance. Recently, iFS has also been shown to give good fairness, although the authors make no attempt to implement it. However the ability of input queued switches to offer bandwidth protection in the way that output buffered switches do is still clearly lacking.

Following the criteria used in other studies of resource allocation schemes such as [29] and [33], the evaluation of input scheduling algorithms should be evaluated with respect to fairness and efficiency. The main considerations for efficiency are throughput and cell delay. The two measures used to measure fairness in output buffered switches, namely the Service Fairness Index and the Worst Case Fairness Index can also be applied to some degree for obtaining results for the performance of input scheduling algorithms. The Service Fairness Index used is identical to that derived in [6]. As in output buffered switching and scheduling the Service Fairness Index gives an indication as to how the scheduler is allocating bandwidth compared to GPS. The WFI gives an indication as to how far the delay an individual cell may differ from what it would receive where the server GPS. In an input buffered switch several flows may be sharing the same source destination port on a switch. A feature of most input scheduling algorithms is that they make no distinction between individual flows belonging to the same input output destination.

4.8.1 Latency

The average queuing delay or latency as a function of the total offered load is the most commonly used method of determining the performance of an input buffered switch scheduling mechanism. In figure 1 we present the results of the simulation, of several input scheduling algorithms in a 16 by 16 switch. Arrivals at each input are Bernoulli i.i.d and the cell destinations are uniformly distributed over all the outputs. This configuration is the standard for evaluating scheduling algorithms for input queued switches and almost identical studies are presented by Mckeown in [17] and by Ni and Bhuyan in [20]. This experiment thus also served as a means to verify and calibrate the simulation environment.

The average latency is traditionally the primary means of comparing input buffered scheduling algorithms. Referring to figure 4.5, it is the performance of the algorithm when the traffic load is between 60 and 95 percent that is most important. As can be seen in figure 4.5 FIFO queueing performs worst, as opposed to output queuing which has the best performance. Maximum Weighted Matching matches output queueing very closely. There is a noticeable gap then between MWM and the iterative algorithms like iSLIP and iFS.

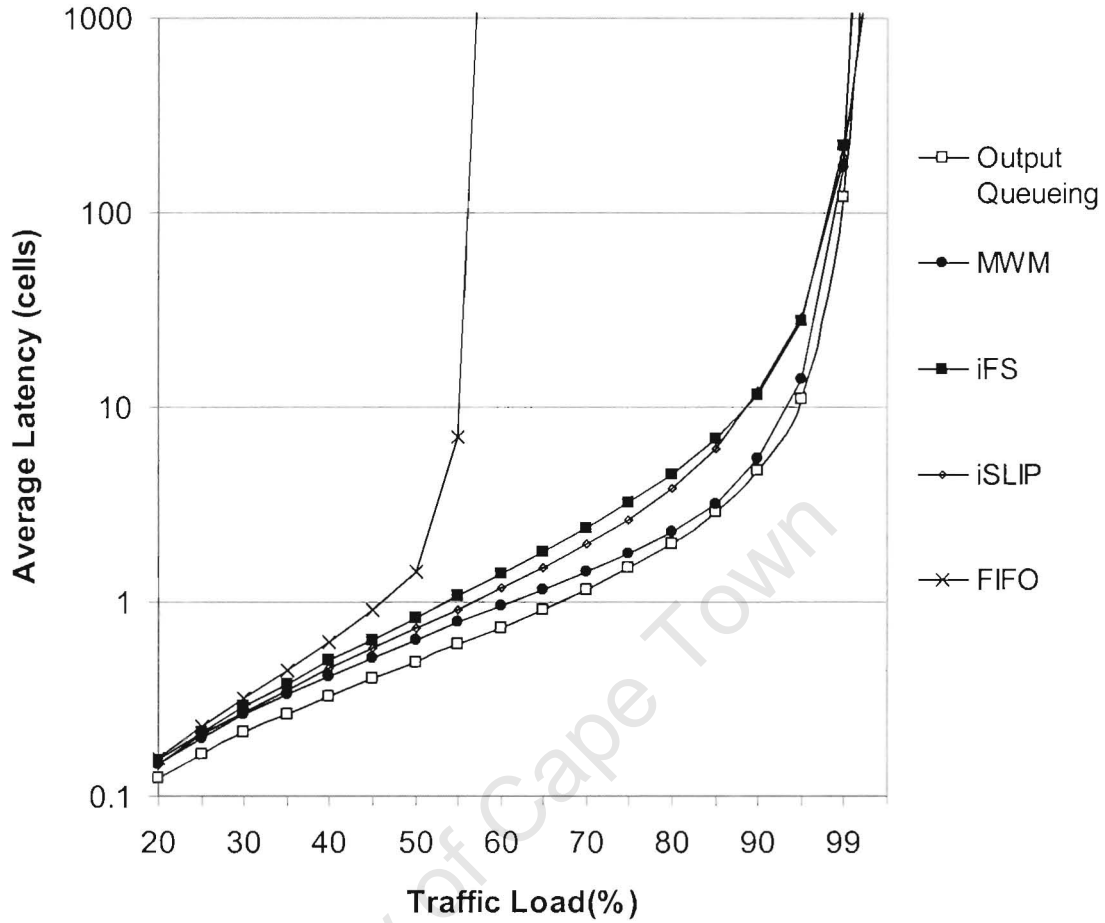

Figure 4.5 Latency characteristics of MWM, iSLIP, iFS and Output Queueing for Bernoulli arrival processes and destinations uniformly distributed across the output ports. The switch size used is 16x16. The graph shows the average delay in cell units. The % traffic load is an indication of the probability of a cell arriving at a specific input port in any time slot. The arrival process is Bernoulli and the cells destinations are uniformly distributed.

4.8.2 Worst-case Fairness

In an input buffered switch several flows may be sharing the same source or destination port on a switch. A feature of most current input scheduling algorithms is that they make no distinction between individual flows originating at the same input port and destined for the same output port.

The most common method of providing connection level allocation of bandwidth is by combining the switch level input scheduling algorithm with a flow level scheduling algorithm. The scheduling problem is then divided into two levels, switch level scheduling

load is the most commonly used method of determining the performance of an input buffered switch scheduling mechanism

While this may give a good indication of the performance of a scheduling mechanism it does not give an indication of the maximum delay a connection might experience. This figure is linked to the jitter and is very important for certain classes. The weighted average maximum delay for all connections is used as an indicator of the worst case performance of an input buffered scheduling mechanism.

One very important point to note here is that we are comparing the performance of an input buffered switch and scheduler with the optimal infinite bandwidth output buffered switch. So while it may be possible to obtain theoretical bounds for scheduling mechanism suitable for output buffered switches, this notion just does not make sense for input buffered switches, since we cannot make the same guarantees because the output port may be in contention. We can however still use the index to compute a figure which gives a good indication of how the input scheduler is performing under a given traffic profile and load.

The worst case fairness index used in input buffered switches is defined as below. This derivation is closely related to that derived for output scheduling algorithms by Zhang in [9]; the main difference is that the index we derive here is not independent of the traffic profile.

Definition 1 *A service discipline s has a cell worst case fairness index* $C_{i,s,p}$ *for traffic profile p and for session i, if for any time* τ, *the delay of a packet arriving at* τ *is bounded by* $\frac{1}{r_i}Q_{i,s,p}(\tau)+C_{i,s,p}$, *i.e.,*

$$d_{i,s}^{k} < a_i^{k} + \frac{Q_{i,s,p}(a_i^{k})}{r_i} + C_{i,s,p} \tag{16}$$

where r_i *is the throughput guarantee to session i,* $Q_{i,s,p}(a_i^{k})$ *is the queue size of session i at time* a_i^{k}

Since $C_{i,s,p}$ is measured in time it is not suitable to compare $C_{i,s,p}$'s of session with different r_i's or bandwidth shares. To perform such a comparison we define the normalized Cell Worst-case Fair Index for session *i* at server *s* and traffic profile *p* to be:

(17)

$$c_{i,s,p} = \frac{r_i C_{i,s,p}}{r_i}$$

Now the servers Normalised Worst-case Fair Index is defined to be

$$c_{s,p} = \max_i \{c_{i,s,p}\} \tag{18}$$

Now $c_{s,p}$ gives us an indication of the maximum delay weighted according to a connections reserved bandwidth. This is an important metric for delay sensitive applications such as voice. A theoretical switch with infinite buffers and infinite switch fabric bandwidth and perfect GPS scheduling at each output would have $c_{s,p} = 0$ under a specific traffic profile *p*. Note that if we want to compare the worst case fairness index of two servers it is only suitable to do this if they are subjected to the identical traffic load and profiles.

4.8.3 Service Fairness Index

As in chapter 3 the Service Fairness Index is defined in the following manner to measure the fairness of a scheduler at allocating bandwidth. Note that ideally unused bandwidth from connections not fully utilizing their shares should be distributed proportionally to backlogged flows. If there are *N* flows each with a bandwidth reservation r_i, and some of these connections are honouring their reservations, while others are oversubscribing their bandwidth reservation, then ideally each oversubscribed connection would receive r_i+B_i, where B_i is the excess bandwidth share for oversubscribed connection *i*. See [33] for a formal definition of excess bandwidth share. Now for every connection define y_i as the minimum of the reserved bandwidth plus excess bandwidth share and the actual arrival rate x_i.

$$y_i = \min\left|x_i, r_i + \beta_i\right| \tag{19}$$

So in the case of an under subscribing flow, y_i equals the arrival rate, and in an oversubscribing flow y_i equals the reserved bandwidth plus excess bandwidth share $(r_i + B_i)$.

$$SFI_{i,j} = \max_{n_1<n_2} \left| \frac{W_i(n_1,n_2)}{y_i} - \frac{W_j(n_1,n_2)}{y_j} \right| \quad (20)$$

where $W_i(n_1,n_2)$ is the number of cells session i transmitted by the server between timeslot n_1 and n_2. Note that if a connection transmits no cells between timeslots n_1 and n_2, we do not consider it in our calculations; its reserved bandwidth is just distributed to other connection excess bandwidth share. The SFI captures the difference in fairness of the scheduler from the ideal fairness of GPS in distributing service to connections that are continuously backlogged.

The evaluation of these scheduling algorithms is conducted with the use of specifically developed software cell based simulator. The simulator was designed to compare various algorithms with one another, and it is easily extensible, should a new algorithm require evaluation. Part of the cell based simulator is the maximum weighted matching module, which performs a maximum weighted match. This required an implementation of a complex mathematical algorithm; however this topic is discussed further in the next chapter.

The two measures used in Output Buffered Switches, namely the Service Fairness Index and the Worst-case Fairness Index can also be applied to some degree in obtaining results for the performance of input scheduling algorithms. The Service Fairness Index used is identical to that derived in [9]. As in output buffered switching and scheduling the Service Fairness Index gives an indication as to how the scheduler is allocating bandwidth compared to GPS. The Worst-case Fairness Index gives an indication how far the delay an individual cell may differ from what it would receive where the server Generalized Processor Sharing.

One very important point to note here is that we are comparing the performance of an input buffered switch and scheduler with the optimal infinite bandwidth output buffered switch. So while it may be possible to obtain theoretical bounds for scheduling mechanism suitable for output buffered switches, this notion just does not make sense for input buffered switches, since we cannot make the same guarantees because the output port may be in contention. We can however still use the index to compute a figure which gives a good indication of how the input scheduler is performing under a given traffic profile and load.

4.9 Traffic models

Investigations have shown that wide area network traffic is correlated form cell to cell and so in practice cells arrive in bursts. This traffic can therefore not be very well characterized by commonly employed Bernoulli or geometrically distributed ON/OFF models. Recently this internet and data traffic has been shown to be bursty over many time scales. Understanding the nature of this sort of traffic and evaluating its impact are important for the design of network elements. Quite a number of ways of modelling bursty network traffic have recently been proposed.

Initial studies have shown [29] that burstiness has as much or more influence on the performance of output queued switches than input queued switches. In other words burstiness tends to concentrate the conflicts on outputs rather than inputs. Intuitively this is what also what happens in real life scenarios; consider for example a single server serving multiple clients or an edge switch connecting to the core network on one port and access devices on the other ports.

Modern packet switches have to be designed to handle bursty traffic. This usually means that more cell buffers are required, although there are other methods of handling burstiness.

Generally burstiness has been shown to increase the queueing delay of a particular connection [29]; however some scheduling algorithms may be more susceptible to decreased performance under bursty traffic.

4.10 Implementation of Input Scheduling Algorithms

One of the most important considerations for input scheduling algorithms is whether they are practical to implement in hardware. Most of the scheduling algorithms presented in this chapter have a proposed hardware implantation. Indeed some of them have actually been used in prototype switches or even commercial products.

In [29] Mckeown proposes an iterative method of implementing iSLIP. He describes an implementation of an NxN switch using $2N$ Arbiters and N^2 memory bits. Stiliadis presents a hardware implementation of WPIM in [35]. The implementation of this algorithm seems

to be more complex than the implementation of iSLIP; it requires random number generators and a complex bandwidth enforcer.

Although Bhuyan and Ni do not provide a description of an iFS implementation IN [36], it should also be possible to implement iFS using the iterative method Mckeown proposes for i-OCF and i-LQF in [29]

4.11 Summary

This section has provided an introduction to scheduling algorithms for input buffered switches. We have highlighted some of the benefits of using input buffered ATM switches and also drawn attention to some of the problems associated with them. None of the existing approaches for traffic scheduling in input buffered switches provide a satisfactory means of providing delay guarantees. The existing approaches are currently only able to offer limited protection to a traffic flow from other possibly misbehaving traffic flows. In chapter 5 we attempt to address this problem, where a simple method of providing bandwidth guarantees is presented. In chapter 6 we present the results of tests which were run to test the operation of this scheduling algorithm and compare it with other current scheduling algorithms.

Chapter 5 : Worst-case Iterative Matching

5.1 Iterative Scheduling Algorithms

Scheduling for input buffered switches has advanced to the point where the throughput and average latency are very close to what is achieved in output buffered switches. However input buffered switches are still not able to offer the hard delay guarantees that output buffered can offer. Current input scheduling algorithms tend to focus more on maximizing the throughput through the switch fabric and assume that some sort of cell dropping mechanism such as Early Packet Discard (EPD) will stop the switch fabric becoming overloaded, which would cause other connections to receive degraded service.

Current scheduling algorithms are not particularly good at preserving bandwidth guarantees in an input buffered switch. The problem is that a cell from a connection with a large bandwidth share has to not only contend with cells from its own port but with cells from other ports as well. This leads to a situation where connections with large bandwidth shares may not suffer degradation in the total bandwidth they receive but they do suffer an increase in the maximum delay that a cell may encounter.

5.2 Using a Ratio to control the delay.

We propose that the choice of weights used in these algorithms may not be ideal for minimizing cell delay. The weights need to be modified in such a way as to force the maximum matching scheduling algorithm to provide the best cell delay protection even under heavy traffic loads. The weights used in OPF are usually weighted according to the connections' bandwidth shares. Thus a packet with a larger bandwidth share will have a larger weight for a cell which has been backlogged the same amount of time as a cell from a connection that has a smaller bandwidth connection. It is this ratio that needs to be modified. Note that if a multiplication factor R is introduced this ratio between the weights will increase.

The idea is then to find the multiplication factor R by which to multiply the weights used in the maximum weighted match, so as to minimize the C-WFI as defined in the previous chapter.

To this end an experiment was conducted with various traffic profiles and loads using the maximum weighted matching algorithm by Edmonds Karp, and using an iterative scheduling algorithm which we call worst case iterative matching which is described below. The results of this experiment are presented in the next chapter.

As stated in the previous chapter virtual output scheduling takes place at the input ports. This means we maintain a queue at each input for each output. From these output queues we have to find some metric to use in the maximum weighted matching algorithm. There are many choices, such as average queue length, maximum queue length or age of the cells. We have chosen to use the age of cell. As stated there are other metrics which could be used, but using the cell timestamp makes the system more sensitive to an individual cells delay.

The weights of a cell are computed as follows:

Virtual output scheduling takes place at the input ports, $r_{i,j}$ is the desired bandwidth for flow i on output port j, $r_{i,j}$ is the actual weighting used for the connection. The weights used in the iterative algorithms are determined as follows. If $P(m)$ is the system Potential, the timestamps $F_{i,j}^{k}$ of cell k on connection i on any input port destined for output port j is computed as:

$$F_{i,j}^{k} = \max(F_{i,j}^{k-1}, P(a_{i,j}^{k})) + \frac{1}{r_k} \tag{21}$$

where $a_{i,j}^{k}$ is the arrival timeslot of the cell. In a switch with N ports, each input port will hold an array of N timestamps holding the oldest timestamps for each output port. These oldest timestamps are called $F_{j,l}$. Using the oldest timestamp first scheme, $W_{j,l}$, the weighting between input l and output j used in the matching part of the algorithm and is defined as:

$$W_{j,l} = \frac{R}{F_{\max} - F_{j,l}} \tag{22}$$

F_{max} is the youngest (or largest) timestamp in the whole system, and F_{max} can also be the current time if a cell has arrived in that slot. So for example a switch with 8 input ports and 8 output ports will calculate a 64 number array holding the weights.

If the ratio R is large then we will be weighting the older timestamps more than the new timestamps. Thus we can use this to give greater importance to scheduling older cells. It is intuitive that the maximum sized match will usually occur when the weights used in the matching algorithm are equal. Thus by modifying R we have control over whether we favour increasing the switch's throughput or decreasing the maximum delay an individual connection might encounter. If the load on a switch is light, or if it is heavy but none of the traffic is delay sensitive there would be no need for a large R value.

Why this scheme would work is intuitive. If the weights are all equal, there is no way for a delay sensitive connection to gain priority. Using a larger ratio makes the system more sensitive to an individual cells delay. The trade off is that the switch may not be maximising its total throughput, it ends up finding the matching which maximises throughput of those cells which are experiencing unwanted delay.

5.3 Worst-case Iterative Matching

As it is impractical to use a maximum weighted matching algorithm in a real switch, what is required is an iterative method which comes as close as possible to approximating MWM. To this end we devised the Worst-case Iterative Matching scheme.

Worst-case iterative Matching is an iterative algorithm. Scheduling takes place in two stages, at the virtual output ports and to maximize the throughput across the fabric.
The maximum weighted matching part of the algorithm consists of two linked iterative algorithms running in parallel. In one part the matching is done at the inputs and the other it is done at the outputs. After each algorithm iterates the total weighted match is compared and the one with the largest is selected.

At the Input Ports:

1. Each unmatched output selects a flow with the smallest virtual time and marks the cell as a candidate for the corresponding input
2. Each unmatched input examines its candidate set as processed by the outputs, and selects a winner according to smallest virtual time. The specific input and output are

then considered a candidate match. The result of this is a preliminary set of candidate matches

At the Output Ports:

1. Each unmatched output selects the largest unmatched input as the winner. The order in which output ports get to select is rotated. The result of this is a second preliminary set of candidate matches

The candidate sets weights are totalled separately. If the set selected at the inputs set has a larger weight than the set selected at the output then the set from the input side is selected, otherwise the output set is selected. The ports selected by the winning set are then considered matched and the whole process is repeated with the unmatched ports

Why this method of choosing winning input and output ports was selected is explained using Figure 5.1

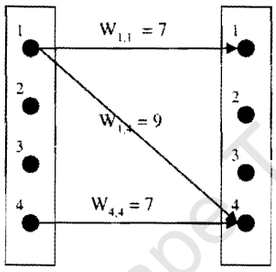

Figure 5.1 Finding a maximum match, selecting at the output ports

If selection of the matching set takes place at the input like it does in iFS or iOFC then the connection $W_{1,4}$ will always get selected over $W_{1,1}$ and $W_{4,4}$. However if the selections are made at the outputs then there is a chance that $W_{1,1}$ and $W_{4,4}$ will be selected. It is not hard to find a situation where selecting at the input gives the largest weighted match.

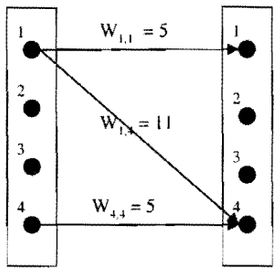

Figure 5.2 Finding a maximum match, selecting at the input ports

In Figure 5.2 with selection at the inputs, as in A, $W_{1,4}$ would be the selected connection, while if selection is done at the output ports $W_{1,1}$ and $W_{4,4}$ could equally likely be selected.

Thus we have shown how selecting the larger weighted matched set of the one matched at the input and the one matched at the output can help increase the total weighted match.

5.4 Parallel Implementation of Worst-case Iterative Matching

To conclude the description of WIM we consider the complexity of implementing the algorithm in hardware. The 'iterative' method of implementing maximum weighted matching scheduling algorithms as presented in [29] and [36] have a serious drawback, in that they need 4 or more iterations to achieve performance comparable to the more sophisticated but unpractical MWM algorithms. It is quite likely that any hardware which has to repeat a step 4 or more times will be too slow

So the requirement was to find a method of implementing WIM using a parallel method, which requires only one iteration, is not too complex, and gives comparable performance to WIM, iFS and SLIP. Other design is influenced by the fact that silicon 'real-estate' has become much cheaper in the recent past.

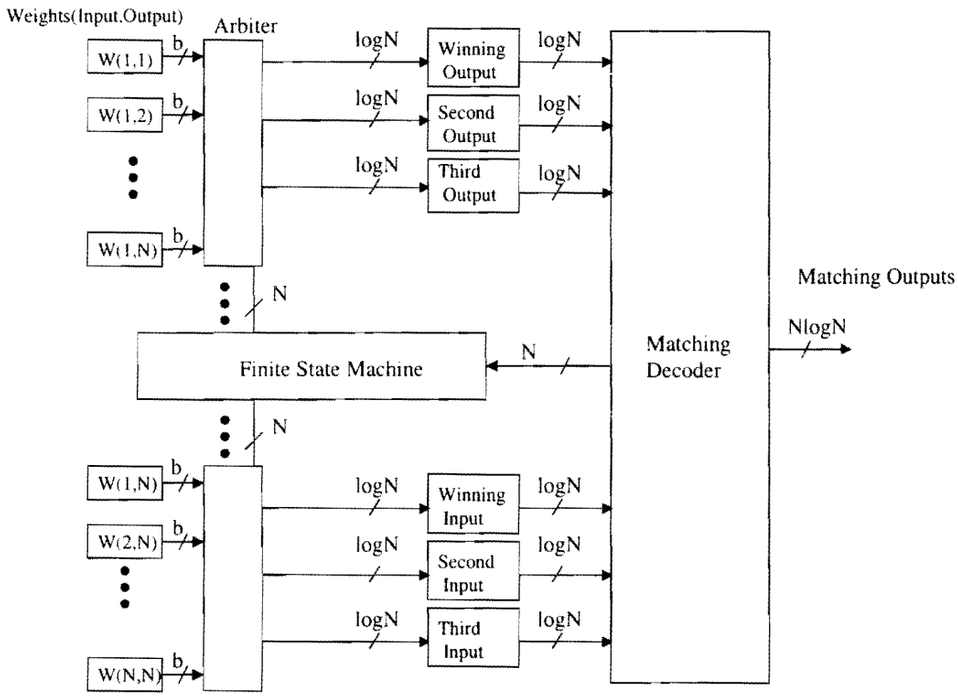

Figure 5.3 Parallel implementation of WIM

Figure 5.3 shows a parallel implementation of WIM. The iterative steps of WIM are performed in parallel in the matching decoder. At each input and output arbiters find the destination ports and originating ports with the largest weights and the second and third largest weights.

The matching decoder consists of multiple input matching decoders and output matching decoders in parallel. At the input matching decoders the largest weighted output at each input is selected. At the output matching decoders the winning input is selected. An input side matching decoder for a 4x4 switch is illustrated in Figure 5.4. The output matching circuitry is almost identical. The matches from the largest weights always get priority.

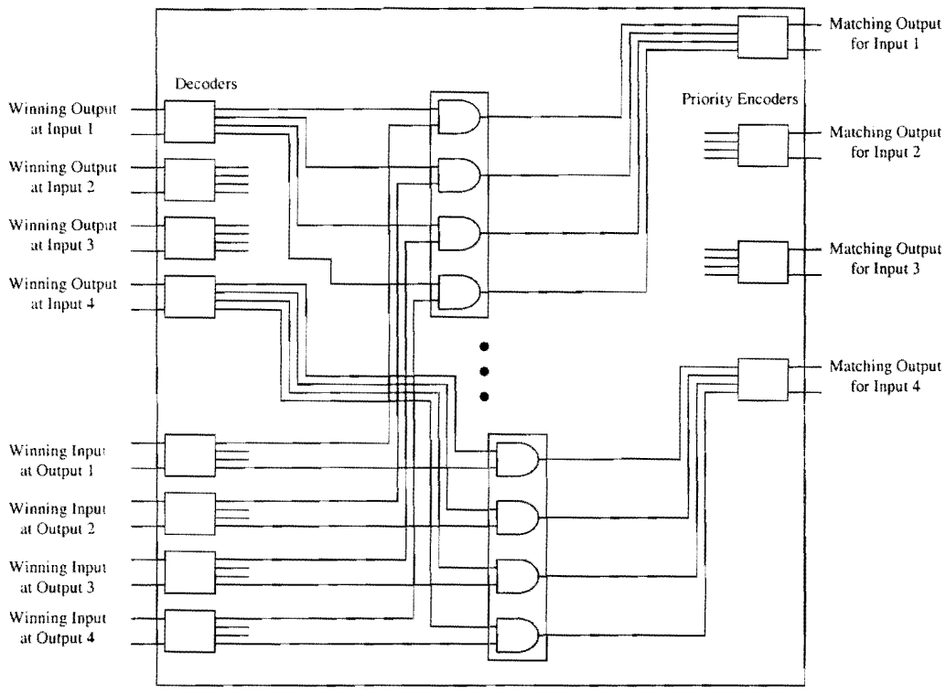

Figure 5.4 An input matching decoder used in a parallel implementation of WIM for a 4x4 switch

Choosing the value for b, the number of bits used to hold the timestamps is an important design decision, affecting the number of different timestamp values can be distinguished, and hence the smallest unit of time that can be scheduled. Basically what this circuitry does is use parallelism to compute the results which could be gained from an iterative approach. One drawback is that it may become too complex for the largest switches.

5.5 Summary

In this chapter we presented 2 major ideas, firstly the use of a multiplying ratio to control the delay that a cell encounters was proposed. In the next chapter we study the effectiveness of this scheme. Secondly we proposed an input scheduling algorithm, called Worst-case iterative Matching. The method computes two matches and selects the larger one in each iteration.

We also presented a possible implementation in hardware. In the next chapter we examine the performance of this scheduling algorithm, comparing it to iFS, iSLIP, MWM and Output Queuing.

Chapter 6 : Evaluation Worst Case Iterative Matching

6.1 Evaluation of Scheduling Algorithms

The evaluation of these scheduling algorithms is conducted with the use of a specifically developed software cell based simulator. The simulator was designed to compare various algorithms with one another, and it is easily extensible, should a new algorithm require evaluation. Part of the cell based simulator is the maximum weighted matching module, which performs a maximum weighted match. This required an implementation of a complex mathematical algorithm, the Edmonds Karp Algorithm.

The method of evaluation is as follows: we examine the performance of the scheduling algorithms using various synthetic workloads. We first look at how the schemes perform with regard to average cell latency. This is followed by an evaluation of the ability of the schemes to perform fair bandwidth distribution. These evaluations are performed on a 16 x 16 switch; that is 16 input ports and 16 output ports. This experiment was conducted to test the ability of an algorithm to provide bandwidth protection. The simulation settings used are as follows. On each port there are 128 connections for a total of 2048 connections. There are 128 connections (8 on each port) evenly distributed which together reserve half the total bandwidth, the next 128 (or the next 8 on each port) connections reserve a quarter of the bandwidth and the next 128 one eighth and so on. Thus the connections with the most bandwidth would expect an average inter cell arrival time of 32 cell units.

First a uniform traffic model is used. Each connection transmits at an equal rate. Each connection transmits at an average rate of 1 cell every 128 intervals. Thus some connections are using much less than their reserved bandwidth, while some are using much more.

The object of this experiment is to test the ability of the scheduling algorithm to maintain bandwidth delays for those connections using less than their total reservation while distributing the remaining bandwidth fairly. The experiment is first performed to determine the value of the ratio which gives the best performance, and then the performance of WIM is compared to other scheduling algorithms.

The experiment is then repeated using a bursty traffic source. As is widely known real network traffic is not uniform. We investigate the performance of WIM using a modulated 2 state Markov chain to drive the traffic source. The traffic sources on each connection produce bursts of cells which are all destined for the same output port. These are followed by idle periods. There are certainly more sophisticated traffic models available, and indeed the use of real network traces has been proposed [36]. However to a first approximation this model has been considered adequate by various studies [29]. This method was thus chosen for its simplicity and speed.

Finally the performance of various algorithms is evaluated for different switch sizes, this is fairly unique and to the best of the author's knowledge no comparison has been conducted like this before.

In total more than 500 hours of simulations where run. The average number of cell periods simulated was roughly 2000000. This figure was limited by the large amount of memory and disk space needed hold backlogged cell data under heavy traffic loads. With a switch size of 16x16 approximately 10GB of hard disk space was required for use as virtual memory to run the simulation for 200000 intervals and at 99% Bernoulli traffic load. The Maximum Weighted Matching algorithm also required large amounts of memory.

The simulations were run on an Intel Pentium IV PC with 512 MB RAM and a 30GB hardrive.

6.2 Performance under Bernoulli Traffic

The performance of the scheduling algorithms was first evaluated using a benign Bernoulli i.i.d traffic model. At any given interval a cell arrives at each port with the probability given by the offered traffic load. The destinations are uniformly distributed among the outputs.

Although this is not the most realistic of models it is the simplest and is a standard method of evaluating the performance of switches and scheduling algorithms.

6.2.1 Latency

We start with perhaps the simplest but perhaps most important measure of a scheduler's performance, the average cell latency.

The WIM algorithm was compared with several algorithms, Pure Output Queuing, Maximum Size Matching, iFS and iSLIP. In figure 1 we present the results of the simulation, using a 16x16 switch. Arrivals at each input are Bernoulli i.i.d and the cell destinations are uniformly distributed over all the outputs.

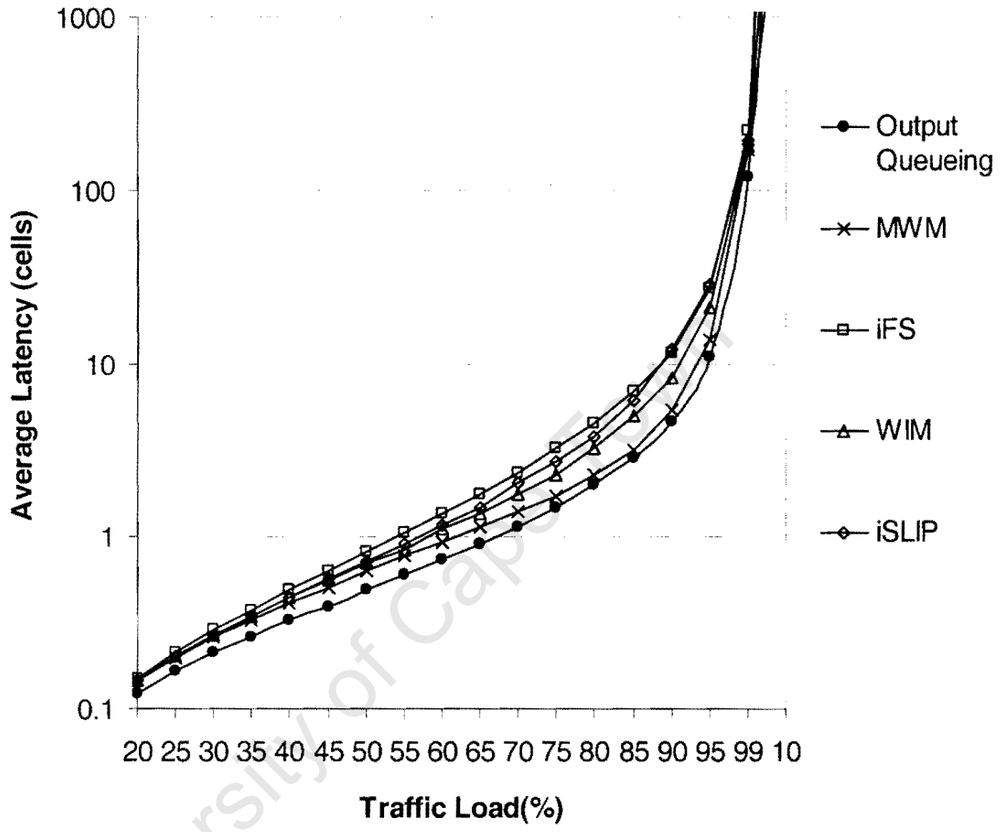

Figure 6.1 Latency characteristics for MWM, iSLIP, WIM, iFS and Output Queueing for Bernoulli arrival processes and destinations uniformly distributed across the output ports. The switch size used is 16x16. The graph shows the average delay in cell units

The WIM algorithm has very similar performance to iFS and iSLIP, although it does give a small performance advantage over these two algorithms. It gets closer to the

performance of MWM than these two algorithms. However it is easy to see that there is still a considerable difference between the performance of WIM and MWM. The ratio *R* (discussed in the previous chapter) used in this experiment for WIM was 100.

Next the performance of WIM was tested under uniform traffic for various ratio values. The values used were 1, 10,100 and 1000. As can be seen the average latency is largely independent of the ratio used, although the latency is lower under heavy traffic if a larger value is used for the ratio

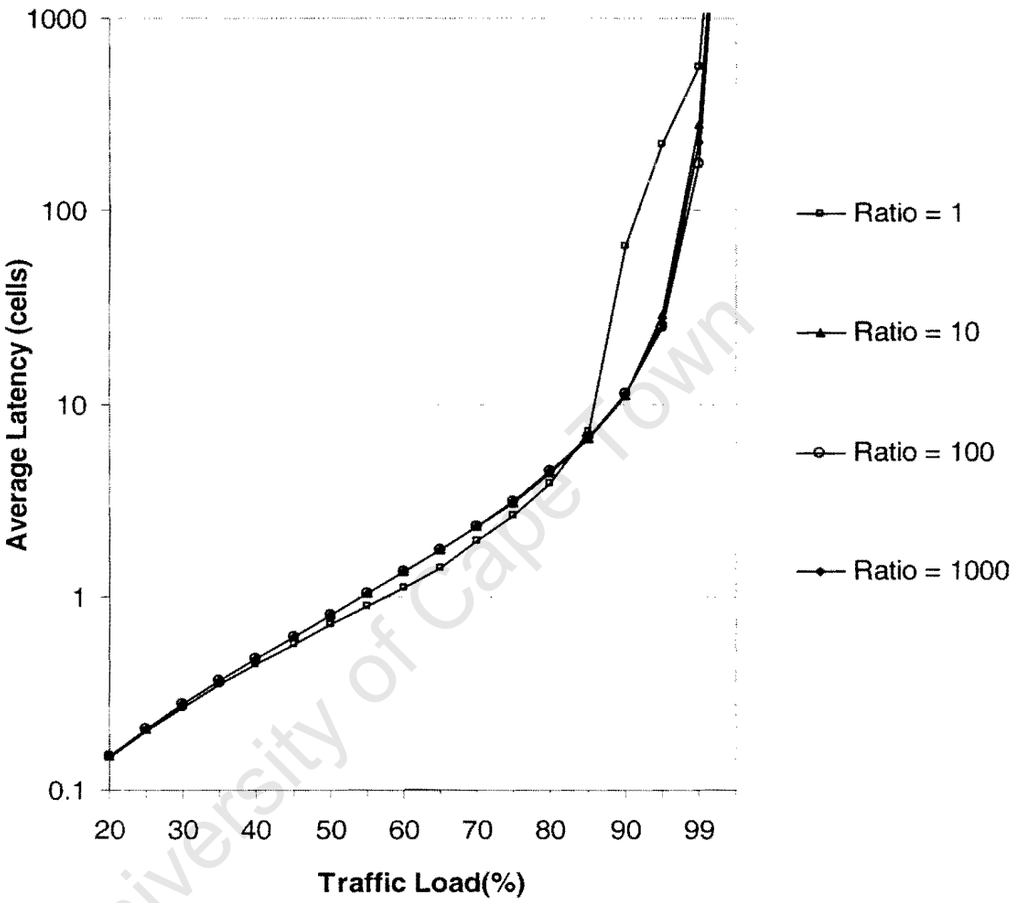

Figure 6.2 Latency performances for WIM scheduling algorithm using different ratio values Arrival processes are Bernoulli and destinations uniformly distributed across the output ports. The switch size used is 16x16. The graph shows the average delay in cell units

6.2.2 Worst-case Fairness

The next experiment was conducted to test the ability of an algorithm to provide bandwidth protection. The simulation was set up as described in section 6.1
Average queue size, total throughput, and average latency were monitored. A uniform traffic model was used and the fairness and worst-case fairness of the WIM algorithm was determined under various ratio's

The worst case performance for various values of R is shown in Figure 6.3.

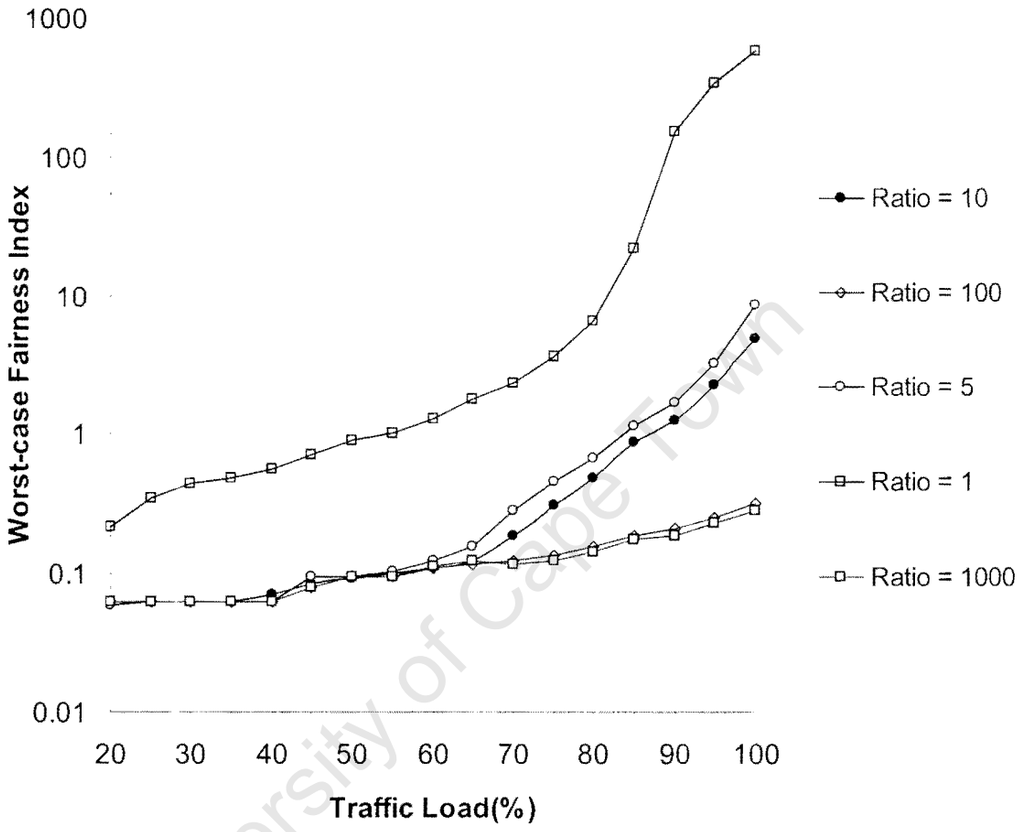

Figure 6.3 Worst-case Fairness Index for WIM scheduling algorithm using different ratio values Arrival processes are Bernoulli and destinations uniformly distributed across the output ports. The switch size used is 16x16. The graph gives an indication of the maximum delay a cell might experience.

As can be seen in Figure 6.3 the worst case index is generally lower for larger ratio values. This is what might be expected, a scheduler with a higher ratio value would tend to give cells which had been queued longer a greater weight than a scheduler using a lower ratio.

cells which had been queued longer a greater weight than a scheduler using a lower ratio. Note that these figures only have relevance with respect to this traffic profile and this switch size, 16x16.

Next the worst-case performance of WIM with a ratio of 100 is compared to SLIP, IFS and Maximum Weighted Matching and Output Queueing. The results are shown in Figure 6.4. As can be seen WIM again performs better than iFS and iSLIP.

What this means is that connections encounter a lower maximum delay and hence a lower jitter when a WIM scheduler is used.

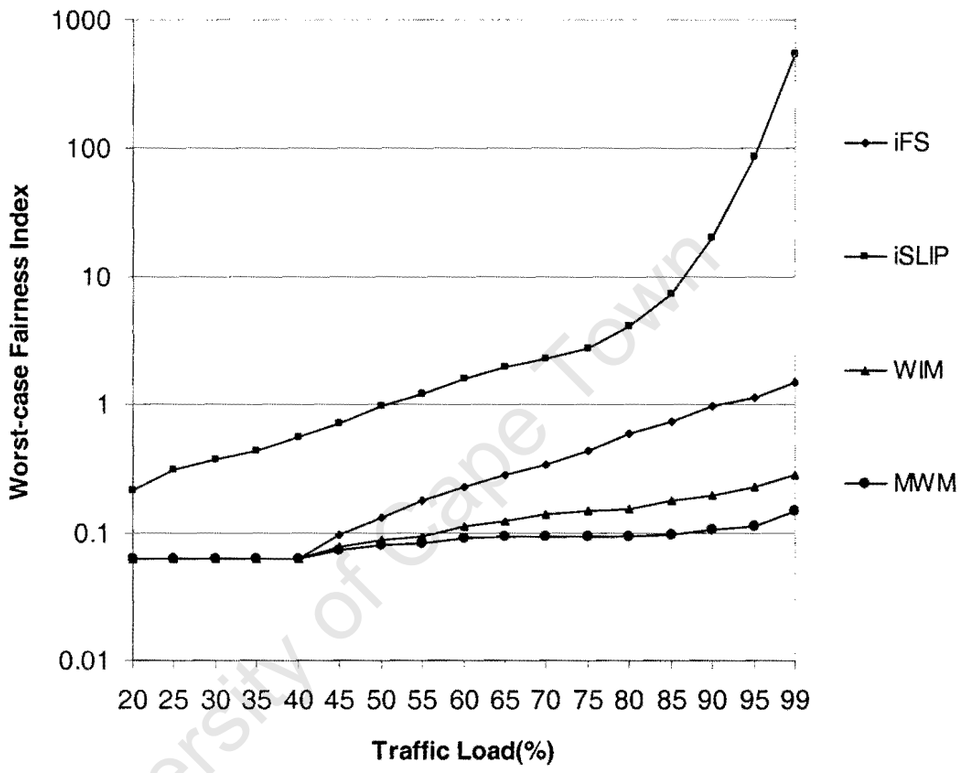

Figure 6.4 Worst-case Fairness Index for the WIM, iSLIP, iFS and MWM scheduling algorithms Arrival processes are Bernoulli and destinations uniformly distributed across the output ports. The switch size used is 16x16. The graph gives an indication of the maximum delay a cell might experience.

6.2.3 Service Fairness Index

Next the performance of WIM in distributing bandwidth equally is investigated again using various ratio values.

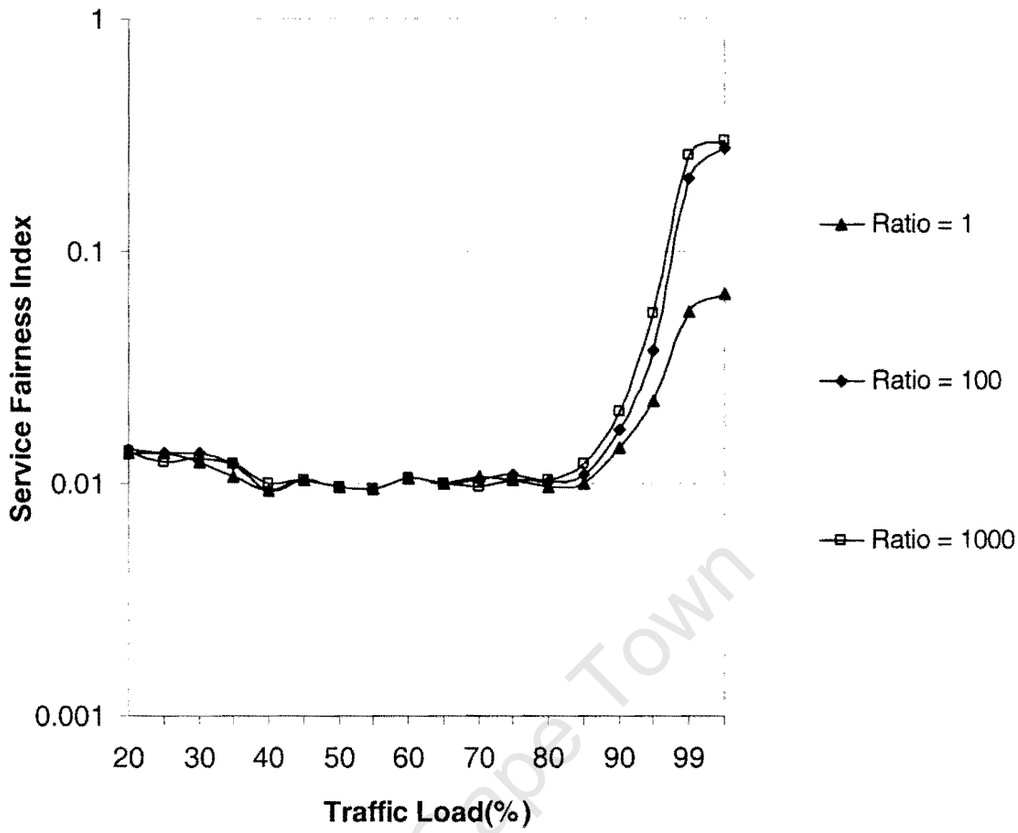

Figure 6.5 Service Fairness Index for the WIM scheduling algorithm using different ratio values. Arrival processes are Bernoulli and destinations uniformly distributed across the output ports. The switch size used is 16x16. The graph gives an indication of the how fairly the bandwidth is distributed under different traffic loads.

As can be seen from Figure 6.5, as far as distributing bandwidth fairly WIM performs similarly under a wide range of ratio values.

The algorithm distributes bandwidth more fairly under a heavy traffic load using smaller ratio values. Comparing WIM with iFS, iSLIP and Maximum Weighted Matching, it gives

very similar performance. iSLIP gives better performance under heavy loads than either WIM or iFS.

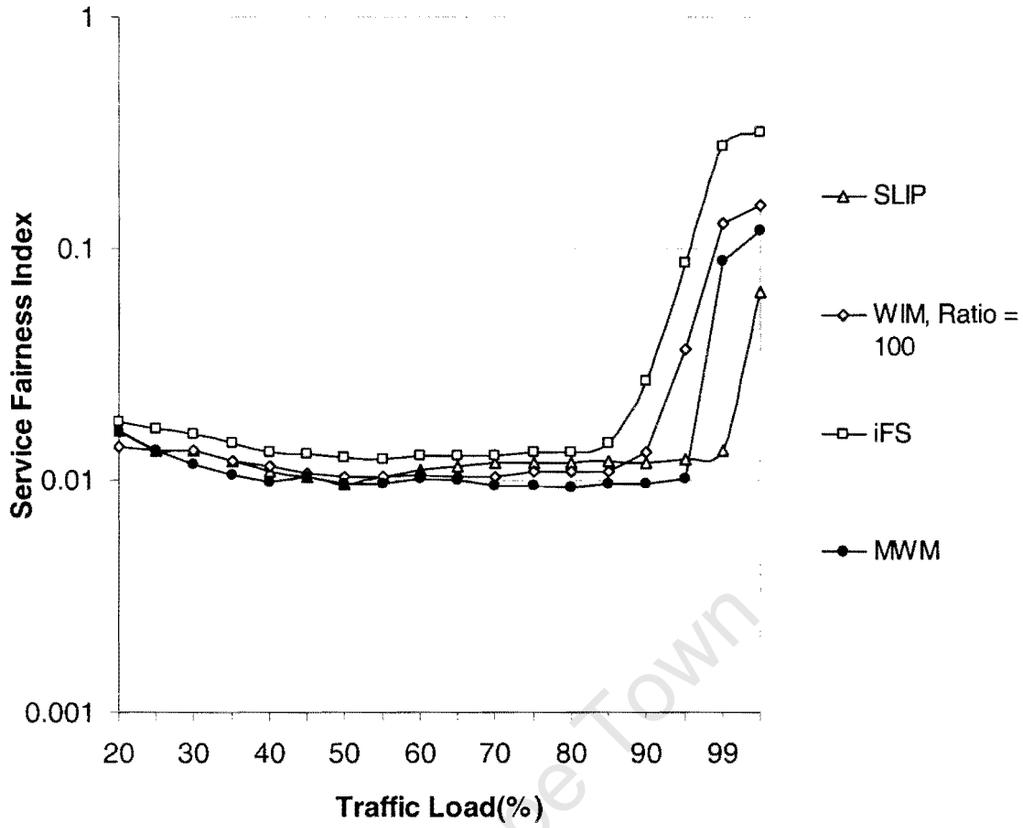

Figure 6.6 Service Fairness Index for the WIM, SLIP, iFS and MWM scheduling algorithms. Arrival processes are Bernoulli and destinations uniformly distributed across the output ports. The switch size used is 16x16. The graph gives an indication of the how fairly the bandwidth is distributed.

6.3 Performance under non-uniform traffic

We now investigate the effect of Burstiness on various schedulers. We chose MWM, WIM, iFS and output queueing as the schedulers to test in this scenario. The arrival process for each port is an on-off arrival process modulated by a two state Markov-chain. The sources produce bursts of cells all destined for the same output port. These bursts are followed by idle periods. Both the idle periods and bursts have geometrically distributed

number of cells. The destinations are uniformly distributed. The switch size used is 16x16. This is a similar configuration to that used by Mckeown in [29].

6.3.1 Latency

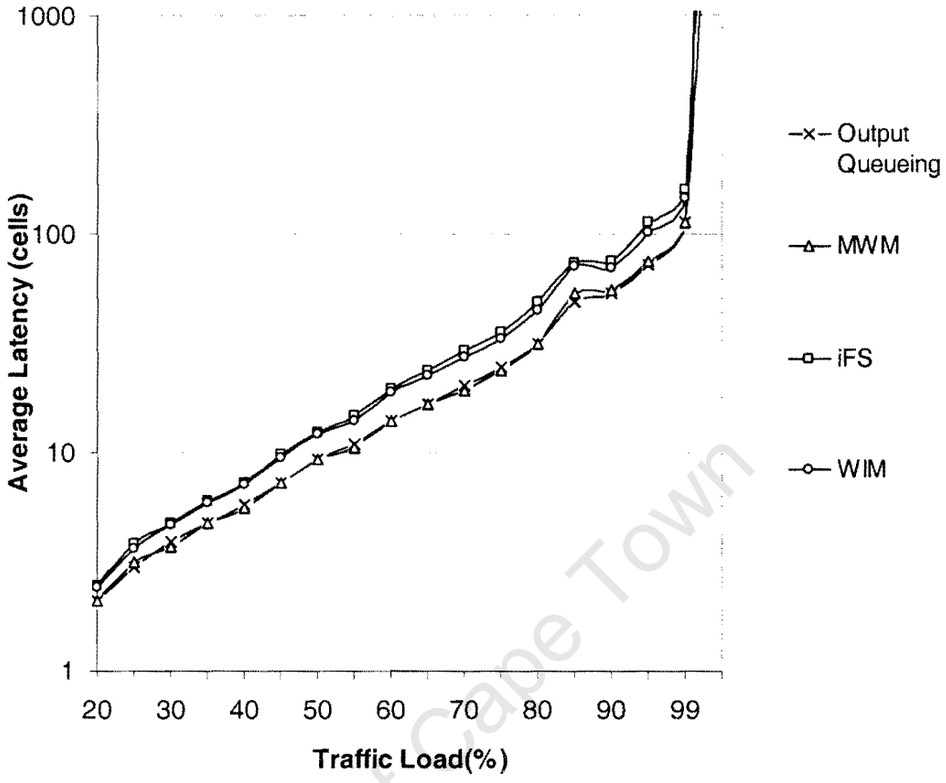

Figure 6.7 Latency performances for MWM, WIM, iFS and Output Queueing for Bursty arrival processes and destinations uniformly distributed across the output ports. The switch size used is 16x16. The graph shows the average delay in cell units

As can be seen in Figure 6.7 the performance of the scheduler decreases significantly under heavy traffic. This clearly illustrates the need for a greater number of buffers in a packet switch which will switch real traffic. Note that while the average latency is much larger under heavy traffic it is still bounded. Comparing this graph with the similar one for Bernoulli traffic also illustrates the sort of problems applications requiring low delay encounter in real world situations.

6.3.2 Service Fairness Performance under Bursty Traffic

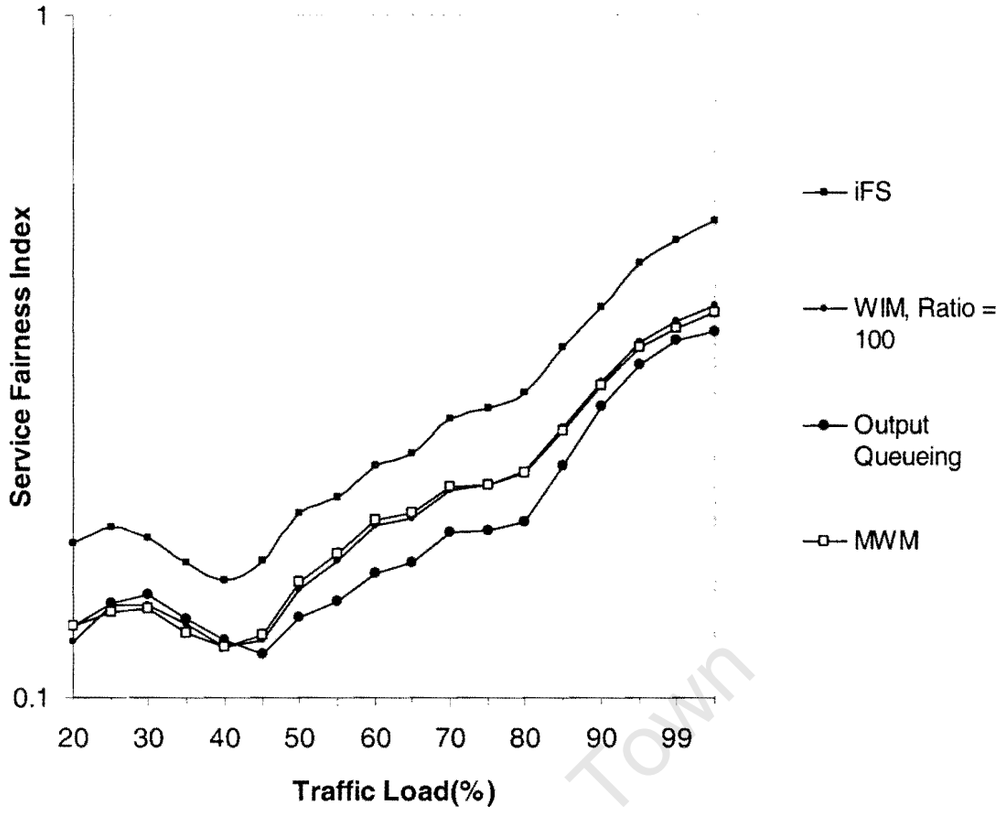

Figure 6.8 Service Fairness Index for WIM, iFS, Output Queueing and MWM. Arrival processes are Bursty and destinations uniformly distributed across the output ports. The switch size used is 16x16. The graph gives an indication of the how fairly the bandwidth is distributed between connections

As can be seen in Figure 6.8, the performance as far as distributing bandwidth fairly under bursty traffic is similar for all the algorithms, although the WIM algorithm does perform slightly better.

6.3.3 Worst-case Fairness under bursty traffic

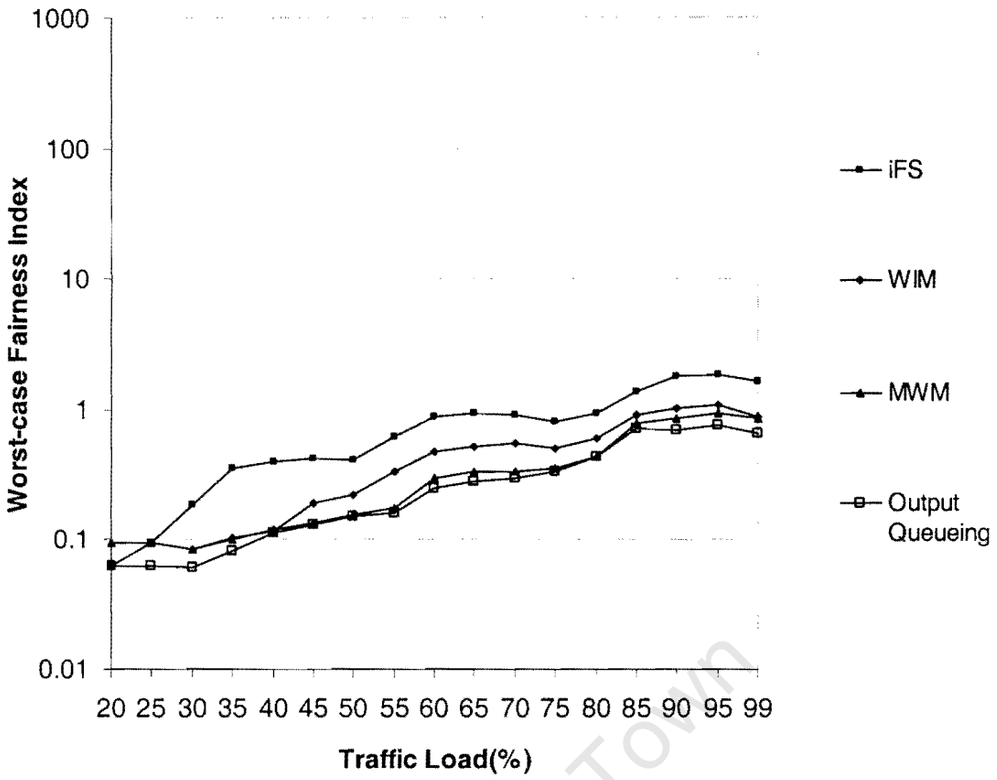

Figure 6.9 Service Fairness Index for the iFS, WIM, MWM and Output Queueing scheduling algorithms. Arrival processes are bursty and destinations uniformly distributed across the output ports. The switch size used is 16x16.

The studies on bursty traffic seem to confirm Mckeown's observation that burstiness has as much or more influence on the performance of output queued switches than input queued switches. The advantages in latency, worst-case performance and service fairness that OQ held over input queued algorithms have largely diminished.

6.4 Switch Size

There have been very few studies to study the impact of the switch size on the performance of the scheduling algorithms. The algorithms which are tested are Output

Queuing, MWM, and WIM. The performance of SLIP with various switch sizes was studied in [29] and this was used as a benchmark for these tests. The switch sizes used where 4x4, 8x8, 16x16, 32x32, and 64x64. No higher number could be run because of large amounts of memory required to run the simulation for this number of ports.

The traffic load used in each simulation is Bernoulli, uniformly distributed among the outputs.

6.4.1 Latency

As can be seen in figures 6.10 to 6.13 the performance of scheduling algorithms for input buffered ATM switches degrades slightly for larger switch sizes are used. This is not unexpected.

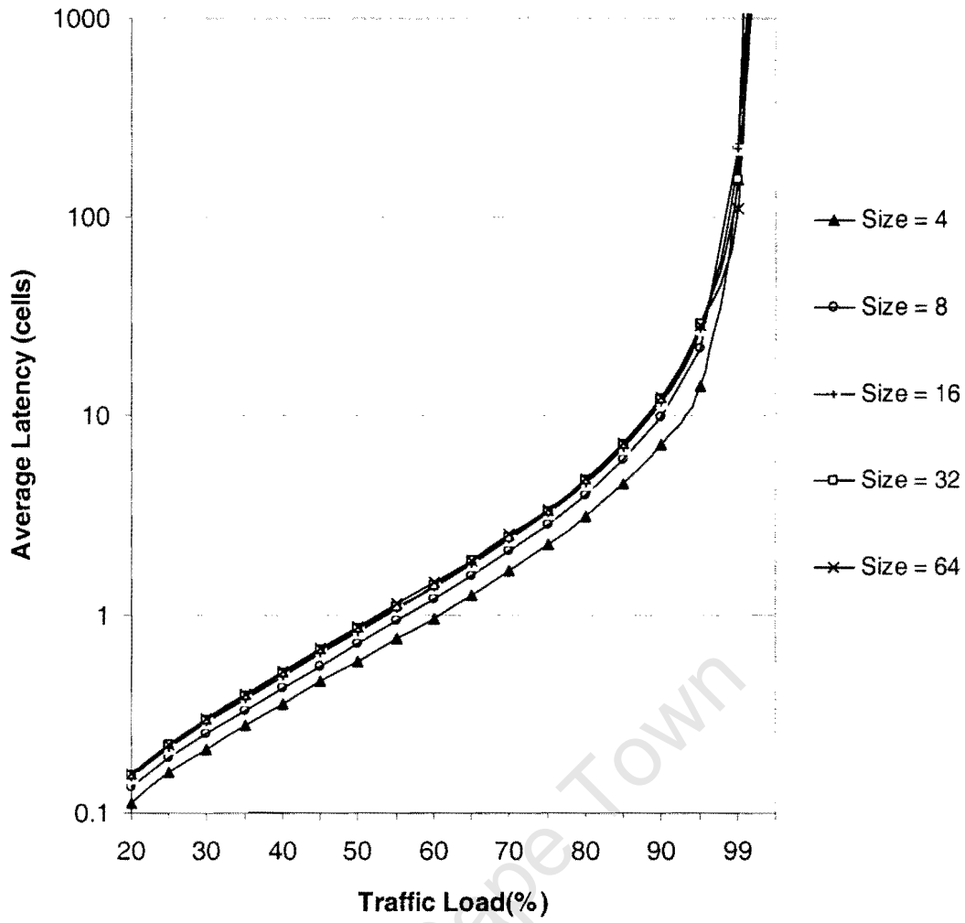

Figure 6.10 Latency characteristics of the WIM scheduling algorithm, for different size switches. Arrival processes are Bernoulli and destinations are uniformly distributed across the output ports. The graph shows the average delay in cell units

Comparing the performance of WIM in Figure 6.10, iFS in Figure 6.11, MWM in Figure 6.12, and Output Queueing in Figure 6.13, it can be seen that the decreased performance witch accompanies larger switch sizes occurs in all the scheduling algorithms. To be more specific the average cell delay at 90% load for a 4x4 switch using WIM was 8.3 cell units and for output queueing it was 5.4. For a 64x64 switch at 90% load the average delay using WIM was 11.8 using output queuing it was 7.3.

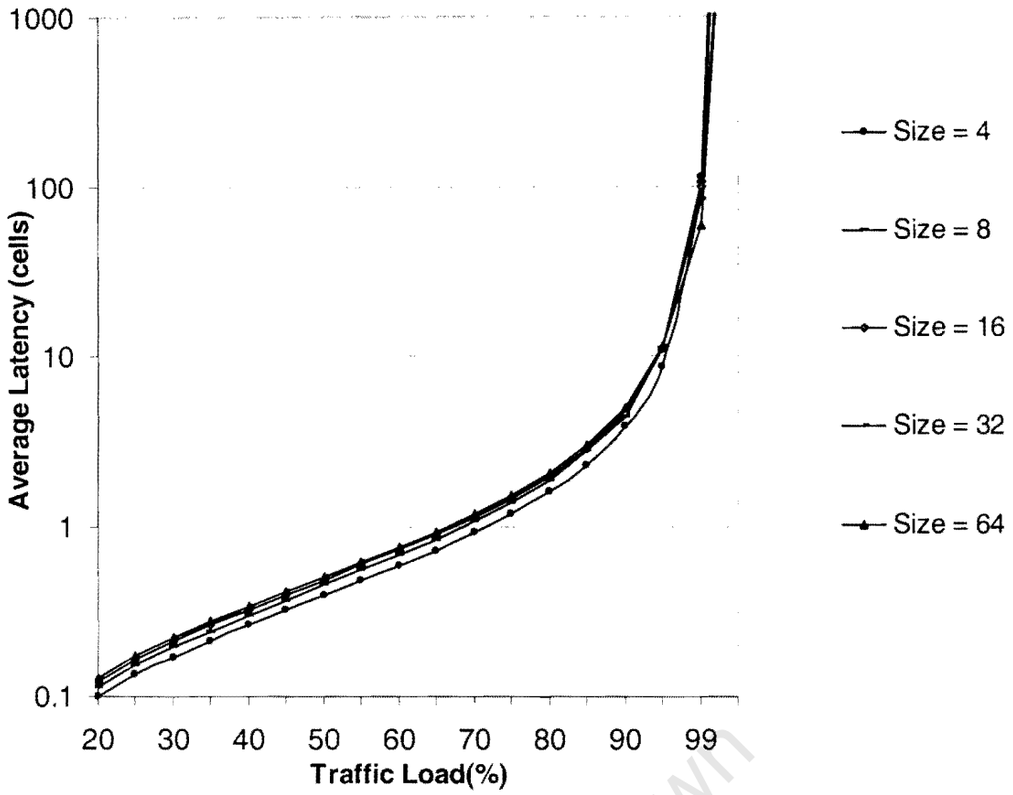

Figure 6.11 Latency characteristics of the iFS scheduling algorithm, for different size switches. Arrival processes are Bernoulli and destinations are uniformly distributed across the output ports. The graph shows the average delay in cell units

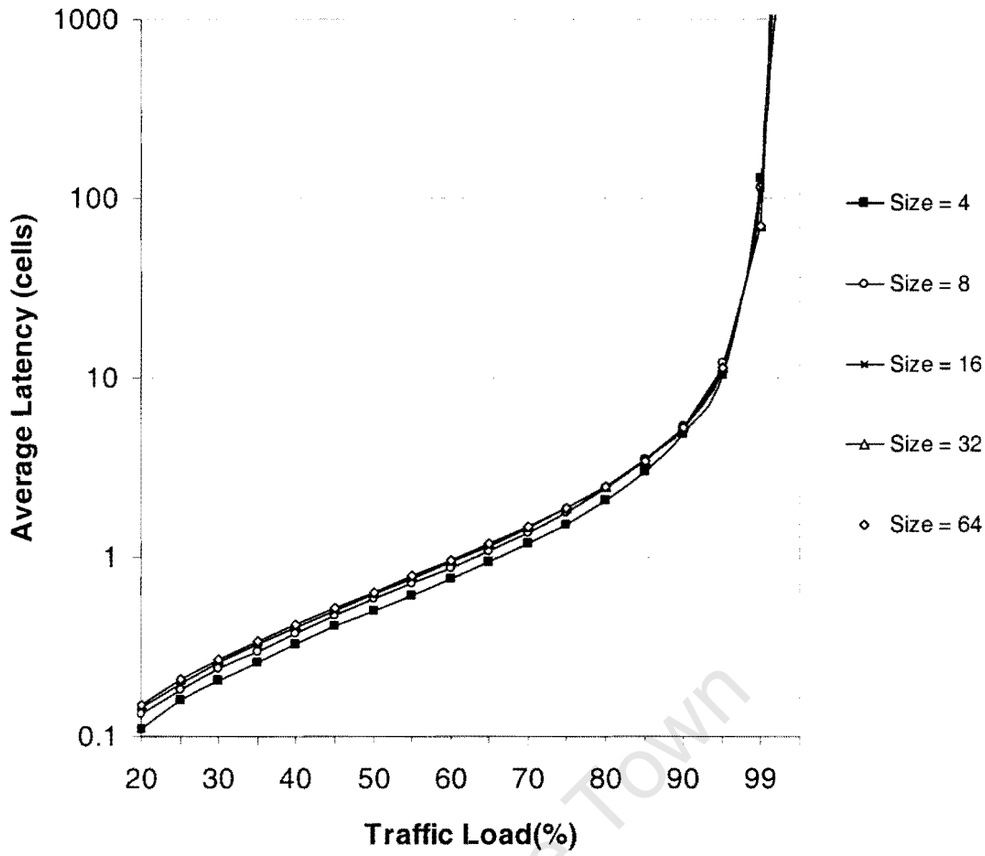

Figure 6.12 Latency characteristics of Maximum Weighted Matching scheduling algorithm, for different size switches. Arrival processes Bernoulli and destinations are uniformly distributed across the output ports. The graph shows the average delay in cell units

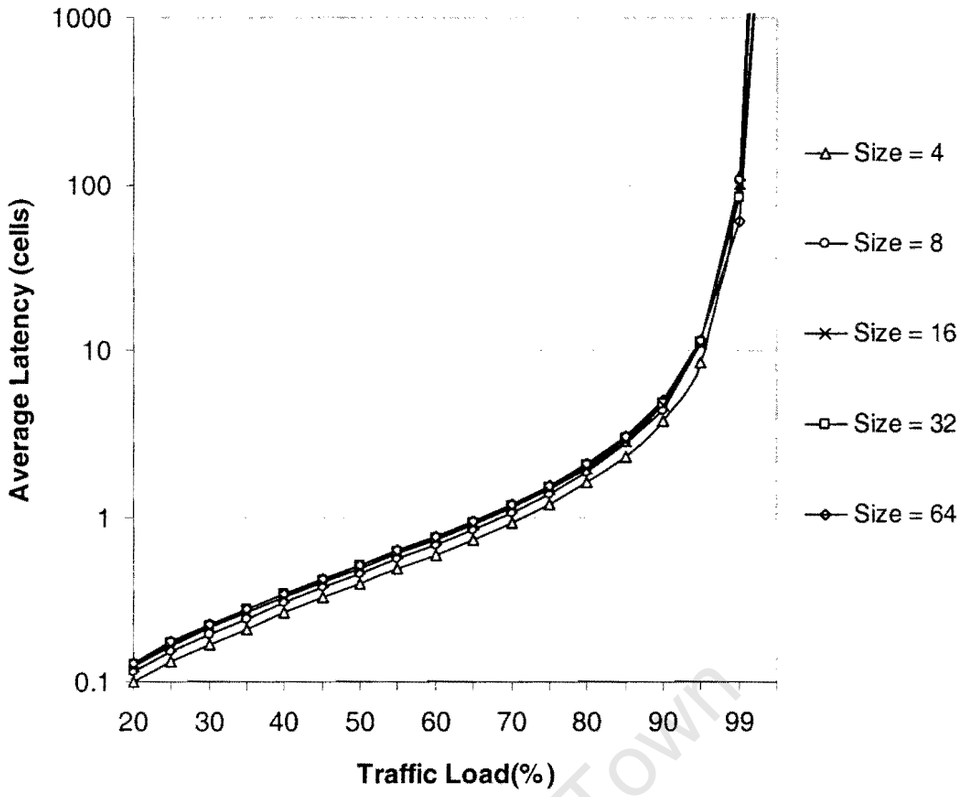

Figure 6.13 Latency characteristics of Output Queueing, for different size switches. Arrival processes are Bernoulli and destinations are uniformly distributed across the output ports. The graph shows the average delay in cell units

6.5 Summary

In this chapter we presented the results of simulations which were run to determine the performance of the WIM scheduling algorithm compared to other scheduling algorithms. We showed first that the average latency of the scheduler was similar if not slightly better than other practical scheduling algorithms. We then determined the worst case and service fairness performance compared to various other input scheduling algorithms. The WIM scheduling algorithm also performed well in terms of fairness and worst-case fairness when compared to the other scheduling algorithms. We also studied the performance of the algorithms under bursty traffic, and noted that the differences between input and output queueing seem to diminish. Finally a study was conducted on the effect of the switch size

on the performance of the algorithm. The performance of all the various scheduling algorithms was found to degrade slightly for larger switch sizes.

Chapter 7 : Summary and Future Research

7.1 Summary

This text has considered the development, analysis, and implementation of scheduling algorithms for input and output buffered packet switches. The focus has been on ATM switches, but the algorithms are applicable to most packet switches. The design of the scheduling algorithm is dictated to a large degree by the architecture of the ATM switch, and for this reason we have studied scheduling algorithms for both input and output buffered switches. Lots of the ideas initially developed for scheduling in output buffered switches can be modified and applied to scheduling in input buffered switches

7.2 Providing QoS support in an Input Buffered Switch

We introduced the idea of using a variable ratio to control the maximum delay a connection may encounter. This allows the switch using a matching algorithm to sacrifice total throughput in order to give priority to cells experiencing undesirable backlog.

We also introduced a scheduling algorithm for input buffered non-blocking switches called Worst-case Iterative Matching(WIM). The scheme offers an improvement in performance over weighted probabilistic matching, iSLIP and iFS.

Results from the simulations show that WIM is able to offer connections a lower worst-case delay than other current algorithms, as well as a slightly lower average latency. Using Virtual Output Queueing and variable weight ratios in the maximal weighted matching algorithm, WIM can provide isolation and protection for any flow. The algorithm also performs well in terms of fairly distributing excess bandwidth among competing connections.

A method of implementing WIM in hardware was proposed. Unlike many other iterative approaches which have been proposed and implement, our proposed implementation is largely a parallel one.

7.3 Worst-case Performance

In this text we have paid specific attention to the worst-case performance of input scheduling algorithms, a topic which has not received much attention in previous studies, although certainly lots of attention has been paid to the worst case performance of output scheduling algorithms.

It has been found in this study and previous work [29] that the worst-case bound gives a very conservative result, and that the average delay performance is much different. However the Worst-case performance does give valuable insights into the operation of the algorithms. In addition as Stiliadis points out in [35], if packet networks are to ultimately replace circuit switched networks, they will have to support a set of applications that do not accept packet losses or unpredictable delays.

7.4 Simulation Study of Input Scheduling Algorithms

The study of scheduling algorithms can not be conducted without taking into account the effect of real network traffic. However the results obtained from simulations seem to confirm Mckeown's observation that burstiness has as much or more influence on the performance of output queued switches than input queued switches.

Another important result obtained was the effect of the switch size on the performance of the scheduling algorithm. The result was obtained that the latency increases slightly for larger switch sizes, although there is not a significant degradation though. Of course this result needs to be compared with the fact that that if a larger switch was implemented using interconnected smaller switches, the latency would at least double. Therefore the conclusion can be made that a single large switch will always give better performance than many smaller interconnected switches.

7.5 Future Work

There is still considerable work to be done in the field of input scheduling algorithms. Currently switches at best use some sort of maximal algorithms, the search for a maximum weighted matching algorithm which can be practically implemented continues. There will be other matching algorithms proposed with lower complexity which provide performance closer to maximum weighted matching than the current algorithms such as WIM, iSLIP etc.

The complexity in studying the performance of these algorithms also presents opportunities for future work. Perhaps the best method of evaluating the performance of a scheduling algorithm is to implement it on a switch and test it under real network conditions. The Washington University Gigabit Switch would be one candidate for this, although a purpose designed switch would probably be the best suited for this algorithm.

There is also work to be done in area of simulation of input scheduling algorithms. It would be useful to run similar simulations using more sophisticated traffic models on a supercomputer or cluster. This would allow a better appreciation of the performance of the scheduling algorithm under conditions closer to what is encountered in a real network.

Another possible area of work is to investigate further the use of a multiplying ratio used in the MWM algorithm. We have used one value for the whole switch, but it may be possible to obtain better control over a connections delay and throughput using separate ratios for each connection.

References

[1] Turner J. and Yamanaka N., "Architectural Choices in Large Scale ATM switches" *WUCS – tn 97-21*, May 1997

[2] Hamid Ahmadi, Wolfgang E. Denzel, "A survey of Modern High Performance Switching," *IEEE Journal on Selected Areas in Communications, vol. 7, no. 7, pp 1091-1103 Sept. 1989*

[3] Chao H.J., Park J., Su W., "Performance study of commercial ATM switches", *Polytechnic University tn*

[4] Demers A., Keshav S., and Shenker S. "Analysis and Simulation of a Fair Queueing Algorithm". *Proc. ACM SIGCOMM'95 , pages 1-12, September 1989*

[5] Zhang H., Srinivasan Keshav, "Comparison of Rate-Based Service Disciplines ", *SIGCOMM '91 pp 113-122, Zurich, Switzerland, September 1991*

[6] Parekh A. K. and Gallacher R. G., "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case," *IEEE/ACM Trans Networking, pp. 344-357, June 1993*

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

[8] Parekh A.K. *"A Generalised Processor Sharing Approach to Flow Control in Integrated Service Networks.* PhD thesis, Department of Electrical Engineering and Computer Science, MIT, 1992.

[9] Bennett J. C. R. and Zhang H, "WF^2Q: Worst-case Fair Weighted Fair Queuing," *Proc. of SIGCOMM '96 pp.120-128, March 1996.*

[10] Goyal P. and Vin H., "Generalised guaranteed rate scheduling algorithms: A framework," *Tech Rep. TR-95-30, Dept. of Computer Science, U Texas at Austin, 1995*

[11] Bennett J.C.R., Stephens D.C., and Zhang H., "High Speed, Scalable, and Accurate Implementation of Fair Queueing Algorithms in ATM Networks," *Proc ICNP' 97, pp 7-14 October 1997.*

[12] Chiussi F. M and Francini A., "A Low-Cost Architecture for the Implementation of Worst-Case-Fair Schedulers in ATM Switches," *Proc. GLOBECOM'98, pp 2905-pp 2911, Nov 1998.*

[13] Ferrari. D., Verma D. "A scheme for real-time channel establishment in wide area networks", *IEEE Journal on Selected Areas in Communications, 8(3):368-379, April 1990*

[14] Verma D., Zhang H., Ferrari D., "Guaranteeing jitter delay bounds in packet switching networks. In *Proc TRICOMM, pp35-46, Chapel Hill, NC, April 1991.*

[15] Sivaraman V., Chiussi F., Gerla M., "Traffic Shaping for End-to-End Delay Guarantees with EDF Scheduling". *Proc IWQOS 2000, pp 10-18, 2000, Pittsburgh, Pennsylvania.*

[16] M.J. Karol, M.G. Hluychy, and S. P. Morgan, "Input versus Output Queuing on a space division packet switch," *IEEE Transactions on Communications, vol. 35, pp 1347-56, December 1987.*

[17] McKeown N. and Mekkitukkul A., "A practical scheduling algorithm to achieve 100% throughput in input queued switches," in *Proc INFOCOM'98*, Vol 2, pp 792-799, *San Francisco, April 1998.*

[18] Ajmone M., Marsan, A. Bianco, P. Giaccone, E. Leonardi, F. Neri, "Packet Scheduling in Input-Queued Cell-Based Switches", *Proc INFOCOM 2001 pp 1085 -- 1094, Anchorage, Alaska, 2001*

[19] M. Ajmone, Marsan, A. Bianco, P. Giaccone, E. Leonardi, F. Neri, "Router Architectures Exploiting Input-Queued, Cell-Based Switching Fabrics " *Lecture Notes in Computer Science, Dipartimento di Elettronica, Politecnico di Torino, Italy*

[20] Gupta P., "A survey of Input Queued Switches" *TN, Department of Computer Science, Stanford University.*

[21] Anderson, T., Owicki, S., Saxe, J. and Thacker, C. "High Speed Switch Scheduling for local Area networks," *ACM Trans. On Computer Systems. Nov 1993, pp.319-352.*

[22] Ian Stoica, Hui Zhang "Exact emulation of an Output Queuing Switch by a Combined Input Output Queuing Switch" Carnegie Mellon University

[23] Shang-Tse Chuang, Ashish Goel, Nick McKeown, Balaji Prabhakar "Matching Output Queueing with a Combined Input Output Queued Switch " *Proc.INFOCOM 1999, pp 1169 -- 1178, IEEE, New York, March 1999.*

[24] J. G. Dai and Balaji Prabhakar, "The Throughput of Data Switches with and without Speedup", pp 556-564", *Proc INFOCOM2000*

[25] C.-Y. Chang, A Paulraj, and T. Kailith, "A Broadband Packet Switch Architecture with Input and Output Queueing," *IEEE, pp 448-452, 1994*

[26] Charny A. Krishna P., Patel N.S., Simcoe R., "Algorithms for providing bandwidth and delay guarantees in input-buffered crossbars with speedup", *6th International Workshop on Quality of Service (IWQoS'98), Napa, CA, May 1998, pp. 235-244*

[27] Gupta A.K, Georganas N.D., "Analysis of a packet switch with input and output buffers and speed constraints", *IEEE INFOCOM'91, Apr 1991, pp 694-700.*

[28] Mckeown N., A Mekkittikul, V Anantharam, J Walrand, "Achieving 100% Throughput in an input queued switch", *IEEE Trans Comm., vol. 47, no 8, pp 1260-1267, 1999.*

[29] McKeown, Nick, "*Scheduling Cells for Input-Queued Cell Switches,*" PhD Thesis, University of California at Berkeley, May 1995.

[30] McKeown, Nick, Thomas E. Anderson, " A Quantitative Comparison of Iterative Scheduling Algorithms for Input-Queued Switches", *Computer Networks and ISDN Systems, v. 30, n. 24, Dec 14, 1998.*

[31] J.Edmonds and R.M.Karp, "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems", *Journal of ACM 19, 1972, pp. 248-264.*

[32] McKeown N., Izzard M., Mekkittikul A., Ellesick B., Horowitz M., "The Tiny Tera: a packet switch core", *IEEE Micro Magazine*, vol. 17, Feb 1997, pp. 27-40

[33] Duan H., Lockwood J.W., Kang S.M., Will J.D., "A high performance OC12/OC48 queue design prototype for input buffered ATM switches", *IEEE INFOCOM'97, vol. 1,1997, pp.20-8, Los Alamitos, CA, USA*

[34] Duan H, Lockwood J.W., Kang S. Mo, "Matrix Unit Cell Scheduler (MUCS) for input-buffered ATM switches" *IEEE Communications Letters, vol.2, n.1, Jan.1998, pp.20-23.",*

[35] D. Stiliadis, A Varma, "Providing bandwidth guarantees in an input buffered crossbar switch, " Proceedings of IEEE INFOCOM'95, pages 960 – 968, Boston, MA, April 1995

[36] Nan Ni, Laxmi N. Bhuyan, "Fair Scheduling and Buffer Management in Internet Routers" , *Proc Infocom 2002, New York, pp 976-984*

[37] Schoenen R, Post G and Sander G "Weighted Arbitration Algorithms with Priorities for Input Queued Switches with 100% throughput" *Institute for Integrated Signal Processing Systems, Achen University of Technology, Templer, Graben, Aachen, Germany.*

[38] Schoenen R, Post G and Sander G "Distributed Cell Scheduling Algorithms for Virtual-Output-Queued Switches" *Institute for Integrated Signal Processing Systems, Achen University of Technology, Templer, Graben, Aachen, Germany.*

[39] Marsan M.A. Bianco A., Leonardo E. Milia L.,"RPA: A Flexible Scheduling Algorithm for Input Buffered Switches", *IEEE Trans Comm. vol. 47, no 12, Dec 1999, pp 1921-1933.*

[40] Y. Tamir and H-.C. Chi, "Symmetric Cross Bar Arbiters for VLSI Communication Switches," *IEEE Transactions on Parallel and Distributed Systems, vol. 4 no. 1, pp. 13-27 1993.*

[41] Shah D.. Kopikare M. "Delay Bounds for Approximate Maximum Weight Matching Algorithms for Input Queued Switches", *Proc INFOCOM 2002, pp 1024-1031, New York, NY, June 2002.*

[42] Bernhard Suter, T.V. Lakshman, Dimitrios Stiliadis, Abhijit Choudhury, " Efficient Active Queue Management for Internet Routers " *Proc INTEROP '98, Las Vegas, NV, May 1998, pp 365-374*

[43] Stiliadis D., "General Methodology for Designing Efficient Traffic Scheduling and Shaping Algorithms:," *Infocom97, pp. 344-357, 1997, Kobe, Japan.*

[44] Anthony C, Kam and Kai-Yeung Siu "Linear complexity algorithms for QoS support in input queued switches with no speedup", *Arbelof Laboratory for Information Systems and technology, Massachusetts Institute of Technology.*

[45] "Cisco 12000 Gigabit Switch Router", *Product Overview, www.cisco.com Apr 2000*

[46] Gabow ``An efficient implementation of Edmond's algorithm for maximum matching on graphs". *Journal of the ACM, Vol. 23, 221-234, 1976*

[47] Chaney T., Fingerhut J., Flucke M., Turner J.: "Design of a Gigabit ATM switch".Washington University", *St Louis, Wucs 96-07*

[48] Iyer S., Mckeown N., "Making Parallel Packet Switches Practical", *Proc IEEE INFOCOM 2001, pp 1680 – 1687 ,Anchorage, Alaska, 2001*

Appendix A: A Quality of Service Framework.

A Quality of Service Framework

Providing QoS in a network has become an active area of work in the last few years, and many approaches have been investigated. Generally implementing QoS in a network requires a number of technologies to work together. These QoS technologies operate at different layers in the protocol stack and provide different services.

Physical Layer Quality of Service

Physical technologies allow for the separation of traffic. The separation may take the form of wavelengths, Virtual Circuits, ports on a device, or frequencies over the air. This is the simplest form of QoS whereby different levels of QoS are provided through traffic separation at the physical layer. For example different wavelengths may provide different services. This type of QoS works well when the transmission media is inexpensive or abundant, but when the resources are limited it becomes inefficient and expensive, for example the Wireless frequency spectrum.

Link Layer Quality of Service

Each type of link layer has a different type of QoS that can be applied. The most common link layers are Ethernet, ATM, PPP, MPLS, Frame Relay and Mobile wireless technologies. ATM and MPLS are described briefly here.

ATM currently has the most comprehensive QoS support. The ATM forum has created ATM service categories, each with a different QoS traffic management parameters and performance levels. The most common ATM service categories are CBR (Constant Bit Rate), rt-VBR (real-time Variable Bit Rate), and UBR (Unspecified Bit rate). CBR is used for circuit emulation, rt-VBR for real time voice or video service and UBR for unspecified data. There are other less widely used services available; however these are not covered here.

ATM also provides a number of traffic management parameters for each of the ATM service categories. The most important of these are the Peak Cell Rate (PCR), Sustained Cell Rate (SCR) and Cell Loss Priority (CLP). The Cell Loss Ratio (CLR) is the ratio of lost cells to total transmitted cells. Cells are lost due to a number of factors including switch malfunctions, discarding due to non-compliance and network congestion. Switches generally implement some specific cell discard policy. This discard policy has a large impact on the performance of the network and is discussed in some detail in the next subsection. The Cell Transfer Delay (CTD) is the elapsed time between a cells entry and departure of an ATM network. This paramcter is especially important for the CBR and rt-VBR service categories, and the maximum Cell Transfer Delay (mCTD) is defined in the ATM Forum Traffic Management specification for these categories. The Cell Delay Variation (CDV) is another name for jitter was discussed in a previous section.

In order to support the guaranteeing of QoS in the network it is important to enforce the compliance of connections at the entrance to the network. In ATM this traffic policing function is implemented using algorithms such as the Generic Cell Rate Algorithm (see Appendix B for an explanation). Traffic policing is implemented by the network usually at the first switch which traffic entering the network encounters.

Traffic shaping smoothes cell streams, eliminating bursty peaks and cell jitter. The result is a more predictable traffic profile, which has economic benefits because it can be accommodated in a virtual circuit with a lower Peak Cell Rate (PCR). It is important to note that Traffic shaping takes usually place on the client equipment before the data enters the network, or on the ingress side of an edge ATM switch.

Because traffic policing enforces connections compliance to a traffic contract, traffic shaping may make quite a big difference in the performance of an application. For example if compressed video is fed into the network without any traffic shaping, some traffic policing algorithm may discard some of the cells during a burst when the cell rate exceeds the negotiated PCR. Because most video information is contained in certain large frames, discarding cells leads to a severe degradation in the quality of the video.

Congestion avoidance is the action a network takes to avoid circumstances in which flows or aggregated flows no longer receive their associated service levels due to excessive traffic loads at points in the network. This is achieved through various means, including the application of a packet discard policy to provide implicit feedback to host systems to reduce network traffic during congestion.

It is inevitable that at some point a port will become so congested that the buffers on a switch become full and packets must be dropped. The policy which is chosen to select which cells to discard can have a large impact on the performance of the network. If the discard policy is based solely on a fixed per connection threshold, it may cause

unnecessary retransmissions which escalate congestion. The majority of upper layer data packets are composed of more than one ATM cell, and when a cell is dropped it causes corruption of the entire packet.

To prevent this congestion escalation, Early Packet Discard (EPD) and Partial Packet Discard (PPD) schemes have been proposed for ATM to discard cells on a Frame Basis. Partial Packet Discard is a reactive measure and acts after a switch drops a cell by dropping the remaining cells in a frame, except for the last cell. The last cell is not discarded, since it is used by the end device to determine the start of the next packet. PPD is not a perfect solution because cells before the first dropped cell are still transmitted.

Early Packet Discard is a proactive measure that acts before a cell is lost due to buffer overflow. EPD recognizes that cells are about to be dropped and begins dropping cells from the new packets. This allows an entire packet to be discarded rather than only a partial packet, thereby minimizing the wasted bandwidth.

MPLS provides for 2 different forms of QoS support. There are 8 service classes which support emission and discard priorities. MPLS also supports a number of traffic management parameters to define the behaviour the traffic will receive as it traverses a particular Label Switched Path. One of MPLS major benefits is that it makes it possible to engineer paths across a network.

As traffic traverses a network it can often take different paths depending on the network technology. For example routed IP networks are connectionless; a packet can take different paths. This can lead to unpredictable QoS. Because network operators want to offer guaranteed service level agreements, network paths are engineered to provide guaranteed QoS performance. Traffic that is within the service level criteria can be steered along the traffic engineered Paths and obtain a predictable QoS level.

Network Signalled QoS

MPLS and ATM use signalling protocols to request a desired QoS from other network nodes prior to connection establishment. This is known as Connection Admission Control or CAC. ATM uses a protocol called PNNI (Private Network - Network Interface) to accomplish this. MPLS uses a protocol called LDP (Label Distribution Protocol) to set up Label Switched Paths (LSP). To Signal QoS for these traffic engineered paths, MPLS uses RSVP-TE (Resource Reservation Protocol for Traffic Engineered paths) or CR-LDP (Constraint based Routed LDP). Both protocols achieve similar goals through different

approaches. The end result is that the network should be able to offer bandwidth and delay guarantees to a connection.

QoS Measurement and Monitoring

In order for service providers need to services such a IP telephony, video on demand etc, the network QoS must be monitored and measured to ensure that the service is being adequately supported. Furthermore since the QoS performance may be specified in a Service Level Agreement (SLA) the service provider needs to ensure that the network is providing the performance as specified. The SLA typically consists of parameters such as maximum packet loss, maximum packet loss, and maximum packet delay.

The SLA specifies the terms and conditions of the service being offered. Once a service provider can accurately measure the network capabilities and provide a guaranteed performance level, he can confidently offer a billable service to his subscribers. Subscribers also want to monitor network performance to ensure that they receive the services to which they subscribe.

Appendix B: Design of an ATM Scheduler Simulator.

This section gives an overview of the software design used in the development of an ATM scheduler test- bed. It is intended as an introduction to the problems encountered when using simulation as a means of evaluating scheduling algorithms and as an introduction to the code.

High Level Design

The design of simulation can be broken up into four main tasks.

1 The generation of a cell stream

2 The queuing and dequeuing of cells in whatever manner is chosen for example FIFO, per VC or per Traffic Class

3. The solution of a maximal or maximum size matching algorithm.

4. The analysis of the switched cells to obtain relevant results

The basic flow of the simulator is shown in figure A.1

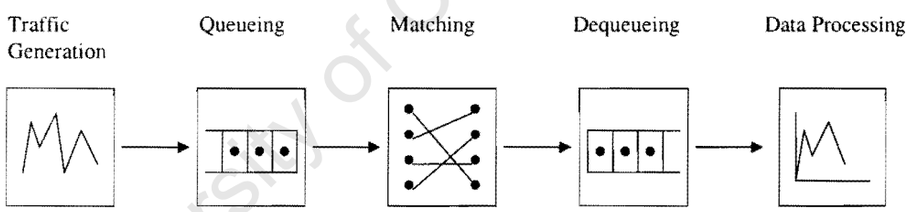

Figure A.1 Modular Flow of Scheduler Simulator

Other requirements that impacted on the design where that the simulation should be scalable and that the scheduler used should be easily interchangeable.

We now discuss each of the above points separately,

Traffic Generation:

Traffic generation is one of the main problems in evaluating switches and schedulers. The models which generate realistic traffic generally run very slowly in software. If real world traffic profiles are used the amount of disk space used is massive and there is a massive slowdown in the speed at which the simulation runs. The approach taken in this project was to use one of the simplest traffic models, Markov modulated chains, other approaches considered were modified Brownian motion . This model allows traffic to be generated on the fly, or as the simulation runs. Another aspect that was designed into the simulator is that it is easy to switch to another traffic model if desired.

Queuing and Dequeuing

The main function of the enqueuing block is to maintain per virtual circuit queues. For each cell that is enqueued a timestamp is stored. This is necessary to calculate the delay through the scheduler. The requirement to store each cells timestamp means that a very large amount of virtual memory is necessary to run the simulation under heavy traffic conditions. The design makes it possible to switch to other queuing methods such as per traffic class and FIFO queueing by changing the command line parameters.

In the case where pure output queuing is simulated, the matching algorithms are bypassed and the dequeueing block finds the best cell to dequeue from the relevant input queues as if they had been switched through the non-blocking infinite bandwidth fabric.

Matching

By far the most complex part of the code is the matching algorithms. The most complex to implement was the maximum weighted matching algorithm. This required an implementation of Gabows ``An efficient implementation of Edmond's algorithm for maximum matching on graphs".[47]. When this matching algorithm is being simulated over 95% of the processor time is spent solving it. Its running time is $O(N^3)$ so the simulation runs very slowly for larger switch sizes. Other algorithms simulated are Maximum Size Matching, iSLIP, PIM, iFS, WiM, and FIFO queueing.

The analysis of the Simulator output.

Once cells have been switched, relevant data from each cell are written to a file. Once the simulation is completed the data analysis program is launched which parses the data, extracting relevant information such as jitter, delay, average delay, total throughput etc.

Configuring the Simulator

The simulator currently uses command line parameters to determine what scheduler, traffic model and traffic load to use for a specific run.
The format of the command is as follows :
Schedsim [scheduler] [load] [traffic model]

The values for scheduler are :

Virtual Output Queueing	0
Per Port Queueing	1
FIFO queueing	2
Longest Queue First	3
Slip	4
iSLIP	5
Parallel Iterative Matching	6
Weighted PIM	7
Iterative Fair Scheduling	8
Maximum Size Matching	9
Maximum Weighted Matching	10
Output Queueing	12
Parallel Weighted Matching	13

The values for load are : 0 to 100

The values for traffic model are :
Bursty 1
Uniform 0
The simulator can also be quickly reconfigured with different switch sizes, number of queues, or number of connections. These are currently set using defines in the code :

```
#define NO_OF_GROUPS 2
#define NO_OF_PORTS 15
#define NO_OF_QUEUES 7
```

Running a Simulation

In order to extract information about the performance of a scheduler, the scheduler must be tested under a range of different traffic loads. It is also necessary to obtain the steady state result for each type of traffic. An automated script is used to run several simulation runs one after another. To obtain a single graph as shown in chapter 6 typically took about 3 or 4 days to run.

First the simulator is run, followed by the data analysis program, which analyses the data written by the simulator, Every time the data analysis program runs it writes its results into the same data file. A couple of iterations would look like the example below. These commands are executed from a script file.

```
mwmsimple.exe 10 20 0
average.exe
del outfile.dat
mwmsimple.exe 10 25 0
average.exe
del outfile.dat
mwmsimple.exe 10 30 0
average.exe
del outfile.dat
mwmsimple.exe 10 35 0
average.exe
del outfile.dat
mwmsimple.exe 10 40 0
```