Multi-stage Queuing with Iterative Probabilistic Scheduling of IP traffic for QoS Provisioning

Oladayo M. Salami

This thesis is submitted in partial fulfillment of the academic requirements for the degree of Master of Science in Electrical Engineering in the Faculty of Engineering and The Built Environment University of Cape Town August 2006
As the candidate's supervisor and co-supervisor, we have approved this dissertation for submission.

Name: Professor H. Anthony Char.

Signed: 
Date: 23 November 2006

Name: Professor Mghele E. Dlodlo

Signed: 
Date: 

ii
Declaration.

I hereby declare that: (1) the above thesis is my own unaided work, both in conception and execution, and that apart from the normal guidance of my supervisor, I have received no assistance apart from that stated below; (2) except as stated below, neither the substance or any part of the thesis has been submitted in the past, or is being, or is to be submitted for a degree in the University or any other University.

I am now presenting the thesis for examination for the Degree M.Sc in Electrical Engineering. I also grant the University free license to reproduce the above thesis in whole or in part, for the purpose of research.

Oladayo M. Salami

Name

November 23, 2006

Date
To

My Heavenly Father,

My Jehovah Nissi,

My Jehovah Jire,

You are everything I long for.

and

My Parents.
Abstract

Queueing and scheduling are packet processing functions, which provide quality of service (QoS) at the first level of the QoS architecture where routers reside. With the continued increase in network speed, queueing and scheduling of traffic needs to be performed at higher speed. In addition, processing co-existing IP traffic from different applications requiring QoS may complicate these functions. Virtual Output Queuing (VOQ) is a strategy that can queue traffic at line rate and achieve 100% throughput with an effective scheduling algorithm. It maintains a separate queue for each output port at each input port. With N input ports and N output ports in the router, there are $N^2$ such virtual output queues at the input. The challenge is to schedule the packets at the head of these $N^2$ queues. Scheduling algorithms have either a deterministic or a probabilistic scheduling policy. QoS guarantee for IP traffic may also be deterministic or probabilistic. Most schedules for VOQ are deterministic algorithms. The deterministic algorithms that guarantee QoS are difficult to implement (e.g. iterative Longest Queue First and iterative Oldest Cell First). Other deterministic algorithms that are relatively simpler to implement do not guarantee QoS (e.g. Parallel Iterative Matching and iterative SLIP). Although deterministic algorithms usually have high throughput they schedule traffic either arbitrarily or by giving priority to some traffic causing starvation of other traffic. These algorithms may become inadequate in the Next Generation Network (NGN) in which converging networks operate at high speed and traffic with different QoS requirement co-exists.

This research proposes a probabilistic approach to QoS provisioning at the first level of the QoS architecture, designs a probabilistic scheduling algorithm for the VOQ strategy and studies its performance through simulation. This scheduling algorithm termed Iterative Probabilistic Scheduling (IPS) is used in a Multi-stage Queuing and Scheduling Architecture (MQAS). It guarantees a probabilistic QoS to ensure fairness and to prevent traffic starvation. Its scheduling policy is a basic mathematical operation. The MQAS, the IPS algorithms and the above-mentioned algorithms were simulated with the OMNeT++ discrete event simulator. From simulation results, IPS achieves a higher scheduling performance than the other algorithms. Future work to implement IPS or a network processor is recommended.
Acknowledgements

My profound gratitude goes to my thesis advisors Professor H. Anthony Chan and Professor Mqhele E. Dlodlo for their support, encouragement and motivation throughout the period of my Master’s degree. Thank you, professors for giving me the opportunity to work with you. The advice, exposure and coaching you gave me created memorable moments which would linger in my mind.

I would like to thank the whole Chan-group, past and present for being there to provide a friendly hand of support whenever I needed it. CRG lab colleagues, thank you for the challenges and motivation during the CRG presentation periods, they made me stronger.

Nathaniel, Nathie, Navonaeli and Nancy, thank you for your prayers and for being there during my unwinding times.

I am infinitely grateful to my family members, for making my dream for a postgraduate education come true. Mum, though you are not here any longer, I am very grateful for the discipline you cultivated in me: you have always been my inspiration. Thank you, Dad, Oladipo, Olawunmi, Olajide, Olanrewaju and Olayemi, for the constant encouragement, prayers, financial support and love given to me over the years. You did not let the seed die, you watered it and now it has blossomed, forging ahead with increased might, it cannot be limited.

Mobolaji Bello, thank you for the constant emotional support, encouragement, and love, you are mostly appreciated.

Finally, and most importantly I wish to thank God for His guidance, strength, wisdom and for His excellent Spirit who assisted me in conducting this research.
# Table of Contents

Abstract............................................................................................................ V

Acknowledgements ............................................................................................. Vi

List of Figures...................................................................................................... X

List of Tables....................................................................................................... 3

Glossary................................................................................................................ viii

1 Introduction..................................................................................................... 1

1.1 The trend in Communication Networks..................................................... 1

1.2 Challenges of Queuing and Scheduling.................................................... 4

1.2.1 Previous Research ................................................................................. 7

1.2.2 Objective and scope of research............................................................ 8

1.3 Thesis Organization.................................................................................... 9

2 Network Layer Quality of Service (QoS)...................................................... 10

2.1 QoS Provisioning...................................................................................... 10

2.2 QoS Parameters....................................................................................... 11

2.3 QoS Guarantees....................................................................................... 13

2.3.1 Deterministic and Probabilistic QoS guarantee.................................... 14

2.4 QoS Architecture..................................................................................... 15

2.4.1 Router Functions................................................................................ 18

2.4.2 Packet Processing.............................................................................. 18

3 Single-stage Queuing and Scheduling......................................................... 22

3.1 Queuing................................................................................................. 22

3.1.1 Input Queuing (IQ)............................................................................ 23

3.1.2 Output Queuing (OQ)........................................................................ 25

3.1.3 Virtual Output Queuing (VOQ)........................................................... 26

3.2 Scheduling and Algorithms for Virtual Output Queuing......................... 27

3.2.1 Deterministic Scheduling Algorithms.............................................. 29

3.2.2 Probabilistic Scheduling Algorithms.............................................. 33

3.3 Issues in Queuing and Scheduling.......................................................... 33
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-Stage Queuing and Scheduling</td>
<td>36</td>
</tr>
<tr>
<td>4.1 Multi-Stage Queuing concept</td>
<td>36</td>
</tr>
<tr>
<td>4.2 Multi-Stage Queuing and Scheduling Architecture</td>
<td>38</td>
</tr>
<tr>
<td>4.3 Proposed Iterative Probabilistic Scheduling (IPS) Algorithm</td>
<td>40</td>
</tr>
<tr>
<td>4.3.1 IPS Model</td>
<td>40</td>
</tr>
<tr>
<td>4.3.2 Initialization Process of the IPS</td>
<td>42</td>
</tr>
<tr>
<td>4.3.3 Operation of the IPS algorithm</td>
<td>43</td>
</tr>
<tr>
<td>4.3.4 IPS Scheduling process</td>
<td>43</td>
</tr>
<tr>
<td>4.3.5 Analysis of the IPS Computational Time Complexity</td>
<td>44</td>
</tr>
<tr>
<td>4.3.6 IPS properties</td>
<td>47</td>
</tr>
<tr>
<td>Performance Analysis of Scheduling Algorithms for the VOQ stage of MQAS</td>
<td>49</td>
</tr>
<tr>
<td>4.4 Existing Scheduling Algorithm</td>
<td>49</td>
</tr>
<tr>
<td>4.4.1 Simulation Scenario</td>
<td>49</td>
</tr>
<tr>
<td>4.4.2 FIFO scheduling</td>
<td>51</td>
</tr>
<tr>
<td>4.4.3 PI3M Scheduling</td>
<td>52</td>
</tr>
<tr>
<td>4.4.4 iLQF Scheduling</td>
<td>54</td>
</tr>
<tr>
<td>4.4.5 iOCF Scheduling</td>
<td>55</td>
</tr>
<tr>
<td>4.4.6 iSLIP Scheduling</td>
<td>56</td>
</tr>
<tr>
<td>4.5 IPS Scheduling</td>
<td>60</td>
</tr>
<tr>
<td>4.5.1 IPS Latency vs. Utilization with different router sizes</td>
<td>64</td>
</tr>
<tr>
<td>4.5.2 Analysis of IPS performance with increase in Number of ports</td>
<td>66</td>
</tr>
<tr>
<td>4.6 Comparison of IPS, PIM, iSLIP, iOCF and iLQF</td>
<td>68</td>
</tr>
<tr>
<td>Conclusions</td>
<td>72</td>
</tr>
<tr>
<td>4.7 Recommendation</td>
<td>74</td>
</tr>
<tr>
<td>4.8 Future Work</td>
<td>74</td>
</tr>
<tr>
<td>References</td>
<td>76</td>
</tr>
<tr>
<td>Appendix A: MQAS Architecture</td>
<td>84</td>
</tr>
<tr>
<td>A.1: Router ned code</td>
<td>84</td>
</tr>
<tr>
<td>A.2: Router network module in OMNeT++ version 3.2</td>
<td>85</td>
</tr>
<tr>
<td>A.3: VOQ ned code</td>
<td>85</td>
</tr>
<tr>
<td>A.4: Input port compound module in OMNeT++ version 3.2</td>
<td>86</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1-1 Relationship between packet processing algorithms and processor speed....... 4
Figure 1-2 Packet processing functions ............................................................... 6
Figure 2-1 Network’s End-to-End QoS Service Levels ........................................ 13
Figure 2-2 Hierarchical QoS Architecture .......................................................... 16
Figure 3-1 Input Queuing .................................................................................. 23
Figure 3-2 Head of Line Blocking ....................................................................... 24
Figure 3-3 Output Queuing ................................................................................ 25
Figure 3-4 Virtual Output Queuing ..................................................................... 26
Figure 4-1 Multi-stage Queuing and Scheduling Architecture ............................ 38
Figure 4-2 Input requests to Output .................................................................... 42
Figure 4-3 Flow Chart of the iPS algorithm ....................................................... 43
Figure 5-1a. Queue model ................................................................................ 50
Figure 5-1b. Simulation Environment .................................................................. 51
Figure 5-2 Latency vs. Utilization for FIFO scheduling algorithm ..................... 52
Figure 5-3 Latency vs. Utilization for PIM-1 scheduling algorithm ..................... 53
Figure 5-4 Latency vs. Utilization for PIM-4 scheduling algorithm ..................... 53
Figure 5-5 Latency vs. Utilization for iLQF-1 scheduling algorithm ..................... 54
Figure 5-6 Latency vs. Utilization for iLQF-4 scheduling algorithm ..................... 55
Figure 5-7 Latency vs. Utilization for iOCCF-1 scheduling algorithm ........................................... 56
Figure 5-8 Latency vs. Utilization for iOCCF-4 scheduling algorithm ........................................... 56
Figure 5-9 Latency vs. Utilization for iSLIP-1 scheduling algorithm .............................................. 57
Figure 5-10 Latency vs. Utilization for iSLIP-4 scheduling algorithm ............................................. 58
Figure 5-11 Latency vs. Number of ports at low utilization for iSLIP-4 scheduling algorithm ............. 59
Figure 5-12 Latency vs. Number of ports at high utilization for iSLIP-4 scheduling algorithm .......... 59
Figure 5-13 Delay of packets at 10% utilization with IPS scheduling algorithm ....................... 61
Figure 5-14 Delay of packets at 30% utilization with IPS scheduling algorithm ....................... 61
Figure 5-15 Delay of packets at 80% utilization with IPS scheduling algorithm ....................... 62
Figure 5-16 Delay of packets at 90% utilization with IPS scheduling algorithm ....................... 62
Figure 5-17 Delay of packets at 100% utilization with IPS scheduling algorithm ....................... 63
Figure 5-18 Latency vs. Utilization for IPS scheduling algorithm ................................................ 64
Figure 5-19 Latency vs. Number of ports for IPS scheduling algorithm ...................................... 66
Figure 5-20 Performance of all algorithms when N=2 ............................................................... 68
Figure 5-21 Performance of all algorithms when N=32 .............................................................. 69
Figure 6-1 Data collection region for Simulation Scenario ......................................................... 75
Figure 6-2 IPS delay performance versus line rate ................................................................. 75
## List of Tables

Table 3-1 Classification of Deterministic VOQ Scheduling Algorithms .................................. 30

Table 4-1 Comparison of existing Queuing Strategies ................................................................. 37

Table 4-2 Execution times of IPS operation ............................................................................... 46

Table 5-1 FIFO Performance as N increases ................................................................................. 52

Table 5-2 i-SLIP -4 Performance as N increases .......................................................................... 58

Table 5-3 IPS Performance as N increases ................................................................................... 65

Table 5-4 IPS delay performance for different line rates when N=2 ........................................ 65

Table 5-5 Comparison of FIFO, i-SLIP -4, PIM-4, iOCF-4 and iLQF-4 ..................................... 71

based on Scheduling requirements .............................................................................................. 71
Glossary

Clock Cycle: the time for one clock period, usually of the processor clock, which runs at a constant time rate.

Clock rate: The speed of a processor or a system measured as the number of cycles per second.

Internet Protocol (IP): A data-oriented protocol used for communicating data across the network.

Packet: A packet is a block of data routed between an origin and a destination.

IP datagram: the structure of an IP datagram consists of a header followed by a variable length data field.

Traffic: Refers to data transmitted over a network.

RFC: Acronym for Requests For Comments which is a series of numbered Internet informational documents and standards.

Data Rate: The amount of data transferred per second by a communication channel. It can also refer to the speed of travel of a given amount of data from one place to another. The greater the bandwidth of a given communication channel the higher the data transfer rate. Data rate is usually measured in bits per seconds.

Latency: it is a synonym for delay, and refers to the amount of time it takes for a data to move from one point to another.

Network speed: the rate at which individual bits are transmitted over a network.

NGN: Next Generation Network.
1 Introduction

1.1 The trend in Communication Networks

Communication technology began its major boost with the Internet. The Internet's concept began in the late 1950s at the Defense Advanced Research Projects Agency (DARPA) for research purposes. Its purpose was to interconnect network systems for data communications [1]. Over the years, the concept went through various developmental stages and today, the Internet refers to a global communication system, which provides a seamless interconnection of networks [2]. It provides a high-speed communication between these networks, which comprises hundreds of millions of computers around the world.

One major technology that came up after the Internet is Ethernet LAN, invented by Robert Metcalfe in the early 1970's. Ethernet LAN interconnects host computers within a building for the exchange of information. Since the inception of Ethernet LAN technology, it has continued to evolve towards higher data rates with the 10 Mbps and 100 Mbps and the Gigabit Ethernet used today [2] [3]. Along with the Ethernet LAN were Metropolitan Area Network (MAN) and Wide Area Network (WAN) for long distance communication between buildings in different areas within a metropolis or a country, and between different countries respectively.

These networks are interconnected with network elements e.g. routers, which virtually create a unified and single network called the INTERconnection of NETworks (INTER-NET) [4]. The router interconnects these networks by transmitting traffic from one part of the network to another. Apart from transmitting traffic, the router also queues and schedules traffic to ensure that the quality of service requirements of these traffic are met. Due to the trend in communication networks there has been a need for better performance of these functions in the router. The next few paragraphs give a summary of this trend.

In the last few decades, a drastic change occurred in the Internet's services and infrastructure. Firstly, there has been a significant growth in its structure and service market segment because of the introduction of new service applications. Secondly, there is a rise in
network speed because of the growth in communication traffic. In turn, the above-mentioned changes introduced several challenges to the Internet's architecture. A major factor contributing to these challenges is the diversity in services created by applications and protocols. These applications may be Voice over IP (VoIP), World Wide Web, streaming media/video, multimedia, video and IP telephony, and Video on Demand (VoD) [2][5].

The rapid global expansion of the Internet has also influenced all other areas of communication. New communication technologies are evaluated based on their potential role in the Internet [2]. A similar expansion in terms of data rate occurred with Ethernet LAN. Ethernet now operates at 10Gbps, emerging from 10Mbps in the 90's, and targeting up to 100Gbps in the near future for high-speed communication [2] [3].

GSM networks known as second-generation networks (2G) revolutionized mobile communication. It is a circuit-switched network originally designed for voice traffic [6]. It supports mobile communication between handheld terminals at low cost and provides good spectral efficiency with data rate up to 9.6kbps [7]. Later, the concept of General Packet Radio Service (GPRS/2.5G) came up to augment GSM-enabled mobile services by providing packet-switched data services to mobile users at 160kbps [8]. These data services include internet-browsing, email, fax and unified messaging [6]. The early part of the new century witnessed the emergence of third generation networks (3G) such as the Universal Mobile Telecommunication System (UMTS). UMTS offers increased bandwidth and target data rates of 144 kbps (for satellite and rural outdoor), 384 kbps (for urban outdoor), and 2048 kbps (for indoor and low range outdoor) [9]. 3G (UMTS) provides broadband capabilities to support more voice and data customers at lower cost than 2G.

An examination of these trends in communication networks shows that future networks known as Next Generation Networks (NGN) and heterogeneous networks like 4G have to target a speed of 70-200Mpbs. NGN is defined by ITU-T as an all IP (packet-based) network which provides telecommunication services to users. According to ITU-T, NGN supports packet-based transfer, generalized mobility and ubiquitous access to services. Both fixed and mobile networks converge in NGN. NGN inter-works with existing legacy networks via open interfaces [10]. The concept of NGN takes into consideration the need to converge the Internet and
telecommunication. This need, which is characterized by the growth in user traffic and the extraordinary increase in data rates, will be followed by a wave of convergence in coming years [11]. Convergence will inevitably give further rise to more and new demands on packet networking technology [11].

With the increasing trend and diversity in user traffic, traffic from applications such as full motion video, Gaming and TV broadcasting are approaching the threshold of what the present network speed can offer. Therefore, a network's ability to support greater capacity and higher speeds becomes paramount to avoid a deluge of congestion and to ensure quality of service (QoS) provisioning to the different traffic types co-existing within the network [12].

The input and output links in a network are usually optical fibres operating at the speed of light. In addition, each fibre may use multiple wavelengths and there are multiple fibres. Therefore, the data rate is very high and can continue to increase in future. On the other hand, routers today are primarily electrical. Thus, it is desirable that the speed of the router be very high and continues to increase in future.

While new applications that differ immensely from their predecessors are emerging and demand for higher network speed is increasing, network infrastructures have to keep improving in performance [13]. The router is a basic building block in communication networks. A router processes user information packets (IP traffic) and forwards them to their appropriate destination network. The processes performed on packets include policing, shaping, classification, queuing and scheduling [14]. Although routers' primary role is to forward IP traffic from a set of input links to a set of output links, they must also deal with heterogeneous link technologies [4]. In addition, the exponential surge in user traffic and the insatiable need for higher network speed require routers to function at a rate of many gigabits per second of aggregate traffic [13]. As a result, the amount of time available to process IP traffic within these network elements decreases with higher network speed [14]. The co-existence of IP traffic generated by diverse services and applications in today's networks also require that routers provide quality-of-service (QoS) to these traffic [11]. Consequently, there is a need for high performance routers, which process IP traffic at line speed (network speed) while providing QoS and which scales with increasing network speed [15].
High speed is achievable in routers through parallelism and pipelining. For example, high-speed routers have multiple processing units to handle traffic that arrives from multiple interfaces. However, high-speed processing still requires intelligent data structures and algorithms. These algorithms need to be optimized to assist in higher speed processing.

1.2 Challenges of Queuing and Scheduling

Due to the challenge for a high-speed, robust and scalable network, this research looks into a better approach to designing the queuing and scheduling packet processing functions of routers.

The basic components of a router include the input port, the output port, the switch fabric, and the processor. The ports are points of attachment to physical links. Input ports are points of entry for incoming packets while output ports are the interface to outgoing transmission links. The switch fabric is a hardware mechanism that interconnects input ports and output ports. The processor is a very important component, which can be implemented in a hybrid of software and hardware known as a network processor.

The processor executes packet-processing algorithms including queuing and scheduling. The performance of routers can be improved by increasing processor speed (path A in Figure 1-1), by implementing efficient packet processing algorithms or protocols (path B in Figure 1-1) or by both (path C in Figure 1-1).

![Figure 1-1 Relationship between packet processing algorithms and processor speed.](image-url)
Several processors are used to perform operations in parallel and in a pipeline to facilitate high speed packet processing. However, to minimize total computation time, CPU designers will still need to optimize processors’ performance by adding efficient instructions that can perform tasks faster [2].

An efficient algorithm can operate at higher speed. Therefore, having an efficient algorithm can be more cost effective than increasing processor speed alone. It may alleviate the amount of work of the processor, which may also help when the router migrate in the future towards a lower cost and smaller size system such as System-on-Chip (SoC).

This thesis proposes and studies an approach to developing efficient queuing and scheduling to aid the router’s performance.

Two issues in designing queuing and scheduling in such routers include:

1. Facilitating high-speed queuing and scheduling of co-existing and different types of IP traffic without violating their QoS requirements.
2. Designing scheduling functions, which can scale with increasing line speed.

The first issue involves the identification of a high-speed router architecture that can assist in providing optimal packet processing. Some proposed architectures for high-speed routers are multi-stage. Some of these multi-stage architectures require scheduling algorithms while others do not. The Multi-stage Queuing and Scheduling (MQAS) architecture simulated in this research is a two-stage queuing architecture, which requires a scheduling algorithm. Chapter four discusses the concept of different types of multi-stage architectures.

The second issue is encountered in scheduling when packets arriving simultaneously on more input ports than one are destined to the same output port [16]. These packets have to be processed at line rate and may require QoS. Therefore, scheduling functions have to be scalable enough to keep up with the pace of increasing network speed and provide QoS to traffic.

Figure 1-2 illustrates some packet processing functions within the router. These include classification, queuing, and scheduling. Queuing and scheduling are the focus of this research. They are packet-processing functions, which ensure QoS provisioning to IP traffic. Queuing is a
strategy for buffering IP traffic while scheduling determines the order for transmitting them to their destination output port in the router. Scheduling is a major function that provides QoS guarantee to traffic [17]. For routers used in high-speed networks, the scheduling algorithm (scheduler) must be able to provide high throughput and low latency [1]. To prevent starvation of traffic, it is essential for routers to have schedulers that are fair in providing QoS at line speed.

![Diagram of packet processing functions](image)

**Figure 1-2 Packet processing functions.**

Virtual Output Queuing (VOQ) is a queuing strategy that can still achieve 100% throughput with an effective scheduling algorithm without requiring higher processing speed than the line rate. It maintains a separate queue for each output port at each input port [20] [24]. With N input ports and N output ports in the router, there are $N^2$ such virtual output queues at the input. Therefore, the large number of queues in VOQ makes traffic scheduling more difficult. VOQ’s fundamental problem is finding the scheduling algorithm that will fairly resolve contention between packets at the head of the $N^2$ virtual queues at the input. The problem is similar to bipartite graph matching and to rooks on chessboard problems [14] [19] [20]. Research works prove that 100% throughput can be achieved in VOQ with an effective scheduling algorithm [1] [19] [20].
1.2.1 Previous Research

A traditional scheduling algorithm is the First In First Out (FIFO) algorithm. FIFO is a simple algorithm, which schedules packet by forwarding only the first packet in each queue at a time slot. A number of authors including Anderson et al, McKeown et al, and Chen et al have discussed and designed various scheduling algorithms for the VOQ. A discussion and analysis of some of these algorithms are provided in this thesis.

The first algorithm discussed is Parallel Iterative Matching (PIM) proposed by Anderson et al. [1] [18] [20]. PIM uses randomness to find a maximal matching between inputs that have traffic queued for transmission and outputs that have queued packets (at the inputs) destined for them.

Another algorithm is the iterative SLIP (iSLIP), which is a variation of PIM and an enhancement to SLIP algorithm proposed by McKeown et al [18]. iSLIP provides better fairness to IP traffic than PIM [18]. It uses round robin to schedule traffic in each active input to their destined output in turns [20] [21].

The last two algorithms are iterative Longest Queue First (iLQF) and iterative Oldest Cell First (iOCF). These are iterative versions of LQF and OCF algorithms. They consider some information from each queue before finding a maximal weight match to schedule packets [18]. The information may be the occupancy of a queue, or the waiting time of queued packets [18] [20]. iLQF gives preference to queues with larger occupancy, while iOCF considers packets that have been waiting longest.

Although a lot of research is going on, and several algorithms have been proposed for VOQ, most of these algorithms have a drawback. For example, they are either high in time complexity or do not provide QoS guarantee to IP traffic. [1] [18]. Nonetheless, the limitations in each algorithm are a trade-off of some requirements in order to achieve others.
1.2.2 Objective and scope of research

The goal of this research is to introduce, describe and evaluate a probabilistic scheduling of traffic queued at the head of N² input queues in the VOQ strategy. To achieve the objective of this thesis, a Multi-stage Queuing and Scheduling (MQAS) architecture was simulated. MQAS is a two-stage Queuing and Scheduling architecture that breaks the VOQ into an Input Queuing (IQ) and an Output Queuing (OQ). Instead of the traditional IQ strategy that tends to match one input port to one output port, VOQ is at the first stage of the architecture while OQ is at the second stage. VOQ is a scalable strategy, which enables the queuing of traffic at line rate and overcomes HoL blocking. Even though various multi-stage architectures exist for routers, the scope of this thesis is limited to the simulation of a two-stage architecture that uses a scheduling algorithm. The use of a scheduling algorithm at the VOQ stage facilitated the realization of a probabilistic scheduling of traffic and the guaranteeing of a probabilistic QoS to traffic. The Iterative Probabilistic Scheduling (IPS) algorithm was used at the VOQ stage and the OQ stage used FIFO to schedule packets. The scheduling processes are within a single router. The proposed IPS algorithm aims at:

- Achieving an effective, high speed and fair scheduling of IP traffic that arrives at the router’s input simultaneously.

- Provisioning probabilistic QoS to all traffic without causing the starvation of any traffic.

- Providing a scalable scheduling policy.

The research aims at achieving a probabilistic approach to scheduling and provisioning QoS at the first level of the QoS architecture. A probabilistic approach prevents the starvation of traffic. It ensures fairness to all traffic because it provides neither strict priority nor reserves resources but allows all traffic to contend for transmission. Since QoS provisioning is a cumulative function of various mechanisms in the QoS architecture, a probabilistic approach to QoS provisioning at the first level of the architecture is proposed for NGN.
This thesis presents the mathematical model and simulation performance analysis of the IPS algorithm. A comparative study and simulation performance analysis of some existing VOQ scheduling algorithms is also presented. From the simulation results, while FIFO became unstable after 58.58% utilization, PIM, iOCT, iSLIP and iLQF maintained a stable throughput until utilization reached 95% and 98%. However, IPS maintained a stable throughput until utilization reached between 95% and 98%. The algorithms were evaluated in a single router, as packets move from the input ports to the output ports. The simulation set-up assumes a router implementing a packet switching system using a crossbar switch fabric.

1.3 Thesis Organization

Chapter two presents a background on the Network Layer Quality of Service, router functions, and packet processing. Chapter three is a literature review of Single Stage Queuing and Scheduling. The Multi-stage Queuing and Scheduling (MQAS) Architecture with the proposed Iterative Probabilistic Scheduling (IPS) algorithm are discussed in chapter four. Chapter five gives a performance analysis of some existing VOQ scheduling algorithms such as FIFO, PIM, iLQF, iOCF and iSLIP. The simulation analysis of IPS with a comparison of all the algorithms discussed is also presented in this chapter. Finally, a concluding discussion, recommendation, and future work are presented in chapter six.
2 Network Layer Quality of Service (QoS)

2.1 QoS Provisioning

This chapter provides an overview of QoS provisioning within a network with a focus on Network Layer QoS and the packet processing functions of a router. The chapter discusses the QoS parameters used in controlling and allocating resources to users’ traffic. The differences between deterministic and probabilistic QoS guarantee are explained. A brief discussion of some of the mechanisms that enable QoS, their placement in the QoS architecture, and the time they function on the QoS architecture is given. Among other functions queuing and scheduling, which are the focus of this thesis, assist in ensuring QoS on the first level of the QoS architecture.

In the upcoming convergence and ubiquitous era, the expected increase in network speed and network traffic makes QoS provisioning imperative in the deployment of NGN. Therefore, interconnecting network elements for Next Generation Networks must be able to provide quality of service to traversing IP traffic.

Quality of Service (QoS) is a generic term, which takes into account several techniques and strategies that assure users of predictable service from the network and its components. QoS also refers to the capability of a network to provide better service to selected traffic [22]. It may be defined in either qualitative or quantitative terms [23].

QoS has been an important area of research and still remains a research issue [22] [24] [25]. One of the challenges of QoS is a major feature of NGN, which is the consolidation of all network infrastructures into a single multi-service platform [26]. This integration of networks is termed convergence and it is a market driver for NGN. Convergence entails one infrastructure transporting various applications such as voice, data, video, mobile TV and VoIP. Another challenge is the proliferation of service applications that demands QoS [27].

Originally, the Internet did not support QoS because early network applications such as e-mail, telnet and FTP were given best effort service [28]. However, best effort QoS can no longer be applied because of the existence of real-time applications. Some examples of these
applications are tele-conferencing and e-commerce [29] [30]. The coexistence of these applications arouses the need for provisioning QoS to traffic that require stringent quality bounds in order to have a quality output. The challenges posed by applications needing QoS have resulted into a demand for more research in this area.

2.2 QoS Parameters

Output quality bounds differ with applications. Applications generate traffic at varying rates and usually require that networks carry their traffic with an awareness of their output quality bound. Therefore, all applications have different requirements regarding the handling of their traffic in the network. These applications are more or less flexible with their traffic requirement bounds. Four main factors that significantly affect QoS requirements at the application level include:

1) Interactive and non-interactive: the network's response time to applications affected by these factors is a key to their success. While interactive factors place a stricter requirement on delay within the network, non-interactive factors are not affected by such delay [31].

2) Tolerant and intolerant: these factors determine the degree to which an application can cope under certain network conditions. Tolerant applications bear with a degrading network condition and intolerant applications may not work even with the least downgrade [31].

3) Adaptive and non-adaptive: refers to the ability of an application to control and adjust its traffic in terms of the network condition. An adaptive video application can slow down its data rate when network feedback indicates the occurrence of congestion. However, a non-adaptive application in the same situation will fail [31].

4) Application criticality: determines the strictness of the QoS requirement. In the case of a video conferencing application, users can express the required quality of service along various critical dimensions such as colour, frame rate and frame resolution. Along each dimension, the application works at different levels of performance that affects the output's quality. For example, in the colour dimension, black and white frames, grayscale frames and coloured frames gives different levels of quality to video application [31].
At the application level, the above factors assign diverse quality requirements, which are mapped onto the network level in terms of QoS parameters [25]. QoS parameters are the performance parameters used for measuring QoS at the network level. These parameters are responsible for characterizing the QoS bounds of traffic generated by varying applications. They include bandwidth, latency, jitter and loss [32].

Bandwidth refers to the rate at which an application’s traffic must be carried through the network. It is also equal to the bit rate (bit per second) generated by the application [23]. RFC 2330 simply defines the bandwidth of a link for packets of size k as the capacity, in bits/second where only the bits of the IP packet are counted, for packets of size k bytes. Latency (Delay) is the delay an application can tolerate when a packet of its data is being delivered. It can also be defined as the time needed for a packet to be transmitted from source and fully received by the destination. Delay within the network may be expressed in terms of propagation delay, network round trip delay, transmission delay, protocol delay and queuing delay [26]. Jitter is the delay variation introduced by network components and communication path. Lastly, loss gives the percentage of data units that fail to arrive at the destination in a certain period. It is a function of instantaneous network load and equipment configuration.

If infinite network resources were available, then all application traffic will be carried at the required bandwidth, with zero latency, zero jitter and zero loss [28]. However, network resources are “non-infinite”. As a result, there are parts of the network in which resources are unable to meet demand. QoS management, in a manner that meets an application’s service requirements, controls the allocation of these resources to applications’ traffic [33]. Network resources such as bandwidth are managed by the operator or service provider. Service providers negotiate an agreed resource allocation measure and network performance expectation with users in the form of Service Level Agreements (SLA) [32] [34].

SLA is a legal contract document that states service provider’s responsibility in ensuring users’ access to the network with pre-specified QoS parameters [33]. SLAs maintain the balance between available network resources and user’s application requirements. Three basic levels of service are provided within a network [16]. These service levels refer to the ability of a network to deliver service needed by specific traffic [22]. They are best effort service also known as lack
of QoS, priority or differentiated service called soft QoS, and guaranteed service known as hard QoS. These three basic levels of end-to-end QoS are shown in Figure 2-1.

![Diagram](image)

**Figure 2-1 Network's End-to-End QoS Service Levels.**

### 2.3 QoS Guarantees

Service levels differ in “QoS strictness”, which describes how tight the service is bound by specific bandwidth, delay, jitter and loss characteristics [16] [22]. The QoS strictness defines different types of QoS guarantee, which may be no guarantee, soft guarantee or hard guarantee.

Best effort services illustrated in Fig 2-1, describes a situation of no guarantee with the service level stating provisioning of basic connectivity only. Best effort implies that all traffic are treated in the same way. Best effort service can be tolerated by data applications like email and file transfer. Service levels bound to guarantee QoS provide soft or hard guarantees in terms of quantitative or qualitative measures. These service levels are termed guaranteed services or differentiated services.

Guaranteed services such as the Integrated Service (IntServ) model provide hard QoS guarantees in quantitative measures. They provide an absolute reservation of network resources
for specific traffic. Resources are held for serving the traffic’s connection while it remains established and released when the connection is torn down. In IntServ, end-to-end QoS guarantee is provided using the Resource Reservation Protocol (RSVP) signalling (IETF Internet standard-RFC2205) [22][35][36]. RSVP enables applications to request a specific QoS for a data flow. IntServ model ensures deterministic QoS for selected traffic within the highly un-deterministic world of IP traffic communication [30]. It provides explicit resource admission control, but its disadvantages can be associated with signalling overhead within the network [30] [36]. Another drawback of the IntServ model is its scalability limitations in a network with large volume of traffic and the complexity in deploying RSVP.

Differentiated services e.g. DiffServ model are less strict and provides soft QoS guarantee. Its approach is qualitative in nature. It classifies traffic into groups and processes them with different priorities based on their group. In this way, it treats some traffic better than the rest, using a statistical preference and not a hard or fast guarantee [22] [25]. It is a simple model, easy to implement and incurs low overhead.

Both IntServ and DiffServ operate solely at packet level, free from the complications of lower-layer protocols [36]. However, the limitations of IntServ are due to its lack of scalability and slow adoption by end-systems and applications, while DiffServ leaves many open issues on the reservation and allocation of network resources [36].

2.3.1 Deterministic and Probabilistic QoS guarantee

Resource allocation for traffic requiring QoS guarantee are stated in deterministic or probabilistic terms, and in some cases both [35][37]. Therefore, networks can assure traffic of either deterministic bandwidth and delay guarantees or probabilistic bandwidth and delay guarantees. Though the deterministic approach provides absolute guarantees, the complexity involved in its implementation is a major disadvantage. In contrast, probabilistic guarantee is a simpler approach because it uses mathematical models to compute the probability of guaranteeing required QoS [38]. The probabilistic approach allows systems to be flexible in allocating resources at different network speed. Despite the simplicity and setbacks of these two approaches, they affect the complexity and performance of a network to different degrees [45].
The co-existence of both deterministic and probabilistic guarantees within a network has been investigated [31] [38]. Tamboli et al used a hybrid approach to ensure deterministic guarantee for the basic level of service and a probabilistic assurance of other service levels for enhanced QoS [31]. The authors of [38] developed mathematical models to evaluate the probability of meeting a specified end-to-end delay. They showed that delays much smaller than the deterministic bound can be guaranteed with probabilities close to one. Lella et al in 2006, explained that deterministic QoS guarantee are based on worst-case analysis and leads to deterministic bounds, which are reached infrequently and often lead to network over-dimensioning [39].

Luca et al presented a probabilistic approach to QoS guarantee. Their approach determines an estimation of the probability that traffic will be transmitted before its absolute deadline (hard guarantee). According to these authors, the estimation of such statistical guarantee is computed based on the inter-arrival and execution times probability distributions [40].

These research works demonstrated that while deterministic guarantees provide traffic with a strong guarantee to meet their deadline, probabilistic guarantees allow the efficient utilization of network resources [31] [38] [40].

For future networks, this research proposes the use of a probabilistic approach to QoS guarantee on the first level of the QoS architecture shown in Figure 2-2. The method allows derivation of resource allocation process that enables all application traffic to receive a certain level of quantitative guarantee adapted to its QoS requirements [31] [40]. In this way, the network resources available on the first level of the architecture are efficiently managed and fairly distributed among traffic. Therefore, the starvation of any application traffic is prevented.

2.4 QoS Architecture

Recently, there has been emphasis on the fact that a global uniform end-to-end IP QoS solution is not realistic [33]. Based on this remark, realizing QoS provisioning in IP networks will be on packet-level basis. Packet-level QoS provisioning involves classifying QoS mechanisms into building blocks [36]. The classification allows each mechanism to fulfill its
objective in different parts of the heterogeneous IP network. This taxonomy is in line with ITU-T initiative toward a QoS architectural framework for IP networks [36]. It gives rise to a thorough presentation of a hierarchical QoS building block [40]. The hierarchy shows how the combination of existing mechanisms jointly provides packet-level QoS to achieve an end-to-end QoS in modern IP networks [36] [41]. Three fundamental hierarchical levels in the QoS architecture for implementing QoS within IP networks are shown in Figure 2-2. They are:

1. Level three of the hierarchy provides QoS policy management and accounting to control and administer end-to-end traffic across a network [22].

2. Level two provides QoS signalling such as admission control and resource reservation, these are performed from end-to-end and between network elements [22].

3. Level one provides QoS within a single network element through classification, queuing, scheduling, and traffic-shaping techniques [22].

Figure 2-2 illustrates these three hierarchical levels, their functional placement within a network, and the time scale in which they take place [15] [22].

Figure 2-2 Hierarchical QoS Architecture

Adapted from [22]
QoS policy management and accounting constitutes the mechanism on the third level of the architecture in Figure 2-2. These mechanisms control and administer QoS provisioning across a network [22]. QoS accounting, also referred to as pricing, is used for maximizing service provider and network operator revenue [36] [37] [42]. It is also a vehicle for discouraging several users from employing quality services, so that network utilization can be optimized [36] [43] [44]. Pricing is a very sensitive QoS issue because an assurance of customers' exact subscribed service is not always possible [37]. Since a service provider does not have control over the whole network, the use of network resources by other users may affect his customers' QoS experience. Therefore, a little deviation from service promised to users is possible [37].

Techniques for QoS signalling performed on level two of the QoS architecture include admission control and resource management. They co-ordinate QoS from end-to-end and between network elements. Admission control determines whether a new traffic flow can be permitted into the network without violating the QoS of already established traffic flows [36] [45]. As a result, admission control ensures that available network resource is sufficient to accommodate the new traffic. An algorithm makes acceptance or rejection decisions. Availability of resources to accommodate the new traffic's target QoS determines the acceptance or rejection decisions. Alternatively, a decision can be made based on whether the new flow violates the QoS of existing flows [46] [47].

Finally, the first level of the QoS architecture is the focus of this thesis. It deals with provisioning QoS within a single network element-the router. QoS provisioning is by techniques such as shaping, policing, classification, queuing and scheduling [22]. These are also known as packet processing functions discussed in the latter part of this chapter.

To conclude this section, this research proposes a probabilistic approach to QoS guarantee at the first level of the QoS architecture because of the cumulative nature of QoS provisioning. A probabilistic approach to QoS guarantee at the first level will ensure high resource utilization and provide fairness to all traffic thereby preventing the starvation of some traffic.
2.4.1 Router Functions

The router is a network element of the first level of the QoS architecture. Three fundamental functions of routers include routing, packet forwarding, and packet buffering. Others are error detection and correction, fragmentation, segmentation and reassembly, frame and protocol de-multiplexing, security, authentication and privacy, and traffic measurement.

Routing involves destination address look-up and computation of the best path that a packet can take through the network to its destination. Routers determine best paths by sharing information about network conditions with neighbouring routers [48]. The routing process is controlled by the processor. A comprehensive route database or routing table is built to enable the forwarding engine to send packets across optimum paths through the network [49].

The routers' second function is to forward packets received on an input interface to the appropriate output interface for transmission across the network. Its third function is to temporarily store packets in its buffer memories to absorb the burst and temporary congestion that may occur. If multiple packets arriving from different interfaces need to be forwarded to the same output interface simultaneously, a buffer must be available as a temporary waiting area in which packets are queued for transmission [49]. The order in which they are transmitted is determined by a policy configured into the router. Ordering policies are provided by an algorithm known as the scheduler. Without sophisticated ordering policies, packets are simply transmitted in the order in which they arrive [49] [50].

2.4.2 Packet Processing

Packet processing functions are classified as the functions performed on the first level of the QoS architecture. They ensure QoS provisioning within a single network element (e.g. router). Without effective packet processing functions, it is almost impossible to deliver meaningful service guarantees [15]. Some packet processing functions include classification, shaping, policing, queuing and scheduling [51].

Classification refers to the process of mapping a packet to a finite set of flows or categories [2]. It allows the provisioning of QoS guarantee to traffic in order to be able to meet
their QoS requirements. Traffic can be categorized based on flow or applications with the same QoS requirements. Classification is a broad and important process. Its concept encompasses sorting traffic flows into a static set of categories determined a-priori or into a dynamic set of categories that changes over time [2].

Traffic shaping is the process of conforming traffic to stated statistical bounds. It enforces particular traffic model at traffic sources or traffic multiplexing points, especially in cases of bursty traffic [36] [52] [53]. Shaping may be applied to aggregate traffic (all traffic from a given site) or to individual flow. Shaping is associated with the existence of a queue and sufficient memory to buffer delayed packets. It retains excess packets, and schedules them for later transmission over increments of time. As a result, it is an outbound concept applied to packets going out of an interface [36]. Moreover, it is also associated with a scheduling function transmitting delayed packets.

Traffic shaping and traffic policing are related but the two differ in the mechanism used and their overall goal. Traffic policing monitors ingress traffic to ensure that they remain compliant with a predefined network profile [52]. A packet is dropped if its traffic goes out of the agreed profile. Policing applies to inbound traffic of an interface at ingress points, and traffic that have been previously shaped to conform to a particular profile. Therefore, policing is a dual function of traffic shaping. While policing uses hard boundary mechanism in which traffic exceeding their bound are automatically discarded, shaping applies soft boundaries. Shaping changes traffic until it meets the desired boundary and does not usually discard packets [2]. Queuing and Scheduling, which are the focus of this thesis, are discussed in the next chapter.

Most of these processing functions are algorithms. They are developed with the assumption of an underlying model for the packet traffic [15].

Traffic modelling is essential in designing majority of the QoS-based packet processing functions. A traffic model is a set of mathematical rules, governing packet generation [36]. The models serve as a basis for applying traffic control functions and estimating network performance. Traffic modelling for packet networks focuses on packet-level models. Packet-level models, in form of stochastic processes model packet inter-arrival times, as well as the packet
sizes [36]. Based on the assumption that traffic is stationary, several stationary models were devised using known stochastic processes e.g., Poisson, Bernoulli, Markov Modulated Processes [54]. Apart from packet-level models, stationary models at burst-level also exist. These models consider batch transmission of groups of packets. Also, fluid flow models which transfer the modelling from discrete to continuous space have been devised [55]. However, burst-level and fluid flow stationary models are in general less accurate than packet-level models [36].

The advantage of traffic modelling with stochastic processes is the application of calculus to traffic control problems. However, studies show that actual traffic have strong non-stationary, uncertainty, and nonlinear properties [36]. LAN (Ethernet) and WAN (internet backbone) traffic are shown to exhibit self-similarity [55] [56]. Self-similar models are more complex to handle, and therefore make traffic control problems more complex. The practical outcome of self-similarity is that traffic are bursty at all timescales. As a result, the buffering capacity at traffic multiplexing points (e.g., routers) must be much larger than that derived by traditional queuing analysis and simulations [36]. However, the trade-off is that large buffers alleviate packet losses at the expense of increasing delay.

From research perspective, self-similarity complicates the task of modelling aggregate traffic in the network [57] [58]. The complexity manifests primarily with Internet traffic, which is now heterogeneous. Generally, the suitability of a conventional Poisson model for IP traffic depends heavily on the level of aggregation in the networks [59]. Regardless of the dynamics of the traffic, traffic shaping can be used to control the traffic profile at certain points in the network. Shaping concepts are mostly based on non-linear spacing laws. These laws result in treating traffic as non self-similar, and thereby support the application of conventional queuing theory [60]. To this end, the traffic model for this research is based on a packet level, stationary stochastic process such as the Poisson model.

QoS provisioning within a network is ensured at the three levels of the QoS architecture. The packet-processing functions discussed in the latter part of this chapter work hand-in-hand to ensure QoS provisioning on the first level of the QoS architecture. As packets arrive at the ingress of the router, policing ensures that the packets are within their SLA while being classified according to their QoS requirement. Queuing buffers these packets within the router as the traffic
shaper ensures that the packet meets the requirement of the downstream network and do not cause congestion. Scheduling determines the optimum order to forward these packets. The attention of this research is on the queuing and scheduling functions. Chapter three and four expounds on different queuing architectures and strategies for scheduling traffic at the first level of the QoS architecture.
3 Single-stage Queuing and Scheduling

3.1 Queuing

With service providers looking at maximizing operating profits, a key requirement for new network build-outs will be the management and billing of bandwidth [61] [62]. To meet this need, equipment manufacturers are looking for proficient solutions to queuing and scheduling in order to meet QoS requirements of traffic.

Queuing and scheduling are two of the packet processing functions at the first level of the QoS architecture mentioned in chapter two. The two fundamental queuing architectures identified by this research are single-stage queuing and scheduling and multi-stage queuing and scheduling. Strategies for single stage queuing and scheduling are discussed in the next sections while multi-stage queuing and scheduling is expatiated on in the next chapter.

As the speed of a network increases, more packets are transmitted depending on the amount of packet generated by the hosts in the network. When the arrival rate of these packets suddenly exceeds the router’s forwarding rate, packets will have to wait [63]. When this happens, the router buffers the waiting packets. The strategy for buffering these packets is known as queuing. Queues are buffer space where packets wait from arrival time to service time [15]. Queuing algorithms provide strategies for buffering packets according to their classification category. Examples of queuing algorithms are Priority Queuing (PQ), First In First Out (FIFO), and Custom Queuing (CQ).

One important issue in queuing is where to buffer packets strategically for optimal performance [15]. Buffering may be at the input or output ports, but the question is which of these ports will give an optimum performance.

Historically, because of the store and forward paradigm in routers, the approach to queuing has been according to the single-stage architecture. Single-stage queuing architecture is characterized by the forming of packet queues only at a single point within routers. Since input ports and output ports configuration support buffering, it is evident that packet queues can be
placed at either the input ports or the output ports. Therefore, the router can implement an input queuing strategy or output queuing strategy.

3.1.1 Input Queuing (IQ)

![Diagram of input queuing](image)

**Figure 3-1 Input Queuing.**

If the switch fabric is not fast enough (relative to the input line speed) in transferring arriving packets immediately, then packet queuing will occur at the input ports. Subsequent packets entering the router join the input port queues and wait for their turn to be transferred through the switching fabric to their destined output port. At anytime slot, only the first packet in any queue is eligible for transmission. Figure 3-1 shows the Input Queuing strategy (IQ) used in this situation. To illustrate an important consequence of the IQ strategy, consider a crossbar switching fabric. Assuming,

1. All link speeds are identical.
2. Only one packet can be transferred from any input port to a given output port in the same amount of time it takes for packet to be received on an input link.
3. Packets are transmitted from a given input queue to their desired output queue in FIFO manner.

Suppose multiple packets are received simultaneously and they are queued at different input ports. If two packets at the front of two input queues are destined to the same output port, then one of them is blocked since the switching fabric can only transfer one packet to a given
output port at a time. The blocked packet must wait at the input queue and consequently, other packets behind it cannot be transmitted even if their destination output port is idle. Figure 3-2 illustrates this scenario further. Here, two packets, tagged 1 at the front of their input queues are destined for the same upper top output port. If the switch fabric chooses to forward the head of queue in the upper left input port, then, the “tag 1” packet in the lower queue must wait. It is not only this packet that will wait, the “tag 2” packet queued behind will also wait, even though there is no contention for its destination (the middle output port). The phenomenon explained above is known as the head of line (HoL) blocking.

![Diagram of switch fabric showing head of line blocking](image)

**Figure 3-2 Head of Line Blocking.**

Karol et al in [64] shows that due to HoL blocking, input queues may grow to unbounded length (meaning that significant packet loss will occur) as soon as packet arrival rate on the input links reaches 58%. A number of solutions to HoL blocking are discussed in [65].
3.1.2 Output Queuing (OQ)

Suppose the input line speeds and output line speeds are all identical, and there are n input ports and n output ports. If the switching fabric speed is at least n times as fast as the input line speed, then no queuing can occur at the input ports. This is because even in the worst case when all n input lines have packets, the switch fabric will be able to transfer n packets from input port to output port in the time it takes each of the n input ports to simultaneously receive a packet.

On the other hand, what happens at the output ports? Transmission runs smoothly as long as packets arriving simultaneously at each of the n input ports are not destined for the same output port. Conversely, if the packets are destined for the same output, then, in the time it takes to receive a single packet, n packets will have arrived at this output port. Since the output port can only transmit a packet at a time, the n arriving packets will have to queue (wait) for transmission over the outgoing link. In this scenario, buffering of packets is at the output ports uses the output queuing (OQ) strategy in Figure 3-3.

![Figure 3-3 Output Queuing](image)

The main disadvantage of output queuing is that for an N-port router, the internal interconnect and output queues must operate at N times the line rate. In cases where the number of ports is large and line rate is high, output queuing may become impractical [18] [64] [65].

In contrast, IQ has no scaling limitations, and it is the simpler single stage queuing strategy. In addition, its queuing speed is the same as the line speed, meaning that packets are
queued at the rate at which they arrive at the input ports [66]. Even though it is simpler, IQ exhibits a performance limitation caused by the HoL blocking.

### 3.1.3 Virtual Output Queuing (VOQ)

![Diagram of Virtual Output Queuing](image)

**Figure 3-4 Virtual Output Queuing**

VOQ, first proposed in [57] is an input queuing strategy where every input port has a separate queue for each output port. At each input, there is a separate FIFO queue for each output port of the router, as shown in Figure 3-4. Therefore, for an N input ports and N output ports router, a total of N^2 VOQs are at its input.

VOQ seem to be slightly more complex than having one FIFO queue per input port as in traditional IQ strategy. Even though it can get rid of HoL blocking, yet the bulk of its complexity lies in the scheduler. The scheduler has to retrieve the state information of all N^2 input queues and compute a pseudo-optimum matching within one cycle [1]. In addition, the scheduler must be able to arbitrate fairly among packets queued at the inputs without starving any queue.

At the beginning of a time slot, the centralized scheduler has to examine the content of all input queues and find a conflict-free match between inputs and outputs. The conflict-free matching is similar to a bipartite graph-matching problem (BMP) of finding conflict-free pairing of inputs to outputs [50]. After the scheduler makes a forwarding decision, the favoured packet is placed in its destination output port.

A key factor in achieving high performance with VOQ lies in the type of scheduling algorithm used. In theory, an effective scheduling algorithm can increase utilization from 58.6% with FIFO queuing to a full 100% if VOQs are used [50]. When compared with the IQ and OQ
strategies, the VOQ strategy has a scalability advantage, but the main challenge is finding an appropriate scheduling algorithm, which can provide a high-speed mapping of packets from inputs to outputs on a cycle-to-cycle basis [18].

3.2 Scheduling and Algorithms for Virtual Output Queuing

Generally, scheduling is a way of controlling access to resources at some kind of server. The server offers and receives service requests while the scheduler determines which service request is dealt with next according to a well-defined policy or algorithm [17]. The scheduler’s role is to make decisions regarding the order in which service request are allowed access to resources.

In the case of a router, it performs packet forwarding as a service and the resource which is under the control of the scheduler is (are) the output port(s). Service requests are from the packets at the input ports. Accordingly, scheduling controls the order in which packets are served, so it may influence service requests’ quality [17]. Therefore, scheduling is a major component in QoS provisioning [15]. The order of serving packets is determined by an algorithm. Four key requirements of any scheduling algorithm are:

1. Simplicity: keeping the algorithm as simple as possible makes it fast, thereby making it suitable for use in high-speed networks [15] [17]. In addition, if the algorithmic complexity is low then it will be easier to implement in hardware. Since a scheduling algorithm is executed every time-slot to determine the next packet to be transmitted, it is very important to keep the complexity at a minimum to ensure faster transmission.

2. Fairness: all traffic should receive a fair allocation of resource. Any traffic should not receive a lesser allocation than other flow requesting to use the same resources. In fact, it is appropriate to be precise about how resources should be allocated [17]. Fairness also prevents the starvation of any traffic. Starvation means that a traffic flow is denied access to resources due to the misbehaviour or characteristics of another flow.
3. A scheduling algorithm can be either work conserving or non-work conserving [17]. Work conserving implies that the algorithm is never idle if there are packets waiting to be served. Consider N packets with mean arrival rate, \( \lambda \) packet/sec, and mean service rate \( \mu \) packet/sec, then the mean utilization rate \( p = \lambda / \mu \). If the mean waiting time due to scheduler is \( q \), then a scheduler is work conserving if \( \sum pq = C \) (constant) [58]. Examples of work conserving schedulers are Fair Queue (FQ), Weighted Fair Queue (WFQ), Delay-EDD, HPFQ [34], and CSFQ [36].

On the contrary, a non-work conserving scheduler can be idle even though there are packets waiting for transmission [17]. The reason for its idle period is that it waits for packets to become eligible for transmission. The scheduler decides whether a packet is eligible based on the time budget that remains for each packet. This allows delay-jitter regulation and makes downstream traffic more predictable. Examples are stop and go queuing [36], hierarchical round robin [2], jitter EDD [43]. However, non-work conserving scheduling disciplines are still considered a research issue.

4. Scalability: an important feature of any scheduling algorithms is its flexibility in efficiently allocating resources to traffic streams as network speed changes.

5. Stated performance bounds: for supporting QoS in networks, the algorithm should specify performance bounds to ensure that the each traffic flow is handled appropriately. The exact nature of these performance bounds depends on the kind of QoS guarantee provisioned. A scheduler providing service guarantee would have to specify either deterministic or probabilistic performance bounds [17].

While there has been a lot of research on developing schedulers that provide QoS guarantees, very little work exists on scheduling algorithms that supports probabilistic QoS guarantee for VOQ. The reason is that the complexity of virtual output queuing makes the requirements of scheduling algorithms more difficult to achieve. Most of the research on providing QoS guarantees assumes the underlying queuing strategy to be OQ [15]. Due to the poor scalability of the OQ, these research efforts have very little practical value.
The difference between guaranteeing QoS in VOQ and OQ is the amount of work that the switching fabric in VOQ will do. In OQ, a packet is available immediately it is scheduled, because no output contention exists. As a result, each output port is independent in guaranteeing QoS. The scheduling of packets queued in different outputs can be isolated from one another [1]. In VOQ, the scheduler's policy may cause some packets not to be promptly transmitted when they should receive service. Consequently, these packets lose the chance of being served, and this result in violating their QoS. Therefore, the key issue in providing QoS guarantees in VOQ lies in the design a scheduling algorithm that can guarantee the prompt transmission of queued packets [1]. If the delay of packets at any port can be guaranteed, the algorithm will not cause starvation.

Various research efforts are on algorithms that are flexible, independent of router size, and traffic type [1]. Several VOQ scheduling algorithms using different methods have been proposed. An important feature of these scheduling algorithms is their approach to the efficient allocation of resources to traffic. This research has classified existing VOQ scheduling algorithms into two main categories according to how they allocate resources to guarantee QoS to traffic. These categories are 1) deterministic algorithms and 2) probabilistic algorithms. The following section gives the details of these algorithms.

This research designs a probabilistic algorithm, which schedules traffic and allocates resources to traffic with a probabilistic approach. It is able to guarantee the prompt transmission of queued packets in the virtual output queuing strategy. The next section discusses some deterministic algorithm, which are analysed and compared with the proposed probabilistic algorithm.

### 3.2.1 Deterministic Scheduling Algorithms

Deterministic algorithms are algorithms with predictable behaviour [69]. That is, each time a certain set of input is presented, the algorithm gives the same results as any other time the same set of input is presented. Apart from having a predictable set of results, the other problem with deterministic algorithm stated in [69] is that they are complex to implement and relatively slower in practice.
Schedulers used in today’s high-performance networks must provide high throughput and low latency. The high throughput and low latency dictates that the algorithm must be able to find a matching of as many conflict-free pairings as possible within a short time range [18].

Existing deterministic algorithms for the VOQ operate under foreseen and known matching outcome and therefore allow the prediction of future packet selection. According to [1] there are various problems associated with any algorithm whose performance is predictable. Firstly, the mathematical complication associated with how it makes decisions gives a high time complexity, thereby making the algorithm’s operation slow. The algorithm becomes difficult to implement in hardware due to this complication. Secondly, such algorithms are biased towards a certain type of traffic i.e. CBR (Constant Bit Rate) or VBR (Variable Bit Rate); therefore, their application is only useful and limited to networks carrying these traffic. Lastly, they underutilize network resources and this makes their average throughput or utilization in any network quite low [76].

Majority of the available VOQ scheduling algorithms are deterministic, some of these algorithms are able to guarantee QoS while others do not. Table 3-1 provides an outline of the QoS guaranteeing algorithms, which are classified into three categories in [24].

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time Slot Assignment</th>
<th>Maximal Matching</th>
<th>Stable Matching</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time complexity</td>
<td>0(N^2.5)</td>
<td>0(N^2)</td>
<td>0(N^2)</td>
</tr>
<tr>
<td>Maximum throughput</td>
<td>100%</td>
<td>50%</td>
<td>50%</td>
</tr>
<tr>
<td>Differentiated Service</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
<tr>
<td>Best supported traffic</td>
<td>CBR</td>
<td>CBR</td>
<td>CBR and VBR</td>
</tr>
</tbody>
</table>
The algorithms based on time slot assignments (TSA) provide guaranteed performance for CBR traffic through a careful planning of the time when the packet have to leave the VOQ [1] [24]. Maximal Matching (MM) algorithms, tries to target the maximal matching of input to output instead of the maximum matching to improve on the time complexity of TSA algorithms. MM algorithms maintain a global input preference list to resolve conflicts occurring at each output port [24]. The preference by input ports can dynamically change with traffic to determine which packet destined to a specific output port is preferred for transmission. Examples of MM algorithms are the Oldest Cell First (OCF) and the Longest Queue First (LQF) [18]. These algorithms are bounded by delay and input preference constraints, which causes problems in guaranteeing QoS to CBR traffic [24]. They are also very complex to implement and require a high running time. SM-based algorithms use the principle of stable matching to derive priority information from packets. They find a stable matching of inputs and outputs based on the preference defined on the input and output preference lists. A matching is unstable if there is at least a pair of input and output, which are unmatched, but each prefers the other to its current matching mate [1].

Two non-QoS guaranteeing deterministic scheduling algorithms for the VOQ discussed in this thesis are the Parallel Iterative Matching (PIM) and the Iterative SLIP. They are categorized as Iterative Maximal Matching algorithms [20] [1].

In comparison with the scheduling time constraint imposed by high-speed gigabit-per-second networks, conventional bipartite maximum matching algorithms have high time complexity. This led Anderson et al. [50] to propose the PIM algorithm, to find a maximal matching. A maximum matching matches a maximum number of paired inputs and outputs while a maximal matching is one in which no unmatched input has a queued packet destined for an unmatched output (i.e., no pairings can be trivially added). PIM uses randomness to find a maximal matching of inputs with packets and outputs that have packets at the inputs destined for them. Specifically, PIM iterates the following three steps until it finds a maximal match or after performing a fixed number of iterations.

1. **Request**: Each unmatched input sends a request to all outputs for which it has a queued packet.
2. **Grant**: If an unmatched output receives any requests, it grants to one by randomly selecting a request uniformly over all requests.

3. **Accept**: If an input receives grants, it accepts one by selecting an output among those that granted to this input.

PIM has a worst case time complexity of $O(N^2)$, and each iteration matches an average of $3/4$ of all possible connections converging to a maximal match in $O(\log N)$ iterations [20][59]. Despite its performance, the main set back of PIM is the random way of scheduling packets. Randomness has some problems attributed to it. First, it is difficult and expensive to implement at high speed. Second, it can lead to unfairness and may cause the starvation of some queued packets. Another downside is that PIM is incapable of supporting traffic with QoS guarantees [1] [24]. However, its parallel and iterative running fashion serves as a common basis for a large number of related algorithms designed for VOQ.

Another non QoS guaranteeing algorithm for the VOQ is the Iterative SLIP (iSLIP) matching algorithm which is an enhancement of the SLIP algorithm introduced by McKeown in [18] [20]. iSLIP is a variation of the PIM and it avoids the use of randomness when making selections. Instead, it schedules packets in a round robin manner. The three steps of an iteration are as follows:

1. **Request**: Each unmatched input sends a request to every output for which it has a queued packet.

2. **Grant**: If an unmatched output receives any requests, it chooses the next input in the round-robin schedule. The output notifies each input whether or not its request was granted.

3. **Accept**: If an unmatched input receives a grant, it accepts the next grant in a fixed, round-robin schedule.
3.2.2 Probabilistic Scheduling Algorithms

Rather than being predictive in its logic of operation, probabilistic algorithms employ some degree of freedom and flexibility in their scheduling operation. In this way, they are able to capture the needs of network traffic [40]. These algorithms require fewer resources to implement than deterministic algorithms. In addition, due to an unpredictable and unscheduled nature of the arrival of packets from different applications to a router, the QoS guarantee of such algorithms is probabilistic. The view of this research is that scheduling packets with probabilistic algorithms is a better way to guarantee QoS at the first level of the QoS architecture in future networks.

An example of probabilistic algorithm for the VOQ strategy in the literature is Weighted Probabilistic Iterative Matching (WPIM) algorithm proposed in [70]. WPIM allows a flexible allocation of bandwidth in a simple manner. The operations of WPIM consist of two consecutive applications of the original PIM algorithm. Consequently, WPIM has a time complexity of $O(N^2)$ for running once in each time slot [1].

WPIM computes the weight of the queued packets to resolve input and output port contentions and achieves probabilistic bandwidth and delay guarantees. Bandwidth guarantees in WPIM is by making reservations during connection setup stage. However, WPIM cannot separately consider the bandwidth allocation to packets with the same input and output ports because its scheduling action is on connection level [1]. Algorithms performing arbitration on connection level supports CBR traffic only. It is difficult for such algorithms to guarantee QoS in any form to VBR. Instead of scheduling at connection-level, some algorithms run packet-level scheduling. Packet level scheduling exhibit distinct advantages in supporting VBR traffic with QoS guarantees [1].

3.3 Issues in Queuing and Scheduling

To conclude this chapter, a brief overview of some issues in queuing and scheduling in routers is given. A router is a shared resource in terms of buffer memory and output bandwidth. Some of the issues faced in today's network are related to the allocation of these limited resources [70]. While queuing is a strategy for buffering packets, scheduling manages access to
the fixed amount of output port bandwidth by selecting the next packet to be transmitted on an output port.

Generally, the queuing issue is a classical problem. When packets arrive at any router in a network, they have to be sent out at once into the right output link. Since a router has multiple inputs and multiple outputs, forwarding may become harder to do as the arrival rate of packets become greater. Therefore, the forwarding of packets will take time and other arriving packets have to wait for their turn. To avoid loss, waiting packets are buffered through a queuing process. Where queuing takes place affects the router’s rate of forwarding packets. In view of the limitations of the two "pure" single stage queuing strategies discussed, the VOQ strategy is a more viable way of queuing packets at the input ports. However, the crux of the problem in VOQ lies in matching the inputs that have packets with the outputs these packets should exit through. Deterministic scheduling algorithms such as PIM and iSLIP that find a maximal matching in O (log N) iterations provide only best-effort service. Even though they are simpler to implement than other deterministic algorithms that guarantee QoS, they can lead to packets’ starvation.

The importance of solving the queuing and scheduling problems cannot be overstated. A solution is vital to the design of queuing and scheduling algorithms for future networks. As with any other computing problem, the solution has to be efficient and scalable for use in any network. The continuous increase in network speed, the growth in user traffic, and the creation of new services calls for faster and intelligent schedulers. VOQ is a simple strategy for overcoming HoL blocking in IQ and it is used in most high-end routers. However, because network speed is growing from giga to tera bits-per-second and QoS requirements are becoming more stringent, current VOQ scheduling algorithms are less adequate for effective scheduling. Robust and efficient scheduling that supports QoS is still lacking. The schemes that support certain level of QoS have limitations. They are too complex, too slow, specific to certain traffic type, not scalable, and not implementable in hardware.

Therefore, the future direction is to design fair and implementable scheduling algorithms with reduced time complexity. Due to the heterogeneity of traffic in NGN, schedulers provisioning probabilistic QoS guarantees are more appropriate for network elements used in NGN.
This chapter provided an overview of some existing single stage queuing and scheduling strategies. Due to the limitations of these strategies, different research efforts are working on the combination of these strategies to form multi-stage architectures. The concepts of some of the existing multi-stage architectures are explained in the next chapter. In addition, the concept of the multi-stage queuing architecture and the proposed scheduling strategy simulated in this research is described in the next chapter.
4 Multi-Stage Queuing and Scheduling

4.1 Multi-Stage Queuing concept

Single stage queuing concepts examined in the last chapter provided a background on some single stage strategies, which are combined in the multi-stage architecture. The Multi-stage queuing architectures form packet queues at several points within the router. Unlike in single stage architectures, a packet goes through multiple sets of queuing stages. In other words, the processing and buffering associated with the forwarding of the packet are physically and logically distributed [49]. The set-up prevents service providers from experiencing performance degradation from their routers as increasing traffic volume begins to drain shared resources. Even though the architecture is a compromise between performance and cost, it has proved scalable in handling IP packets [71] [49]. Since NGN is based on an all IP platform and many new services like Video on Demand (VoD) and Voice over IP (VoIP) will traverse the network, multi-stage architectures are becoming more popular [71].

This research implements a multi-stage queuing architecture, which combines VOQ and OQ strategies. Lately, several researchers have been working on combining single stage queuing strategies into multistage. [72] combines a strategy known as Combined Shared Queuing (CSQ) with IQ while three-stage architecture is proposed in [71].

Table 4-1 gives a comparison of a few single stage queuing strategies combined in today's high-end routers. The number of memory units, total memory cost, throughput, and complexity cost are the terms of comparison.

In table 4-1, shared buffer and cross-point queuing are a variation of the output queuing strategy while VOQ is a variation of the IQ strategy. Further analysis of these single stage queuing strategies is available in [73]. From the literature, the VOQ strategy is shown to improve the throughput of a router if an effective scheduler is used. Hence, it is a focus of attention in the queuing research community. Nevertheless, its complexity is not making it a very attractive option and thus the design of queuing strategies is still an open issue.
### Table 4-1 Comparison of some Queuing Strategies.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>No of Memory units</th>
<th>Max Throughput per memory unit</th>
<th>Total Memory Throughput</th>
<th>Memory utilization</th>
<th>Performance</th>
<th>Complexity cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Queuing</td>
<td>N</td>
<td>m+1</td>
<td>(m+1)n</td>
<td>Medium</td>
<td>best</td>
<td>Memory Cost &amp; *LQT</td>
</tr>
<tr>
<td>Cross-point queuing</td>
<td>Mn</td>
<td>2</td>
<td>2nm</td>
<td>Worst</td>
<td>best</td>
<td>Many memories</td>
</tr>
<tr>
<td>Shared Buffer</td>
<td>1</td>
<td>m+n</td>
<td>m+n</td>
<td>Best</td>
<td>Best</td>
<td>*LQT</td>
</tr>
<tr>
<td>Internal speed up of S</td>
<td>m+n</td>
<td>S+1</td>
<td>(S+1)(m+n)</td>
<td>Medium</td>
<td>Very good</td>
<td>Expensive Switch fabric</td>
</tr>
<tr>
<td>Input queuing</td>
<td>M</td>
<td>2</td>
<td>2m</td>
<td>Medium</td>
<td>**HQT but Worst (HoL)</td>
<td>None</td>
</tr>
<tr>
<td>Virtual Output queuing</td>
<td>M</td>
<td>2</td>
<td>2m</td>
<td>***fair</td>
<td>Very good &amp; **HQT</td>
<td>Scheduling algorithm</td>
</tr>
</tbody>
</table>

* *LQT indicates Low Queuing Throughput. **HQT indicates High Queuing Throughput.

*** “Fair” is a metric between medium and worst memory utilization.

“m” is the number of input ports and “n” is the number of output ports.

A number of research efforts have been exploiting the merits of the VOQ. In [74] C.S. Chang et al proposed a two-stage architecture using VOQs at the router’s input and output without a scheduler. The two VOQ stages were incorporated without a separate scheduler because scheduling of packets is through a trivial deterministic sequence of configurations at each stage. In this way, the scheduling process is by distributing a timing reference to the line cards. In the architecture, arriving packets are placed in the first VOQ according to their output. They are transferred through a deterministic sequence to the second stage. The VOQs at the second stage are all serviced at the same rate by a second deterministic sequence. Chang et al proved that the first stage effectively makes non-uniform traffic uniform by spreading it evenly over the second stage. Hence, the two stages provides 100% throughput. A disadvantage of this architecture pointed out by the authors of [74] is that packets can be mis-sequenced, which can
cause problems for current versions of TCP, and some common rules of practice dictate that routers should not mis-sequence packets.

4.2 Multi-Stage Queuing and Scheduling Architecture

![Diagram of Multi-Stage Queuing and Scheduling Architecture](image)

Figure 4-1 Multistage Queuing and Scheduling Architecture.

To study the performance of the proposed probabilistic scheduling algorithm, this research simulated a multistage queuing and scheduling (MQAS) architecture in Figure 4-1. MQAS is a two-stage queuing and scheduling architecture. Contrary to the approach by Chang et al., it combines the scalability of the VOQ with the high performance of OQ [15]. The queuing and scheduling of packets are in two separate stages. The first stage occurs at the input while the second stage is at the output of the router. The simulation of an architecture that uses a scheduler facilitated the studying of the performance of the proposed algorithm.

At the input, every port maintains a separate FIFO virtual output queue for each output port. Therefore, a router with N-input ports and N-output ports will have a total of N² virtual output queues. Achieving optimal throughput performance after queuing incoming packets in the VOQ stage depends on the scheduler.
The scheduler has to serve packets in the same VOQ on first come first served basis and resolve contention among N² VOQ head of queue (HoQ) packet. These packets request for transmission to their respective outputs simultaneously and may request for transmission to the same output. Therefore, the scheduler has to be able to make a fast decision about the VOQ to send a packet to an output at a time slot. In addition, it should be able to arbitrate fairly among inputs queued packets. It should also meet the basic requirements of a scheduling algorithm. The scheduling algorithm proposed for the first stage of MQAS is an Iterative Probabilistic Scheduling (IPS) discussed in the next session.

Consider a hypothetical router with N input ports and N output ports implementing the MQAS architecture with an internal speed up of one. The queuing model is based on the M/M/1 queue model. From queuing theory, the meaning of M/M/1 is stated below:

- The first ‘M’ indicates that the arrival rate (λ) of packets is Poisson distributed.
- The second ‘M’ indicates an exponentially distributed service rate (μ).
- ‘1’ implies a single server system.

The hypothetical router under consideration uses an N x N non-blocking crossbar switch fabric. That is, the N inputs are connected to the N outputs via a non-blocking interconnection network.

In simulation, for each input port, the VOQs are memory buffers so that as packets arrive and are written into the receive memory; they are automatically placed into a VOQ associated with their destination.

At a given time slot, the scheduler selects packets in each queue for transmission with the following constraints:

- Only the HoQ packets from any of the N queues in an input port are eligible.
- Only one packet from the input section can be transmitted to an output queue. In other words, at most one packet is received by an output queue.
The second stage of MQAS uses an OQ strategy. Here, packets are queued as they arrive in the output queue on FIFO basis. Each output port has one queue using an M/M/1 queue model and the scheduler uses FIFO algorithm to match packets to the output link. Scheduling is based on FIFO because the major contention is already resolved at the input ports. The main advantage of output queuing is that throughput is maximized [75] [76].

4.3 Proposed Iterative Probabilistic Scheduling (IPS) Algorithm

This section presents the analysis of the Iterative Probabilistic Scheduling (IPS) algorithm proposed for the VOQ stage of MQAS. IPS is a probabilistic algorithm. IPS aims at improving on the PIM, iLQF, iOCF and i-SLIP. Even though it combines the properties of Maximal Matching (MM) and Stable Matching (SM) algorithms, IPS can be considered as a Probabilistic Iterative Maximal Matching algorithm. Similar to MM algorithms, IPS considers the state information of queues and like SM algorithms; it derives priority information from queued packets. In this way, it uses information from input queues and packets to find a maximal matching and it resolves contention among packets dynamically by assigning weights to the state information.

IPS focuses on two quality-of-service parameters, which are delay and bandwidth. The scheduling policy and QoS guarantee are probabilistic in nature. The aim of IPS is to ensure fairness to all traffic types while guaranteeing QoS. This makes IPS scalable for implementation in all networks [77]. Contention is made stringent and fair by the weights assigned to parameters retrieved as state information. IPS runs its scheduling process at packet-level, which have better performance in environments where traffic changes frequently.

4.3.1 IPS Model

Akin to iOCF and iLQF, IPS schedules packets by calculating the weight $w_{ij}$, where $w_{ij}$ is the weight of every head of queue (HoQ) packet at each input queue.

In the MQAS architecture, since each input port has N VOQs, let ‘i’ represent an input port and ‘j’ an output port within the router. Therefore, VOQ$_{ij}$ denotes a VOQ in input port ‘i’ queuing packets for output port ‘j’, and OQ$_i$ represents an output queue in output port ‘j’. For all
non-empty queue at the input the weight of a packet in VOQ_{ij} denoted WP_{voq}_{ij} is given by equation 1.

\[ eBW_{voq}_{ij} \cdot 2 + eQ_{voq}_{ij} \cdot 1 \]  

Where \( eBW_{voq}_{ij} \) is an estimation of the packets bandwidth and \( eQ_{voq}_{ij} \) is the estimated waiting time of the HoQ packet. \( eQ_{voq}_{ij} \) is calculated by subtracting the time the queue buffering the packet was last served from the current time \([92]\). Packets contend for transmission based on an estimation of the bandwidth and waiting time in the queue and IPS orders packets using these two state information. The weight factor of ‘2’ is given to \( eBW \) to prevent queues from overflowing. In addition, according to the internet traffic mix specified by RFC2544 and NASA Ames Internet Exchange, 56% of Internet traffic range around 40-byte, which is a minimum size for TCP packets carrying TCP acknowledgements but no payload. 23% are a range of 1500-byte, the maximum Ethernet payload size from TCP implementations that use path MTU discovery. 17%, which range between 552 byte and 576 byte are from TCP implementations that do not use path MTU discovery and 5% are 52-byte packets. From the CDF plot in appendix C.1, an average traffic carrying payload ranges from 500-1500 bytes, therefore, \( eBW \) is given a greater weight than \( eQ \), so that IP packets carrying real traffic can be given precedence.

After the weight of each packet is calculated using these two parameters which are retrieved during the scheduling process. IPS determines the probability of transmitting each HoQ packet during that time slot. The probability of transmitting any packet is given by equation 2. During any contention time, if the \( P_{voq}_{ij} \) computed for two different HoQ packets are equal, the packet with a higher \( eQ \) is scheduled, so that other packets behind it are not delayed for too long.

\[ P_{voq}_{ij} = \frac{WP_{voq}_{ij}}{\sum_i WP_{voq}_{ij}} \]  

\( 0 \leq WP_{voq}_{ij} \leq 1 \) for all time slots. \( \sum WP_{voq}_{ij} \) in equation 2 is the total \( WP_{voq}_{ij} \) of all packets retrieved at an instant. For example, if two packets have been retrieved and a third is being retrieved, the \( \sum WP_{voq}_{ij} \) is for these three packets. The \( WP_{voq}_{ij} \) of other packets are not considered until their parameters are retrieved.
4.3.2 Initialization Process of the IPS

At a time slot, input ports with packets in their VOQs send requests for transmission (REQi,j) to corresponding OQi. If an input port has packets in all its VOQs it sends a request to each output port for which it has a packet queued as shown in Figure 4-2.

![Diagram showing the initialization process of the IPS]

Figure 4-2. Input requests to Output.

After all requests have been sent, IPS forms a set \( Z \) of inputs with REQi,j to OQj such that for any j chosen after iteration, all elements in \( Z \) must be \( \leq N \). It then iterates only through the output queues with requests made to them and chooses an output queue (OQj) to serve a packet. IPS grants only one request per input port in a time slot using the algorithm described in the next subsection. For example, if an input port has made N requests to all the N output ports only one of these requests will be granted at that time slot.
4.3.3 Operation of the IPS algorithm

The flow chart in Fig 4-3 outlines the operation of the IPS algorithm. The calculation of $P_{VQO_i}$ in the flow chart is subject to the normalizing condition: $\sum P_{QO_i,j} = 1$

![Flow Chart of the IPS algorithm](image)

Figure 4-3 Flow Chart of the IPS algorithm.

4.3.4 IPS Scheduling process

Once IPS gets the maximum $P_{VQO_i}$, it performs the following process to schedule a packet to send to the iterated OQ$_j$.

STEP 1: Pick the packet with the maximum $P_{VQO_i}$.

STEP 2: Append the (Highest Probability Packet) HPP tag to the packet.
STEP 3: Transmit the packet to chosen OQi.

During any scheduling time slot, only the packet with the HPP tag is scheduled. After each scheduling process, it goes back to iterate for the next output port to which a packet is to be scheduled. IPS is idle only when input ports stop sending requests to the output ports; meaning there are no queued packets at the input ports.

At the output of the MQAS architecture, packets are mapped to the output port based on their arrival time at the output queue with a FIFO algorithm. There is an increased throughput at the output ports because no packet major contention exists between packets at this stage.

4.3.5 Analysis of the IPS Computational Time Complexity

One of the critical resources considered in the design of an algorithm is time [78]. The amount of time an algorithm takes to solve a problem is referred to as the running time. The running time is influenced by several factors, which includes:

1. Speed of the machine running the program

2. The language in which the program is written, for example, programs written in assembly language generally run faster than those written in C or C++, which in turn tend to run faster than those written in Java.

3. Efficiency of the compiler that created the program

4. Size of the input e.g. processing 1000 records will take more time than processing 10 records.

The first three factors are not used to measure the efficiency of an algorithm because they vary with type and are manufacturer dependent. An important consideration is the fourth factor, which is an independent factor [79]. An algorithm will run slower if it must process more data but this decrease in speed is not because of the construction of the algorithm. It is simply because there are more data to be processed.
Due to this consideration and because absolute time cannot be measured without a host
machine, an algorithm’s running time is expressed as a function of the input size and the number
of primitive operations it must perform to complete a task [80]. These primitive operations
include addition and multiplication. Therefore, the analysis of an algorithm’s running time
measures the number of times its instructions are executed when solving a problem and n is
denoted by T (n).

An estimation of how T (n) varies proportionally with the input size n is called time
complexity. The notation commonly used is the “big-oh notation” which describes the
asymptotic performance of an algorithm. It gives a relationship between how the growth of the
number of inputs affects the execution time [81]. As n grows to infinity, the growth rate may be
countant, linear, logarithmic, quadratic or exponential. Due to the relationship, execution time
may be slower or faster as more inputs are added. A low time complexity shows a fast execution
time while a high time complexity shows a slow execution time.

A summarized pseudo code of the operation of the Iterative Probabilistic Scheduling
algorithm is as below:

1. insert all inputs with REQ into an Array[Z]
2. iterate to choose OQ
3. Set n=number of objects in Array[Z]
4. Set Pmax=0
5. Set counter: for ( i=1; i<n; i++) {
  6. Picks the first input port in Array [Z].
  7. get eBW and eQ of the RoQ packet of this input
  8. calculate WP and P
  9. If P > Pmax
10. Pmax=P
11. Pick the packet with Pmax value append HPP
12. Transmit packet.

A worst-case scenario occurs if all inputs send a request. In this situation, steps 6 to 10
are executed a maximum of N times, where N is the number of ports. The number of execution
times of each step is analyzed in table 4-2.
Table 4-2 Execution times of IPS operation.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Number of times executed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>N</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>N</td>
</tr>
<tr>
<td>7</td>
<td>N</td>
</tr>
<tr>
<td>8</td>
<td>N</td>
</tr>
<tr>
<td>9</td>
<td>N</td>
</tr>
<tr>
<td>10</td>
<td>N</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>T(n)</td>
<td>6n+9</td>
</tr>
</tbody>
</table>

Statements 3, 4, 5, 11, and 12 are executed once, while statement 2 can be executed a maximum of 4 times, and all others a maximum of N times. Therefore, the total running time is 6n+9.

From the “Big-Oh” theorem, n is a non-negative integer parameter describing the number of requests made by the input ports. f(n) is a function describing the performance of the algorithm. The performance is in terms of the number of steps needed to be executed multiplied by the number of times they must be re-executed. g(n) is another function that represents an upper bound on f(n).

If there is a real constant c > 0 and an integer constant n₀ ≥ 1, such that f(n) ≤ c g(n) for every integer n ≥ n₀, then g(n) is the “order” of the algorithm. The order of the algorithm written in “Big-Oh” notation is O (g(n)).

**Definition:** Let f(n) and g(n) be two functions f(n) = O (g(n)) or f = O (g)

(Read, "f of n is big oh of g of n" or "f is big oh of g"), if there is a positive integer C such that f(n) ≤ C * g(n) for all positive integers n.
For IPS, the computational running time, \( T(n) = 6n + 9 \). Mathematically, according to the Big-Oh theorem, \( f(n) = 6n + 9 = O(n) \).

**Proof:** consider that a constant \( C \) is needed such that \( f(n) \leq C \cdot n \) for all \( n \). Trying \( C = 6 \), does not work because \( 6n + 9 \) is not less than \( 6n \), therefore \( C \) has to be at least 15 to cover all \( n \). If \( n = 1 \), \( C \) has to be 15, but \( C \) can be smaller for greater values of \( n \) (i.e., if \( n = 100 \), \( C \) can be 5). Since the chosen \( C \) must work for all \( n \), then \( C \) must be 15 so that \( 6n + 9 \leq 6n + 9n = 15n \). Therefore,

\[
T(6n + 9) = O(n)
\]  

From the falsal equation above, intuitively, the running time grows linearly as \( n \) grows.
Therefore, the time complexity for IPS is given by: \( T(n) = O(N) \).

4.3.6 IPS properties

The following are the properties of the IPS algorithm.

a) Simple: the scheduling process of IPS involves basic mathematical operations (e.g. addition, multiplication) so it is simple to understand and implement.

b) Fair: it is independent of traffic type because it does not provide priority, so it is fair to all traffic. The weights assigned to the eBW, eQ ensures fairness and allows all queued packets to contend for transmission with their packet size and the last time its queue was served. Contention is stringent based on the weight 2:1, the weight of 2 is given to packet size parameter to prevent buffer overflow. If two packets happen to have the same probability after computation, the one that has a higher value for the eQ parameter is scheduled. This is to prevent a longer delay for other packets behind it.

c) Prevents starvation: To ensure that no queue is starved, each queue maintains a last time service time stamps. In this way, the estimated waiting time of the HoQ of each VOQ eligible for transmission during a time slot is given by the difference between the current time and the last service time of the queue. The waiting time (eQ parameter) is based on
the queue last service time not the actual packet waiting time. IPS cannot starve a queued packet under any offered load because of the weight assigned to the eBW and eQ. Packets at the head of queues that have not been served recently increase in their last service time until they are eventually served. Therefore, no queued packet is left indefinitely without being served.

d) Scalable: Due to its independence on traffic type, it is flexible for use in networks conveying only voice, data or video or a combination of traffic.

e) Work conserving: IPS is a work conserving scheduler because it continues to serve packets as long as input ports keep sending requests to the output ports. If there are no requests, that indicates no packets in the input ports.

f) QoS guarantee: IPS's QoS guarantee is probabilistic
IPS iterates on the choice of the output queue to send a packet to, as opposed to iLQF, iOCF, PIM, and iSLIP. These algorithms iterate their operations and therefore their time complexities are quadratic and cubic. IPS's iteration for the OQ to schedule packet to contributes to making its time complexity linear.

Chapter four expounded on the concept of the MQAS architecture and the IPS algorithm simulated in this research. The performance of some scheduling algorithms discussed so far in this thesis and their simulation results are analysed in chapter five.
5 Performance Analysis of Scheduling Algorithms for VOQ stage of MQAS.

5.1 Existing Scheduling Algorithm

This chapter presents the performance analysis of FIFO, PIM, iLQF, iOCF, iSLIP, and IPS algorithm. The above-mentioned algorithms apart from FIFO and IPS are examples of deterministic iterative maximal matching algorithms [1] [18]. IPS is a probabilistic iterative maximal matching algorithm while FIFO is a pure deterministic algorithm.

iOCF and iLQF are the iterative versions of the Oldest Cell First (OCF) and Longest Queue First (LQF) algorithms respectively. They attempt to find maximal weight matching in a similar manner to PIM and iSLIP. iOCF and iLQF retrieve state information from queues before matching packets to output ports. iOCF considers the waiting time of queued packets while iLQF considers the occupancy of any queue. They find a maximal matching by giving preference to packets that have been waiting longest or to queues with a larger occupancy [18]. While iOCF and iLQF guarantees QoS, PIM and iSLIP do not [1].

5.1.1 Simulation Scenario

The simulation scenario was a case of implementing a single processor where the processor speed is equal to line-rate. In this case, the simulation clock rate is comparable to the processor speed with no pipelining and parallelism. The scenario was taken for the sake of comparing the algorithms. Implementation of a real-time scenario with the use of network processors will be achieved in future work. The advantages that pipelining and parallelism techniques provide will then be used.

The OMNeT++ Discrete Event Simulation System (version 3.2) developed by András Varga, Technical University of Budapest, was used to develop the simulations. A traffic source generated packets as input (offered) load to the hypothetical router with MQAS architecture. These packets were generated with Poisson distributed arrival rate and exponentially distributed inter-arrival rate. The offered load was increased per time by decreasing the inter-arrival time of
packets, thereby increasing the arrival rate of packets. Increasing the arrival rate automatically increases the hypothetical router’s utilization. Utilization (U) is the fraction of time for which a router is busy.

The utilization was increased from 10% to 100% to get the average delay or latency (L) of packets caused by each algorithm used in the MQAS architecture. This set-up allowed the study of the scheduling performance of each algorithm. The scheduling performance of each algorithm was studied by taking a measurement of the average delay of packets as arrival rate increases. The plot of the latency against utilization allowed a measurement of the utilization at which the average delay of packets begins to grow to infinity. Latency and average delay have been used interchangeably. Latency is expressed in clock cycle units. A clock cycle is the discrete time when an event takes place in the simulation. Latency is given by the addition of the queuing delay and service time as shown in Figure 5-1a. Incoming packets have their destinations uniformly distributed over all output ports and the maximum capacity of each input queue was set to 5000 packets.

![Figure 5-1a. Queue model](image)

The OMNeT++ version 3.2 discrete event simulation environment for the multi-stage queuing and scheduling (MQAS) architecture is shown in Figure 5-1b. The algorithms were implemented in the simple module of the MQAS architecture. These simple modules and the sub-modules for the MQAS were developed with C++ programming language and the NED language of the OMNeT++ simulator. Details of the sub-module codes are provided in appendix A: 1 to A: 4. The MQAS general architecture is the screen shot above in Figure 5-1b and the architecture for a router with N=2 is the screen below it where N is the number of ports in a router.
Figure 5-1b. Simulation Environment

5.1.2 FIFO Scheduling

The limitation of FIFO algorithm for VOQ is first illustrated. The FIFO Latency-Utilization (L-U) curve as the number of ports (N) increases is shown in Figure 5-2. Table 5-1 summarizes the maximum utilization (U) with increase in N that the FIFO algorithm can sustain before being unstable. Instability shows that the input queues are beginning to grow and the average delay of packets is tending to infinity. Figure 5-2 shows that for N=8, FIFO offers a maximum utilization (U) limited to approximately 58.58%. Therefore, simulation confirmed that when N is large, the performance of FIFO is limited to 58.58%. The simulation also shows that the 58.58% utilization is asymptotic with N, where small N gives asymptotes above 60%. In the legend for all the plots, N x N indicates N input ports and N output ports, where N=2, 4, 8, 16, and 32.
Table 5-1 FIFO Performance as N increases.

<table>
<thead>
<tr>
<th>Number of Pons (N)</th>
<th>Utilization (U)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.75</td>
</tr>
<tr>
<td>4</td>
<td>0.6533</td>
</tr>
<tr>
<td>8</td>
<td>0.5990</td>
</tr>
<tr>
<td>16</td>
<td>0.5858</td>
</tr>
<tr>
<td>$\infty$</td>
<td>0.5858</td>
</tr>
</tbody>
</table>

Figure 5-2 Latency vs. Utilization for FIFO scheduling algorithm

5.1.3 PIM Scheduling

PIM algorithm considers no time-varying state information but only the occupancy of the input queues [18]. Therefore, it does not require any memory to keep track of the matching of packets from input to output. This makes PIM simple and straightforward to understand. Nevertheless, the effect of the arbitrary manner of matching packets is seen in the L-U curve in Figure 5-3.
The \( L-U \) curve for PIM with a single iteration (PIM-1) in Figure 5-3 shows a poor utilization as \( N \) increases. PIM-1 gives a maximum of 63\% utilization with \( N>8 \). This is a slight increase to the 58.58\% offered by FIFO. However, PIM with four iterations (PIM-4) gives a significant improvement by reducing latency, thereby increasing utilization. PIM-4 remains stable with 95\% utilization as shown in Figure 5-4.

**Figure 5-3 Latency vs. Utilization for PIM-1 scheduling algorithm**

**Figure 5-4 Latency vs. Utilization for PIM-4 scheduling algorithm**
5.1.4 iLQF Scheduling

iLQF algorithm gives preferential service to input queues that are heavily occupied. It uses the matching weight \( \Sigma w_{ij} \) by defining \( w_{ij} \) to be equal to the queue occupancy \( L_{in} \). The algorithm maintains a word memory where each entry indicates the occupancy of an input queue. The word width, \( b \), is determined by the maximum queue length \( L_{\text{max}} \) where \( 2^b \geq L_{\text{max}} \) [18]. Similar to PIM, each iteration consists of three steps. An unmatched input sends a request with its word width bits to each output for which it has a queued packet. The word width indicates the number of packets that it has queued to that output.

When an unmatched output receives a request, it chooses the largest valued request and if an unmatched input receives one or more grants, it accepts the one to which it made the largest valued request. A packet at the head of the longest queue is served first, independent of the number of iterations. However, this scheduling operation starves other input queues. The performance of iLQF with one iteration (iLQF-1) and iLQF with four iterations (iLQF-4) are illustrated in Figures 5-5 and 5-6 respectively.

![Latency vs. Utilization for iLQF-1 algorithm](image)

*Figure 5-5 Latency vs. Utilization for iLQF-1 scheduling algorithm*
5.1.5 iOFC Scheduling

iOFC algorithm gives preferential service to packets that have been queued for the longest time. This is achieved by defining \( w_{i,j} \) to be equal to the waiting time \( (W_{i,j}) \) of packet at the HoQ. For iOFC, the value of the request from an input to an output is equal to the waiting time, \( W_{i,j} \), of the HoQ packet. An iteration also consists of three steps as in PBM and iLQF. Independent of the number of iterations, the packet that has waited the longest time in the input queues is served provided it is a HoQ packet.

iOFC eliminates the starvation problem of iLQF under any offered load because packets at the HoQ that have not been served increase in weight until, eventually, they are served [18]. The performance of iOFC is shown in Figures 5-7 and 5-8. iOFC with one iteration (iOFC-1) in Figure 5-7 gives almost the same performance as iLQF-1. However, iOFC with four iterations (iOFC-4) in Figure 5-8 shows an improvement of the algorithm.
5.1.6 iSLIP Scheduling

As discovered with the other algorithms, and demonstrated in Figures 5.9 and 5.10, increasing iteration improved the performance of iSLIP. iSLIP’s performance moved up to 95% with four iterations because the latency reduces with increasing number of iterations [18]. Like
PIM, iLQF and iOCF, its performance also degrades as N increases. In addition, the algorithm gave the same result for iterations >4 for all N meaning that the algorithm becomes stable with four iterations. PIM, iLQF and iOCF also became stable after four iterations.

Significantly, iSLIP-4 exhibits almost the same performance as PIM-4, iOCF-4, iLQF-4. Unlike these algorithms, iSLIP-4 performance fluctuates as the number of ports (N) increases. Figure 5-11 shows that for all N, the latency is almost constant for very low utilization and increases with utilization increase. Figure 5-12 shows that at high utilization, the latency fluctuates with N. For small values of N, e.g., when N is between 2 and 12, and keeping the utilization between 80% and 100%, as N increases, intuitively utilization decreases because latency increases. However, for values of N, when N is between 12 and 32, as N is increased, utilization and latency fluctuates. The details of the utilization fluctuation are shown in table 5-2.

![Latency vs. Utilization for iSLIP-1 algorithm](image)

Figure 5-9 Latency vs. Utilization for iSLIP-1 scheduling algorithm
Figure 5-10 Latency vs. Utilization for iSLIP-4 scheduling algorithm

Table 5-2 i-SLIP -4 Performance as N increases

<table>
<thead>
<tr>
<th>Number of Ports (N)</th>
<th>Utilization (U)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.952</td>
</tr>
<tr>
<td>4</td>
<td>0.900</td>
</tr>
<tr>
<td>8</td>
<td>0.910</td>
</tr>
<tr>
<td>16</td>
<td>0.875</td>
</tr>
<tr>
<td>32</td>
<td>0.885</td>
</tr>
</tbody>
</table>
From simulations, increasing the number of iterations of the algorithms improves latency and results in better utilization. Although these deterministic algorithms showed a performance of >90%, iOCF and iLQF have high time complexity and are difficult to implement in hardware [18]. However, PIM and iSLIP are simpler to implement, but they do not guarantee QoS. These
algorithms do not provide fairness and can starve traffic in a network conveying different traffic types.

5.2 IPS Scheduling

The IPS algorithm is the proposed scheduling algorithm for the VOQ strategy. In the OMNeT++ version 3.2 simulation environment, arrival of packet was controlled by an exponentially generated inter-arrival rate and a Poisson distributed arrival rate ($\lambda$) "lambda", as generated for other algorithms. The same simulation scenario used for other algorithms was set up. For an arriving packet, its size is its length in bytes, which was chosen randomly between 40 bytes and 1500 bytes according to the Internet packet size distribution proportion specified by RFC2544 and NASA Antes Internet Exchange. The cumulative distribution plot of the packet sizes observed by NASA is in appendix C.1. Traffic was generated as packets with traffic intensity increased in percentage proportion from 10% to 100% by decreasing the inter-arrival times of packets.

The C++ code for the IPS is provided in appendix B. After the next destination output queue (OQ) is chosen by iteration, IPS matches a packet to the chosen OQ by computing the probability of transmission of the packets with the weight of the packet as explained in chapter 4. IPS became stable with one iteration of the destination OQ, i.e. it gave the same result for one or greater than one iteration.

When the number of ports (N) is equal to 2, as utilization increases, the delay experienced by each packet was observed. Figures 5-13 to 5-17 illustrates the behaviour of the delay for 10%, 30%, 80%, 90% and 100% utilization respectively.
Figure 5-13 Delay of packets at 10% utilization with IPS scheduling algorithm

Figure 5-14 Delay of packets at 30% utilization with IPS scheduling algorithm
Figure 5-15 Delay of packets at 80% utilization with IPS scheduling algorithm

Figure 5-16 Delay of packets at 90% utilization with IPS scheduling algorithm
Figure 5-17 Delay of packets at 100% utilization with IPS scheduling algorithm

A small inter-arrival time shows a high arrival rate and a high utilization. The Figures in 5-13 to 5-17 explains that as utilization is increased with more arrival of packets, the delay experienced by each packet increases. At small arrival rate, the delay per packet is minimal as shown in Figure 5-13, the progression of the delay per packet as the arrival rate increases is illustrated with Figures 5-14, 5-15, 5-16 and 5-17. Instability occurs when queues begin to grow indefinitely and the delay of packets grows to infinity, as shown in Figure 5-17. Therefore, the performance of IPS degrades gracefully with network traffic increase. Appendix C.2 gives the distribution of packet delay when N=2.
5.2.1 IPS Latency vs. Utilization with different router sizes

![Latency vs. Utilization for IPS algorithm](image)

**Figure 5-18** Latency vs. Utilization for IPS scheduling algorithm

The performance of IPS became stable between 95% and 98% with increase in the number of ports (N). Delay starts to grow to infinity after 95% and 98% utilization for large values of N and small values of N respectively as shown in Figure 5-18. Details of the latency obtained for each utilization with different number of ports considered are given in Appendix C.3. Table 5-3 gives the maximum utilization before instability for N=2, 4, 8, 16 and 32.
Table 5-3 IPS Performance as N increases.

<table>
<thead>
<tr>
<th>Number of Ports (N)</th>
<th>Utilization (U)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.980</td>
</tr>
<tr>
<td>4</td>
<td>0.9790</td>
</tr>
<tr>
<td>8</td>
<td>0.9643</td>
</tr>
<tr>
<td>16</td>
<td>0.9542</td>
</tr>
<tr>
<td>32</td>
<td>0.9511</td>
</tr>
</tbody>
</table>

From Figure 5.18 and appendix C: 3, IPS gave a delay of approximately 4 clock cycles at 98% utilization for N=2. The IPS delay performance with line rate (clock rate) increase is provided in Table 5-4.

Table 5-4 IPS delay performance as line rate increases while keeping clock rate the same as line rate

<table>
<thead>
<tr>
<th>Line rate</th>
<th>Clock rate</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>0.5</td>
</tr>
</tbody>
</table>
5.2.2 Analysis of IPS performance with increase in Number of ports.

![Latency vs. Number of ports for IPS algorithm](image)

**Figure 5-19 Latency vs. Number of ports for IPS scheduling algorithm**

From Figures 5-3, 5-4, 5-11 and 5-12, PIM's performance degrades as the number of ports increases while jSiJIP's performance fluctuates, because of their scheduling strategy. The input and output ports participate in matching packets by sensing grants and accepting grants randomly or in a round robin fashion. Also, their service policy is not constant. When a queue changes between empty and non-empty the scheduler must adapt to the new set of queues that require service and this increases latency [18]. In IPS, ports are not involved in the matching process and its scheduling policy is a constant mathematical process, so its performance does not degrade increase in number of ports. However, IPS's performance degrades as the number of request (REQs) increases.

For low arrival rate ($\lambda$), the delay converges to a constant as shown in Figure 5-19, but under a high arrival rate the delay begins to rise as $N$ increases. The reason for the rise in delay is because the number of REQ's increases at a high arrival rate.

A packet's delay is a function of the number of elements in set $Z$ formed by IPS. The number of elements in set $Z$ is determined by the number of requests (REQs) made by the input ports to the iterated output queue (OQ) and the number of REQs determines the number of
packets whose parameters will be retrieved during that time slot. The higher the number of parameters to be retrieved the higher the delay. Obviously, for large number of ports, a high arrival rate increases the number of REQs.

For any value of N, the number of requests is \(1 < k < N\) where \(k\) is the number of requests and \(N\) is the number of ports. Therefore, when \(N = 2\), the maximum number of requests at any time is 2 while for \(N = 64\) the maximum is 64.

A key difference between IPS scheduling approach and other algorithms' approach is that the scheduling is probabilistic, which makes the service time probabilistic. The probabilistic nature is because \(k\) takes a random value at any time slot. IPS makes the service time a function of \(k\), and can be related to the queuing delay. From queuing theory, for an \(M/G/1\) queue with random service time, the waiting time in the queue for a packet is given by the Pollaczek-Khinchine or P-K Formula in equation 4 [68]:

\[
W_q = \frac{\lambda X^2}{2(1 - \rho)} \tag{4}
\]

Where \(\lambda\) is the arrival rate, \(\rho\) is the utilization, and \(X^2\) is the second moment of the service time.

For iSLIP, \(W_q\) is given by equation 5 and it is proportional to \(N\), where \(N\) is the number of ports [18].

For IPS \(W_q\) is as expressed in equation 6.

\[
W_q = \frac{\lambda \varphi(k)}{2(1 - \rho)} \tag{6}
\]

Where \(\varphi(k)\) is the IPS service time, which is a function of the number of REQ's at a time. \(k\) is a random value with a probability of \(1/N\) and this probability is the same for all \(1 < k < N\). Therefore, for a fixed high utilization, queuing delay is proportional to the service time, which is a function of the number of REQs.
5.3 Comparison of IPS, PIM, iSLIP, iOCF and iLQF

There are a number of scheduling algorithms, which attempt to find the correct balance between performance, complexity, QoS provisioning and fairness. A router designer may choose a scheduling algorithm or combine the features of several algorithms in order to optimize router's performance. Several factors lead router designers to select one scheduling algorithm over another. First, it must be simple to implement. A complex implementation is not only expensive in area and power, but in most cases provide less performance. Second, the algorithm should provide high throughput and avoid starvation for any traffic type. However, all algorithms are expected to perform worse as utilization increases. Utilization rises when arrival rate of traffic increases. The degradation in performance is not necessarily because of the algorithm, but rather because of the increase in queue length that occurs when the arrival rate suddenly exceeds the service rate.

In Figures 5-20 and 5-21 all the algorithms degraded in performance as N increased from 2 to 32. FIFO has the worst performance while iSLIP-4, PIM-4, iOCF-4, iLQF-4 and IPS showed a better performance in an increasing order.

![Performance of all algorithms when N=2](image)

*Figure 5-20 Performance of all algorithms when N=2.*
Figure 5-21 Performance of all algorithms when N=32.

An advantage IPS has over the other algorithms is its simple scheduling policy, which is an easy mathematical calculation. Since the implementation of mathematical operations is faster than implementing randomness and pointers, IPS has a faster scheduling speed. Its linear time complexity, O(n), also contributes to the fast scheduling speed. In addition, the delay caused by packet's information retrieval is minimal compared with the delay caused by updating pointers for the other algorithms. Furthermore, IPS's simplicity makes it practical enough for implementation in hardware. However, IPS utilizes more memories than the other algorithms. The other limitation in IPS is its need to be able to retrieve packets' sizes on time, any delay in the retrieval of a packet's size adds to the queuing delay.

The operation and scheduling processes of the IPS algorithm can be supported by parallelization and pipelining. Parallelization and pipelining facilitates the optimization of its scheduling performance.

IPS does not need feedback from the output ports and the input ports to make grant decisions. Therefore, its performance does not depend on the number of ports. In PIM, iSLIP, iOCF, and iLQF grants from the input and output ports are needed to match an input to output before a packet is scheduled. With IPS, matching is done using the probability and weights calculated. Therefore, delay in IPS is not affected by the value of N, but by the number of
requests (REQs) in set Z. However, for any value of N, the maximum number of request at a time slot is the same as the number of ports.

Within the MQAS architecture, IPS fairly arbitrates between the $N^2$ VOQ input queued packets by allowing packets to contend based on their estimated bandwidth (eBW) and estimated waiting time (eQ). IPS is independent of traffic type so it does not provide priority in arbitration. Once priority is considered in scheduling, some packets are bound to be starved and treated unfairly. Due to MQAS distributed approach to queuing packets at the input and output ports; it is possible to categorize traffic with priority at the classification level, before packets are queued in the first stage of MQAS. Packets may be classified and queued according to their class of service (CoS), so that VOQs are maintained at each input port for different CoS rather than destination output ports. IPS may then be used to iterate on the destination OQ for the different class of service or flow. In this manner, priority service may be allowed without being unfair to other traffic. The fairness property of IPS implies that a stream of traffic going from an input port, 'i' always get a fair share of service.

Since QoS is a cumulative hierarchical process, for future convergent networks, the only way to guarantee QoS at the first level of the architecture without being unfair to any traffic is by a probabilistic approach rather than a deterministic approach. Therefore, contrary to guaranteeing QoS in a deterministic manner like PIM, ISLIP etc. IPS guarantees a probabilistic quality of service to traffic. Guaranteeing deterministic QoS is unfair and starves packets. Due to its probabilistic approach to QoS guarantee, IPS algorithm fairly allocates the output bandwidth to queued packets and as a result, it does not starve any traffic type. IPS is a scalable algorithm, which can be used within any network carrying any type of traffic or a combination of different types of traffic. Table 6-1 gives a comparison of the properties of all the scheduling algorithms discussed in this thesis.
<table>
<thead>
<tr>
<th>Property</th>
<th>FIFO</th>
<th>iSLIP-4</th>
<th>PIM-4</th>
<th>iOCF-4</th>
<th>iLQF-4</th>
<th>IPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplicity</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Fairness</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Starvation</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>Seableability</td>
<td>☒</td>
<td>☐</td>
<td>☒</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>QoS guarantee</td>
<td>☒</td>
<td>☐</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☐</td>
</tr>
<tr>
<td>Work Conserving</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>Simple to Implement in hardware</td>
<td>☑</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Algorithm type</td>
<td>Deterministic</td>
<td>Deterministic</td>
<td>Deterministic</td>
<td>Deterministic</td>
<td>Deterministic</td>
<td>Probabilistic</td>
</tr>
<tr>
<td>Maximum Scheduling Performance</td>
<td>58.58%</td>
<td>95%</td>
<td>95%</td>
<td>95%</td>
<td>95%</td>
<td>98%</td>
</tr>
<tr>
<td>Time complexity</td>
<td>O(1)</td>
<td>O(N^3logN)</td>
<td>O(N^3)</td>
<td>O(N^3logN)</td>
<td>O(N^3logN)</td>
<td>O(N)</td>
</tr>
<tr>
<td>Limitation</td>
<td>Input blocking</td>
<td>Round Robin scheduling policy</td>
<td>Random scheduling policy</td>
<td>Complex scheduling policy</td>
<td>Complex scheduling policy</td>
<td>Utilizes a lot of processing memory</td>
</tr>
</tbody>
</table>

☑ Indicates that an algorithm exhibits the property.
☒ Indicates that an algorithm does not exhibit the property.

IPS uses the eQ state information to deal with input blocking effectively. Input blocking is a phenomenon that happens when an input port is the only one sending packets to output ports at every scheduling time slot or at most scheduling time slot. If ω denotes the frequency of occurrence of input blocking for any N, in IPS, the occurrence of ω>2 is less than 7% in most cases. Instances of ω>5 virtually do not occur because it considers the last service time of each VOQ in any scheduling process.

The simulation results, performance analysis and comparison of the FIFO, iOCF, iLQF, PIM, iSLIP and IPS were studied in this chapter. Chapter six presents concluding remarks about the study carried out.
6 Conclusions

To achieve a high speed queuing and scheduling of traffic, this research simulated a multi-stage queuing and scheduling architecture (MQAS). The two-stage architecture used the VOQ strategy at the first stage and the OQ strategy at the second stage. The MQAS architecture facilitated the study of the proposed scheduling algorithm. Packets at the VOQ section were served with a probabilistic approach to scheduling and guaranteeing QoS. Probabilistic scheduling is proposed because future NGN is a convergent network that will have to carry traffic generated by varying applications at high speed.

From the QoS hierarchical architecture, QoS provisioning for traffic traversing a network is cumulative. Since QoS is jointly provisioned from end-to-end by all the QoS mechanisms in the architecture, a probabilistic QoS guarantee at the first level of the QoS architecture is more reasonable in NGN. A probabilistic approach will prevent the starvation of traffic at this level because all traffic are allowed to have a fair share of the resources. The simulation results of the probabilistic scheduler termed Iterative Probabilistic Scheduling (IPS) algorithm proved the proposition. IPS allows traffic to contend for transmission based on the state information retrieved by the algorithm. These information are the estimated waiting time (eQ) and the estimated bandwidth (eBW).

FIFO, iOCF, iLQF, PIM and iSLIP are representatives of well known deterministic algorithms. iOCF and iLQF guarantee deterministic QoS but they are complex to implement. PIM and iSLIP are relatively simpler to implement but do not guarantee QoS. They were chosen for analysis in order to compare their scheduling performance with the scheduling performance of the IPS.

Algorithms that guarantee deterministic QoS do so based on priority scheduling of traffic. However, scheduling traffic with priority causes starvation. In addition, most of the algorithms that support priority have high time complexity, are complex to implement in hardware and are not scalable for any network type. Though their scheduling performance can enable a high utilization and they guarantee QoS, the guarantee is for some specific traffic types. Therefore,
some packets are eventually starved of service because priority is given to selected traffic over others or to flows with some particular state information.

iOCF and iLQF exhibited high scheduling performance but their scheduling policy gives priority to queues with particular state information. Therefore, traffic in some queues are starved of resources. iSLIP and PIM also exhibited high scheduling performance but they do not guarantee QoS because of their scheduling policy. A limitation common to these algorithms is their high algorithmic time complexity. Algorithms with high time complexity make the processor do more execution work [93].

The research shows that IPS can achieve a high scheduling speed and a scalable scheduling of traffic in the MQAS architecture. IPS was designed based on the above-mentioned deterministic algorithms but it guarantees probabilistic QoS rather than deterministic QoS. From analysis, the algorithmic time complexity of the IPS is linear-O (n). A linear time complexity not only lowers the execution time of the processor it also reduces the number of operations the processor needs to perform. The linear time complexity enabled IPS to achieve better scheduling performance than the deterministic algorithms. IPS guarantees probabilistic QoS based on its mathematical scheduling policy, which is quite simple to implement in hardware. Contrary to the iOCF and iLQF, IPS is independent of traffic type and fair to all traffic in its scheduling process because it allows all traffic to contend for transmission.

However, the IPS utilized more memory than the other algorithms mentioned. The high memory utilization is because IPS needs to keep track of the state information of all the N² virtual output queues at the input port.

Probabilistic scheduling is proposed for implementation at the first level of the QoS architecture because implementing a strict priority scheduling at the first level of the architecture may starve some traffic. Due to the cumulative nature of QoS over the architecture, strict priority QoS guarantees may be carried out on the other levels of the hierarchy. A probabilistic approach to scheduling will ensure a non-priority QoS and allow all traffic to contend for transmission at the first level of the QoS architecture in NGN environment.
6.1 Recommendation

Firstly, the author recommends more work on probabilistic scheduling algorithms that can provide a probabilistic QoS guarantee to IP traffic at the first level of the QoS architecture in NGN. The probabilistic approach allows traffic to contend for transmission based on the major QoS parameters. The recommendation is given because deterministic scheduling algorithms that guarantee QoS have scheduling policies, which allows priority and they are difficult to implement in hardware. In addition, the relatively simpler deterministic algorithms do not support QoS guarantee [82].

Secondly, the proposed algorithm should be implemented on a processor with pipelining and parallelism techniques for a more efficient scheduling performance.

6.2 Future Work.

Future work on this research involves the implementation of the IPS algorithm on a Network Processor. Among other concepts, parallelism and pipelining will be used in order to optimize the scheduling performance of the IPS. Table 5-1 shows the delay performance of IPS from the scenario simulated. Table 5-4 also illustrates that IPS is able to scale as line rate (clock speed) continues to increase in future.

As the line rate increases, it is possible to improve router performance by increasing the processing clock rate or alternatively by using pipelining. For the sake of comparing the algorithms mentioned in this thesis, data was collected along the dotted line, where the clock rate equals the line rate as shown in Figure 6-1.
Figure 6-1 Data collection region for Simulation Scenario.

For example, if line rate increases from 10Gbps to 60Gbps while keeping processing clock rate to match it, the delay performance of IPS improves as shown in Figure 6-2. Increasing the clock rate will improve the scheduling performance in each algorithm. Yet having an efficient algorithm alleviates the work of the processor and may reduce the cost of the processor. This can also be helpful when the router may migrate in the future towards lower cost and smaller size system such as in System on Chip (SoC).

Figure 6-2 IPS delay performance versus line rate.
References


Packet Radio Service,” Proceeding of the IEE Telecommunications Quality of Service: The


m.htm.

www.itu-t.org

“Network processor requirements and benchmarking,” Elsevier Computer Networks


/ module routerIPS
parameters:
    numPorts: numeric;
submodules:
    hinput: hinput[numPorts];
    display: "i=block/subqueue;p=96,64,column";
    ips: IPS;
    display: "i=block/switch;p=184,72";
    fSink: Sink[numPorts];
    display: "i=block/passiveq;p=288,72,column";
connections:
    for i=0..numPorts-1 do
        hinput[i].out --> ips.in++;
        ips.out++ --> fSink[i].in;
        hinput[i].out' --> fSink[i].fromVoq;
        fSink[i].toIPS --> ips.fromoq++;
        ips.fronips++ --> hinput[i].fromips; //new
    endfor;
    display: "b=812,552";
endmodule
A.3: VOQ ned code

An "input port for the router.
module hiinput
parameters:
    numPorts: numeric const;
gates:
    out: out;
    out: out1;
in: fromips;
submodules:
switch: Switch;
gatesizes:
in[numPorts];
display: "p=60,60,m,1;1=block/switch";
queue: Queue[numPorts];
parameters:
   packetsize = intuniform(x),
   laTime = exponential(y);
   display: "p=150,90,m,1;numPorts,1=block/queue";
connections:
   for s=0..numPorts-1 do
      queue[s].out --> switch.in[s];
      queue[s].fromVoq --> switch.in1++;
      switch.outfromips++ --> queue[s].fromips;
   endfor;

A.4: Input port compound module in OMNeT++ version 3.2
Appendix B: Iterative Probabilistic Scheduling Algorithm
code

```cpp
#include <omnetpp.h>
#include "cSimpleModule.h"
#include "cGate.h"
#include "cArray.h"
#include "cModule.h"
class IPS: public cSimpleModule
{
private:
    int n;
    int z;
    int k;
    double TWP1;  // int sim;
    double sim;
    double prob;
    int x;
    int i;
    int j;
    int y;
    int p;
    double hPP;
    cMessage *nu;
    cMessage *pcktVOQ;
    cMessage *calc;
    cMessage *msg;
    cMessage *copySent;
    cMessage *prmsg;
    cModule *input;
    cMessage *twpp;
    cMessage *packet;
    cQueue IPS_queue;
    cQueue queue;
    cQueue OO_queue;
    int numGates;
    double hpp;
    cQueue twpp_queue;
protected:
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
    virtual void finish();
};
```
Define_Module(IPS)

IPS::IPS()
{
}

IPS::~IPS()
{
}

void IPS::initialize()
{
    int z=par("size");
    int i=0;
    OQ_queue.setName("OQ_queue");
    numGates = gate("fromips")-size();
    double TWP=0.00;
    double hpp=0.00;
}

void IPS::handleMessage(cMessage *msg)
{
    if (msg->arrivedOn("frommq"))
    {
        int k=msg->kind();
        delete msg;
        cMessage *nextEvent=new cMessage("QQ");
        OQ_queue.insert(nextEvent);
        ev <<"IPS starts iterating for OQn"
        for (OQ_queue::Iterator iter(OQ_queue,1); iter.end(), iter++)
        {
            cMessage *msg = (cMessage *) iter;
        }
        ev <<"end iterating for OQn"
        int gate=runif(0,1);
        packet=(cMessage *)OQ_queue.pop();
        packet->setKind(k);
        send(packet,"fromips",gate);
    }
    if (msg->arrivedOn("in"))
    {
        ev <<"IPS starts operationn"
        double eBW=msg ->priority();
        ev << "eBW"<< eBW << endl;
        double eQ=simTime()-msg->length();
        ev << "eQ"<< eQ << endl;
        double WPP=(2*eBW)+eQ;
    }
ev "WPP WPP endl;
double sim=WPP;
double n=msg->timestamp();
ev "ArrivalTime n endl;
int d=msg->kind();
delete msg;
cMessage *pmsg=new cMessage("calculation");
pmsg->setPriority(sim);
pmsg->setKind(d);
pmsg->setTimestamp(n);
queue.insert(pmsg);
double TWPP=WPP+WPP; //new
ev "TWPP TWPP endl;
cMessage *twpp=new cMessage("twpp");
twpp->setLength(TWPP);
twpp_queue.insert(twpp);

{calc = (cMessage *)queue.pop();
double WP=calc->priority();
ev "WP WP endl;
double k=calc->timestamp();
delete calc;
twp=(cMessage *)twpp_queue.pop();
double TWPP1=twp->length();
double prob=WP/TWPP1;
ev "Probability=" prob endl;
double hpp=0.00;
if (prob>hpp) //new
{
hpp=prob;
double x=k;
cMessage *pckt=new cMessage("pckt");
pckt->setTimestamp(x); //Length(x);
init gate = intuniform(0,numGates-1);
ev "Scheduling to OQ" gate endl;
send(pckt,"out",gate);
}
}

void IPS::finish()
{
}
Appendix C: Distribution Plots

C.1: NASA AIX Cumulative Distribution of Internet Packet sizes.

C.2: Distribution of Packet delay when the number of ports=2
### Distribution of packet delay at 30% utilization

- Frequency
- Bin

### Distribution of packet delay at 80% utilization

- Frequency
- Bin

---

#### C.3: IPS Latency table.

<table>
<thead>
<tr>
<th>N</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.0528</td>
<td>0.0758</td>
<td>0.1209</td>
<td>0.1478</td>
<td>0.188</td>
<td>0.3508</td>
<td>0.5084</td>
<td>0.76483</td>
<td>1.52399</td>
<td>9.01292</td>
</tr>
<tr>
<td>4</td>
<td>0.08826</td>
<td>0.126307</td>
<td>0.169427</td>
<td>0.229023</td>
<td>0.271884</td>
<td>0.530212</td>
<td>0.741255</td>
<td>1.16642</td>
<td>1.84503</td>
<td>6.75914</td>
</tr>
<tr>
<td>8</td>
<td>0.085273</td>
<td>0.101392</td>
<td>0.166987</td>
<td>0.234852</td>
<td>0.251066</td>
<td>0.520299</td>
<td>0.732556</td>
<td>1.18859</td>
<td>2.09496</td>
<td>9.5033</td>
</tr>
<tr>
<td>16</td>
<td>0.074191</td>
<td>0.108104</td>
<td>0.173464</td>
<td>0.220275</td>
<td>0.262007</td>
<td>0.533551</td>
<td>0.777388</td>
<td>1.27109</td>
<td>3.60172</td>
<td>12.3283</td>
</tr>
<tr>
<td>32</td>
<td>0.075616</td>
<td>0.106644</td>
<td>0.176788</td>
<td>0.23129</td>
<td>0.293183</td>
<td>0.657152</td>
<td>1.1007</td>
<td>1.8371</td>
<td>4.26568</td>
<td>16.9917</td>
</tr>
</tbody>
</table>

* U: Utilization
* N: Number of ports.
* Latency (unit in clock cycle).
Appendix D: List of Standards.

RFC 791------------------- Internet Protocol.


RFC 2679------------------- A One-way Delay Metric for IP Performance Metric (IPPM).

RFC 1754 / ATM forum ------ IP over ATM.

RFC 2544------------------- Benchmarking Methodology for Network Interconnect Devices.

ITU-T Recommendation Y.1219------------ An Architectural Framework for support of Quality of Service in Packet Networks.

Appendix E: Research Publications.

Parts of this thesis have been published as peer reviewed original papers for international and local conferences. These publications include:


