IMPLEMENTATION AND PERFORMANCE ANALYSIS OF SWITCH FABRIC SCHEDULERS WITH A NEW ACCURATE SIMULATOR SOFTWARE

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

BY

AHMET ADA

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

SEPTEMBER 2014

Approval of the thesis:

IMPLEMENTATION AND PERFORMANCE ANALYSIS OF SWITCH FABRIC SCHEDULERS WITH A NEW ACCURATE SIMULATOR SOFTWARE

submitted by **AHMET ADA** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Gönül Turhan Sayan _____
Head of Department, **Electrical and Electronics Engineering**

Assoc. Prof. Dr. Şenan Ece Schmidt _____
Supervisor, **Electrical and Electronics Engineering Department, METU**

Examining Committee Members:

Prof. Dr. Semih Bilgen _____
Electrical and Electronics Engineering Department, METU

Assoc. Prof. Dr. Şenan Ece Schmidt _____
Electrical and Electronics Engineering Department, METU

Assoc. Prof. Dr. Cüneyt Bazlamaçcı _____
Electrical and Electronics Engineering Department, METU

Dr. Nizam Ayyıldız _____
REHİS, ASELSAN

MSc. İlker Sönmez _____
HBT, ASELSAN

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: AHMET ADA

Signature :

ABSTRACT

IMPLEMENTATION AND PERFORMANCE ANALYSIS OF SWITCH FABRIC SCHEDULERS WITH A NEW ACCURATE SIMULATOR SOFTWARE

ADA, Ahmet
M.S., Department of Electrical and Electronics Engineering
Supervisor : Assoc. Prof. Dr. Şenan Ece Schmidt

September 2014, 64 pages

The switches and routers in computer networks forward the incoming packets that arrive at input ports to their output ports where the connections between input lines and output lines are made by a switch fabric. If the fabric speed can match the aggregate capacity of all input ports, the queuing of the packets is at the output ports. Such output queued arrangements yield the best throughput and delay for the packets together with different levels of Quality of Service Support (QoS) to different flows. However, the speed limits for these fabrics result in queuing at the input ports in practical switch/router implementations. Such devices require the scheduling of the switch fabric which is the decision of the matched input output port pairs. To this end, the design of these fabric schedulers for achieving high throughput, low delay as well as QoS support is an important research problem. The first contribution of this thesis is a software simulator that is called SwitchSim that accurately simulates switch fabric schedulers. The design of the simulator is modular with well defined interfaces following an Object Oriented Approach to enable integrating different scheduler algorithms and traffic generation patterns. It is important to note that SwitchSim is verified by comparing its results to a hardware scheduler together with to the results of the legacy ISLIP scheduler. The second contribution of the thesis is extending ISLIP to support different priority flow to support QoS. Experiments are carried out using SwitchSim to evaluate the proposed fabric schedulers with QoS support and their results are presented with discussions. The results show that upto loads of 70%

the proposed algorithms can provide less delay to the high priority flows without starving the low priority flows.

Keywords: switch fabric schedulers, QoS support, flow prioritization, software simulator

ÖZ

ANAHTAR ÖRGÜSÜ ÇİZELGELEYİCİLERİNİN YENİ, DOĞRULANMIŞ BİR BENZETİM YAZILIMI İLE GERÇEKLENMESİ VE BAŞARIM İNCELEMESİ

ADA, Ahmet
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. Şenan Ece Schmidt

Eylül 2014 , 64 sayfa

Bilgisayar ağlarında anahtarlama cihazları ve yönlendiriciler, giriş kapılarından gelen paketleri çıkış kapılarına iletirler. Bu cihazlarda giriş ve çıkış kapıları arasındaki bağlantı bir anahtarlama örgüsü tarafından sağlanır. Eğer anahtarlama örgüsü hızı bütün giriş kapıların kapasitesinin toplamını eşleştirmeye yeterli ise, kuyruklama çıkış kapılarında yapılır. Bunun gibi çıkışta kuyruklamalı yapılar, paketler için farklı akışlar için farklı seviyelerde hizmet kalitesi sağlamanın yanında en iyi çıkan iş oranı ve gecikme değerlerini verir. Ama bu anahtarlama örgüleri için hız limitinin olması, tatbiki yönlendiricilerde giriş kapısında kuyruklamalı yapının kullanılması ile sonuçlanmıştır. Bu cihazlarda giriş ve çıkış kapılarının eşleştirilmesini sağlamak amacıyla anahtarlama örgülerinin çizelgeleyicilere ihtiyacı vardır. Bu bağlamda, bu anahtarlama örgüsü çizelgeleyecilerinin hizmet kalitesi desteğiyle birlikte yüksek çıkan iş oranı ve düşük gecikme ile tasarlanaması önemli bir araştırma problemidir. Bu tezin ilk katkısı anahtarlama örgüsü çizelgeleleyicilerinin doğru bir şekilde benzetimini yapan SwitchSim adındaki bir benzetim yazılımıdır. Benzetim yazılımı tasarımının Nesne Tabanlı bir yaklaşımla iyi tanımlanmış arayüzleriyle birimsel olarak yapılması farklı çizelgeleleyici algoritmalarının ve trafik desenlerinin entegre edilmesine olanak tanır. SwitchSim yazılımının, donanınmsal bir çizelgeleleyici ile ISLIP çizelgeleleyicisi algoritması sonuçlarıyla doğrulanmış olması önemli bir noktadır. Bu tezin ikinci katkısı ise ISLIP algoritmasının farklı seviyelerde hizmet kalitesi sunacak şekilde ge-

nişletilmesidir. Hizmet kalitesi desteği veren bu anahtarlama örgüsü çizelgeleleyicilerini değerlendirmek üzere deneyler yapılmış ve deneylerin sonuçları tartışmalarıyla birlikte verilmiştir. Sonuçlar %70 yüküne kadar, önerilen algoritmaların düşük öncelikli akışlara servis açlığı çektirmeden yüksek öncelikli akışlara daha az gecikme sağladığını göstermiştir.

Anahtar Kelimeler: Anahtarlama örgüsü çizelgeleyicileri, hizmet kalitesi desteği, akış önceliklendirmesi,benzetim yazılımı

To my wife-to-be and my family

ACKNOWLEDGMENTS

I would like express the deepest appreciation to my supervisor Associate Professor Şenan Ece Schmidt for her constant support, guidance and friendship. It was great honor to work with her for the last three years. Our cooperation has influenced my academical life and my world view deeply.

There are a lot of people that were with me in these three years. Their love, help and understanding were the only power I had. They are true owners of this work. It is not possible to write down why each of them important to me, because it takes more space than the work itself.

I would like to thank my manager and supervisor at work, Sıdıka Bengür and İlker Sönmez; my colleagues Veysel Çakır, Deniz Bardak, Talip Küçükkılıç, Mehmet Ali Akyol and Şenol Ergenç for their support and tolerance to me. I am also deeply grateful to Volkan Çağlayan, Kaan Çetinkaya, Gökmen Cengiz, Orhan Can Görür, Mehmet Bülbüldere, Onur Ünver, Ozan Bektaş, Raşit Nedim Tok, Osman Arslan, Bekir Said Çiftler and Abdullah Kaya for their encouragement and friendship.

Special thanks to my beloved girlfriend Ayşe Soylu and my family members Mustafa Ada, Emine Ada, Mehmet Ada and Merve Ada. Words cannot express how grateful I am to them. Their love and prayer incented me to strive towards my goal.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

CB	Central-stage Buffered
DCCP	Datagram Congestion Control Protocol
DRR	Deficit Round Robin
FTP	File Transfer Protocol
HOL	Head of Line
IMC	Intelligent Multi-class Controller
IQ	Input Queued
IP	Internet Protocol
IETF	Internet Engineering Task Force
ISLIP	Iterative Round Robin with SLIP
MAC	Medium Access Control
MSM	Memory-Space-Memory
NS	Networ Simulator
OQ	Output Queued
OSI	Open Systems Interconnection
OPNET	Optimized Network Engineering Tools
QoS	Quality of Service
PIM	Parallel Iterative Matching
RRR	Recursive Round Robin
SCTP	Stream Control Transmission Protocol
SPES	Scalable Simulation Platform for Switching and Scheduling
SFQ	Stochastic Fair Queuing
TCP	Transmission Control Protocol
UDP	Unified Datagram Protocol
VOQ	Virtual Output Queue
VC	Visual C
WFQ	Weighted Fair Queuing
WF2Q	Worst-case Fair Weighted Fair Queueing

WF2Q+	Worst-case Fair Weighted Fair Queueing Plus
WRR	Weighted Round Robin

CHAPTER 1

INTRODUCTION

With recent developments on the network infrastructure and the end devices ; IP traffic in the world is increasing rapidly. Annual traffic in the world will exceed zettabyte level by 2015. Furthermore, the fore cast is more than 70% of the IP traffic will be video [1]. Delivery of video traffic requires end-to-end Quality of Service (QoS) support such as bounded delay and guaranteed bandwidth. While QoS facilities exist at the end systems at transport and application layers, the effects of such solutions are diminished if they are not supported by the network layer. To this end, network switches and routers implement flow-based queuing and the scheduling of these queues at their output ports to provide differential service to flows.

It is important to note that, the switch fabric has to operate at the aggregate speed of the input ports to get the full benefit from these output queue schedulers. However, it is very expensive and often not possible to reach such fabric speeds at hundreds of Gbps of line speeds at the network core. Hence, the switches and routers are built with queues at the input ports. Such architectures are called Input Queued (IQ) where the queues at the input are organized as Virtual Output Queues (VOQ) and a fabric scheduler algorithm decides the matching of the inputs and outputs. The previous work in the literature focuses on fabric scheduling algorithms that maximize the throughput. To this end, a very prominent algorithm is ISLIP [13] which aims to achieve a maximal matching of inputs and outputs by employing a request-grant-accept arbitration between inputs and outputs with round robin prioritization of the ports to avoid starvation.

The current IP traffic requirements and the line speeds motivate this thesis to inves-

tigate fabric schedulers and QoS support in fabric schedulers. To this end, the first contribution of the thesis is a software simulator for fabric schedulers SwitchSim. The well-known and widely used network simulators such as NS2 and OPNET do not provide detailed implementation and evaluation support of fabric schedulers. SwitchSim is a modular cell-level simulator that implements the IQ switches in detail together with traffic generator, fabric scheduler, packet to cell segmentation and reassembly modules. It has well defined module interfaces enabling the integration of any fabric scheduler algorithm. SwitchSim is verified by comparing its results with the results of [13] together with to the results of a hardware fabric scheduler implementation.

The second contribution of the thesis is to propose two extensions to the ISLIP algorithm which support per flow QoS support to prioritize traffic without starving the lower priority flows. The thesis presents extensive simulation results and their discussions to evaluate these algorithms under different traffic scenarios.

The rest of the thesis is outlined as follows.

In Chapter 2, overview of previous related works are given with their features and drawbacks.

In Chapter 3, as background information, general concepts of IQ switching and ISLIP algorithm basics are given. Operation cycle of ISLIP algorithm which consists of request, grant and accept phases is explained. In ISLIP, fairness between input/output ports are obtained by using pointers. Working principles of these grant and accept pointers are investigated. Maximal matching which is used by ISLIP is explained.

In Chapter 4, features of the SwitchSim simulation tool which is developed in an object oriented manner with C++ language, are listed. Assumptions made during development are given. The main modules, the submodules and the interaction among them are explained in detail.

In Chapter 5, two proposed extensions to the ISLIP algorithm are given and a new VOQ structure with "Intelligent Multi-Class(IMC)" controller, is presented. Names of the proposed algorithms are “Strictly Prioritized ISLIP” and “Limited Prioritized ISLIP”. Both of these algorithms deploy IMC Controller with new proposed Multi–Class Virtual Output Queuing structure.

In Chapter 6, conducted experiments and results are given. Some of the experiments in [13] are repeated and compared for the sake of the reliability of the simulation tool.Moreover, performance analysis of the new proposed algorithms is made and compared with conventional ISLIP and PIM algorithms. In conventional ISLIP, effects of the maximum iteration number and effects of the number of input/output ports (switch size) on average cell delay performance are observed. An experiment showing the behavior of standard ISLIP with bursty traffic is conducted and results are given. PIM algorithm and ISLIP algorithm are compared in terms of average cell delay performances.

In Chapter 7, the conclusion of the thesis work is presented with possible future work.

CHAPTER 2

RELATED WORK

2.1 Switch Fabric Scheduler Simulators

The main contribution of this thesis work, is to develop a software simulator for *input queued* switch fabric schedulers with its traffic generator, segmentation and reassembly modules. To this end, in this section, we present an overview of the existing network simulators.

NS-2[3] and OPNET[4] are well-known, widely used network simulation and modeler tools. Hence, their correctness is verified by a very large number of studies that use these tools with consistent results. Their capabilities are so rich that they support simulation of almost all well-known protocols of all layers of the computer networks. Moreover they allow users to make customizations of these protocols. However, their features are very poor in switching technologies. Users have no chance to analyze switch fabric structures and the scheduler performances related to them. Making modification of the simulated switch/router models on switch fabric level is not possible.

In [17], a general purpose switch fabric scheduler and simulator called SPES is developed with support of variety of fabric architecture and fabric scheduler algorithm. SPES supports Output Queued, Input Queued, Combined Input-Output Queued, MSM Clos Network, CB Clos Network and Benes switch fabric architectures. It supports all of the well-known switch fabric algorithms such as DRR, WRR, RRR, SFQ, WFQ, WF^2Q, WF^2Q+, ISLIP, PIM, etc. Moreover SPES supports variety of traffic model and it allows users to add new traffic models as well. SPES tool is developed in

object-oriented manner with VC++. Despite these advertised features, the authors do not provide a verification of the tool that gives confidence to the user about its correctness. No citations are found to [17] in our literature survey.

In [5], a C++ based simulator developed in order to examine the Clos Network switch fabrics. Clos Network is a multi-stage switching network that deploys multiple crossbar switches in parallel or pipelined. The authors verify their research on Clos Network switches with the simulation tool developed during their work. However, the authors do not provide a verification of the simulator that gives confidence about its correctness.

2.2 Flow Prioritization and QoS Support on Different Layers

Another contribution of this thesis work is the prioritization of specific flows on switch fabric level. As an example, multimedia packets which have more stringent QoS metrics should be prioritized at first place.

Various studies have been done on transport, network and application layers for QoS support. One approach is SCTP(Stream Control Transmission Protocol). In SCTP datagram oriented feature of UDP and reliability and sequencing properties of TCP are combined for QoS purposes [16]. Another transport layer protocol published by IETF at 2006 is DCCP (Datagram Congestion Control Protocol)[10]. DCCP has improved congestion control mechanism of TCP with using explicit congestion notification. Moreover, at application layer great effort has been given to meet QoS requirements of multimedia applications; data buffering at the clients is only one of them.

In[14] VAIPA, a video aware Internet protocol architecture is proposed. VAIPA offers better bandwidth utilization for multimedia traffic, by adding video aware intelligence to label switching routers which are part of the service provider's core network. In[11], a fast rerouting for strategy for IPTV networks is proposed in order to eliminate link failures on the network infrastructure.

For the current networks, quality of service requirements at network layer are satis-

fied at the output by using Output-Queued (OQ) switches. OQ switches guarantee 100 % throughput; because in OQ switches, packets are directly put into queues at the output according to their destination port. This creates an advantage of separating switching works and QoS works. A lot of efforts have been spent and various algorithms have been developed in order to guarantee QoS requirements at output. For example: Weighted Fair Queuing (WFQ) [9], Deficit Round Robin (DRR)[15], Worst-Case Weighted fair Queuing (WF^2Q)[8]. The OQ switches, it is required that the switch fabric is *N* (port number) times faster than line rate. That makes either impossible or impractical to use OQ switching in core networks, while switches have large number of ports operating at very high speed.

CHAPTER 3

SWITCHING ALGORITHMS AND ISLIP

3.1 IP Routers/Switches and Their Functionality

Routers and switches are both computer networking devices. They both provide connection between computers or between networks. Actually, switches operate at layer-2 of the OSI model with using MAC-addresses. Routers operate at layer-3 of OSI model with using IP addresses.

IP routers/switches' functions can be considered in two categories: Data functions and control plane functions. The control plane functions include system management and drawing the network map by exchanging routing table information. Routers exchange their network topology information by some routing protocol such as: OSPF, RIP, and BGP. By doing so, they have the ability to create forwarding tables. These functions are carried on by control plane and give service to the data plane. Data plane functions consist of switching IP packets from input port to the output ports. By looking at the forwarding table and the destination address of the IP packets, source/destination ports of the packet inside the router are decided. Here we need to note that difference between switches and routers is at the control plane where their data planes employ identical hardware designs. In terms of switching functionalities,a switch and a router operate in same manner. Therefore "switch" and "router" terms are used interchangeably in this thesis.

3.2 Definition of Basic Concepts

Output Contention:

For an $N \times N$ IP switch, there is always a possibility to have concurrent multiple IP packets that are destined to the same output port. This situation is called *output contention*, in other words, multiple input ports are contending for the same output port at the same time.

Arbitration:

Deciding the winner(s) of the output contention is called arbitration. Arbitration has critical importance in terms of fairness and complexity of a switching algorithm.

Head-of-Line (HOL) Blocking:

A FIFO IP packet buffer would contain packets that are destined to different output ports. Since this buffer is FIFO, only the first packet can be served blocking the other packets that can be possibly switched. This phenomenon is called Head-of-Line blocking.

Virtual Output Queuing:

In order to eliminate HOL blocking problem, Virtual Output Queuing (VOQ) strategy is used. If a switch uses VOQs, each input port has separate queues for each output port. That also means that there are $N \times N$ queues at the switch inputs.

Speed-Up:

Speed-Up (S) is defined as the ratio between line rate and the switching rate of a switch. In other words, switching capability (as number of packets) of a switch during a packet time is defined as Speed-Up.

Cell:

In the past years, a lot of academic research has been done in order to develop efficient ATM switches[7]. These research efforts lead switching technology to use ATM-like cells. Consequently, many commercial Internet routers/switches segment IP packets

into ATM-like cells at the input interfaces and transfer them to the output interfaces to be reassembled. For example,Cisco Inc, whose market share is over 60 % [2] , uses ATM-like cells for their commercial switches as well.

Packet/Cell-Time:

Time required in order to switch one fixed size packet/cell from an input port to an output port.

3.3 Switches by Their Queuing Types/Strategies

For a switch that have *N* input ports, there is always output contention. Therefore, packets that lose contention must be buffered. Switch/Router architectures can be categorized by their buffering strategies. Each one has their own advantages and disadvantages.

3.3.1 Output-Queued Switches

In this kind of switches, all packets that are present at the time must be switched before a new packet arrives; regardless of how many packets are destined to the same output port. Since switching rate is much faster than line rate, buffering at the output port is a must. Queuing at the output port causes limitation of the switch size. In an IP switch there is always a chance that all input ports have packets destined to the same output. Since there are N input ports, N concurrent packets that are all destined to the same output port might be present in *a packet time*. Therefore switching capability of the switch must be at least N times faster than the line rate. Especially at core networks which operate at very high speeds, this limitation makes it either impractical or very costly to implement this kind of switches. Queuing at the output ports brings the advantage of ordering of IP packets before they are sent to the lines. Therefore QoS requirements are easy to meet when Output-Queued(OQ) switches are used.

3.3.2 Input-Queued Switches

For the Input-Queued (IQ) switches, the output port contention at the input ports of the switch is very critical. Because unlike OQ switches, at the IQ switches packets that lose contention must be buffered at the input ports. According to the ratio between line rate and switching rate (Speed-Up), multiple packets might be switched at a packet time. This kind of switches are designed in order to prevent size limitation problem of the OQ switches. Therefore their Speed-Up(S) never reaches to N (input port number). Since N>S and there is always a chance that all input ports have packets destined for the same output port; (N - S) packets lose contention at maximum. These packets are queued at the input ports. IQ switches might have FIFO queues for each input port or they might have VOQ for each input port. Mostly VOQs are used for commercial IQ switches. Because FIFO queues suffers from HOL Blocking problem. Since output port contention is very critical at IQ switches, fabric scheduling algorithms that manage this contention have gained great importance and have become a hot research topic. According to the design requirements these algorithms might get extremely complex. For commercial uses, the switch vendors generally choose simple and fair algorithms like ISLIP (Iterative round-robin matching with SLIP.)[13] The interest areas of this thesis are includes IQ switches and IQ switch fabric scheduler algorithms, especially ISLIP, therefore ISLIP algorithm will be explained in details.

3.4 Fabric Scheduling Switching Algorithms

At core networks, switches/routers operate at “very high speed”.However the perception of fastness, is highly related to the current technology of the time. In other words; 1 Gbit/s was very high speed fifteen years ago, considering the technology available at those years. For example, high speed switches were first designed to operate at couple of hundreds of Mbit/s. Since the network data speed was at that level, the OQ switches were able to handle that amount of data. Because of their throughput and delay performance; the OQ switches were the focus at that time. Most of the commercial switches are designed as OQ switch by vendors. However as the data rates are

getting higher and higher with the evolving technology and widening networks, the OQ switches have become impractical or very costly to implement. The IQ switches have become alternative to the OQ switches, because IQ switches don't suffer size limitation. However there were two major issues with the IQ switches as well. The first one was HOL Blocking problem. This problem was limiting the throughput to %58.6 and is prevented by using VOQ at the input ports. The second problem is the output port contention and this problem brings the necessity to use complex scheduler algorithms for arbitration. Following metrics have great importance when choosing or designing a fabric scheduler algorithm for an IQ switch.

•**Efficiency:** A fabric scheduler algorithm should provide high throughput and low delay.

•**Fairness:** VOQs at each input should not starve. Every input port should get as fair as possible service.

•**Stability:** Queues at each input port should not grow without bound under any traffic pattern.

•**Complexity of Hardware Implementation:** Algorithm should not be too complex to implement on the hardware. Complex hardware implementation brings high scheduling times which means limitation of the switch line speeds.

As the above metrics are being considered, the ISLIP algorithm which is proposed by Nick McKeown[13] is the most preferred algorithm in today's switches. Many leading vendors deploy ISLIP or ISLIP based scheduler algorithms for their commercial switches. The developed simulation software supports algorithms based on ISLIP algorithm. ISLIP algorithm is a fabric scheduler algorithm making matching between input ports and output ports. ISLIP is developed for IQ switches which uses fixed-size packets (cells) as switching unit. This algorithm configures the crossbar switch, by determining the serving order of the arriving packets.[13]

3.5 ISLIP Algorithm

ISLIP uses maximal matching. Maximal matching is a type of matching which incrementally adds connections without breaking the previous connections. Maximal matching actually makes less or equivalent connections than the maximum matching (maximum number of connections among all possible connection sets) and tries to converge to it. However it is very simple to implement and to compare to the maximum match. ISLIP reaches maximal matching iteratively. In each iteration new connections can be made. These iterations are composed of three main phases. Phases of iterations:

1. Request: In this phase, all of the input ports send request to the output ports for those they have at least one cell.

2. Grant: In this phase, each output port chooses an input port among the ones that made request for itself at the first phase of the iteration. There might be no request for an output port; in that case no grants will be made.

3. Accept: In this phase, each input port chooses an output port among the ones that made grant for itself at the second phase of the iteration. There might be no grant for an input port; in that case no accepts will be made.

In phase two and phase three there must be a selection algorithm for choosing among multiple request/grants. ISLIP deploys round-robin scheme to schedule input and output ports in turn. The actual switching of the cells take place after the iterations are completed.

Round-Robin Scheme:

Round Robin scheduling scheme is a circular order scheme and provides equal share scheduling. The fairness or starvation-free properties of ISLIP comes from Round-Robin scheduling scheme.

In order to apply round-robin scheme; there are grant pointers for each output port and accept pointers for each input port. Grant and accept pointers are updated only if accept phase is completed successfully in the first iteration. In the subsequent

iterations pointers are not updated.

Operation cycle:

One operation cycle should contain matching(s) and switching(s). Operation cycle is the time that elapsed during a fixed sized cell is switched if the speed up of the switch is one. If the speed-up(S) of the switch is more than one; in one operation cycle S times matching and S times switching are made. In one matching process, at least one iteration occurs; if the maximal matching has not been reached extra iterations might be tried. This is a parametric variable, according to the switch's traffic conditions and size, maximum iteration number can be optimized.

CHAPTER 4

DEVOLOPED SOFTWARE SIMULATION TOOL: SWITCHSIM

SwitchSim is the simulation tool that is developed within the framework of this thesis. The purpose of SwitchSim is to analyze the performance of commercial switch fabric algorithms and new proposed algorithms. In this chapter; firstly, features of SwitchSim and assumptions made during development will be listed. Then functionality and working principles of the simulation will be explained module by module. Finally, software development details will be given.

Features of SwitchSim:

• Written in C++ language in object-oriented manner.

• Simulates real behavior of a switch fabric of CICQ and IQ switches.

• Purpose is to measure switch fabric algorithms' performance.

• Supports different number of input or output port numbers and port numbers are adjustable.

• Supports variable size packets: Packet sizes are measured in multiple of a size-unit: namely *cell*.

• Packets are segmented into cells at inputs and reassembled at outputs.

• Time is not real: a time unit is defined as cycle. Cycle is defined as time elapsed during a single size-unit (one cell) packet is switched from input port to output.

• Multiple speed-up is supported and is adjustable.

• Multiple iterations are supported and the number of iterations is configurable.

• Different fabric scheduling algorithms can be adapted to the software.

Assumptions:

• Line rate of the input and output are normalized to 1 cell/cycle.

• The processing time required to match input ports and output ports, is ignored.

• Time required to segment packets to cells, is ignored.

• Time required to reassemble cell into packets, is ignored.

Modules of the Software

SwitchSim consists of different software modules and these modules consist of their submodules. These submodules and modules interact and exchange information with each other. In this Section functionality of these modules and the interaction between them will be explained. The list of software modules is given below.

• Builder Module

• Traffic Generator Module

• Packet Segmentation Module

• Input Port Module

• Switch Fabric Module

• Output Port Module

• Packet Reassembly Module

• Statistics Module

4.1 Traffic Generator Module

The properties of the *traffic generator module* are listed as follows:

• Generates traffic in order to assist switch fabric simulation.

• Packets are not real IP packets, just objects of *packet class* defined in the software.

• Packets contain input and output port number, birth time and size

• Supports variable size packet generation.

• Each input port is treated equally, input port number can be configured.

• Supports different types of traffic shapes: Poisson traffic, bursty traffic.

• Packets can be labeled with different flow numbers.

• Traffic load can be adjusted between 0 and 1.

Submodules of the *traffic generator module* and the interaction between them are given in Figure 4.1.

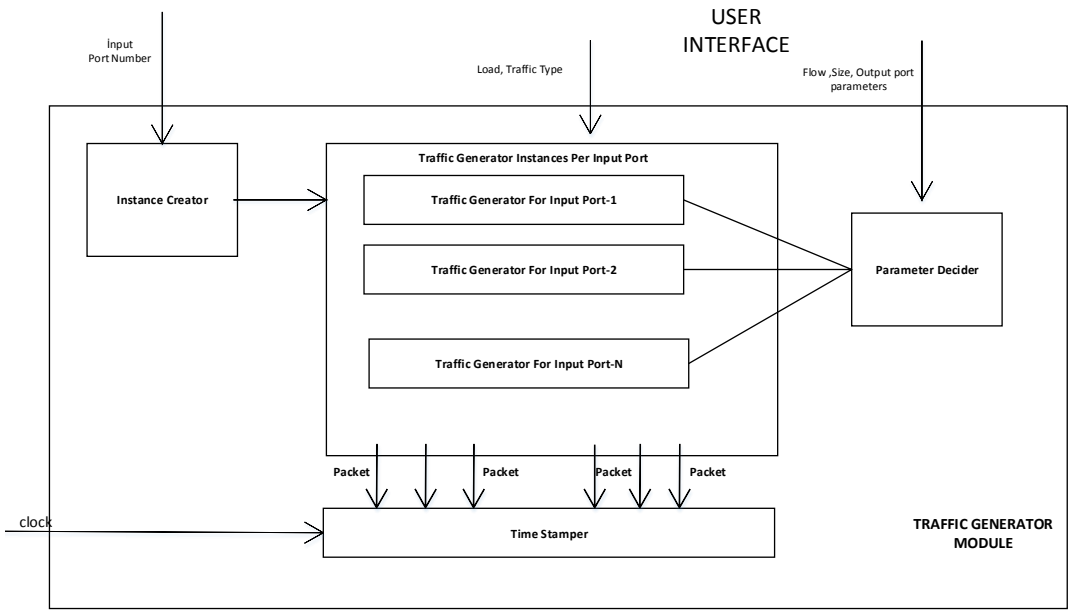

Figure 4.1: Traffic Generator Submodules

Configurable parameters like input port number of the switch, traffic type, load, output port distribution, flow number and flow distribution are gathered by user choice.

According to this information:

A submodule of traffic generator *instance creator* creates traffic generator instances for each input port. These traffic generator instances are considered to work independently.

Traffic generator instances create packets according to two parameters; namely, *traffic load* and *traffic type*. Traffic load is a value between 0 and 1 and the offered traffic load for the simulation is determined by looking at this parameter.*traffic type* determines the shape of the traffic. The developed SwitchSim supports Poisson and bursty traffic models. New traffic models can be added to the simulator easily.

After packets are created by traffic generator instances; *parameter decision* submodule decides what the destination output port of the packet is, what size the packet is, which flow the packet belongs to. Moreover this submodules give every packet a unique packet number. Destination ports of the packets are determined randomly and uniformly. Sizes of the packets are determined by an average value chosen by the user. Flow number is an extra label that is required for testing prioritization of flows. The number of flows and ratios to the total offered load are determined by user choice.

After parameters are set to packets, the *time stamper* submodule stamps every packet at the birth. This time stamp is called birth time and used for calculating queuing delays and end-to-end delays.

4.2 Packet Segmentation Module

The properties of the *packet segmentation module* are listed as follows:

- Variable-sized packets are segmented to cells.
- Information in the packets transferred to each cell.
- Assumption: packets are exact multiples of a cell-size.

The submodules of the *packet segmentation module* and the interaction between them

are given in Figure 4.2.

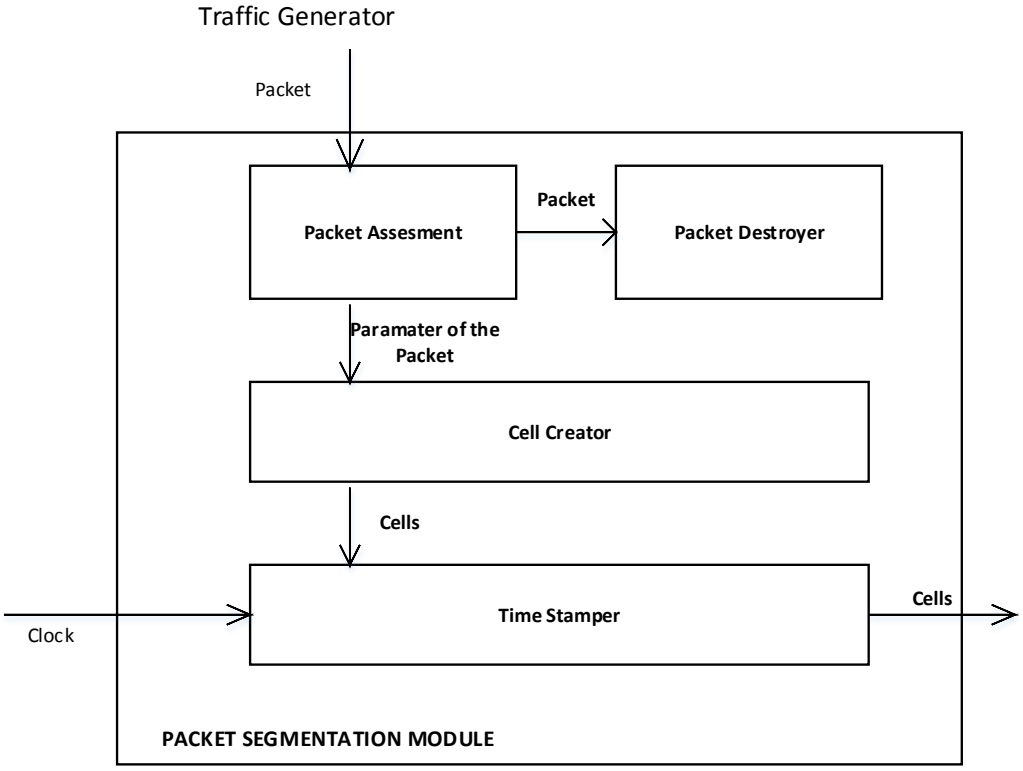

Figure 4.2: Packet Segmentation Submodules

Generated packets arrive directly to the packet segmentation module after the traffic generator is done with them.

Firstly, *packet assessment* submodule reads the information in the packet such as: source port no, destination port no, packet no, flow no, birth time.

The above parameters read by packet assessment submodule, are forwarded to the *cell creator module*. In the meantime; the original packet is destroyed by *packet destroyer* submodule.

Cell creator submodule creates cells according to the size of the packet. If a packet with size "n cells" is being processed; cell creator, creates n cells with same parameters (source port no, destination port no, packet no, flow no, birth time).The only difference of those n cells is the cell no which is given by cell creator module itself.*cell no* is given starting from 0 and incremented for each different cell. This

parameter is packet based. In other words different cell may have the same *cell no* if they are cells of distinct packets.

After the cells are created, the *time stamper* submodule stamps each cell with the current time and these time stamps are kept in the cells as a parameter, namely:*segmentation time*.

4.3 Input Ports Module

The properties of the *input ports module* are listed as follows:

• Variable port number is supported.

• Cells are queued in the input ports.

• VOQs are used in order to discard HOL-Blocking problem.

• Each input port has its own VOQ for cells.

• VOQs are modified in order to give priority to specific flows.

Submodules of the *input ports module* and the interaction between them are given in Figure 4.3.

According to the input port number of the router which is set by user; builder module of software builds input ports with their Virtual Output Queues (VOQ) at the beginning of the simulation. VOQ structure also depends on the algorithm type. The differentiation of the VOQ structure for supporting flow prioritization is given at Chapter 5.

After segmentation, cells arrive to input ports. When a cell comes to an input port, it should be classified. This is necessary in order to place the cell in the right queue. Each input port has its own VOQ. In conventional usage of these queues, cells are queued, according to their destination ports only. A cell with input port i $(0 < i \leq N)$ and output port j $(0 < j \leq N)$ is placed in the queue $Q_{(i,j)}$. Each queue is a FIFO queue. In addition to that conventional usage, our design classifies the cells according to their flows too. A cell with input port i $(0 < i \leq N)$ and output port j $(0 < j \leq N)$

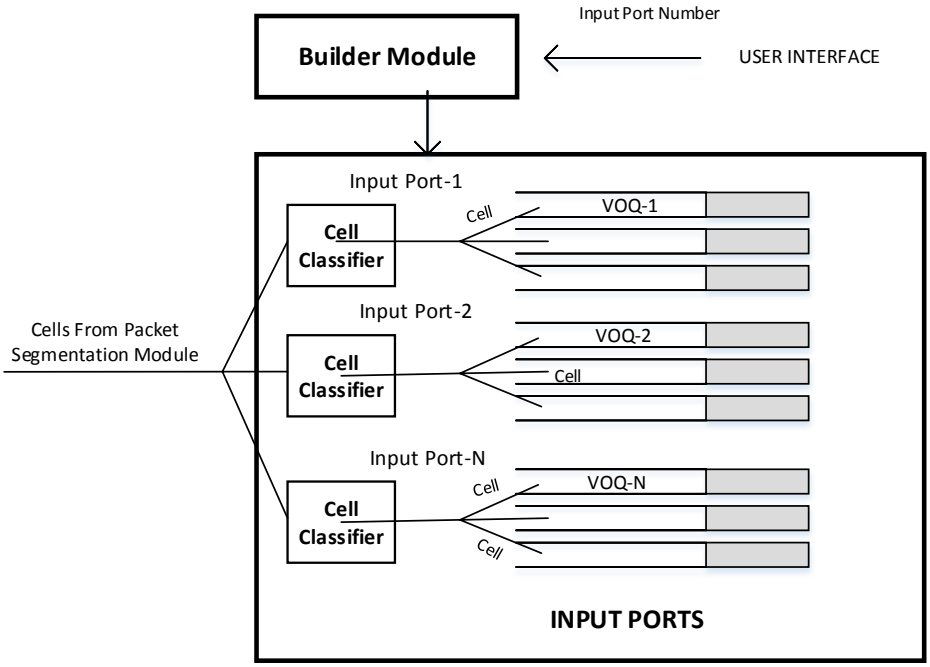

Figure 4.3: Input Ports Submodules

and flow number f $(0 < f \leq F)$ is placed in the queue $Q_{(i,j,f)}$. Each $Q_{(i,j,f)}$is a FIFO queue. Cell classifier submodule reads the required parameters and puts the incoming cell to the appropriate queue.F is the *maximum flow number* and N is the *input/output port number*.

After classification cells are ready to be switched and waiting their turn.

4.4 Switch Fabric Module

The properties of the *switch fabric module* are listed as follows:

• Simulates NxM switch fabric scheduler. Input/output port numbers are variable

• Not a specific algorithm. Adaptive to new algorithms. Supports PIM,ISLIP, SP-ISLIP, LP-ISLIP.

• Multiple speed-up is possible

Submodules of the *switch fabric module* and the interaction between them are given

in Figure 4.4.

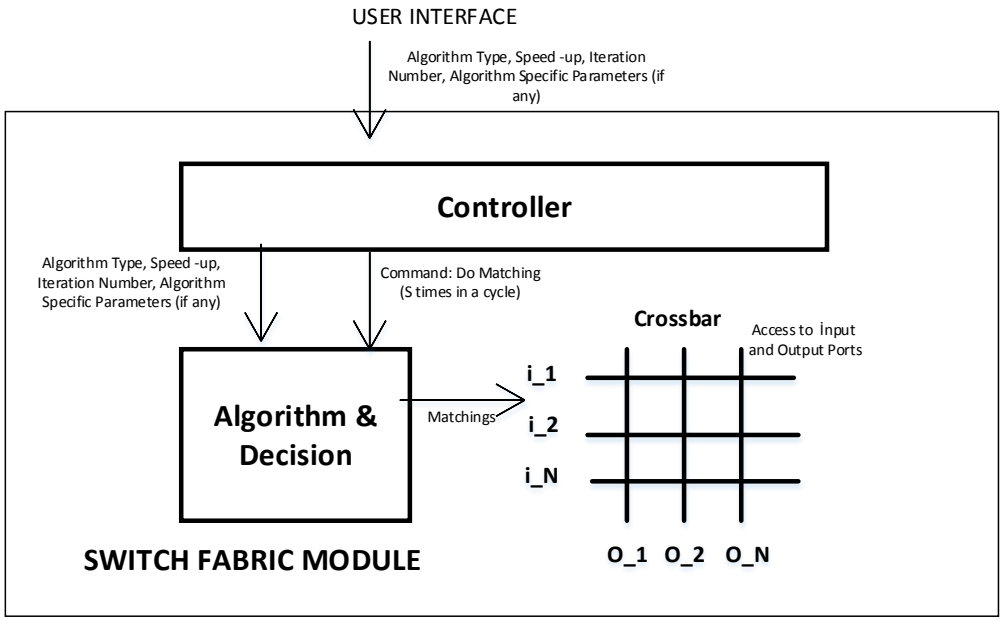

Figure 4.4: Switch Fabric Submodules

Switch fabric module supports variable input/output port numbers. After user choice of input/output ports number, NXM type crossbar is built by builder module.

This module is written highly modular in order to adapt new algorithms. *Algorithm and decision* sub-block can be easily replaced with a new one.

Algorithm and decision block decides the matchings between input and output ports. Then it gives these matchings to the *crossbar* submodule.

Crossbar submodule does the switching process according to the information coming from *algorithm and decision* submodule. *Crossbar* submodule has right to access to input ports and output ports.

The switching process can be done multiple times during one cycle. This is called speed-up. The speed-up number is controlled by the *controller* submodule. And the number of the speed-ups are determined by the user choice.

The last step of the switching process is time stamping. The switching time of each cell is stamped as switching time to the cells. This is necessary in order to the statistics.

4.5 Algorithm Module

This module is a submodule of the *switch fabric module*. It is the heart of SwitchSim and has its own submodules. According to the user choice the structure of *algorithm module* can be different. Therefore all of the supported algorithms will be explained separately.

4.5.1 ISLIP

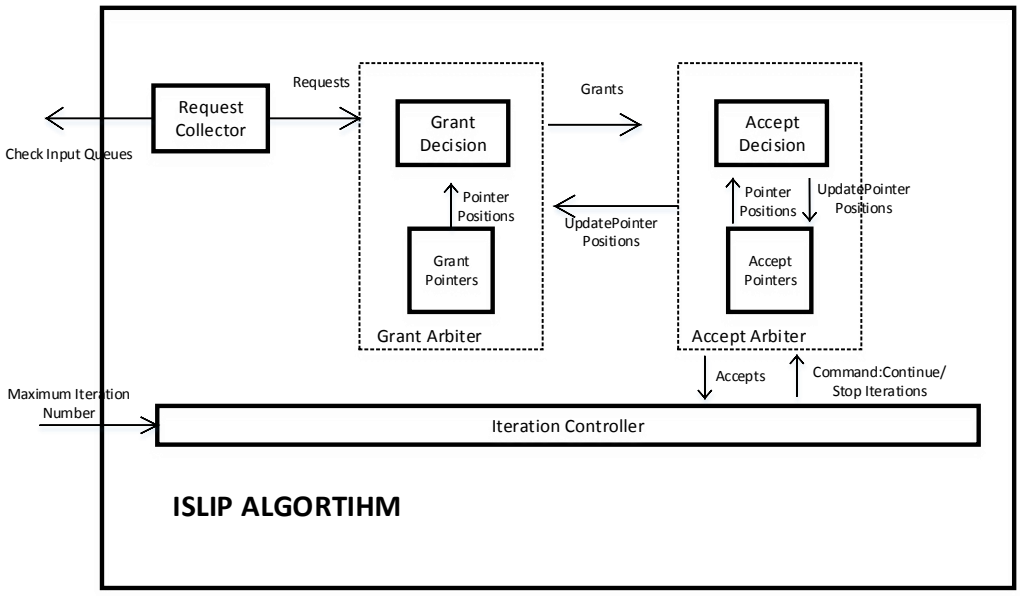

Figure 4.5: ISLIP Submodules

SwitchSim supports well known algorithm ISLIP (Iterative Round-Robin with SLIP).

Submodules of the ISLIP module and the interaction between them are given in Figure 4.5.The details of the ISLIP algorithm can be found at Section 3.5.

4.5.2 SP-ISLIP: Strictly Prioritized ISLIP

As it can be seen from figures 4.5 and 4.6 *Strictly Prioritized ISLIP* algorithm is an extension of ISLIP algorithm. Actually *Strictly Prioritized ISLIP* carries all the features of the ISLIP algorithm. In addition to that, it has *flow check* submodule. This module is used in order to give priority to the specific flows. For example video

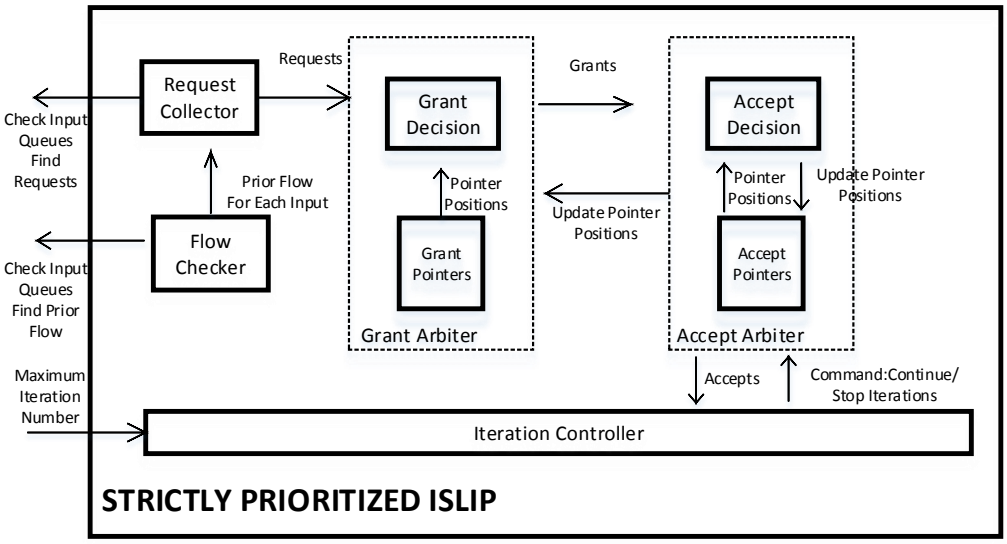

Figure 4.6: SP-ISLIP Submodules

packets can be considered as high priority flows. *Flow check* submodule and conventional *request collector* submodule comprise the Intelligent Multi-Class (IMC) VOQ Controller block which are explained at Chapter 5 in details.

Flow checking mechanism works with *request collector* submodule. Before requests are collected, the *flow check* submodule checks every input port in order to decide which priority level joins to the contention. According to the existence of a highest-priority-level-flow cell, *flow check* notifies the *request collector* submodule.

Request collector submodule collects requests from input ports that only have the highest priority cell at that time. Rest of the process is same as classical ISLIP. Details of this algorithm are presented in Chapter 5.

4.5.3 LP-ISLIP: Limited Prioritized ISLIP

Submodules of the LP-ISLIP and the interaction between them are given in Figure 4.7.

In addition to the *Strictly Prioritized ISLIP (SP-ISLIP)*; *Limited Prioritized ISLIP(LP-ISLIP)* has *window check* submodule. This algorithm is implemented in order to avoid starvation of the low-priority flows. At SP-ISLIP, if high priority flows have busy

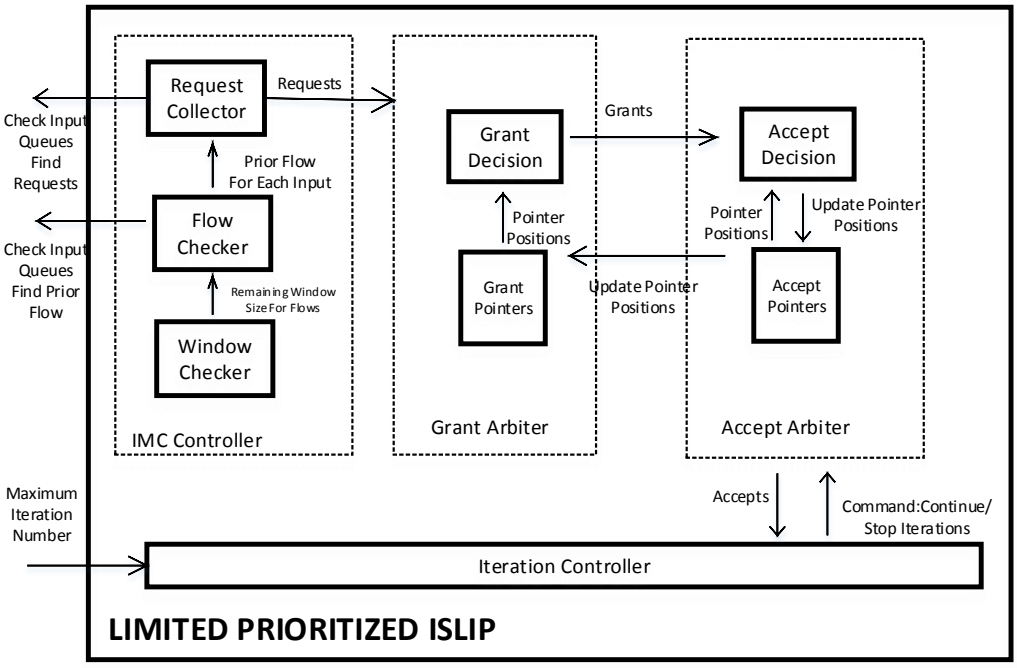

Figure 4.7: LP-ISLIP Submodules

traffic the low priority flows have no chance to be switched. That causes the overall delay performance to get worse as load is increasing.

Window check mechanism has a window size and a counter.*Window size* is variable and is set at the beginning of the simulation. This *window size* parameter is actually determines the level of limitation to the prioritization.

IMC VOQ block is also used in this algorithm. Different than the SP-ISLIP algorithm, this block includes *window check* submodule for LP-ISLIP. Details of this algorithm is given at Chapter 5.

4.6 Output Port Module

- Variable port number is supported.
- Switched cells queued in Output Ports.
- At the output port queues, cells classified according to their source port number.

Submodules of the *output port module* and the interaction between them are given in Figure 4.8.

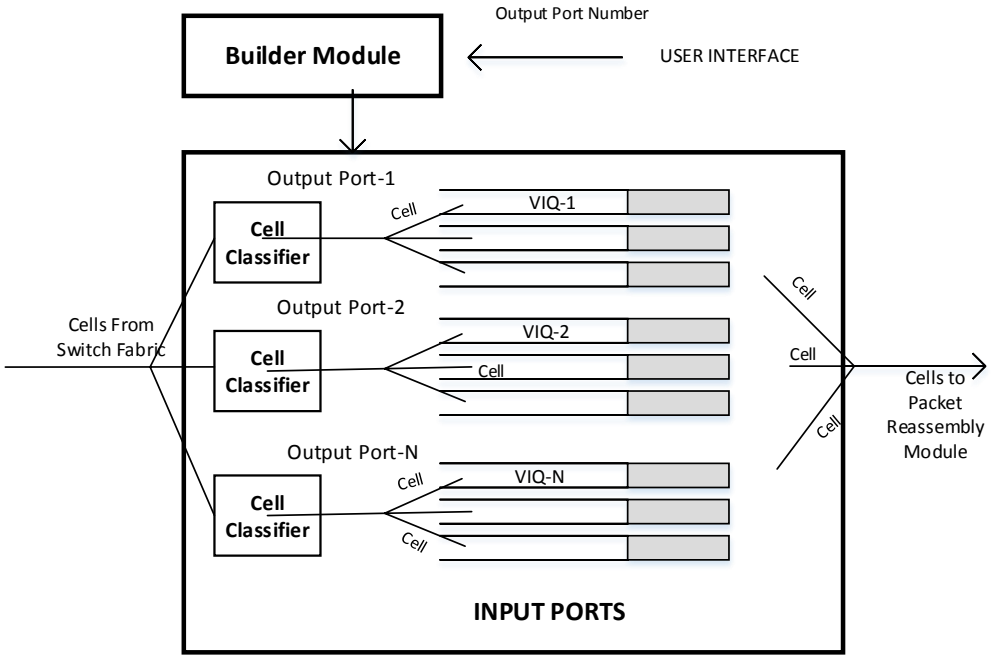

Figure 4.8: Output Ports Submodules

Output ports are very similar to the input ports. According to the output port number of the router which is set by user; builder module of the software builds output ports with their 2 tuple queues at the beginning of the simulation. A cell with source port i $(0 < i \leq N)$ and output port j $(0 < j \leq N)$ is placed to the queue $OQ_{(i,j)}$. The placement is done by cell classifier submodules.

After classification, cells are ready to be switched, reassembled by *packet reassembly module* waiting for all cells of a packet accumulates at the output queue.

4.7 Packet Reassembly Module

- Cells are needed to be reassembled to retrieve the original packet.
- Already classified cells are reassembled by reassembler block.

• A packet is ready to leave the switch after it is reassembled by its cells.

Submodules of the *packet reassembly module* and the interaction between them are given in Figure 4.9.

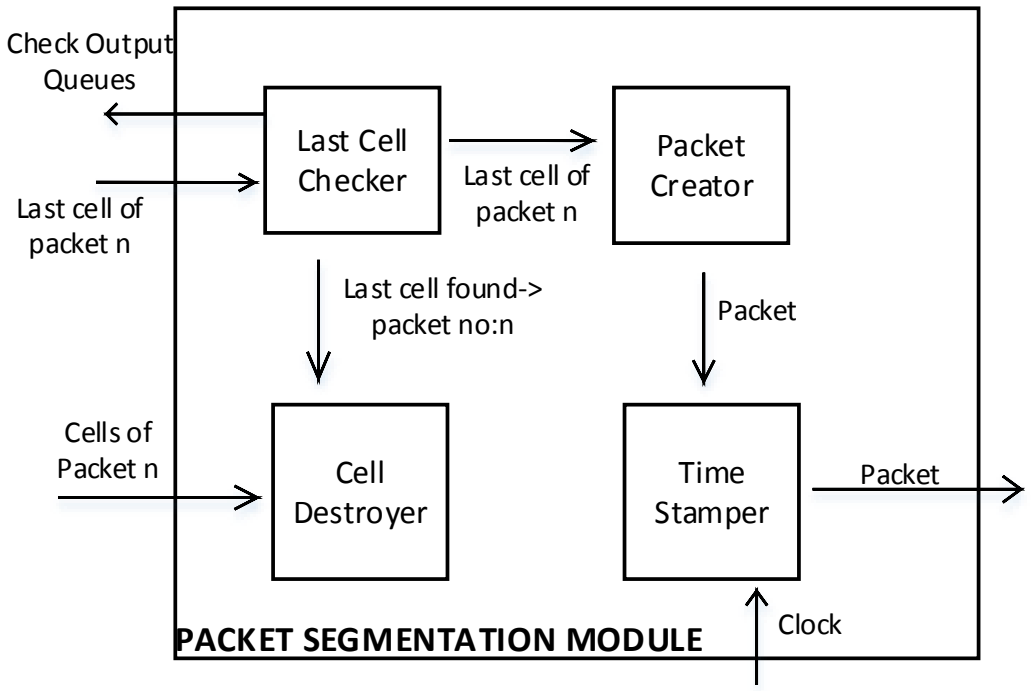

Figure 4.9: Packet Reassembly Submodules

At every cycle;*packet reassembly module* checks the output ports' queues if there are any cells ready to reassemble. Cells are ready to be reassembled if all the cells belonging to the same packet are already switched.

Last cell checker checks all the cells that are already in the output ports' queues. If it notices a cell that is last cell of a packet. It notifies the *cell destroyer* and *packet creator* submodules.

Cell destroyer submodule destroys all cells belonging to a packet whose all cells are switched.

Packet creator module recreates the packet according to the information gathered from the last cell. Since all the cells contain all properties of a packet, packet can be retrieved without losing any information.

4.8 Statistics Module

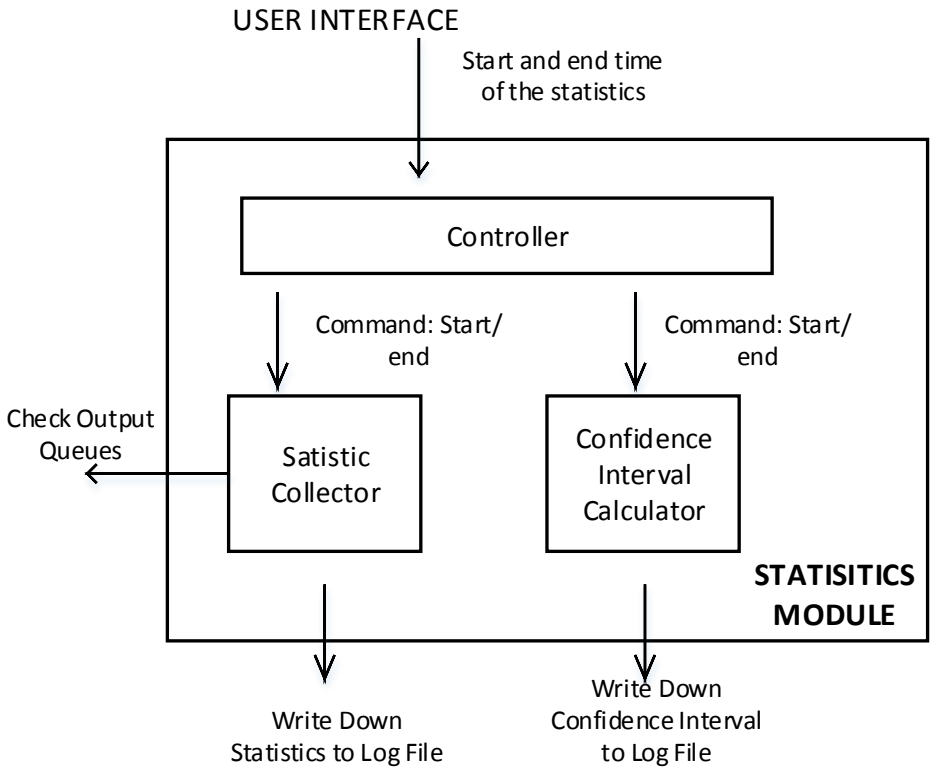

Figure 4.10: Statistics Submodules

User may not want to collect statistics at all time. For example, at the beginning of the simulation, collecting delay statistic is not a good decision, because average delay values need to be saturated after beginning. *Controller* submodule manages the start/end time of the collecting statistics according to user choice.

Statistic collector submodule collects the desired statistics and calculates the average of them. At every cycle, it checks the output queues in order to collect queuing delay (fabric delay), and read all switched cells' information (birth time, switching time).Moreover, in order to collect end-to-end delays *statistic collector* submodule reads the reassembled packets.

Since statistics are given in average values, the confidence interval of these averages should be calculated at the end of the simulation. After the simulation ends, *confidence interval calculator* submodule calculates the confidence interval of the average values with confidence level of 95% or 99%.

Submodules of the *statistics module* and the interaction between them are given in Figure 4.10.

CHAPTER 5

FLOW PRIORITIZATION AND PROPOSED ALGORITHMS

In Chapter 3, the most widely used scheduler algorithm for IQ switches, ISLIP, is summarized. It has been emphasized that at very high speeds (especially at core networks), using IQ switches have several advantages including switch size. Implementation complexity of an IQ switch fabric scheduler algorithm has vital importance. That is why; the ISLIP is one of the widely used algorithm for commercial purposes. However, using an IQ switch instead of an OQ switch causes losing the QoS ability of the switch. Since ISLIP is an IQ switch scheduler algorithm, this problem is also valid for ISLIP as well.

In this thesis, an extension is made to the ISLIP algorithm and two different algorithms are proposed. With these algorithms, giving QoS capability to the ISLIP algorithm to differentiate among different traffic classes with different priorities, is suggested. While doing this, the complexity of the algorithm at the hardware is also considered[12]. ISLIP is used as base IQ switch fabric algorithm for extension, because it is the most preferred and widely used algorithm due to its efficiency, fairness, stability and complexity features.

As it is explained in Chapter 3, OQ switches are not suitable at very high speeds and large port numbers. When IQ switches are used, packets of a lost-intolerant traffic flow might be dropped due to buffer size limitation and temporary bursty traffic load. A cell of a packet of a delay intolerant flow might lose contention at the arbitration phase of the scheduling at the input ports.

Giving priority to some flows over the other ones at the arbitration phase might give

useful results for the problem that's defined above. Giving priority to some specific flows should not starve other flows that have lenient QoS requirements.

5.1 Proposed Algorithms

In this Section, two different algorithms that are implemented on SwitchSim and the results are compared. First algorithm simply gives the priority for the specific flows. Second algorithm limits this prioritization by keeping a predefined window size. Except giving priority to the some specific flows both of these switch fabric algorithms work like ISLIP. Another important feature of proposed new/extended switch fabric algorithms, is to keep the implementation complexity on the hardware as easy as possible. These algorithms are implemented on the FPGA and results are cross-checked with simulation results for some cases.[12]

5.1.1 SP-ISLIP: Strictly Prioritized ISLIP

Strictly Prioritized ISLIP is implemented for just seeing the step that giving blindly priority to some specific flows makes the other flows to starve. This starvation not only causes the loss of the packets of the less important flows' but also affects the overall delay/ throughput performance of the switch fabric.

Experiments are conducted to compare algorithms. The results of the delay performance of the switch for different priority levels will be shown under different traffic conditions at Chapter 6.

In order to implement this algorithm; a new VOQ structure, for input ports, is proposed. ISLIP algorithm uses virtual output queuing which keeps different FIFO queues for different input-output pairs. In other words, for virtual output queuing there should be N^2 FIFO queue available. Giving special priority to specific traffic flows requires more differentiated queues. There should be a distinct FIFO queue for any input-output-flow triplet. This is a must to avoid HOL blocking.As a result; for a switch with size N and P different priority level(can be considered as different flows) there will be $N^2 \times P$ distinct FIFO queue. In addition to that a new algorithm block

is added to the system to manage the requests. This block is called “Intelligent Multi-Class VOQ Controller. Figure 5.1 shows an example of this structure with switch size N for two different flows(priority levels).

IMC block manages the requests of the input ports. Due to the nature of the IQ switches, an input port i cannot request more than one times for any specific output j during one cycle of the operation. However since there are P flows, there may be P requests from input port i to output port j at maximum. IMC manages these requests and decide which request will be sent from input port i and output port j. IMC basically determines the valid request according to the cells' flow priority. For one input port only one priority level request can be sent. This means that a cell priority p can send request only if there is no request with higher priority level from that input port to any output port. In other words, if a single request from input port i with priority level p is made, other request from i with priority level lower than p are discarded by IMC.

The IMC block is independent of the rest of the ISLIP algorithm. Therefore this VOQ structure with IMC can be used any other IQ switch fabric algorithms.

5.1.2 LP ISLIP:Limited Prioritized ISLIP

After implementing the first algorithm and observing the results of giving blind priority to some specific flows; the second step is limiting the level of the priority. As it can be seen from the experiment results discussed at Chapter 6; giving full priority to some specific flow(s) causes the lower priority flow to starve. Although some flows are less delay tolerant, packet loss of these flows,of course, is undesirable. Moreover giving this much priority to some specific flows causes overall delay and throughput performances of the switch decrease.

Therefore at LP-ISLIP algorithm, which is again based on well-known ISLIP algorithm, the priority level of high importance packets is limited by a number called window limit. VOQ structure of the input ports are the same as the “Strictly Prioritized ISLIP”.For switch size N and maximum level of priority P. There are $N \times P$ distinct VOQs at this algorithm too.However, in this algorithm, the IMC block is used in order

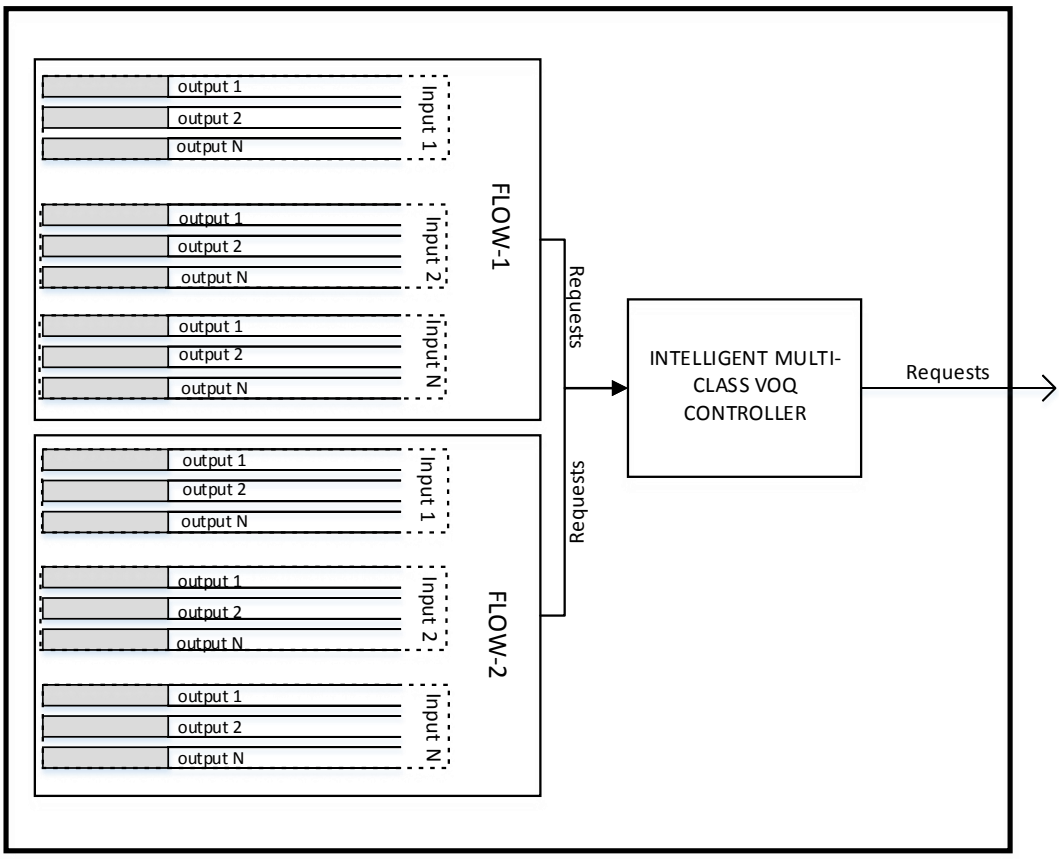

Figure 5.1: New Multi-Class VOQ structure with IMC

to limit successive requests of the high priority flows for each input separately.

Delay performance comparisons are made with basic ISLIP and with SP-ISLIP algorithm under different conditions. Results of these comparisons are discussed at Chapter 6.

Algorithm works as described below for two level priority levels. These priority levels are named as low priority and high priority. A new counter is defined to count given in-a-row priorities to the high priority level packets over low priority packets.

• Request from input ports are checked to determine the maximum priority level.

• If there is any high priority packet, only the high priority level packets have right to join the contention unless the window counter of the high priority level is at window limit. Window counter is incremented by one at the end of the process.(If there is no low priority request at the time, window counter is not incremented.)

• If the window counter is at limit, only low priority level packets joins the contention and window counter is set to zero.

• If only high priority level packets are present at the input ports queues, the window counter is not incremented.

Note that, this algorithm is applied to all input ports separately.

CHAPTER 6

SIMULATION RESULTS AND PERFORMANCE ANALYSIS

In this Chapter, a number of various experiments are conducted to compare different algorithms, namely, ISLIP, PIM, SP-ISLIP and LP-ISLIP . Since this thesis work is mainly based on ISLIP algorithm, some of the experiments which are conducted by the author of [13] are repeated and same results are gathered. In addition to those experiments the algorithm comparisons for delay performance are made for the new proposed algorithms.

Simulation durations of all experiments are 100000 cycles. 100000-cycles duration is enough for each calculated average to remain within $\pm$ 5% interval with a confidence level of 95% except for the cases in which corresponding algorithm is not stable under load values that are very high. These situations are remarked within the comments of the related experiment. Since the average values do not saturate, calculating confidence interval or even averages is not meaningful. However, for the sake of drawing the graphs which gives insight for commenting the experiments; unsaturated averages are given for those cases.

Before giving the results of the experiment, it should be noted that the repeated experiments from [13] is a strong proof the reliability of SwitchSim. Moreover, the functionality of the ISLIP, SP-ISLIP and LP-ISLIP algorithms; are cross checked with hardware implementation of these algorithms in[12]. Tests are made with 4×4 ISLIP, 4×4 2-Level SP-ISLIP and 4×4 2-Level LP-ISLIP. In these tests, all the created packets for both software and hardware are the same and the switched packets are compared one by one using MATLAB.

6.1 Experiment 1: ISLIP Algorithm – Effect of Maximum Iteration Number on Delay Performance

In this experiment, standard ISLIP algorithm's delay performance is analyzed under different traffic loads. Results are given in Figure 6.1. Horizontal axis of the Figure 6.1 is offered load. Vertical axis of the Figure 6.1 is average cell delay. Switch size is 16x16. Statistics are gathered with a new accurate SwitchSim which is explained in details at Chapter 4.

The variable parameter other than the offered load is the maximum iteration number. As it is explained at Chapter 3, iteration is a compilation of request, grant, and accept phases. Due to output contention maximal match may not be reached in single iteration. In ISLIP algorithm, multiple iterations may be conducted in order to reach maximal match. However continuously doing iterations, until the maximal match is reached, may not be a wise decision. Although SwitchSim ignores the time elapsed during iterations (matching time); in hardware, running matching algorithm many times, can cause delays that can be comparable to the fabric delay. Therefore in the algorithm, maximum iteration number must be limited. Maximum iteration number does not imply the exact iteration numbers that occurs in every cycle. It is just a limitation. For example; algorithm can reach maximal match with single iteration while maximum iteration number is 4.

In this experiment; three different cases are examined. Three different lines in Figure 6.1 represent the ISLIP algorithm with maximum iteration number 1, 2, and 4. This experiment is also conducted by Nick McKeown in [13]. Corresponding experiment results from [13] is also represented with dashed, black lines in Figure 6.1. Comparing those results with the experiment conducted within this thesis work shows the reliability of SwitchSim that is developed in the scope of this thesis work.

As it is clear from Figure 6.1 that under the light traffic (0 - 0,2): all three lines follow the same path. This is because under the light traffic; mostly one single iteration is enough to reach maximal match. In order to justify this assumption an extra statistic is collected called average iteration per cycle. In Table 6.1, results confirm the assumption that even if the maximum iteration number more than one; the required

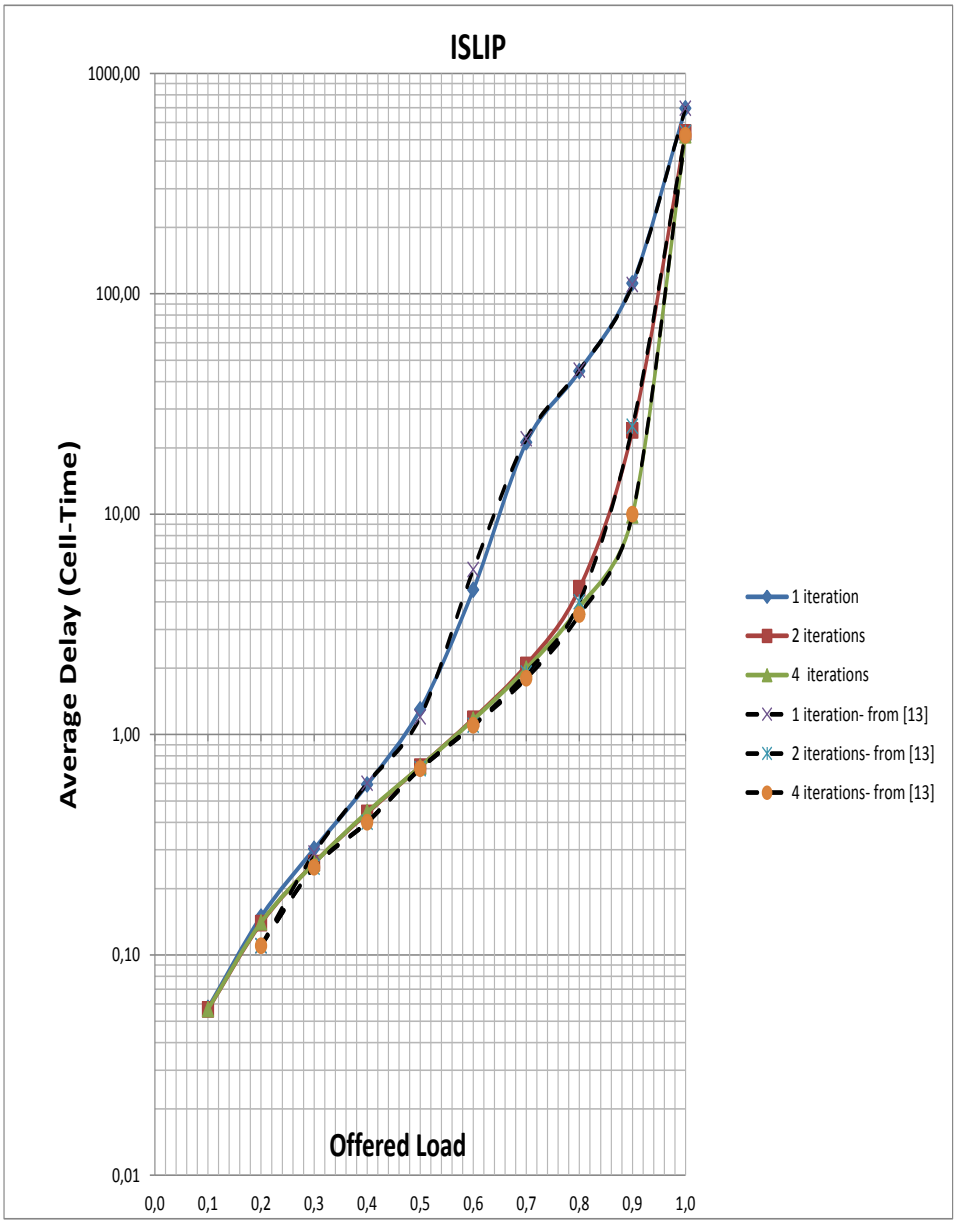

Figure 6.1: ISLIP Algorithm – Effect of Maximum Iteration Number on Delay Performance

average iteration to reach maximal match is very close to 1.

Table 6.1: Average Realized Iteration Numbers under Light Load

Load 0,1		Load 0,2	
Maximum Iteration Number	Average Realized Iteration Number	Maximum Iteration Number	Average Realized Iteration Number
1	1.0000	1	1.0000
2	1.0005	2	1.0070
4	1.0005	4	1.0077

As it can be seen from Figure 6.1 that under medium traffic load (0.3 -0.7);two lines for iteration number 2 and 4 follow the same path but the line for iteration number 1 follows different path. This is because under medium traffic, the average required iteration to reach maximal match is more than 1 and is less or very close to 2. To verify this comment, the average iteration numbers for the medium load are given at Table 6.2.

Table 6.2: Average Realized Iteration Numbers under Medium Load

Load 0,3		Load 0,5	
Maximum Iteration Number	Average Realized Iteration Number	Maximum Iteration Number	Average Realized Iteration Number
1	1.0000	1	1.0000
2	1.044	2	1.3266
4	1.045	4	1.3251

Under heavy traffic, average required iteration number increases. Because the higher offered load means more ports are joining the output contention. As the number of ports that joins contention increases, finding matches becomes harder. At Figure 6.1, it can be clearly seen that three different lines follow three different paths. Because of the limitation of the iteration is lower than required iteration, at first two cases (when iteration number 1 and 2) more average cell delay occurs when compared to the third one. In order to verify this comment the average iteration numbers for each case under heavy traffic is given at Table 6.3. At first two cases (when iteration number 1 and 2), realized iteration numbers are at the limit. This means that the limitation of the iterations causes extra delay when compared to the third case(when iteration number is 4).

Table 6.3: Average Realized Iteration Numbers under Heavy Load

Load 0,8		Load 0,9	
Maximum Iteration Number	Average Realized Iteration Number	Maximum Iteration Number	Average Realized Iteration Number
1	1.0000	1	1.0000
2	1.9688	2	1.9946
4	2.4808	4	2.48381

Looking from another perspective: if this algorithm is stable under offered load “L” and with switch size "N" (for iteration 1, 2, 4) the average matching number per cycle should be equal to (offered load) x (switch size). NxL corresponds to the average number of incoming cells to the switch fabric. If average matching number per cycle would be less than NxL, input queues would have constantly increased and the algorithm would have been unstable.

Delays are basically proportional to the average number of cells that are present at the input queues. Beginning from the start, the average cell number at the input queues increases until the system reaches NxL matching per cycle. After system reaches its stable region; the average number of requests per cycle remains constant. In other words; for different iteration numbers there should be a different average request number per cycle. Moreover, the more iterations the less average request number per cycle. For example for offered load 0.8, delay performances are different for each iteration number. In order to verify the comment that is made above the statistics are given at Table 6.4.

Table 6.4: Average Requests per Cycle

Maximum Iteration Number	Average Request per Cycle
1	183.6
2	55.16
4	54.7

6.2 Experiment 2: ISLIP Algorithm – Effect of Switch Size on Delay Performance

In this experiment, ISLIP algorithm is examined with different switch sizes. Switch size means number of input and output ports. Horizontal axis of the Figure 6.2 is offered load and vertical axis is the average cell delay. Time unit of the cell delay is one cell-time. Iteration number is limited to 1 for all cases.

In this experiment, only switch size is variable other than the offered load. In Figure 6.2 there are four distinct lines. They are representing the same experiments with 4x4, 8x8, 16x16, 32x32 switch sizes. As switch size increases, average number of input ports contending for an output port and average number of output ports contending for an input port increases.

This experiment is also conducted by Nick McKeown in[13].Corresponding experiment results from [13] is also represented with dashed, black lines in Figure 6.2. Comparing those results with the experiment conducted within this thesis work shows the reliability of SwitchSim that is developed in the scope of this thesis work.

The effect of the switch size differs under heavy load and light load. Under low load the probability of one cell is transmitted without delay is proportional to the probability that no other cell is waiting to be transmitted for the same output.

Under the light load we can ignore the fabric delay. In other words, we can formulate the number of contending cell for each output considering there is no accumulation in the queues.

If we ignore the small fabric delay under light load, the expected number of the contending cells for each output is approximately: $\lambda(1 - ((N-1)/N)^{N-1})$

Where λ is offered load and N is number of ports. When N goes to infinity the number of contending cells for each output port converges to a constant. Therefore the fabric delay converges to a constant with increasing N too.

$\lim_{N\to\infty} \lambda(1 - ((N-1)/N)^{N-1}) = \lambda(1 - (1/e))$ which is equal to 0.63λ. This convergence occurs quite fast. For example:

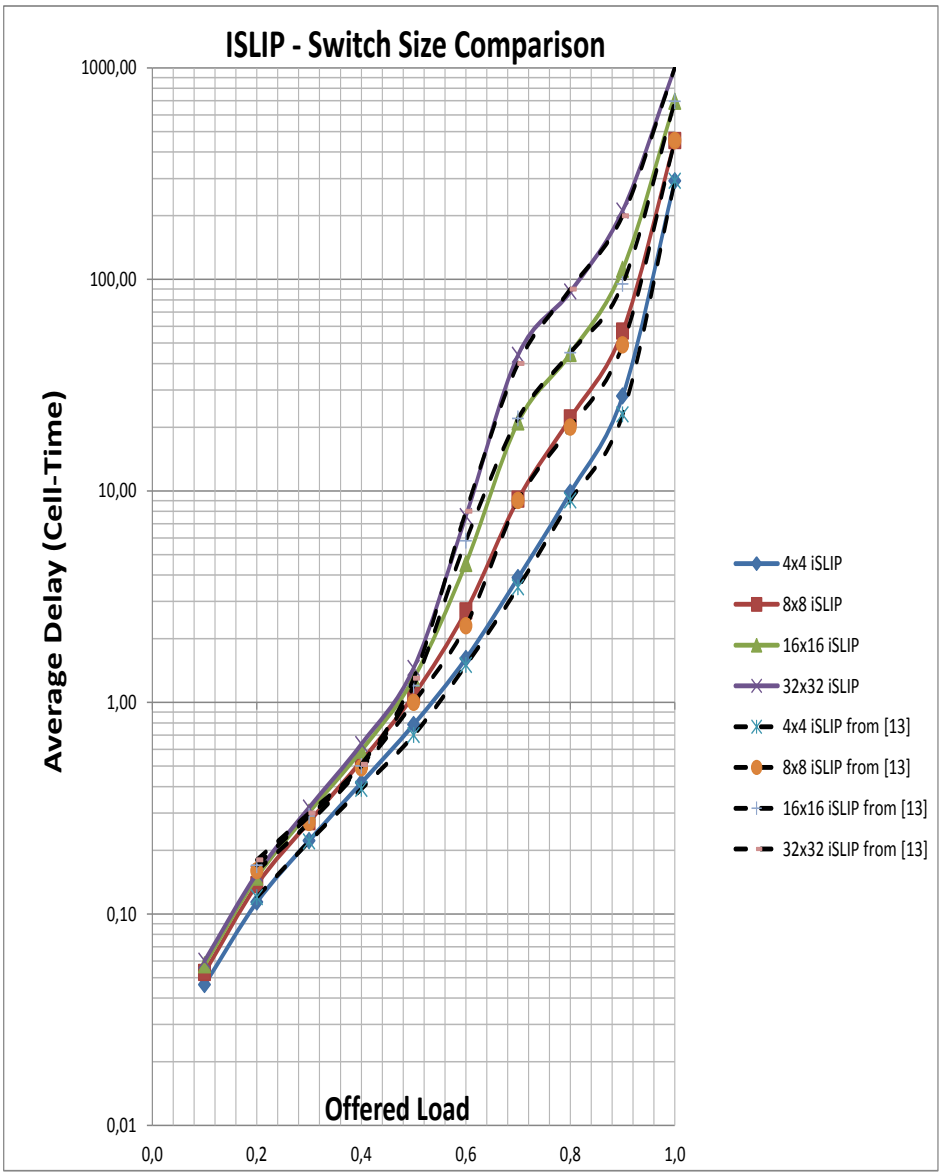

Figure 6.2: ISLIP Algorithm – Effect of Switch Size on Delay Performance

For N=4 the expected number of contending cells for each output is 0.57λ

For N=8 the expected number of contending cells for each output is 0.607λ

For N=16 the expected number of contending cells for each output is 0.62λ

N=32 expected number of contending cells for each output is 0.626λ)

N=64 expected number of contending cells for each output is 0.629λ

N=∞ expected number of contending cells for each output is 0.63λ

Figure 6.2 shows that under low load, all four lines follow approximately the same path. There is a little difference between consequent lines. But this difference diminishes when N increases.

However under the heavy load we cannot ignore the fabric delay. This means that beside the new coming cells, the accumulated cells should be considered while calculated number of contending cells for each output. Therefore the above formula does not apply under the heavy load. From the different perspective; assuming that all input queues have at least one cell for any output queue. It takes N cycles for a FIFO queue to be served. Therefore under heavy load, the fabric delay is proportional to switch size N. At Figure 6.2, after a certain point; lines start to follow different paths.

6.3 Experiment 3: ISLIP Performance Analysis under Bursty Traffic

In first two experiments, the created traffic was Poisson traffic. In order to see the ISLIP's delay performance under the bursty traffic 4 different cases are studied with average burst lengths of 16 and 64 and maximum iteration numbers of 2 and 4. In [13] the author states that bursty traffic analysis shows the performance of the ISLIP algorithm for variable-size incoming packets. Definition of the generated bursty traffic is given below:

-Traffic generator alternates between *active* and *silent* periods

- Generates packets back to back during the *active* periods and stays idle during *silent* periods.

- Durations of active and silent periods are geometrically distributed.

Given offered load and mean burst length probability of leaving active period "p" and probability of leaving silent period can be calculated as follows.

$p = 1/BurstLength$ and

$q = OfferedLoad/(BurstLength \times (1 - OfferedLoad))$

Results are given in Figure 6.3. Vertical axis of the Figure 6.3 is average cell delay. Switch size is 16x16. There are four lines in the graph. Each of them represents one of the combinations of burst lengths of 16 and 64 and maximum iteration numbers of 2 and 4. This experiment is also conducted by in [13]. Corresponding experiment results from [13] is also represented with dashed, black lines in Figure 6.3. Comparing the results with the experiment conducted within this thesis work shows a little difference. This difference might be due to a possible different implementation of the bursty traffic model.

If every parameter is kept constant except for the burst sizes, it is seen that average cell delay is proportional to the burst size. Increased burst size causes increasing delays because of two reasons. Firstly, as already mentioned, the implementation of the bursty traffic, states that all of the cells within a burst must be put to input queues at the same time. This brings some delay directly. Even if there is no other cell accumulation in the queues (under light load), at least burst-size amount of time must be elapsed in order to last cell of the burst to be transmitted. In other words, due to the nature of bursty traffic; there are two states named active/and passive. In active state cells are coming; in passive states the traffic generator remains idle. In passive period there are no incoming cells. This means that in passive period the resource of the fabric remains idle. This situation causes inefficiency. The bigger is the burst size the longer the idle period. The inefficient use of the fabric resources causes an increase in average delay. Secondly, in bursty traffic model, all cells within a burst should be destined to the same output. This feature of the bursty traffic model also causes delay.

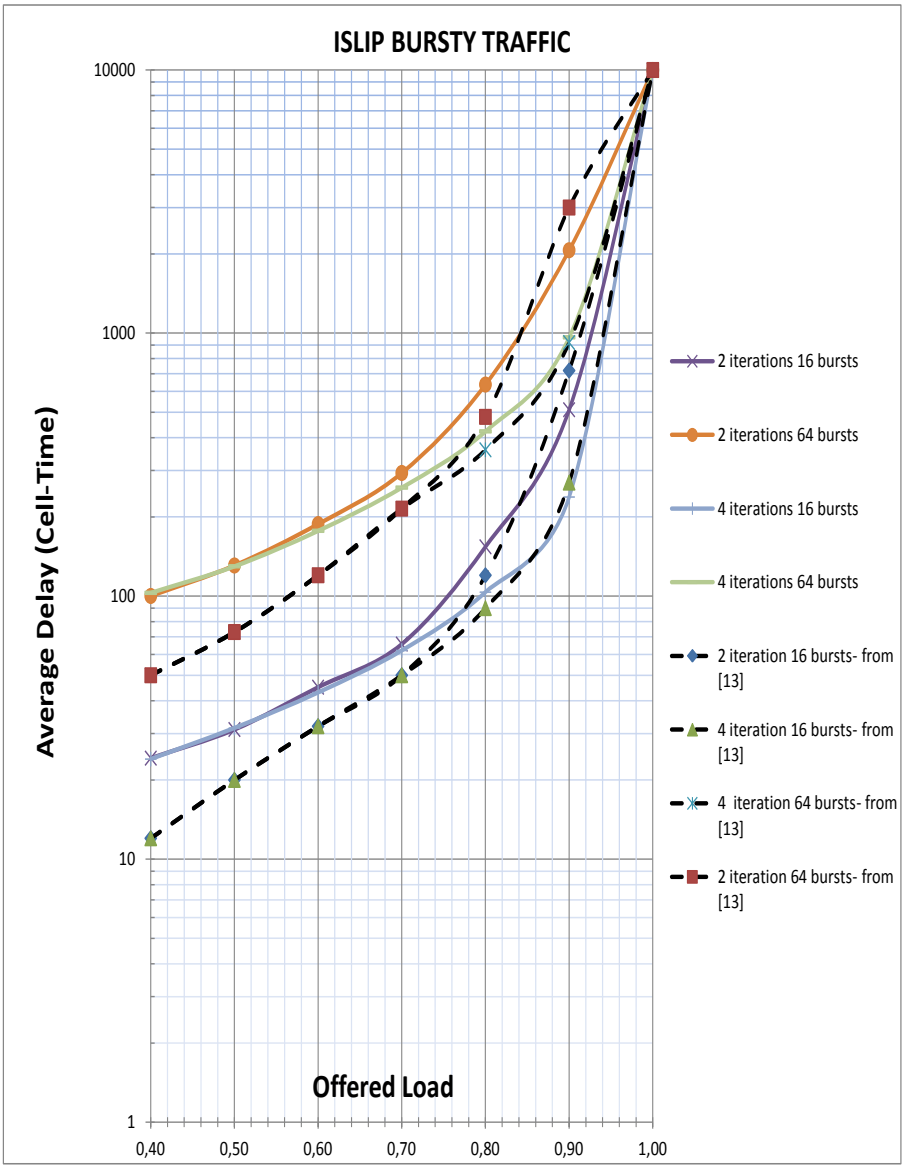

Figure 6.3: ISLIP Performance Analysis under Bursty Traffic

6.4 Experiment 4: Parallel Iterative Matching vs ISLIP

In this thesis work, in addition to the ISLIP algorithm three other algorithms are implemented. One of them is PIM[6] (Parallel iterative Matching). ISLIP algorithm is actually based on PIM algorithm. This algorithm is implemented in order to get better understanding of ISLIP algorithm. Comparing these two algorithms' can give us a deeper understanding about ISLIP

In this experiment, delay performance of PIM and ISLIP are compared. Results are given in Figure 6.4. Horizontal axis of the Figure 6.4 is offered load. Vertical axis of the Figure 6.4 is average cell delay. Switch size is 16x16. Maximum iteration number is 1.

Instead of the round robin arbiters in ISLIP, PIM employs a random selection at both input and output. Comparing the average delays between two algorithms shows how fairness between ports has great effect on the overall performance of the algorithm. At Figure 6.4 under light loads, there are no such big differences between two algorithms. Because of lacking fairness, under medium or heavy loads, starting from load is 0.6, PIM algorithm becomes unstable.

6.5 Experiment 5: Variable Sized Packets

In the previous experiments, fixed sized packets with size equals to "1 cell" are used and average cell delays due to fabric scheduler algorithm (queuing delay) are given. Those experiments gave insight about the efficiency of the ISLIP algorithm. However, in this experiment packet sizes are variable with an average value. This time measured metric is end-to-end delay. Since packets are multiple-sized, switched cells of a packet must wait at the output queues until the last cell of the packet is switched. This waiting time contributes to the end-to-end delay while it does not count in the calculation of the queuing delay. Moreover, when packets come in multiple sizes, all the cells belonging to a given packet are placed in the respective queue simultaneously. This causes an increase in end-to-end delay. In addition to these two factors, while computing end-to-end delays, time elapsed during transmission of the packets

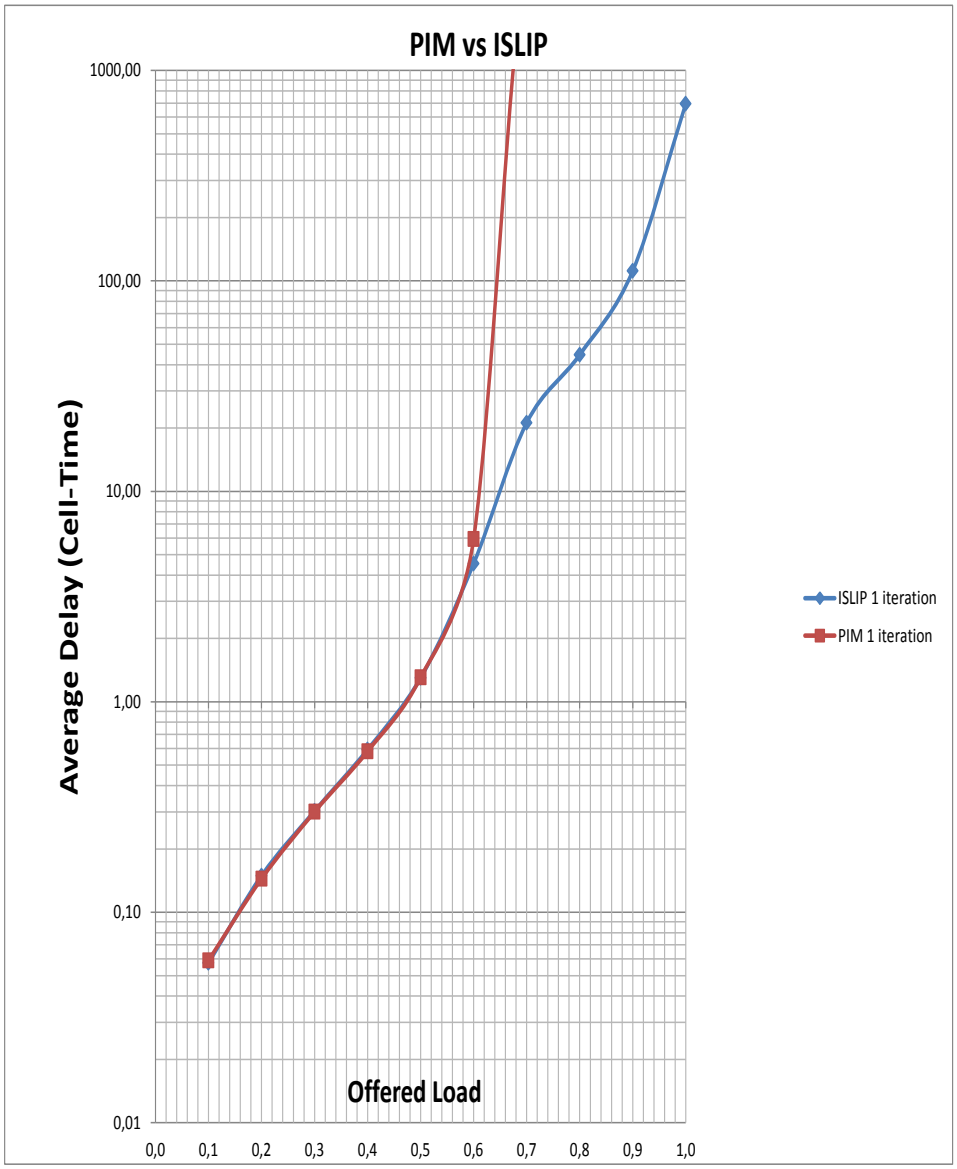

Figure 6.4: Parallel Iterative Matching vs ISLIP

out of the switch are counted. Note that segmentation and reassembling times are ignored.

In this experiment, end-to-end delays are calculated with three different average packet sizes: 1,2 and 4. Given average packet size "n", generated packets are sized uniformly distributed within the interval of [1, 2n-1]. Maximum iteration number is set to be 2. Switch size is 16×16. Results are given in Figure 6.5.

6.6 Experiment 6: Prioritizing ISLIP Algorithm

In this thesis work two different prioritized ISLIP algorithms are implemented which are called “Strictly Prioritized ISLIP” and “Limited Prioritized ISLIP”. These algorithms are explained in Chapter 4 and Chapter 5 in details.

In [13], the author also defines a “Prioritized algorithm”. This algorithm and the algorithms that are implemented in this thesis work, have differences.

In the “Prioritized ISLIP” in[13] cells that are belongs to prioritized flow has priority while requesting, granting and accepting.

Request Priority: In ISLIP algorithm, every input port that has at least one cell destined to a specific output port requests for that output port. In this algorithm, an input port sends request to an output port only if the present cell that is destined to that output port has the biggest priority. For example; in 16x16 switch, an input port has 10 cells that are destined to different output ports; but 5 of them are belong to low priority flow and 5 of them are belong to high priority flow. Only 5 requests occur

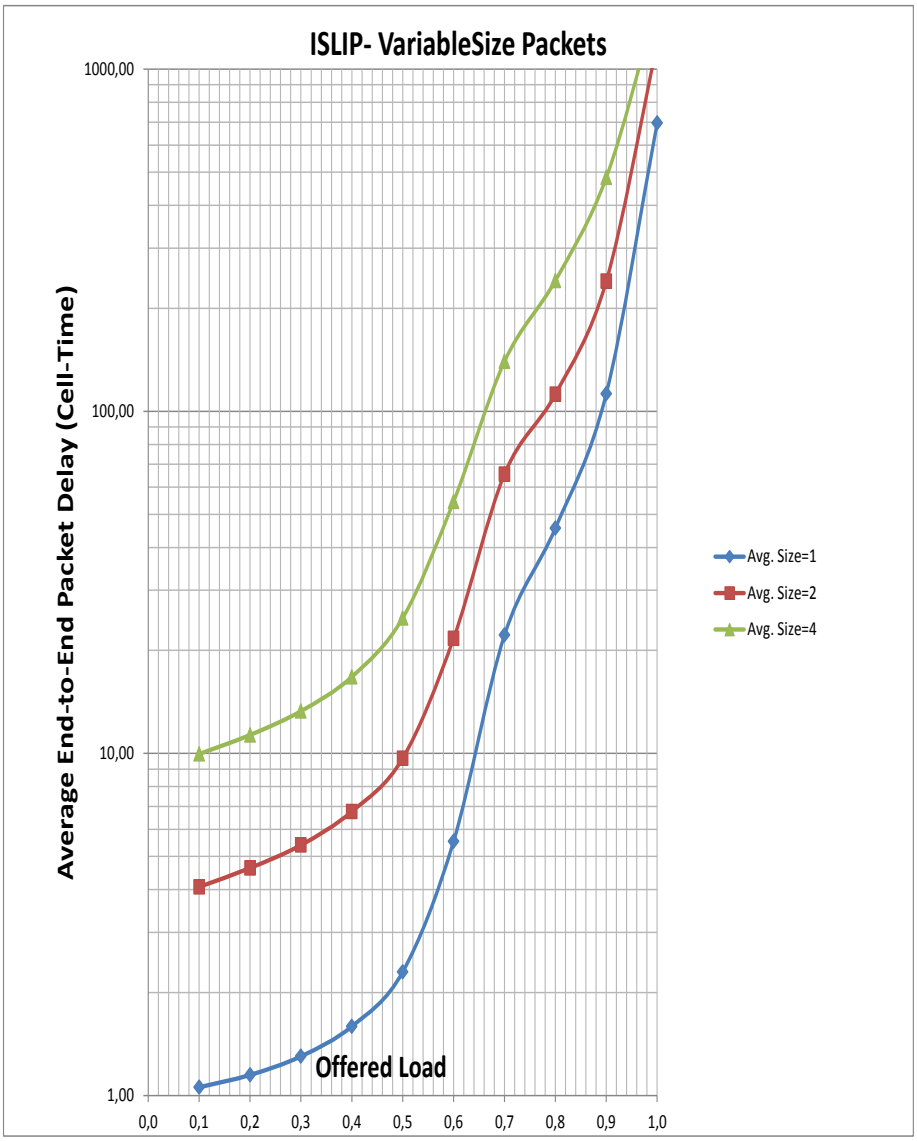

Figure 6.5: ISLIP-Variable Sized Packets, End-to-End Delays

corresponding to “high priority flow cells”. Requests are made corresponding to “low priority flow cells” only if there is no “high priority low cells” which are present in that input port.

Grant Priority: In ISLIP algorithm each output port makes one single grant even if it receives multiple requests from different input ports. The choice is made according to grant pointer which works in round-robin distribution scheme. However, in this algorithm, the choice is made according to flow priority. This feature makes ISLIP less fair between ports.

Accept Priority: In ISLIP algorithm each input port makes one single accept even if it receives multiple grants from different output ports. The choice is made according to accept pointer which works in round-robin distribution scheme. However, in this algorithm, the choice is made according to flow priority. This feature makes ISLIP less fair between ports too.

“Strictly Prioritized ISLIP” algorithm only deploys request priority. This algorithm is implemented because the “Prioritized ISLIP” that is defined in [13] makes ISLIP algorithm less fair between ports. In experiment 4; the effects of fairness between ports, on delay performance are shown via comparing ISLIP with a less fair algorithm, PIM. Moreover the request selector which is the algorithm block that is implemented for this algorithm can be used with other fabric scheduler algorithms too. Grant and accept mechanism of the ISLIP algorithm is the same as proposed.

“Limited Prioritized ISLIP” is implemented because the observations from the “Strictly Prioritized ISLIP” show that giving priority to a specific flow might cause the other flows to starve. The level of limitation can be adjusted considering the required QoS metrics and traffic characteristics.

In this experiment, two-level prioritization is simulated. Cells are labeled with different two flow numbers and the delay performance for each flows are observed. Flows are named as video and data. Cells that are belonging to video flow are called video cells. Cells that are belonging to data flow are called data cells.

The ratio of the cells belonging to different flows may vary. Therefore two different cases are simulated with different ratios. These are 30% video-70% data and 70%

video-30% data. Experiments 6.1 and 6.2 show the average delays of the video and data cells separately. Experiment 6.3 and 6.4 show the average delays of the all cells and compare the results with non-prioritized (standard) ISLIP algorithm.

6.6.1 Experiment 6.1 30% video-70% data - Flow Delays

In this experiment the ratio of video cells is 30%; 16x16 switch configuration with 4 iterations, is used. Figure 6.6 shows the average cell delays of data cells and video cells which are calculated separately. Overall delays will be shown at experiment 6.3. In this experiment the following algorithms are compared: "Strictly Prioritized ISLIP" and "Limited Prioritized ISLIP". Moreover, "Limited Prioritized ISLIP" is tested twice with "priority window" size 2 and 4. The "priority window" is the number of giving priority to the high priority flows in a row.

The difference between average data cell delays and average video cell delays for each algorithm can be seen from Figure 6.6. Under low load all three different algorithms follow the same path for video cells and for data cells. With increasing load, lines start to differ. The most important point at this graph is that after some point, the switch becomes unstable for data cells. The reason of this situation is request selector. It selects the video cells during the request collecting phase. Under low loads, the request selector has chance to select data cells. Because at low loads; ignoring very low video cell delays, there is no video cell accumulation at the input port. As offered load increases, video cell delay increases too. Delay means accumulation of the video cells at the input ports. As this accumulation increases, the chance for data cells to request decreases proportionally. After a certain point, although some data cells are transmitted to the output ports, the incoming data cells are always more than the transmitted ones. Therefore after that point data cells starts to starve and switch becomes useless for data cells.

Another interesting point is that when data cells are starving, video delays become constant with increasing load (when load is bigger than 0.7). Because, due to the request selector mechanism, after that point data cells requesting less and less even if the amount of incoming data cells is increasing. Fewer requests number means fewer matches for data cells. On the hand, with increasing load incoming video cell

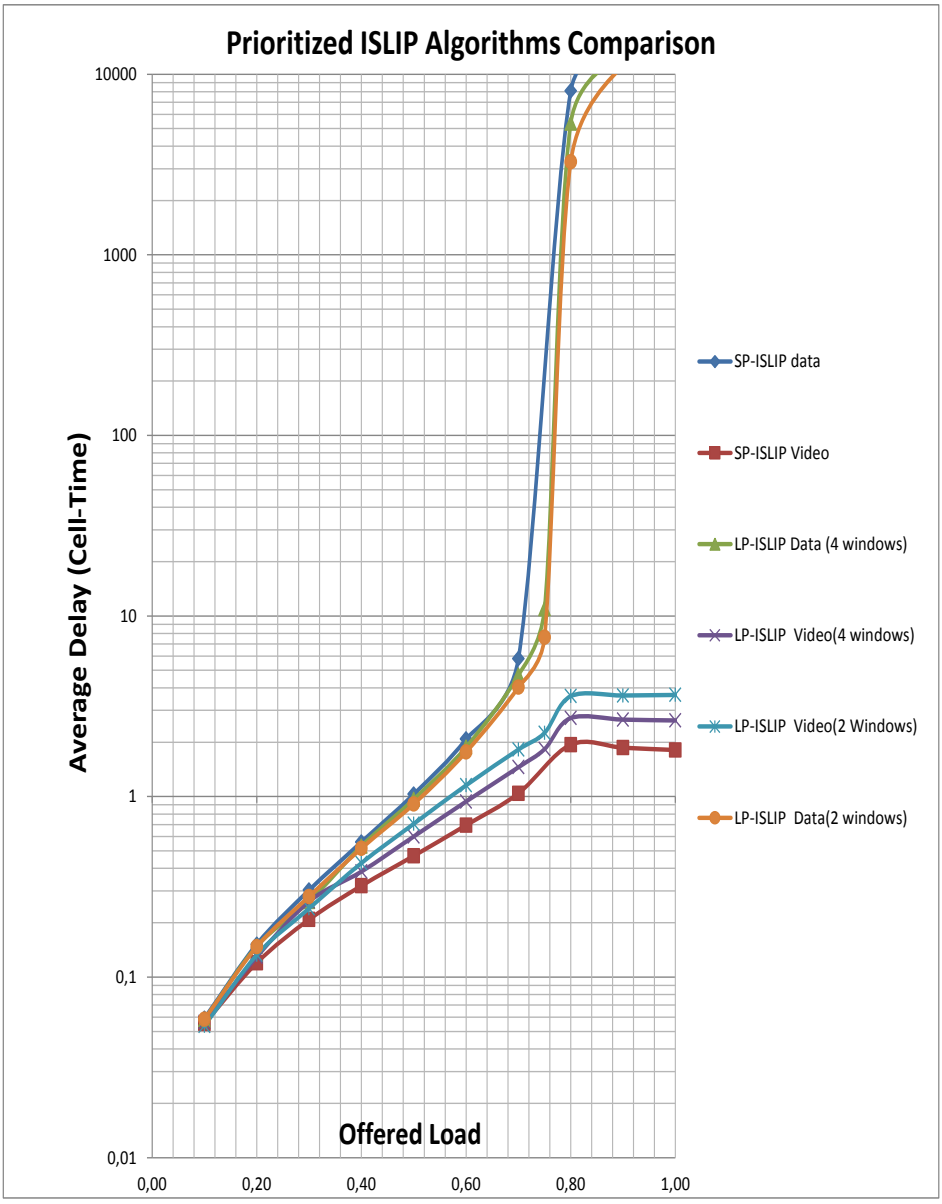

Figure 6.6: 30% video-70% data - Flow Delays

number increases too. Since video cells have priority; they request more. In order to have deeper understanding of this situation request numbers that are made for video and data delays are observed under different offered load values. Results are given at Table 6.5 and Table 6.6

Table 6.5: Average Requests For “Strictly Prioritized ISLIP”

Load	Average Requets For Data Cells	Average Requests For Video Cells
0.7	27.71	6.74
0.8	119.75	10.63
0.9	110.33	11.92
1.0	101.80	13.24

Table 6.6: Average Requests For “Limited Prioritized ISLIP” with Window Size=4

Load	Average Requets For Data Cells	Average Requests For Video Cells
0.7	25.63	6.68
0.8	129.28	11.02
0.9	121.47	12.26
1.0	113.53	13.72

Table 6.5 and Table 6.6 show the average request numbers for data and video cells separately. As mentioned above, for both algorithms, even when offered load is increasing the requests that are made for data cell is decreasing.

Moreover, requests numbers of video cells verify the average delays in the graph. In order to compare the numbers under different loads, they should be normalized according to the load.

For “Strictly Prioritized ISLIP”

Load 0.7 -> 6.74/0.7= 9.63

Load 0.8 -> 10.63/0.8=13.29

Load 0.9-> 11.92/0.9=13.24

Load 1-> 3.24/1=13.24

For “Strictly Prioritized ISLIP” Comparing normalized video request numbers for 0.7 and 0.8 load implies increase in average cell delay. However comparing normalized video request numbers for 0.8, 0.9 and 1 load implies no increase in average cell delay. Similar results are observed for the “Limited Prioritized ISLIP” too.

Comparing the average cell delay results which is given in Figure 6.6 shows; and comparing the average request numbers given at Table 6.5 and Table 6.6, implies that limiting priority by a window size, causes increase in average video cell delays and decrease in average data cell delays.

6.6.2 Experiment 6.2 70% video-30% data -Flow Delays

In this experiment the ratio of video cells is 70%, 16x16 switch configuration with 4 iterations is used. Figure 6.7 shows the average cell delays of data cells and video cells which are calculated separately. Overall delays will be shown at experiment 6.4. In this experiment following algorithms are compared: “Strictly Prioritized ISLIP” and “Limited Prioritized ISLIP”. Moreover, “Limited Prioritized ISLIP” is tested twice with “priority window” size 2 and 4. The “priority window” is the number of giving priority to the high priority flows in a row.

The only difference of this experiment from experiment 6.1 is the ratio of incoming video cells. When two graphs are compared (Figure 6.6 and Figure 6.7); it is seen that in Figure 6.7, video delays are not constant even under heavy load. The reason of this behavior is the ratio of the video cells.

Moreover, the point that data cells start to starve is has become a little later at the horizontal axis. Because, the number of incoming data cells are far fewer when compared to experiment 6.1.

6.6.3 Experiment 6.3 30% video-70%- Overall Delays

In this experiment algorithms’ overall performances are compared. The statistics are collected from the same simulation run of the experiment 6.1. However, in Figure 6.8 overall performances of the all incoming cells are given. When thinking of giving

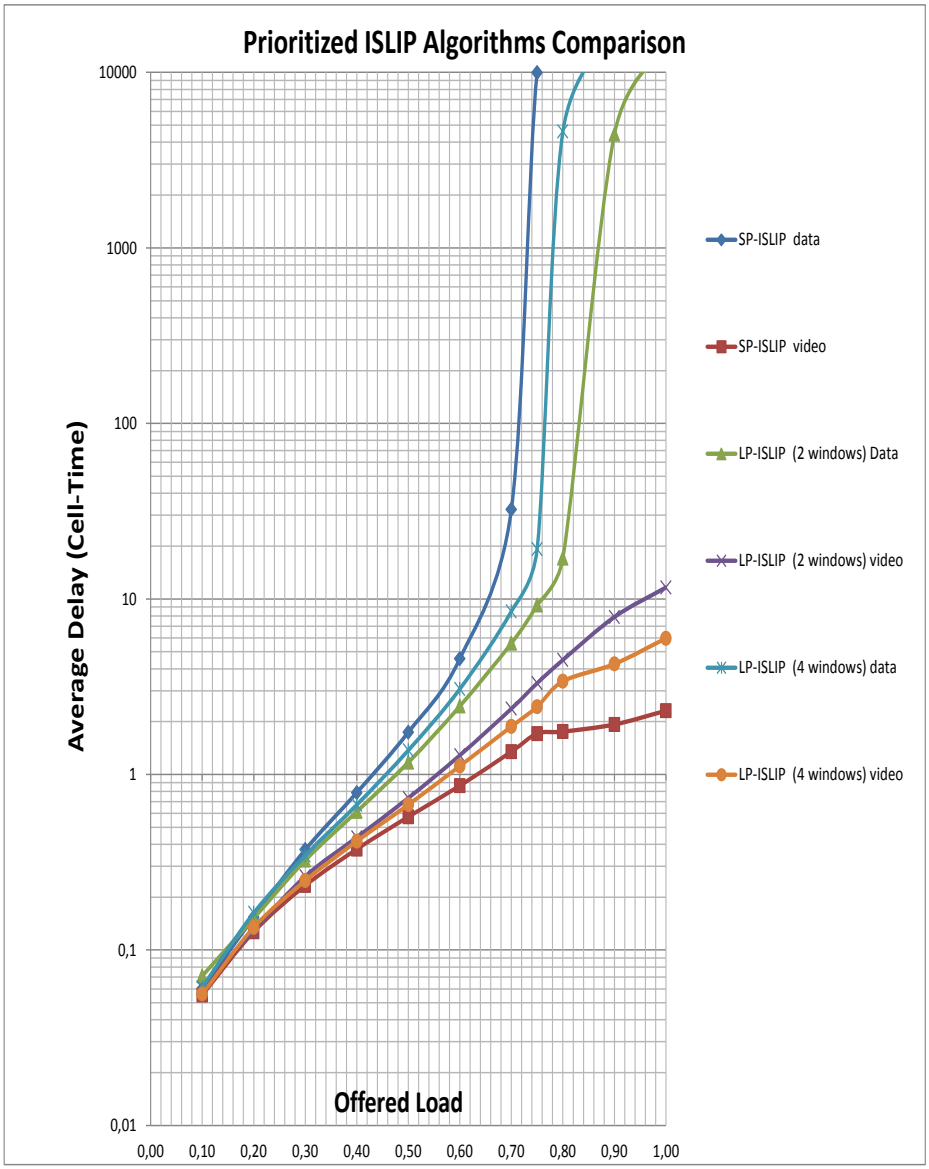

Figure 6.7: 70% video-30% data -Flow Delays

priority of the video cells in order to meet the QoS metrics of the video flows, the overall delay performance of the switch can be jeopardized. While constructing a switch configuration; video, data and overall performance should be considered.

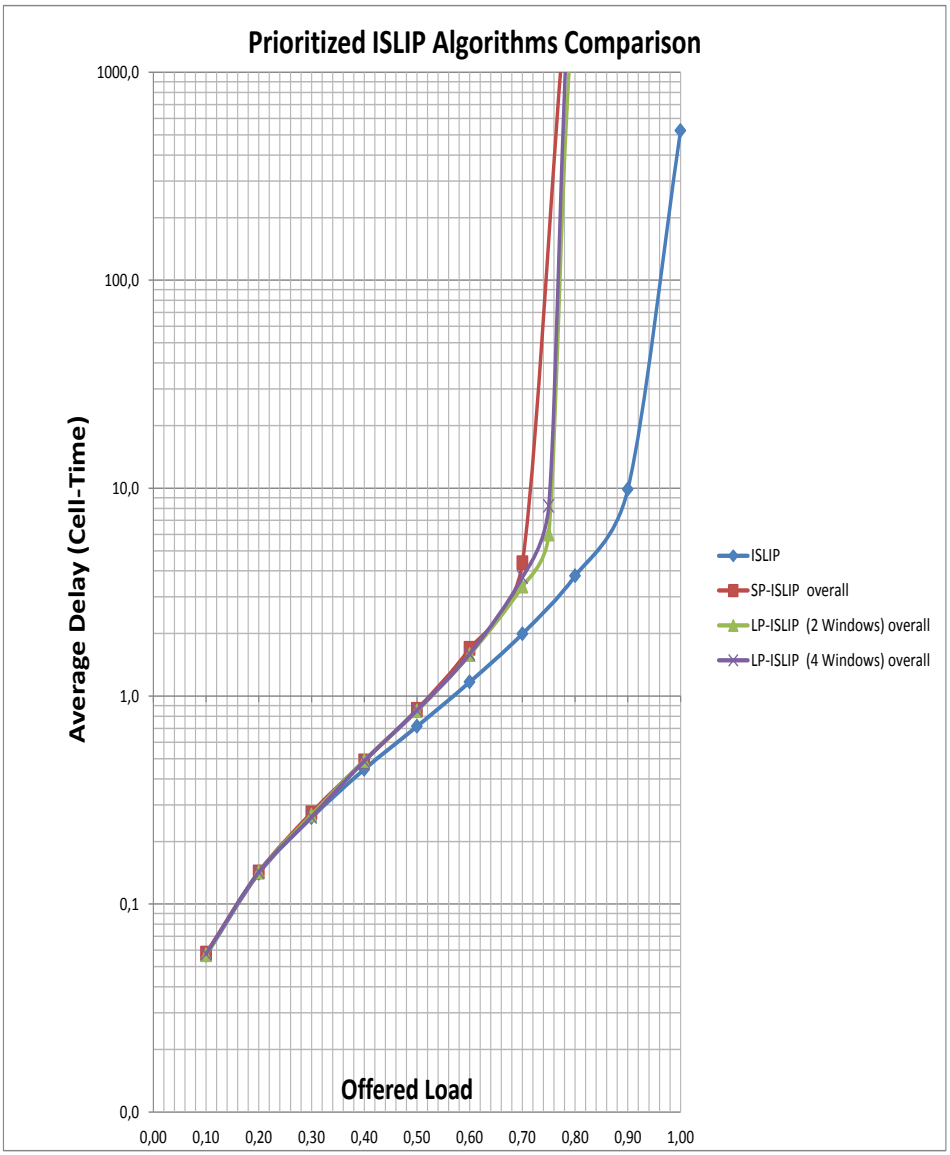

Figure 6.8: 30% video-70%- Overall Delays

As it can be seen from the Figure 6.8 all variations of the prioritized ISLIP algorithms become unstable under heavy load (after around 0.7-0.75 offered load). However ISLIP is stable until offered load becomes 0.95.

6.6.4 Experiment 6.4 70% video-30% data Overall Delays

This experiment again shows the overall delay performances of the algorithms. Only difference from experiment 6.3 is the ratio of the video cells.

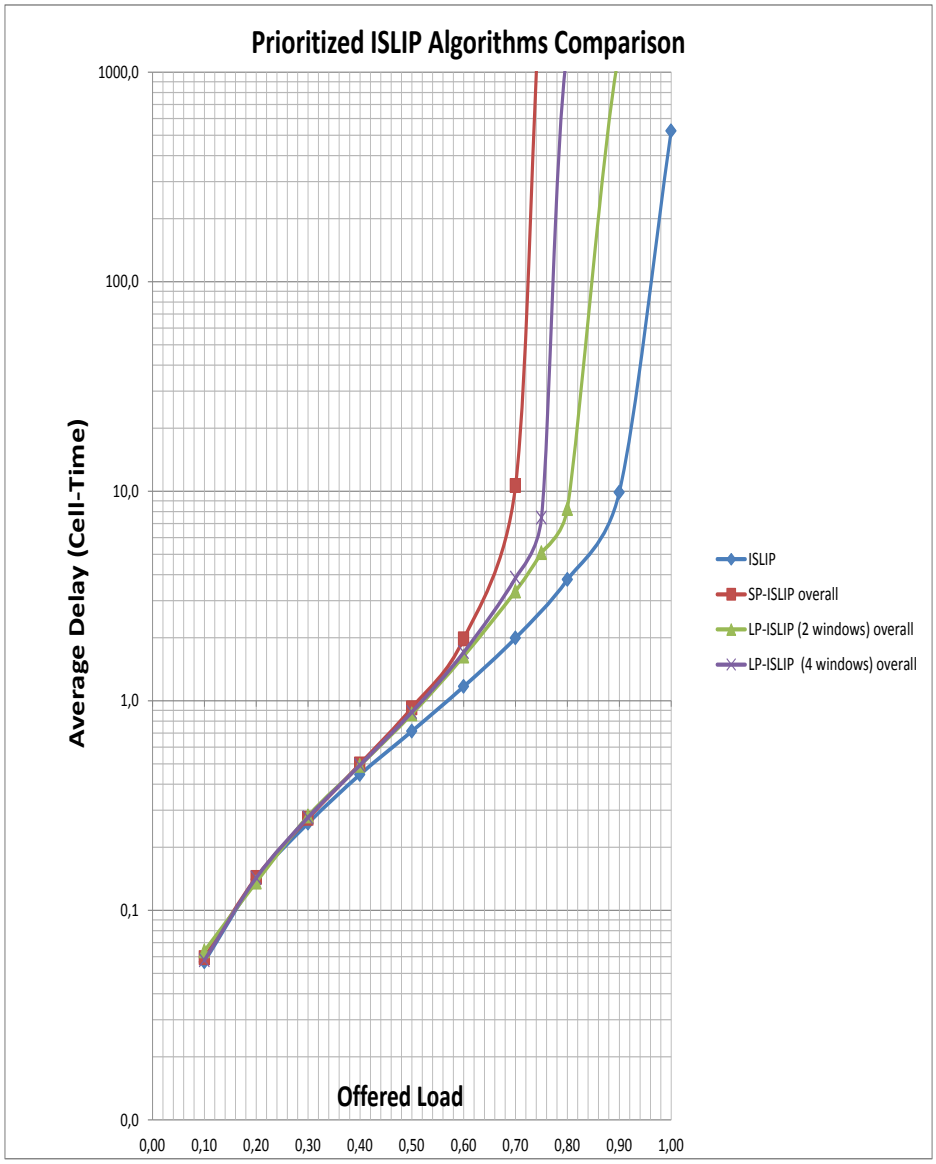

Figure 6.9: 70% video-30% data Overall Delays

When Figure 6.8 and Figure 6.9 are compared, it can be seen that, prioritized algorithms stay stable a little more when the ratio of the video cells are high.

CHAPTER 7

CONCLUSION AND FUTURE WORK

Simulation platforms have become critically important for performance analysis in every field of research. For fabric scheduler algorithms, using software simulators in order to evaluate the performance of a given algorithm under different traffic scenarios and according to various metrics is indispensable. Today's famous network simulation tools such as OPNET and NS-2 do not provide support in order to evaluate switch fabric scheduler algorithms.

The first contribution of this thesis work is *SwitchSim* which is a software tool developed in order to simulate IQ switches' fabric scheduler algorithms. SwitchSim is designed with a modular, object oriented architecture with well defined interfaces to enable plugging in different fabric scheduler algorithms and traffic models. SwitchSim currently implements the well-known and most used IQ switch fabric scheduler algorithm ISLIP and PIM. SwitchSim is verified by comparing the results for a given traffic arrival trace to a hardware switch fabric scheduler implementation as well as the results provided in the original ISLIP work.

As the second contribution, in order to provide QoS support to IQ switch architectures, ISLIP algorithm is extended with two different algorithms: SP-ISLIP and LP-ISLIP. Development of these algorithms includes adding a new block, named IMC, to the algorithm which manages requests, and altering the structure of VOQs.

We present the results of a number of experiments that are performed by SwitchSim both for the verification of the correctness and to evaluate the performances of SP-ISLIP and LP-ISLIP algorithms. New proposed extensions of ISLIP are tested with

2-level prioritization. Considering rapidly increasing importance of video traffic over the Internet these two levels can be interpreted as video and data packets.

The proposed architecture of the simulation tool supports fabric scheduler algorithms of IQ switches. As a future work; implementing other well-known IQ switch fabric scheduler algorithms other than ISLIP and PIM will increase contribution of the simulator.

As another future work, the simulator can be extended to support fabric architectures other than IQ switch fabrics, such as OQ ,CIOQ, Clos Network fabric architectures.

REFERENCES

[1] Cisco the zettabyte era-trends and analysis. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/VNI_Hyperconnectivity_WP.html. Acessed: 2014-08-20.

[2] GIGAOM.com chart: Cisco owns the switching and routing world. https://gigaom.com/2013/02/27/chart-cisco-owns-the-switching-and-routing-world/. Acessed: 2014-08-20.

[3] NS-2 the network simulator- 2. http://www.isi.edu/nsnam/ns/. Acessed: 2014-08-20.

[4] OPNET optimized network engineering tools. http://www.riverbed.com/products/performance-management-control/opnet.html. Acessed: 2014-08-20.

[5] WeiXiang Tang and Yarsun Hsu . Design and Analysis of a Pipelined Clos Network with Late Release Scheme. *International Journal of Digital Content Technology and its Applications*, 6(13):482–498, July 2012.

[6] Thomas E. Anderson, Susan S. Owicki, James B. Saxe, and Charles P. Thacker. High-speed switch scheduling for local-area networks. *ACM Transactions on Computer Systems*, 11(4):319–352, November 1993.

[7] R. Y. Awdeh and H. T. Mouftah. Survey of ATM switch architectures. *Computer Networks and ISDN Systems*, 27(12):1567–1613, 1995.

[8] Jon C R Bennett and Hui Zhang. WF2Q: worst-case fair weighted fair queueing. In *Proceedings - IEEE INFOCOM*, volume 1, pages 120–128. IEEE, 1996.

[9] Alan Demers, Srinivasan Keshav, and Scott Shenker. Analysis and simulation of a fair queueing algorithm. *Computer Communication Review*, 25(1):174–187, 1995.

[10] Eddie Kohler, Mark Handley, and Sally Floyd. Designing DCCP. *ACM SIGCOMM Computer Communication Review*, 36(4):27, August 2006.

[11] Ralf Luebben and Robert Doverspike. Fast rerouting for IP multicast in managed IPTV networks. In *2009 17th International Workshop on Quality of Service*, pages 1–5. IEEE, July 2009.

[12] M.Akpinar. Switch fabric schedulers with intelligent multi-class support: Design, implementation and evaluation on fpga. M.S. Thesis, METU, sep 2014.

[13] N. McKeown. The ISLIP scheduling algorithm for input-queued switches. *IEEE/ACM Transactions on Networking*, 7(2):188–201, April 1999.

[14] Victor Murcia, Meli Delgado, Tito R. Vargas, Juan C. Guerri, and Javier Antich. VAIPA: A video-aware internet protocol architecture. In *2011 IEEE 12th International Conference on High Performance Switching and Routing*, pages 140–145. IEEE, July 2011.

[15] M. Shreedhar and G. Varghese. Efficient fair queuing using deficit round-robin. *IEEE/ACM Transactions on Networking*, 4(3):375–385, June 1996.

[16] R. Stewart and C. Metz. SCTP: new transport protocol for TCP/IP. *IEEE Internet Computing*, 5(6):64–69, 2001.

[17] Haoxue Wang, Yuxiang Hu, and Zhu Ke. A scalable simulation platform for switching and scheduling. In *2008 International Conference on Information and Automation*, pages 1389–1394. IEEE, June 2008.