

**INTEGRATED QOS MANAGEMENT TECHNIQUE FOR INTERNET  
PROTOCOL STORAGE AREA NETWORKS**

**JOSEPH KITHINJI**

**A THESIS SUBMITTED IN PARTIAL FULFILMENT OF  
REQUIREMENTS FOR AWARD OF THE DEGREE OF DOCTOR OF  
PHILOSOPHY IN COMPUTER SCIENCE IN THE MERU  
UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**2022**

### DECLARATION AND CERTIFICATION

This thesis is my original work and has not been presented for a degree in any other Institution.

Name: Joseph Kithinji

Registration No: CT501/3006/14

Sign.....

Date.....



30/9/2022

### CERTIFICATION

This thesis has been submitted for examination with our approval as University Supervisors.

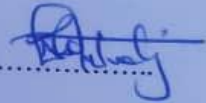
1. Dr. Makau Mutua (Ph.D)

Senior Lecturer, Dept. of Computer Science

Meru University of Science and Technology, Kenya.

Sign.....

Date.....



03.10.2022

2. Dr. David Gitonga Mwathi (Ph.D)

Senior Lecturer, Dept. of Computer Science

Chuka University, Kenya.

Sign.....

Date.....



30/9/2022

## **ACKNOWLEDGEMENT**

The writing of this thesis would not have been possible without the support of many people. Firstly, I would like to register my sincere gratitude to my supervisors Dr. Makau S. Mutua and Dr. Gitonga D.Mwathi for their continuous support and guidance during the whole lifetime of my PhD study. Their perpetual energy and enthusiasm motivated me to complete this study. I could not have imagined having better supervisors and mentors for my PhD study. Secondly, my deepest appreciation goes to my dearest family and friends for their never-ending support and love. They are my true motivation in achieving all my dreams. Lastly thanks to School of Computing and Informatics, Meru University of science and technology for giving me the opportunity to further my doctorate.

## **DEDICATION**

To all my loved ones.

## TABLE OF CONTENTS

<b>DECLARATION AND CERTIFICATION ....</b>	<b>Error! Bookmark not defined.</b>
<b>ACKNOWLEDGEMENT.....</b>	<b>ii</b>
<b>DEDICATION.....</b>	<b>iii</b>
<b>TABLE OF CONTENTS .....</b>	<b>iv</b>
<b>LIST OF TABLES .....</b>	<b>xi</b>
<b>LIST OF FIGURES .....</b>	<b>xii</b>
<b>DEFINATION OF TERMS .....</b>	<b>xiii</b>
<b>LIST OF ACRONYMS .....</b>	<b>xiv</b>
<b>ABSTRACT.....</b>	<b>xvii</b>
<b>CHAPTER ONE: INTRODUCTION.....</b>	<b>1</b>
1.0 Overview of the Chapter.....	1
1.1 Background to the Study.....	1
1.2 Problem Statement.....	6
1.3 Objectives of the Study.....	7
1.3.1 Main Objective of the Study.....	7
1.3.2 Specific Objectives of the Study.....	7
1.4 Research Questions.....	8
1.5 Significance.....	8
1.6 Scope.....	8
1.7 Thesis Organization.....	9
<b>CHAPTER TWO: LITERATURE REVIEW.....</b>	<b>11</b>
2.0 Overview of the Chapter.....	11
2.1 Evolution of Storage Systems.....	11
2.2 Directly Attached Storage (DAS).....	13
2.2.1 Small Computer System Interface (SCSI).....	15
2.2.2 Parallel Advanced Technology Attachment (PATA).....	15
2.2.3 Serial Attached Technology Attachment (SATA).....	16
2.2.4 Serial Attached SCSI (SAS).....	16
2.2.5 Flash.....	17
2.3 Network Attached Storage (NAS).....	17

2.4 Storage Area Network (SAN).....	19
2.4.1 Fibre Channel (FC) .....	20
2.4.2 Internet Small Computer System Interface.....	21
2.4.3 ATA over Ethernet (AOE).....	21
2.5 Storage Area Network Building Blocks .....	22
2.5.1 Host Layer.....	22
2.5.2 Fabric Layer .....	23
2.5.3 The Storage Layer.....	25
2.6 Classification of Storage Arrays .....	27
2.7 Storage Area IP Networking.....	28
2.7.1 Fibre Channel over Internet Protocol (FCIP).....	29
2.7.2 Internet Fiber Channel Protocol.....	29
2.7.3 Internet Small Computer System Interface.....	31
2.7.4 The iSCSI Structure.....	32
2.7.5 Internet Small Computer System Interface Protocol Stack .....	33
2.7.6 Internet Small Computer System Interface Naming.....	34
2.7.7 Internet Small Computer System Interface Host Connectivity....	35
2.7.8 Internet Small Computer System Interface Discovery .....	36
2.7.9 Internet Small Computer System Interface Session.....	36
2.7.10 iSCSI Session Logout and Shutdown .....	39
2.7.11 iSCSI Protocol Data Unit.....	40
2.7.12 iSCSI Read and Write Operations .....	41
2.8 Topologies for iSCSI Connectivity.....	42
2.8.1 Native iSCSI Connectivity .....	43
2.8.2 Bridged iSCSI Connectivity .....	44
2.9 Introduction to Quality of Service .....	44
2.10 Levels of Quality of Service .....	45
2.10.1 Best effort QOS.....	45
2.10.2 Differentiated Service .....	45
2.10.3 Guaranteed Service .....	45
2.11 Quality of Service Functions .....	46
2.11.1 Packet Classification and Marking .....	46
2.11.2 Traffic Rate Management .....	47

2.11.3 Resource Allocation.....	48
2.12 Quality of Service Metrics .....	49
2.12.1 Packet Loss .....	49
2.12.2 Latency.....	51
2.12.3 Jitter.....	53
2.12.4 Throughput.....	54
2.13 Quality of Service Architectures.....	55
2.13.1 Integrated Services (IntServe) Architecture.....	55
2.13.2 Differentiated Services (DiffServ) Architecture .....	56
2.13.3 Multi-Protocol Label Switching .....	57
2.14 Techniques for Providing QOS in Internet Protocol Networks.....	57
2.14.1 First in First out Queuing (FIFO) .....	57
2.14.2 Priority Queuing.....	59
2.14.3 Class Based Queuing .....	61
2.14.4 Fair Queuing and Weighted Fair Queuing.....	62
2.14.5 Class Based Weighted Fair Queuing .....	64
2.14.6 Custom Queuing (CQ).....	65
2.14.7 Modified Weighted Round Robin and Deficit Weighted Round Robin.....	66
2.14.8 Hybrid Waiting Queues .....	68
2.14.9 Custom Class Based Weighted Fair Queuing and Priority Class Based Weighted Fair Queuing .....	68
2.14.10 Weighted Fair Queuing and Class Based Weighted Fair Queuing.....	69
2.15 Admission Control for QOS .....	70
2.15.1 Measurement Based Admission Control .....	71
2.15.2 Parameter Based Admission Control.....	72
2.16 Admission Control Algorithms.....	73
2.16.1 Simple Sum.....	73
2.16.2 Measured Sum .....	73
2.16.3 Acceptance Region .....	74
2.16.4 Equivalent Bandwidth Algorithm .....	74
2.16.5 End Point Admission Control .....	74
2.17 Congestion Avoidance Mechanisms for QOS .....	75

2.17.1 Random Early Detection (RED) .....	75
2.17.2 Weighted RED .....	76
2.18 Packet Classification .....	77
2.18.1 Port-Based Approach .....	77
2.18.2 Deep Packet Inspection .....	78
2.18.3 Statistical Signature Based Classification .....	79
2.18.4 IP Address Based Classification .....	79
2.19 QOS for Storage Area Networks .....	80
2.19.1 Stonehenge .....	80
2.19.2 PClock .....	81
2.19.3 Argon .....	83
2.19.4 Facade .....	84
2.19.5 Proportional Allocation of Resources for Distributed Storage Access (PARDA) .....	87
2.20 QOS Optimization Theories .....	90
2.21 Performance Isolation .....	93
2.21.1 The Flower Classifier .....	94
2.21.2 Berkeley Packet Filter .....	96
2.21.3 Iptables Packet Filtering .....	97
2.21.4 RSVP & RSVP6 classifiers .....	100
2.21.5 Traffic Control Index Classifier .....	100
2.21.6 Routing Table Based Classifier .....	101
2.21.7 U32 Classifier .....	101
2.22 Limitations of Linear Search Based Classifiers .....	103
2.22.1 Shadowed Rule Limitation .....	104
2.22.2 Swapping Position between Rules Limitation .....	105
2.22.3 Redundant Rules Limitation .....	105
2.22.4 Bigger Rule Problem .....	105
2.22.5 Sequential Computation Limitation .....	106
2.23 Performance Isolation Optimization .....	106
2.24 Bandwidth Management for QOS .....	110
2.25 Layer One and Two Bandwidth Management .....	112
2.26 Layer Three Bandwidth Management .....	112



2.26.1 Upgrading Cables and Ethernet Hubs .....	113
2.26.2 Network Segmentation and Full Duplex Ethernet .....	113
2.26.3 Bandwidth Allocation, Sharing and Reservation.....	114
2.26.4 Load Shedding and Buffer Allocation .....	115
2.26.5 Flow Control Using Choke Packets and VLANs .....	116
2.27 Layer Four Bandwidth Management .....	117
2.28 Layer Five Bandwidth Management (Application layer) .....	118
2.29 Dynamic Bandwidth Management Algorithms .....	119
2.30 Burst Handling for QOS .....	124
2.30.1 Leaky-Bucket Traffic Shaping.....	125
2.30.2 Token Bucket Algorithm .....	127
2.30.3 Combining Token Bucket and Leaky Bucket.....	128
2.31 Optimization of Bandwidth Management and Burst Handling .....	128
2.32 Integration of QOS Technique.....	132
2.32.1 IQMIS .....	135
2.33 IP Networks Validation.....	137
<b>CHAPTER THREE: METHODOLOGY .....</b>	<b>141</b>
3.0 Overview of the Chapter.....	141
3.1 Research Philosophy .....	141
3.2 Research Design.....	142
3.2.1 Research Strategy.....	142
3.2.2 Data Collection and Analysis.....	143
3.3 Experimental Setup .....	145
3.3.1 Traffic Generation.....	146
3.3.2 Optimization of Performance Isolation.....	147
3.3.3 Optimization of Bandwidth Management and Burst Handling .	148
3.3.4 Integration of Performance Isolation, Bandwidth Management and Traffic Shaping.....	149
3.3.5 Validation of the IQMIS .....	150
3.4 Quality Control .....	150
3.4.1 Validity .....	151
3.4.2 Reliability.....	152
3.5 Review of Objectives .....	152

3.6 Ethical Considerations .....	154
3.7 Summary .....	154
<b>CHAPTER FOUR: PERFORMANCE ISOLATION OPTIMIZATION.....</b>	<b>155</b>
4.1 Chapter Overview .....	155
4.2 Problem Definition.....	155
4.3 Proposed Solution .....	156
4.3.1 Packets Feature Extraction and Selection .....	158
4.3.2 User Classes and Operational Metrics .....	159
4.4 Performance Isolation Optimization Techniques.....	164
4.4.1 Rules Priority Estimation and Sorting.....	164
4.4.2 Partitioning the Rule List .....	167
4.4.3 Linear Tree Rule Structure Design.....	171
4.4.4 Time Complexity Analysis.....	175
4.4.5 Performance Evaluation .....	176
4.4.6 Throughput .....	177
4.4.7 Latency .....	180
4.5 Classifier Accuracy .....	182
4.6 Summary .....	184
<b>CHAPTER FIVE: OPTIMIZATION OF BANDWIDTH MANAGEMENT AND BURST HANDLING .....</b>	<b>185</b>
5.1 Chapter Overview .....	185
5.2 Problem Definition.....	185
5.3 Proposed Solution .....	191
5.4 Bandwidth Management Optimization .....	197
5.4.1 Bandwidth Allocation .....	198
5.4.2 Bandwidth Borrowing.....	200
5.5 Handling Bursts .....	203
5.6 Summary .....	206
<b>CHAPTER SIX: INTEGRATION OF QOS TECHNIQUES AND VALIDATION.....</b>	<b>208</b>
6.1 Chapter Overview .....	208
6.2 Integrated QOS Management Technique.....	208
6.2.1 Priority Estimation Module.....	209

6.2.2 Performance Isolation Module .....	210
6.2.3 Burst Handling Module .....	211
6.2.4 Bandwidth Manager .....	211
6.3 Validation of IQMIS .....	213
6.3.1 Validation Metrics .....	214
6.3.2 User QOS Mapping .....	214
6.3.3 Validation Setup .....	215
6.4 Validation Results .....	215
6.4.1 Throughput versus IO Size .....	216
6.4.2 Latency and IO size .....	221
6.4.3 Jitter and IO size .....	226
6.5 Summary .....	229
<b>CHAPTER SEVEN: CONCLUSION, RECOMMENDATIONS AND FUTURE WORK .....</b>	<b>230</b>
7.0 Chapter Overview .....	230
7.1 Conclusions .....	230
7.2 Future Work .....	233
7.3 Publications .....	233
7.3.1 First Original Research Article Publication Titled .....	233
7.3.1 Second Original Research Article Publication Titled .....	234
<b>REFERENCES .....</b>	<b>235</b>
<b>Appendix A: Research Permit .....</b>	<b>263</b>
<b>Appendix B: Wireshark Packets Capture .....</b>	<b>264</b>
<b>Appendix C: Parkdale Output Screen .....</b>	<b>265</b>
<b>Appendix D: First Publication .....</b>	<b>266</b>
<b>Appendix D: Second Publication .....</b>	<b>268</b>

## LIST OF TABLES

Table 2.1: Quality standards TiPhone TR 101 329 for Packet Loss.....	51
Table 2.2: Quality Standards ITU-T G.114 for Delay .....	53
Table 2.3: Quality Standards ITU-T G.114 for Jitter.....	54
Table 2.4: Quality Standards ITU-T G.114 for Throughput.....	54
Table 2.5: Comparison of Storage Specific QOS Solutions .....	90
Table 2.6: Example of a Sequential Rule List Policy .....	104
Table 3.1: Hardware and Software specifications to be used for the Experiment .....	146
Table 3.2: Summary of Objectives .....	153
Table 4.1: Packet Features Used for Classification .....	159
Table 4.2: Estimated Operational Resource Per User.....	160
Table 4.3: SLO for Classes of Storage Users .....	162
Table 4.4: Sample Classifier Policy with 325 Rules.....	163
Table 4.5: Partitioned Rule List .....	170
Table 4.6: Statistics of Packet Classification .....	183
Table 6.1 Total Number of Packets Generated.....	216
Table 6.2: Scenario 1 with IO size of 4KB .....	218
Table 6.3: Scenario 2 with IO size of 64KB .....	219
Table 6.4: Scenario 3 with IO size of 1MB .....	220
Table 6.5: Scenario 1 with IO size of 4KB .....	223
Table 6.6: Scenario 2 with IO size of 64KB .....	224
Table 6.7: Scenario 3 with IO size of 1MB .....	224
Table 6.8: Average Jitter in Milliseconds .....	226

## LIST OF FIGURES

Figure 2.1: SAN Building Blocks .....	22
Figure 2.2: IP SAN Based on ISCSI.....	33
Figure 2.3: ISCSI Protocol Stack.....	34
Figure 2.4: ISCSI Session Error Handling.....	38
Figure 2.5: ISCSI PDU Encapsulation in an IP Packet .....	40
Figure:2.6:ISCSI Connectivity .....	43
Figure 2.7:Overview of DiffServ operation.....	56
Figure 2.8:First-In-First-Out (FIFO) queuing.....	59
Figure 2.9:Priority Queuing .....	60
Figure 2.10:Weighted Fair Queuing (WFQ).....	64
Figure 2.11: Class-based queuing (CBQ) .....	65
Figure 2.12: Facade Structure .....	85
Figure 2.13: The Flower Classifier Operation .....	95
Figure 2.14: BPF usage overview .....	97
Figure 2.15: Functioning of HTB .....	123
Figure 2.16:Bursty traffic handling for QOS .....	125
Figure 2.17:Leaky Bucket Algorithm .....	126
Figure 2.18:Token Bucket Algorithm.....	127
Figure 2.19: IQMIS Architecture.....	137
Figure 3.1: Experimental Setup .....	145
Figure.4.1: ELPCIS Methodology .....	157
Figure 4.2: Rule Hits Distribution over Varied Block Sizes .....	165
Figure 4.3: Linear Tree Rule Structure Building .....	172
Figure 4.4: Sequential Tree Rule Structure Based on Table 4.5.....	173
Figure 4.5 Writes throughput comparison .....	177
Figure 4.6 Reads throughput comparison .....	178
Figure 4.7 Latency comparison for writes .....	180
Figure 4.8 Latency comparison for writes .....	181
Figure 5.1: Architecture of the HPDDRR.....	194
Figure 5.2 Bandwidth Allocation.....	198
Figure 5.3: Bandwidth Borrowing .....	201
Figure 5.4:Burst handling .....	204
Figure 6.1: IQMIS Architecture.....	209
Figure 6.2: Throughput for 200 seconds.....	217
Figure 6.3: Latency for 200 seconds.....	222
Figure 6.4: Jitter for 200 seconds.....	227

## DEFINATION OF TERMS

**Bandwidth:** Capacity of a network measured in bits per second.

**Burst handling:** Process of regulating traffic to a certain rate.

**Optimization:** Designing an algorithm to work in the most effective way

**Packet classification:** Process of associating packets to classes.

**Performance isolation:** Process of segregating traffic using reservations and limits to avoid interferences between classes of users.

**Linear search:** Sequential search from the top of a list until a match is found.

**Quality of service:** Management of data transmission capabilities of a network in order to offer prioritization.

## LIST OF ACRONYMS

<b>AOE</b>	Advanced Technology Attachment (ATA) Over Ethernet
<b>ATAPI</b>	Attachment Packet Interface
<b>BIOS</b>	Basic Input Output System
<b>BPF</b>	Berkeley Packet Filter
<b>CID</b>	Connection Identity
<b>CIFS</b>	Common Internet File System
<b>CPU</b>	Central Processing Unit
<b>CQ</b>	Custom Queuing
<b>DAS</b>	Direct Attached Storage
<b>DRR</b>	Deficit Round Robin
<b>DSCP</b>	Differentiated Service Code Point
<b>DWRR</b>	Deficit Weighted Round Robin
<b>EDF</b>	Earliest Deadline First
<b>ELPCIS</b>	Enhanced List Based Packet classifier for Internet Protocol Storage Area Networks
<b>FC</b>	Fibre Channel
<b>FC SAN</b>	Fibre channel Storage Area Network
<b>FCIP</b>	Fibre Channel Internet Protocol
<b>FLOGI</b>	Fabric Login
<b>GBIC</b>	Gigabit Interface Converter
<b>HBA</b>	Host Bus Adapter
<b>HDD</b>	Hard Disk Drive
<b>HPDDRR</b>	Hierarchical Priority Dynamic Deficit Round Robin
<b>HTB</b>	Hierarchical Token Bucket
<b>IBM</b>	International Business Machines
<b>IDE</b>	Integrated Drive Electronics
<b>IEEE</b>	Internet Electrical Engineering
<b>IETF</b>	Internet Engineering Task Force
<b>IFCIP</b>	Internet Fibre Channel Internet protocol

<b>IP</b>	Internet Protocol
<b>IP SAN</b>	Internet Protocol Storage Area Network
<b>IQMIS</b>	integrated Quality of Service Management Technique for Internet Protocol Storage Area Networks
<b>ISCSI</b>	Internet Small Computer System Interface
<b>ISID</b>	Initiator Identity
<b>LAN</b>	Local Area Network
<b>LER</b>	Label Edge Router
<b>LUN</b>	Logical Unit Number
<b>MAC</b>	Media Access Control
<b>MDRR</b>	Modified Deficit Round Robin
<b>MPLS</b>	Multiprotocol Label Switching
<b>NAS</b>	Network Attached Storage
<b>NFS</b>	Network File System
<b>NIC</b>	Network Interface Card
<b>OSD</b>	Object Storage Devices
<b>P2P</b>	Peer to Peer
<b>PARDA</b>	Proportional Allocation of Resources for Distributed Storage Access
<b>PATA</b>	Parallel Advanced Technology Attachment
<b>PCQ</b>	Per Connection Queue
<b>PDU</b>	Protocol Data Unit
<b>PFIFO</b>	Priority First in First Out
<b>PRIQ</b>	Priority Qdisc
<b>QFULL</b>	Queue Full
<b>QOS</b>	Quality of Service
<b>RAID</b>	Redundant Array of Independent Disks
<b>RPC</b>	Remote Procedure Call
<b>RSVP</b>	Resource Reservation Protocol
<b>SAN</b>	Storage Area Network



<b>SAS</b>	Serial Attached Small Computer System Interface
<b>SATA</b>	Serial Attached Technology Attachment
<b>SLED</b>	Service Level Enforcement Discipline for Storage
<b>TBF</b>	Token Bucket Filter
<b>TC</b>	Traffic Control
<b>TOE</b>	TCP/IP Offload Engine
<b>TSID</b>	Target Identity
<b>U32</b>	Universal 32 Bit
<b>VBR</b>	Variable Bit Rate
<b>VLAN</b>	Virtual Local Area Network
<b>WRED</b>	Weighted Random Early Detection
<b>YFQ</b>	Yet Another Fair Queuing

## ABSTRACT

The increasing number of Information technology Users around the world has led to tremendous increase in the amount of data that requires storage. In response to this challenge, new storage area network architectures based on Ethernet (IP) have evolved. With the coexistence of storage traffic with other types of traffic in the same IP network, it is important to offer storage traffic QOS guarantees to prevent performance degradation for storage users. Regrettably, the storage device itself does not provide any capability of guaranteeing storage QOS. QOS is a vital issue in environment of mixed works like IP SANS. The main aim of the study was to analyses the QOS techniques used in IP networks, design, develop and validate an Integrated QOS management technique for IP SANs. The study first analyzed the various techniques for achieving QOS in IP Networks. By decomposing QOS problem into an integration of four techniques of performance isolation, bandwidth management and burst handling the study designed and developed IQMIS, an integrated quality of service management technique for IP SANS. The study adopted experimental research design. Simulations were used as the source of data where Park dale tool was used for simulating reads and writes to the targets. The study generated quantitative results which were analyzed using descriptive statistics and results presented in tables and charts. Empirical results show that IQMIS enables users to fairly share the aggregate system throughput even in environment of contention of resources with a small implementation cost of 6%. In the implementation of bandwidth management and burst handling, IQMIS was found to be work conserving and quickly adopts to network changes with a convergence time of 10 seconds. Further the results show that IQMIS can provide strong performance isolation, superior latency, throughput and jitter compared to best effort. Ultimately IQMIS can be used to provide end to end QOS management in IPSANS and at the same time provide building blocks for providing QOS in IPSANS.to tremendous increase in the amount of data that requires storage.

## **CHAPTER ONE: INTRODUCTION**

### **1.0 Overview of the Chapter**

This chapter presents the introduction to the work presented herein. It provides the crucial background to the study, statement of the problem, research objectives, and the research questions. Further, the chapter outlines the significance of the study, its scope and finally the overall organization of the thesis.

### **1.1 Background to the Study**

The growing number of Information and Communication Technology (ICT) users all around the world has led to a steady growth in the volume of data that needs storage. A modern approach is the use of the Cloud Computing technology which largely employs the use of Storage Area Networks (SAN). A SAN can be defined as a dedicated high-speed network of storage devices and switches connected to computer systems. A SAN provides a common pool of storage to multiple servers where each server is able to connect to the storage devices as if they were directly connected to it(Fang et al., 2019). In addition to providing a way of managing storage in a centralized place, SANs also provide a way of sharing data, backing up and restoring data and moving data between storage devices(Ghazal, Ben, & Claudé, 2012). This is in contrast with the conventional storage area networks that includes the technologies of Small Computer Systems Interface(SCSI) and Fibre Channel(FC) but yet are not able to meet the requirement to increase capacity as well as reduce capital and operational expenditures(Azadegan & Beheshti, 2014).

However, (SCSI) and (FC) are among most prevalent technologies used in SANs. Although these implementations are widely used, they come with a number of challenges. First the SCSI and (FC) implementations use customized network components and therefore are not able to take advantage of the readily available and low cost technologies used in IP-based networks. In addition they involve dedicated equipment that leads to the creation of data centers and storage systems using dissimilar interconnects(Nunome, 2014). Consequently, this creates a necessity to explore alternative solutions that use the IP technology to ease the cost as well as take advantage of fast developments in IP based networking equipment's(Shimano, 2019). In response to these challenges of SCSI and FC SANs, architectures based on Ethernet have been developed such as the fiber channel over IP protocol (FCIP) and the Internet Small Computer System Interface (iSCSI)

With fiber channel being one of the most popular technology used in SANs because of its high performance there has been efforts to extend it beyond the local area network. However it is inhibited in terms of reach as it uses a flow control algorithm which needs to acknowledge every single transmission(Wang et al., 2015). To increase the coverage of FC SANs past the LAN thus, FCIP or internet fiber channel protocol (IFCP) is used to interconnect FC networks over IP networks. However, these technologies utilize specialized hardware for the transfer of FC traffic over a wide area network which results in extra complexity as well as susceptibility to security vulnerability associated with IP network (Yahya-imam, 2014).

The ISCSI is an architecture standard standardized by the Internet Engineering Task Force(IETF) which is designed to transport SCSI application data over Transmission Control/Internet Protocol(TCP/IP) networks (Gauger, 2005). ISCSI has the advantage of putting together messaging network and storage networks into a single communication network at the same time deliver improved scalability of IP networks. This means that storage networking is converging to familiar TCP/IP environment (Paulraj & Kannigadevi, 2019). ISCSI enables the creation of IP SANs (Internet Protocol Storage Area Network). IP SAN is a storage area network that utilizes the IP technology. The ISCSI protocol makes it conceivable for TCP/IP networks to link hosts to their associated storage devices in the SAN (Paulraj & Kannigadevi, 2019). These combination of messaging and storage traffic in IP SANs brings about contention for network resources between messaging traffic and storage traffic which if not managed would lead to degradation of quality of service (QOS) of IP SANs.

In a SAN, QOS guarantee is necessary for ensuring performance guarantees for clients using it (Salmani, 2015). QOS is the control and management the resources in a network by allocating resources based on priority to a group of users in the network. Time sensitive users are given higher priority than those perceived to be less time sensitive. (Mary & Jayapriya, 2019). Primarily, QOS makes it possible to provide differentiated service to flows in network. This is done by assigning priority to flows based on their importance (Shimano, 2015). However, over the recent years, storage network technology has evolved with the focus being that of improving the client quality of service as well as

providing reliability in storage area networks. Moreover users need fast and reliable access to information stored SANs (Jiang, 2019). Although SANs are able to present to the client a virtualized amount of storage, the storage devices do not have QOS features. In addition the storage service agreements included in the storage are not able to provide predictability of service delivery (Bigang, Jiwu, & Weimin, 2006). Due to the sharing of a single storage pool by many users, a single user may flood the network with requests causing performance degradation to other users on the network (Ramaswamy, 2008). Therefore the performance of a given user accessing a storage network is erratic due to the sharing of network resources (Fang et al., 2019). For example in an enterprise network, web hosting, data analysis and data editing may be running at the same time (Mary & Jayapriya, 2019).

To address the problem of service unpredictability in storage area networks, a mechanism of providing QOS based on some policy is required (Mahajan & Mahajan, 2015). QOS is essential in the mixed environment where various users with different levels of priorities and preferences are accessing the storage systems simultaneously (Salmani, 2015). As a result, the need to implement predictable performance in IP SANs led to research which resulted in the development of various approaches including but not limited to Proportional Allocation of Resources for Distributed Storage Access (PARDA), Façade and Stonehenge.

Developed by (Gulati & Waldspurger, 2007), PARDA uses latency measurements to make adjustments to queue lengths to ensure fairness in the

allocation of network resources. However PARDA algorithm has to run on every storage device which introduces overhead. In a further attempt, (Lumb, Merchant, & Alvarez, 2003) developed the façade tool which provides performance guarantees through performance isolation (Nam, Ryu, Park, & Ahn, 2004). Though, Façade was found to be able to utilize resources more efficiently and balance load among the storage devices, its performance was found to degrade with increasing workloads. In addition, Facade also requires to run in multiple storage devices which causes overhead in an attempt to synchronize the independent algorithms.

All the mentioned techniques require that multiple instances of the same algorithm runs on every storage device which increases overhead which is caused by the processing of the individual algorithms(Shimano, 2015). Furthermore, these techniques are implemented on the storage device and therefore do not provide service guarantees when storage traffic is traversing the network which is important in IP SAN storage (Mary & Jayapriya, 2019).

This study therefore developed an integrated approach for providing quality of service in IP-SANS using ISCSI since it's mechanisms for enforcing fairness amongst storage consumers are well tested (Billaud & Gulati, 2017) and are increasingly being used for QOS research. Similarly, integration of the well-known TCP/IP throttling mechanisms and storage systems internals provided a good method for solving the issue of QOS in storage area networks. Since ISCSI uses TCP for data transfer(Mary & Jayapriya, 2019), the use of TCP/IP as transport mechanisms call for use of traffic shaping (Mary & Jayapriya, 2019)

which were integrated in the proposed model. Experimental results clearly indicates that the proposed technique can provide high level QOS management of bandwidth, provide scalable performance isolation and high levels of burst handling in IP-SANs.

## **1.2 Problem Statement**

The concept of creating SANS with IP networking is compelling due to savings gains in terms of management and reduction in cost of deployment. This has been largely supported by the increasing popularity of the Internet Small Computer System Interface (iSCSI) protocol that enables storage read and write commands to be run over the Internet Protocol (IP) network. Regrettably, the IP protocol and the storage device itself do not provide any capability of guaranteeing Quality of Service (QOS) (Salmani, 2015) leading to the need for techniques to address this open problem.

Whereas a number of techniques such as PARDA(Gulati & Waldspurger, 2009), Argon(Wachs et al., 2007),Façade (Lumb et al., 2003) and EdgeIso(Nam, Choi, Yoo, Eom, & Son, 2020), PTrans(Peng & Varman, 2020), pShift(Peng, Liu, & Varman, 2019) have been proposed, these techniques focused on implementing QOS specifically in the storage device itself and assuming that the SAN has no QOS issue. The absence of network QOS in SANs creates an avenue for uncontrolled contention for network resources by storage users and if not addressed would eventually lead poor user performance isolation, poor burst handling and unfair bandwidth management which may lead to throttling of storage priority services.



User performance isolation is an important feature for minimizing disruptions that might be caused by busy flows. Without user performance isolation a greedy user may send huge volumes of data denying services to other users that happen to share the same network. On the other hand bandwidth management as realized by best effort include static allocations for bandwidth which are exceedingly conservative and cannot adopt to network changes achieving poor bandwidth utilization.

An ideal QOS mechanism needs have the following qualities. Firstly meet throughput and latency requirements for well behaving flows without interference from ill behaving flows that is user performance isolation. Secondly allocate bandwidth to users based on the current need. Thirdly use the spare capacity to handle bursty traffic without penalizing user's latency and throughput requirements.

### **1.3 Objectives of the Study**

#### **1.3.1 Main Objective of the Study**

The main objective of the study was to analyze the QOS techniques used in IP networks, design, develop and validate an Integrated QOS management technique for IP SANS.

#### **1.3.2 Specific Objectives of the Study**

- i. To analyze techniques of providing QOS in IP networks.
- ii. To optimize techniques for performance isolation, bandwidth management and burst handling for QOS in IP SANS.
- iii. To develop an integrated QOS management technique for IP-SANS.

- iv. To validate the integrated technique for providing QOS management in IP-SANs.

#### **1.4 Research Questions**

- i. What techniques are used to provide QOS in IP networks?
- ii. How can an optimization design of the QOS techniques of performance isolation, bandwidth management and burst handling be achieved for providing QOS management in IP SANs?
- iii. How can an integrated QOS management be developed in IP SANs using performance isolation, bandwidth management and burst handling?
- iv. Is the developed technique valid for managing QOS in IP SANs?

#### **1.5 Significance**

By using the concepts and findings from this research, network administrators can provide differentiated QOS storage clients with a guarantee earlier not easy to realize. Again the integration of the well-known TCP/IP throttling mechanisms provides researchers a good approach to solving the issue of QOS in IP storage systems, instead of creating new algorithms which are bound to cause uncertainty and overhead. Moreover the outcome of this research acts as a catalyst for further research about QOS in IP SANs.

#### **1.6 Scope**

This study was delimited to the provisioning of QOS to storage traffic generated by users in an IP SAN. This was based on the assumption that any other traffic does not have QOS issues in the SAN. The classes of users studied include the task,

knowledge and power users. The study was also delimited to the QOS techniques for implementing performance isolation, bandwidth management and burst handling.

### **1.7 Thesis Organization**

In the following sections, the thesis provides a summary of what is contained in the rest of the chapters of this thesis:

**Chapter Two.** Literature review: In this chapter the thesis provide a review of the existing literature in storage area networks and QOS. The chapter further presents the framework of the proposed solution that is used in the rest of the thesis.

**Chapter Three.** Methodology: This chapter presents the methods and tools used in achieving the objectives of the thesis. It also includes the metrics to be used for measuring the performance of the proposed system together with the methods of data analysis.

**Chapter Four.** Performance isolation optimization: In this chapter the thesis analyzes the classification of packets for performance isolation. Performance isolation is achieved through classifying packets and binding resources to the packets to prevent interference between classes of packets. The thesis hypothesized that the classification for packets if not optimized would lead to performance degradation when implementing performance isolation. To investigate this the thesis derived a set of classification rules and analyzed the cost of classification using linear search while varying the number of rules.

**Chapter Five.** Optimization of bandwidth management and burst handling: This chapter embarked on the optimization of dynamic bandwidth management

which is achieved through HTB. HTB uses DRR as a scheduler which incurs a lot of delays due to the use of FIFO queues. In addition DRR suffers from head of line queues where big packets delay smaller packets. Therefore to optimize bandwidth management the thesis implemented HPDDRR which is a scheduler shaper that uses hierarchy of queues arranged according to priority to ensure high priority queues are not mixed with low priority queues and are served first. The optimization further uses a dynamic quantum which is generated based on priority and network statistics to ensure more packets are sent per round in contrast to the conventional DRR where the quantum is static.

**Chapter Six.** Integration and validation of performance isolation, bandwidth management and burst handling: In the review of literature in chapter two the thesis found that there is need of integration of QOS techniques to achieve the benefits of all the techniques. Therefore this chapter integrated the techniques of performance isolation, bandwidth management and burst handling. The performance of the integrated technique was evaluated using QOS metrics of throughput, latency and jitter.

**Chapter Seven.** Conclusion, recommendations and publications: This chapter presents the conclusion on contributions and findings, and presents recommendations for future work. It further presents the publications.

## **CHAPTER TWO: LITERATURE REVIEW**

### **2.0 Overview of the Chapter**

This chapter discusses in detail the literature that was used in this study. The chapter begins by presenting a brief evolution of storage systems over the years up to the inception of IP SANs. It further reviews the various IP SANS QOS techniques available and the efforts done over the years to implement them. Similarly, existing gaps are identified which the study aims to fill with the proposed solution. Finally, the proposed solution is briefly introduced as a method that presents promising empirical results in the subsequent chapters.

### **2.1 Evolution of Storage Systems**

Storage systems are built by incorporating layers of hardware and software to provide reliability, manageability and high performance. IBM is credited to have created the first storage device in 1956 and from this invention storage systems have evolved to include new services as well as different forms of interconnection(Wu, Wang, Hua, Member, & Feng, 2017). However, the initial storage devices were attached to the central processing unit (CPU) thus limiting total amount of data that could be held at any given time. Consequently, to address this challenge, in 1964 IBM developed external hard disks which would be managed independent of the CPU(Jaichandra & Prasannakumar, 2015). Nevertheless, over time it was established that single disk drives may not provide the variety of storage capabilities required by modern enterprise systems.

In the 1980s the development of low cost LAN technology lead to a major trend in storage systems. This is because as computers became networked the client server model for computing also emerged. To achieve data sharing storage servers also emerged. This development led to the emergence of network attached storage(NAS) which provided network storage sharing capability through protocols such as NFS (Network File System), HTTP(Hypertext transfer protocol),FTP (File Transfer Protocol) and CIFS (Common Internet File System)(Vishvanath & Nasreen, 2014).

In the 1990s storage systems evolved further with the introduction of the redundant array of independent disks (RAID). It provided performance as well as high availability that could not be achieved in a single drive by means of parity that would be used to restore lost data(Romli, 2019). However, as time went by, disaster recovery became crucial for all IT systems and this further drove the evolution for the design of storage systems. In this case techniques such as point in time copy and mirroring were developed (Puters, 2012) which involved the creation of a virtual copy which could later be used to create a real copy in case of failure. Additionally, mirroring also known as continuous copy is a technique in which a duplicate copy is continuously made at a local site which is primary at a secondary site for recovery of data (Kozhedub & Air, 2018).

At the same time the IT companies were trying to regain control of the decentralized nature of storage brought about by the NAS leading to the creation of data centers. The main aim was to have access to storage without necessarily

having that storage directly attached to a server hence the emergence of storage area networks (SANs). SANs provide the advantage of managing storage separate from the server and having the storage independent of the server hardware and software (Wang, Gilligan, Green, & Raubitschek, 2003). This decoupling of storage from the server brought about advantages of connectivity, scalability and cost. The fiber channel became the technology of choice for the interconnection of storage to servers, however the fiber technology is expensive (Noertjahyana et al., 2020).

The increased speeds in Ethernet LANS using TCP/IP led to an interest in Ethernet SANS in an attempt to reduce implementation costs as well as management costs since network managers need to be familiar with only one type of technology (Chiu, Singh, Wang, Lee, & Park, 2017). The iSCSI was introduced to facilitate the transmission of SCSI commands over the TCP/IP network. However in order to have all benefits of TCP/IP SANs, issues of performance and security need to be addressed (Mistry, Prajapati, Patel, & Saxena, 2020). The following sections look at storage models including NAS, DAS and SANs.

## **2.2 Directly Attached Storage (DAS)**

Direct attached storage (DAS) comprises of a computer or a server which is directly attached to a hard disk drive or an array of drives. Buses such as SCSI, Fibre channel, Advanced Technology Attachment (ATA) and serial ATA(SATA) are used to connect the computer to the storage device (Wu et al., 2017). DAS is a popular storage model in most enterprise networks due to its

low cost and simplicity. DAS is the most suitable solution for attaching storage to computers and servers. However it is not suitable for backup availability and performance requirements. Beside the mentioned downside, DAS is still a popular choice for small enterprise as some of the issues such as those of reliability and performance can be addressed using advancement in Hard-Disk Drive (HDD) and bus technologies. The introduction of standards such as FC BUS, SATA-2, ULTRA SATA and Serial Attached SCSI (SAS) has reduced some of the performance disadvantages of the bus interface. In addition the HDD technology has improved over the years which has addressed some of the requirements of storage users(Vishvanath & Nasreen, 2014).

The DAS ties the storage resources to a given server which becomes a limitation when client applications demand higher requirements on the access of storage data. The number of HDD a certain DAS can support is limited by the bus of the server. In addition in a case of maintenance the server has to be put offline for a period of the maintenance(Preethi, 2017). The high distribution of storage means that data is highly replicated and a free storage in one computer cannot be accessed by another computer(Romli, 2019).

The availability of information stored in storage devices is not always guaranteed since if a server attached to a storage device fails the storage becomes inaccessible. The use of parallel processing for improved performance is not possible in DAS since sharing of workload among servers is not possible. Performance of a DAS is also limited by the processing capability of the server(Noertjahyana et al., 2020).



The cost of maintenance of a DAS is increased by the fact that for a simple backup all the servers must be backed up in addition any repairs to be done are done on all the servers making the job tedious and time consuming(Wu et al., 2017).The DAS uses one or a combination of protocols. These include; Small Computer System Interface (SCSI), Parallel Advanced Technology Attachment(PATA), Serial Attached Technology Attachment (SATA), Serial Attached SCSI (SAS), Fibre Channel and FLASH. These protocols are discussed in the following sections.

### **2.2.1 Small Computer System Interface (SCSI)**

Small computer system interface is an interface used for servers and work stations. However, over the recent years, it has significantly lost its market but is still sparingly used in some modern servers with the evolution from SCSI-1 to ULTRA-640 SCSI. Most of the latest versions of SCSI can handle more than fifteen hard drives (Vishvanath & Nasreen, 2014).

### **2.2.2 Parallel Advanced Technology Attachment (PATA)**

Parallel advanced technology attachment (PATA) was originally known as ATA (Advanced Technology Attachment (ATA) Over Ethernet), IDE (Integrated Drive Electronics) or ATAPI had been the predominant computer storage interface until it has been overtaken by SATA (Serial Attached Technology Attachment). PATA storage drives are still in use today especially for external disk drive boxes (Romli, 2019). Like others PATA has also gone through numerous revisions. PATA supports a master/slave configuration

however sharing of the same port is not recommended if performance is vital (Kozhedub & Air, 2018).

### **2.2.3 Serial Attached Technology Attachment (SATA)**

Serial attached technology attachment (SATA) is the predecessor to PATA. SATA doesn't allow port sharing therefore does not experience performance problems associated with PATA (Noertjahyana et al., 2020). However SATA is more expensive than PATA which uses a very small pin connector attached to a thin cable which reduces the space occupied thus providing sufficient airflow for more denser installations. SATA is used in small servers and inexpensive storage arrays (Chiu et al., 2017).

### **2.2.4 Serial Attached SCSI (SAS)**

Serial Attached SCSI (SAS) is one of the storage interfaces commonly used in servers and storage devices. SAS is seen as the merging of SCSI and SATA is due to the fact that it uses SCSI commands and is pin compatible with SATA. SATA drives can connect to SAS but SAS devices are not able to connect to SATA ports (Wu et al., 2017).

Another difference between SAS and SATA is that SATA cables are limited to one meter long whereas SAS can be able to run up to eight meters long. SAS cables are long due to the high signal voltage but when a SATA is connected the voltage level is lowered. SAS is mainly used for storage arrays and high performance servers however SATA is primarily used in personal computers (Chiu et al., 2017). SAS can be linked to numerous hard drives by means of expanders unlike SATA, however sharing a SATA experiences low

overhead than SCSI. Due to the fact SATA ports are faster, SAS provides superior performance as compared to SCSI and SATA (Wu et al., 2017).

### **2.2.5 Flash**

Flash is not exactly a storage interface however it can be packaged in a hard drive to help reduce the latency associated with seek and rotational latency of hard disk. Flash offers the benefits of 100 times read/write IOPS compared to hard drives and therefore suitable when database application are used(Wu et al., 2017). The limitation of flash memory is that it is limited in terms of writes and rewrites and also is very expensive. When it comes to the limitation of writes and rewrites the flash memory starts to fail when writes and rewrites to the flash memory reach a maximum ranging between ten thousand to one million writes. To deal with the limitation of writes and rewrites flash memory uses wear levelling of which also has some limits(Puters, 2012).

### **2.3 Network Attached Storage (NAS)**

Network attached storage (NAS) is a storage system where storage is accessed by a server which acts as gateway(Vishvanath & Nasreen, 2014). The advantage of NAS is that it offers the central management of storage as well as backup. Other advantages of Network attached storage over Direct attached storage (DAS) is that NAS is not affected by server downtime as in DAS. In addition NAS offers easy way of installing devices for better performance and offers more reliability than DAS. NAS is a way of connecting to the storage via LAN using file systems like NFS and CIFS. The main distinction between NAS and

SAN include that, NAS provides file level I/O access while SAN offers block level I/O access over the network(Chiu et al., 2017).

In a NAS data is transmitted in form of file data stream whereas in a SAN it is transmitted in form of blocks (Vishvanath & Nasreen, 2014). The file access model found in NAS require extra processing in the host as well as in the NAS box. This processing results in overhead which is detrimental to processing speed as well as an increase in data transfer overhead. Although these solutions can be solved using Moore's law, however I/O throughput processing latency cannot be solved by Moore's law. Block level access found in SANs can be used to solve the problem of extra layer processing found in NAS(Noertjahyana et al., 2020).

The client application generate I/O requests which are then handled by the client operating system as system calls similar to those generated by a DAS system(Kozhedub & Air, 2018). The difference between NAS and DAS system calls is how they are processed by the operating system(Romli, 2019). When the system calls are generated an I/O redirector determines the location of the file if it is local or remote (Mistry et al., 2020). If it is included in a DAS, the system calls are processed by the local file system, on the other hand if the file are remote the system calls are handled by network file system which include the NFS or CIFS. The file requests are then forwarded to the TCP/IP protocol stack for reliable transmission across the network( Wu et al., 2017). Network file system protocol provides a mechanism for a client host to access files over a network. For parallel access of storage system parallel NFS (PNFS) is used as

a standard protocol(Chiu et al., 2017). On the other hand Common Internet File System (CIFS) provides mechanism for the sharing of files over internet and intranets. CIFS operates in the application layer and is mainly used for file and printer sharing(Noertjahyana et al., 2020).

When it comes to NAS, a NAS device NIC receives the file access commands then passes them as data grams to the TCP/IP protocol stack(Mistry et al., 2020). The TCP/IP then unwraps the datagrams to access the NFS or CIFS message sent by the client. Then the CIFS and NFS system calls are handled by the NFS file system where the NFS/CIFS commands are mapped to file access system calls from the file system of the NAS storage device(Jacob, 2017). The disk system, the file system and the volume manager in NAS operate in similar manner as in DAS where they translate the file I/O commands into block I/O transfers between the disk system and disk controller. It is key to understand that a disk system cannot be termed as one device as it is an array of devices. Storage devices in a NAS are accessed through technologies such as HBA or the disk controller using block level I/O(Vishvanath & Nasreen, 2014).

#### **2.4 Storage Area Network (SAN)**

A SAN offers block level input/output access for hosts to target storage systems. In a SAN the interconnection between the target storage and host is done either using the Ethernet ISCSI or Fibre channel(Jaichandra & Prasannakumar, 2015). In either FC or ISCSI SAN the storage is separated from the hosts. The hosts and storage are connected in a manner that all the devices are at the same level with the benefits of high availability, high bandwidth and long distance

reach(Romli, 2019). In an ideal setup, the FC SAN and IP SAN the SAN fabric is separated from the LAN. However in an IP SAN it is possible to have a shared infrastructure between a SAN and LAN. Since the SAN requires very high QOS its mixing with the LAN requires efficient techniques for QOS(Wu et al., 2017).

The SAN internal operation uses a number of protocols including Fibre channel, Internet small computer system interface (ISCSI) and ATA over Ethernet (AOE)(Brahneborg, Duvignau, Afzal, Mubeen, & Member, 2022).The following sections looks ate these protocols in details.

#### **2.4.1 Fibre Channel (FC)**

FC provides a mechanism for channeling and networking technology which is used for interconnection between computers and storage devices for high speed data transfer (Romli, 2019). It is used as a transport for multiple protocols such as IP and SCSI for high speed I/O and networking capability. FC incorporates both the channeling and networking capabilities(Wu et al., 2017). A channel provides a dedicated link for moving data from one end to another with the smallest amount of latency(Zhang, Wang, Xiao, Xiong, & Chang, 2019). The networking capability includes packet and circuit switching, ability to act as a transport protocol for other protocols. In its simplest form a Fibre channel network consists of bidirectional point to point channels(Lim & Choi, 2005).

FC is the most popular technology used to implement SAN due to a number of reasons. First FC supports transport media technologies such as copper and optics. Copper implementations offer low cost configurations and fiber optics for high speed at higher cost. Secondly FC technology supports other important

capabilities vital for SANs that is reliability, self-configuration and fault isolation which ensures maintenance does not affect all SAN operations(Wu et al., 2017).

#### **2.4.2 Internet Small Computer System Interface**

Internet small computer system interface (iSCSI) is a cheaper alternative to FC since it runs on the TCP/IP protocol and commonly used Ethernet switches. In addition since most of the network managers are familiar with TCP/IP technology and the Ethernet devices are shared and therefore IP SANs provide immense cost advantages. In addition since IP SANs use TCP/IP, traffic can be rerouted to different sub nets providing wide area network access for back up and disaster recovery. The downside of iSCSI is that since it has to encapsulate the SCSI protocol into TCP packets it is computationally expensive of which this problem can be solved using modern multicore processors(Xiong, Wu, Zhao, & Wang, 2019).

iSCSI targets are the source of the storage and can either be hardware storage array or software running on a server. This means a server running iSCSI target software is like a hardware target. Use of the iSCSI software target provides for the configuration of varied types of devices as targets and also access by a variety of initiators (clients) are the IP network. The iSCSI initiator is found in many operating systems (Dhabake, 2016).

#### **2.4.3 ATA over Ethernet (AOE)**

ATA over Ethernet (AOE) is created as a cheaper alternative to iSCSI. AOE does not use TCP/IP however it encapsulates low level Ethernet

frames(Gaonkar, Bojewar, & Das, 2013). This makes it cheaper also in terms of computations, however it is not routable. AOE is supported in most Linux implementations however it requires purchase of initiator software in windows(Malviya, 2016).

## 2.5 Storage Area Network Building Blocks

This section looks at the SAN building blocks divided into host layer, fabric layer and storage layer. Figure 2.1 illustrates the association between SANs building blocks.

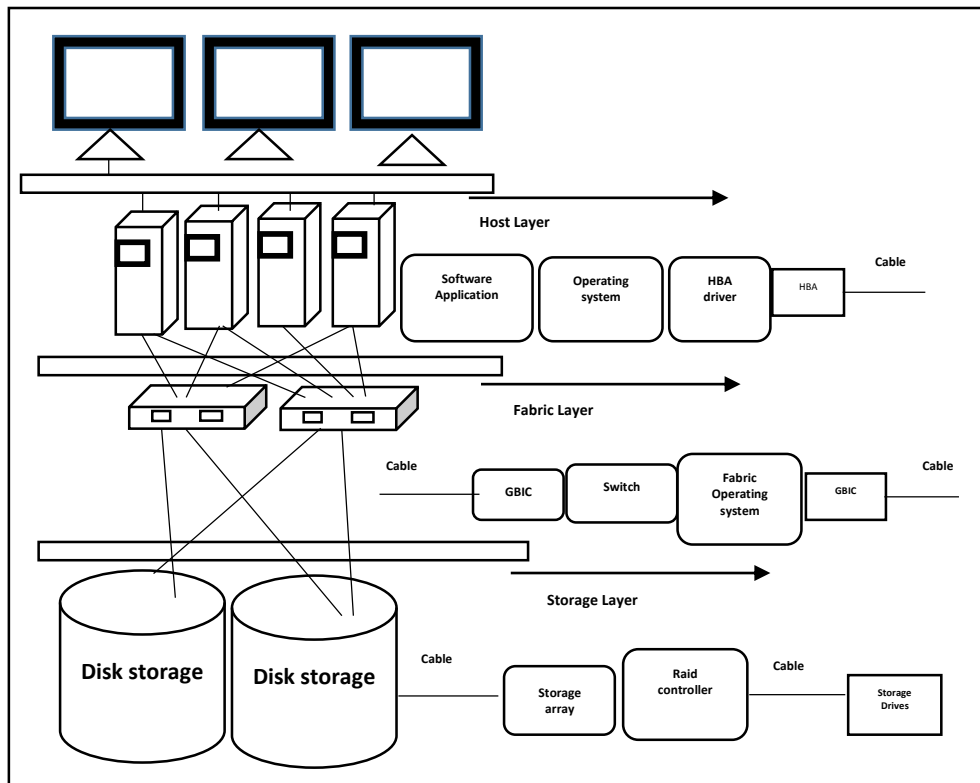


Figure 2.1: SAN Building Blocks

### 2.5.1 Host Layer

The host layer consists of the servers (hosts) and devices that facilitate the host to connect to the SAN. These components include host bus adapter (HBA), host bus adapter drivers and gigabit interface converter (GBIC)(Riabov, 2004).



The host bus adapter is an intelligent device that is affixed in a slot inside the server (Preethi, 2017). The HBA enable the server to connect to the fabric layer as well as communicate with storage devices in the SAN. The HBA intelligence is achieved through hardware and software. The software consists of a driver which allows the HBA and the operating system to communicate as well as the BIOS firmware, that is used in updating the HBA functionality in the SAN(Vishvanath & Nasreen, 2014).

Since a SAN deals with transmission and storage of huge amounts of data, a gigabit interface converter is required to transmit as well as receive data. Data coming from the SAN enters the host via gigabit interface converter in the HBA. The GBIC converts the analog signal into digital signal that the server can understand(Romli, 2019).

### **2.5.2 Fabric Layer**

The fabric layer includes devices such as SAN hubs, switches, routers, protocol bridges, gateway and the cables. The fabric layer is the hardware part of the SAN and the main task of devices found at this layer is to move data from the initiator to the target(Biswas, 2010). The SAN hubs connect the HBAs to the storage devices. A hub contains connection points where each device connects to one of those points(Salmani, 2015). When using a hub each device takes turns to transmit. This is because the hub creates a single loop wire where only one device can transmit at the same time. Devices negotiate for the use of the loop with other device. Hubs have the disadvantage of lower speeds compared to switches. In addition when a device is added to a hub network it interrupts all

the transmitting devices since they will need to negotiate for the use of the link(Hemke, Gawande, Gautum, & Email, 2013).

A SAN switch provides a central connection for the devices that require connection to storage(Jaichandra & Prasannakumar, 2015). A switch allows for the connection of devices in a point to point manner and all devices can communicate at the same time. A single switch could be used to link devices in a SAN however it is recommended to use more than one switch to avoid creating a single point of failure(Salmani, 2015). A switch allows devices to communicate simultaneously by creating a dedicated link between the communicating devices. SAN switches come in categories of 8 ports to hundreds of ports for every switch. In a SAN there are two types of switches that are used that is modular switches and enterprise switches. Standard modular switches are used to in small SAN fabrics, consequently to scale for the network you have to add more switches(Osama, 2011). The enterprise switches are mostly used at the core of the network. This is because they are built for high resilience and can be serviced without putting them offline. In addition parts can be replaced when the switch is on to ensure high availability(Malviya, 2016).

The data routers are best known as bridges or gateways and are used to interconnect ISCSI servers and other devices in a SAN. The data routers enables for intelligent bridging where the servers in a SAN were able to address older disks and tape drives. A data router can also be used to bridge between an IP SAN and an FC SAN.

### **2.5.3 The Storage Layer**

The storage layer contains the types and disks for storing data. They include storage arrays and RAID(Gode, Kashalkar, Kale, & Bhingarkar, 2014). Storage arrays is a collection of disk drives where data is stored. It is known as storage array because it consists of a group of independent disk (RAID)(Jaichandra & Prasannakumar, 2015). For example if there are three 100GB drives which are combined so that there is only one 300GB disk, this is called disk arrays. If data is striped across all the disk drives and include an extra disk with a copy of the data then this is known as a RAID. If there are many disks combined in their hundreds results in the creation of a huge disk drive known as logical unit. The difference between storage array and internal hard drive in a computer is that a storage array can be accessed by all the hosts in a SAN(Hemke et al., 2013).

Redundant array of independent disks (RAID) is a combination of independent disks used to create a bigger drive known as RAID set. Use of RAID comes with two advantages that is high availability and good performance. Performance is improved due to the ability of a host to be able to access more than one disk. On the other hand availability is improved due to the ability to restore data using parity information which exists in either of the disks in a RAID set. Therefore if one disk fails parity information in other disks can be used to restore information(Noertjahyana et al., 2020).

RAID can be numbered from 0 to 6.The numbering symbolize the level of RAID used. RAID sets 0, 1 and 5 are the most popular levels of grouping drives since they have the best variation of redundancy and performance. In situations

where data loss is unimaginable RAID6 is preferable since it provides two parity drives for restoring lost data, however it is slower than other types of RAID(H. Lim & Choi, 2005).

RAID 0 is also known as data stripping. Data stripping means chunks of data are spread across multiple disks(at least two disks) in a RAID set which increases performance since the workload of storing and retrieving data is shared among drives (Jacob, 2017). RAID 1 is also known as disk mirroring due to the fact that data is written on all the physical devices to create mirror images of the data(Chiu et al., 2017). This is important because in a case where one disk fails the other mirror images can be used to restore data lost (Noertjahyana et al., 2020). RAID 1+0 uses a combination of disks mirroring and stripping. One can either strip first or then mirror or mirror then strip(Lim & Choi, 2005).

In a RAID 3 environment there is a dedicated disk storing parity information. RAID 3 is more suitable for long sequential data transfer requests however it performs poorly with very small requests of data(Hemke et al., 2013). The configurations of RAID 4 are similar to those of RAID 3, however the point of distinction between the two is that RAID 4 uses block level stripping while RAID 3 uses bit level stripping(Wu et al., 2017). RAID 5 is a combination of parity and disk stripping and is used to achieve fault tolerance(Jacob, 2017). The RAID 6 setup is used where there is need to store the data over prolonged time periods. The suitability of RAID 6 for long storage of data is due to the fact that it has a double parity which reduces the chances of data loss(Romli, 2019). In an adaptive RAID set the controllers choose between either RAID3 or

RAID 5 based on performance according to the data being written to the disks(Wu et al., 2017). A logical unit number(LUN) can be defined as a unit of storage created from a RAID set(Gaonkar et al., 2013). It could be the whole RAID set or a partition of the RAID set. To effectively use RAID set it is recommended to partition the RAID set into LUNs to avoid wastage of storage(Mistry et al., 2020).

## **2.6 Classification of Storage Arrays**

Storage arrays can be classified based on the size, as either monolithic or modular arrays(Malviya, 2016). Monolithic arrays are big and expensive with a lot of redundant features for fault tolerance(Romli, 2019). Monolithic arrays are mainly used in mainframe computers in addition they have huge cache memory for support of many servers accessing data at the same time. Monolithic arrays require well-conditioned room with suitable power supply. Due to the high cost of monolithic arrays they are mainly used in data centers. To make management of data more efficient monolithic arrays have more advanced intelligent firmware(Toyoda, Yamaguchi, & Oguchi, 2005).

Modular arrays contain the same redundant features as those found in monolithic arrays however, they differ in the ability to connect to mainframe computers and the size of cache memory(Preethi, 2017). Modular arrays cannot connect to mainframe computers and they have less cache memory compared to monolithic arrays. In addition modular arrays have fewer ports which means they can be able to connect to fewer servers. There are two main approaches for storing data that is centralized and decentralized. In centralized there is one big

monolithic storage array while in decentralized approach there could be many modular arrays spread across the departments(Hemke et al., 2013).

## **2.7 Storage Area IP Networking**

The need for more scalable storage solutions is driving many organizations to move away from directly attached storage solutions towards SANS. A SAN is a high speed storage pool that consists of different vendor storage systems, application servers, storage management software and network hardware(Salmani, 2015). SANs offer many advantages including: flexible management due to the fact that you can add servers without affecting stored data and storage can be effortlessly increased or reduced. In addition SANs offer flexibility of being able to reconfigure storage without interrupting their services. SANs reduce business risks through disaster recovery and reduced revenue loss that may result from downtime(Ren et al., 2015).

Currently the most popularly used transmission media in SANS is fiber channel (FC) that is aimed at providing high data rate transmissions, low latency and reliable data transmission between servers(Puters, 2018). Despite the advantages of reliability, low latency and high speeds, FC SANs are expensive that most medium sized organizations cannot afford. In contrast to FC SANs that require different and costly network infrastructure, IP SANs utilize the existing IP network infrastructure which can provide a noteworthy cost reduction in hardware, deployment and operations. To create IP SANs, protocols that are able to transport storage commands over the IP network are required. These protocols include the Fiber Channel over IP (FCIP), Internet

Fiber Channel Protocol (iFCP) and the Internet SCSI Protocols (iSCSI) (Liang, Long, Mei, & Wang, 2019).

### **2.7.1 Fibre Channel over Internet Protocol (FCIP)**

Fibre Channel over Internet Protocol (FCIP) operates as a tunnel protocol for fiber channel frames by wrapping them within TCP/IP. The main application of FCIP is interconnecting FC SANs over long geographical areas where TCP/IP services are only used for interconnection of remote SANs(Hemke et al., 2019). Therefore FCIP contains minimal IP information and only creates a Fiber channel extension. This way storage is consolidated and creates a larger storage capacity out of the many earlier separate ones(Martins & Zucch, 2019). Error handling is done by TCP/IP services as well as congestion control management. The most important benefit of using FCIP is it's ease of set up and the fact that it overcomes distance limitations associated with Fiber channel(Li & Cao, 2017). However the shortcoming of using FCIP is its vulnerability to disruptions and requires creation of two separate networks using two separate platforms and does not provide a direct migration path to IP SANs(Liang et al., 2019).

### **2.7.2 Internet Fiber Channel Protocol**

Internet Fiber Channel Protocol (IFCP) provides a way of transmitting data from one FC SAN to the other through the internet or IP network using TCP/IP services. Whereas FCIP is a tunneling protocol IFCP provides routing services(Dapeng, Chuanyi, & Dongsheng, 2010). This implies that FCIP links FC SANs over the IP networks whereas the IFCP protocol provides TCP/IP

interconnections between end devices directly hence eliminating the need to have fiber switches by providing an IP storage switched network. IFCP was originally developed by Nishan Systems, acquired by McDATA in September 2003(Dhaini & Shami, 2008).

Internet Fibre channel protocol and ISCSI use the same iSNS mechanism. IFCP makes it possible for data to be transmitted as IP packets and also allows for the sharing of packets. Some FCIP configurations when using software compression can achieve similar results, but not otherwise. IFCIP generally breaks the existing Fiber Channel packet into dedicated IP packets. IFCP is only able to compress the payload but not the header information. Compressing the header information would have been useful as it would simplify diagnostics. (Datsika et al., 2018). IFCP uses one TCP connection per fabric login (*FLOGI*), while FCIP typically uses one connection per router link although more are possible( Fang et al., 2019). A FLOGI is the process by which an N\_PORT acquires a class of service as well as the address. Because under IFCP there is a separate TCP connection for each N\_PORT to N\_PORT couple, each connection can be managed to have its own QOS identity(Mebarkia & Zsóka, 2019). A lone occurrence of congestion does not have to reduce the sending rate for all connections on the link. While all IFCP traffic between a given remote and local N\_PORT pair must use the same iFCP session, that IFCP session can be shared across multiple gateways or routers (Murizah & Hafizoah, 2011).

Advantages of IFCIP includes overcoming scalability issues associated with Fibre channel since it is not reliant on Fibre channel routing protocols. Another advantage is that iFCIP uses open shortest path first to ensure autonomy so that



disruptions in one SAN are not propagated to others. Another appeal for IFCP is that it enables the interconnection of a wide variety of FC devices to the IP network(Noorshams, Kounev & Reussner, 2013).

### **2.7.3 Internet Small Computer System Interface**

The internet small computer system interface (ISCSI) protocol implements a client/server model, with clients also called initiators issuing requests commands to the server also called the target(Li & Cao, 2017). The SCSI transport mechanism maps the ISCSI protocol to a particular interconnect. The SCSI protocol is mapped over various transports, including Parallel SCSI, Intelligent Peripheral Interface (IPI), IEEE-1394 fire wire, and Fibre Channel(Nleya & Mutsvangwa, 2018). All of these modes of transport provides the mechanisms necessary to transmit SCSI commands over TCP/IP in order to exploit the advantage of the already established internet infrastructure. A session identity uniquely identifies a session established between an initiator and a target and is made up of the initiator ID and the target tag (Ravindran, Rabby & Liu, 2009). The iSCSI commands direction of transfer is stipulated based on the initiator position. Packets are termed as outgoing when they originate from the initiator towards the target .On the other hand inbound packets are those that are produced by the target towards the initiator(Yang et al., 2018). To ensure performance is not compromised, ISCS uses phase collapse, a mechanism where a command and its data can be transmitted together from the initiator to the target. The phase collapse also ensures that data and its acknowledgements are transmitted together(Mebarkia & Zsóka, 2019).

An iSCSI name is not tied to a specific port or address but it specifies a logical initiator or target. In a case where many NICs are used, they point to the same iSCSI initiator name for the targets as they are paths to the same SCSI layer. In the context of operating systems the name entity is used to refer to the operating system image (Iqbal & Rikli, 2011). iSCSI is a storage network transport protocol that transmits SCSI packets over the IP network by encapsulating them into TCP packets.

#### **2.7.4 The iSCSI Structure**

The iSCSI structure links initiator and target nodes over IP. The iSCSI initiators include devices like file servers and hosts that transmits data to iSCSI target nodes and encapsulate storage data into TCP/IP for transmission over the IP network (Martins & Zucch, 2019). The iSCSI targets breaks down iSCSI commands from target and processes them (Salmani, 2015). The iSCSI targets include devices that accept iSCSI commands and transmit data across the IP network through equipment such as routers and switches. Figure 2.2 illustrates an IP-SAN based on iSCSI. Examples of iSCSI target include disk arrays, RAID devices and tape libraries (Hemke et al., 2019).

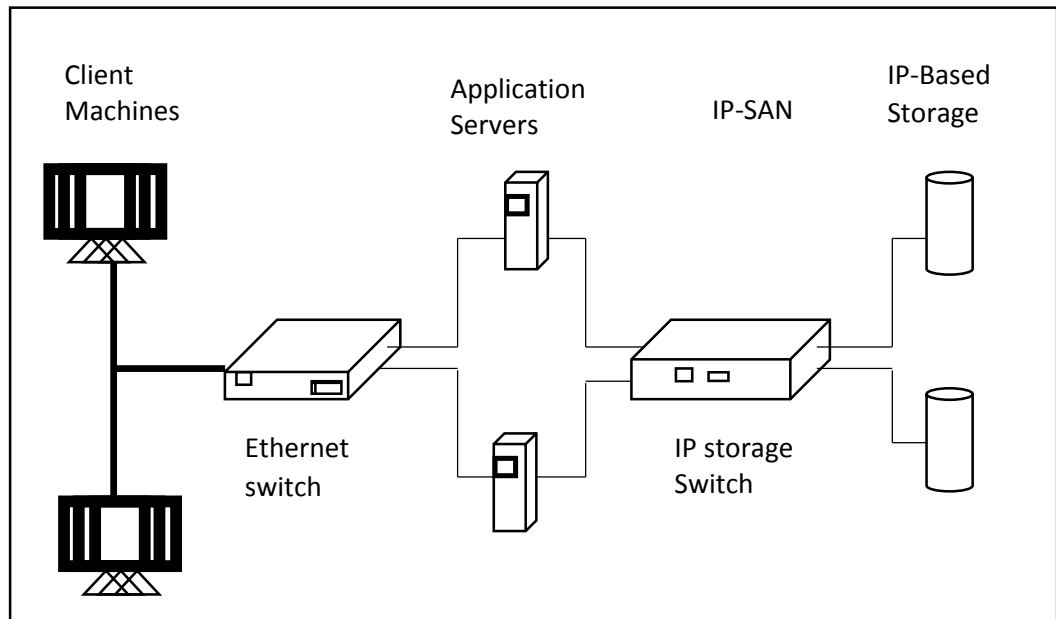
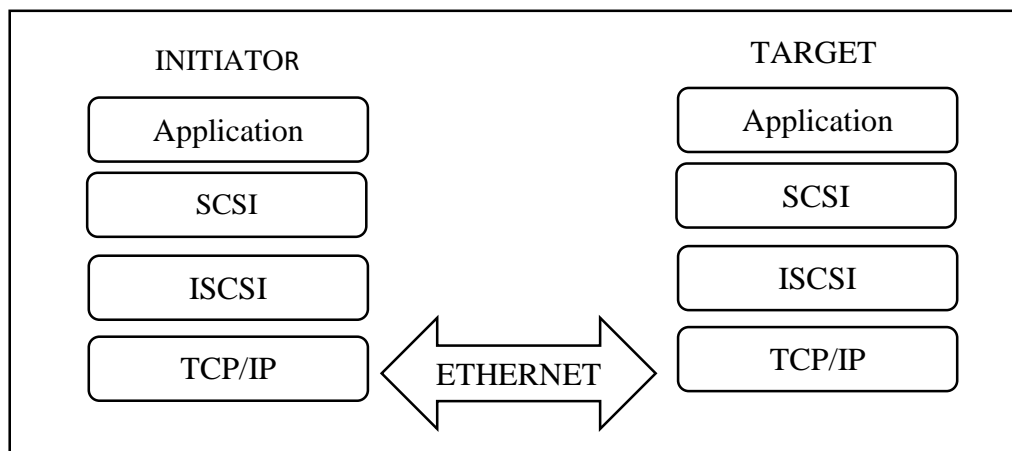


Figure 2.2: IP SAN Based on iSCSI(Source:Nleya & Mutsvangwa, 2018).

### 2.7.5 Internet Small Computer System Interface Protocol Stack

The iSCSI architecture is based on a client server model. Figure 2.3 illustrates the iSCSI protocol stack. At the application layer there is the iSCSI protocol of which the iSCSI commands are used by the target and initiator communication. The SCSI commands are encapsulated into TCP/IP for transmission across the network linking the target and initiator(Gauger, 2005). At the session layer there is the iSCSI session which establishes a session between a storage device and TCP/IP. The iSCSI session is responsible for login, target discovery, authentication and session management. TCP is used to provide reliable communication (Nleya & Mutsvangwa, 2018).

The ISCSI encapsulates the request into TCP/IP adding routing, control information and error checking. The message is then sent the network using the HBA(Neto, Fonseca, & Member, 2007). At the destination the packets go through a reverse process of reassembling data which is then passed to the SCSI controller. The SCSI controller then proceed to either read or write data to the target device(Noertjahyana et al., 2020).



*Figure 2.3: ISCSI Protocol Stack(Source:Nleya & Mutsvangwa, 2018).*

### **2.7.6 Internet Small Computer System Interface Naming**

Internet Small Computer System Interface name is a unique identifier for initiators and targets in an IP SAN. An ISCSI name could be a combination of an asset tag, department names and manufacturer name. There are two types of ISCSI names that is the ISCSI qualified name and extended unique identifier(Mebarkia & Zsóka, 2019).The ISCSI qualified name is generated by use of a domain name which is reserved for a particular organization. The domain name need to be preserved for that organization to avoid conflicts where other organizations cold use the same domain name(Hemke et al., 2019).

Extended unique identifier is a unique identifier based on the IEEE naming standard. Extended unique identifier consists of 16 characters. The ISCSI qualified name provides for naming storage devices for easy management. Network address authority is a type of name that allows for worldwide naming of storage using the international committee for information technology standards(Martins & Zucch, 2019).

### **2.7.7 Internet Small Computer System Interface Host Connectivity**

To connect an ISCSI host to a SAN the host requires to have a NIC with the ISCSI initiator software. For a host to use the ISCSI protocol the initiator software is installed for routing SCSI commands to the TCP/IP stack(Nleya & Mutsvangwa, 2018).There are three options for ISCSI connectivity that is a standard NIC,ISCSI HBA and a TCP/IP offload engine(TOE) NIC card(Li & Cao, 2017). To save on costs a standard NIC can be used. Since most servers come with at least one.

The downside of using standard network interface card is that the processing workload is done by the host CPU as the NIC does not provide any processing capability(Hassan, Albakr, & Al-dossari, 2017). Therefore if a standard NIC is used in a heavy workload situation the host CPU could become a bottle neck. The TCP/IP offload engine could be used as the alternative to NIC in cases of heavy workload as it does all the TCP management and leaving the ISCSI management to the host(Nleya & Mutsvangwa, 2018). However ISCSI management still needs the host CPU. To offload the TCP/IP ISCSI processing form the host an ISCSI HBA is used. In addition the ISCSI HBA provides a

boot from the SAN option using iSCSI otherwise modifications would require to be performed to the operating system. To achieve fault tolerance it is essential to use multipathing multiple network interface cards(NICs) for link aggregation which provides failover or load balancing(Hassan et al., 2017).

### **2.7.8 Internet Small Computer System Interface Discovery**

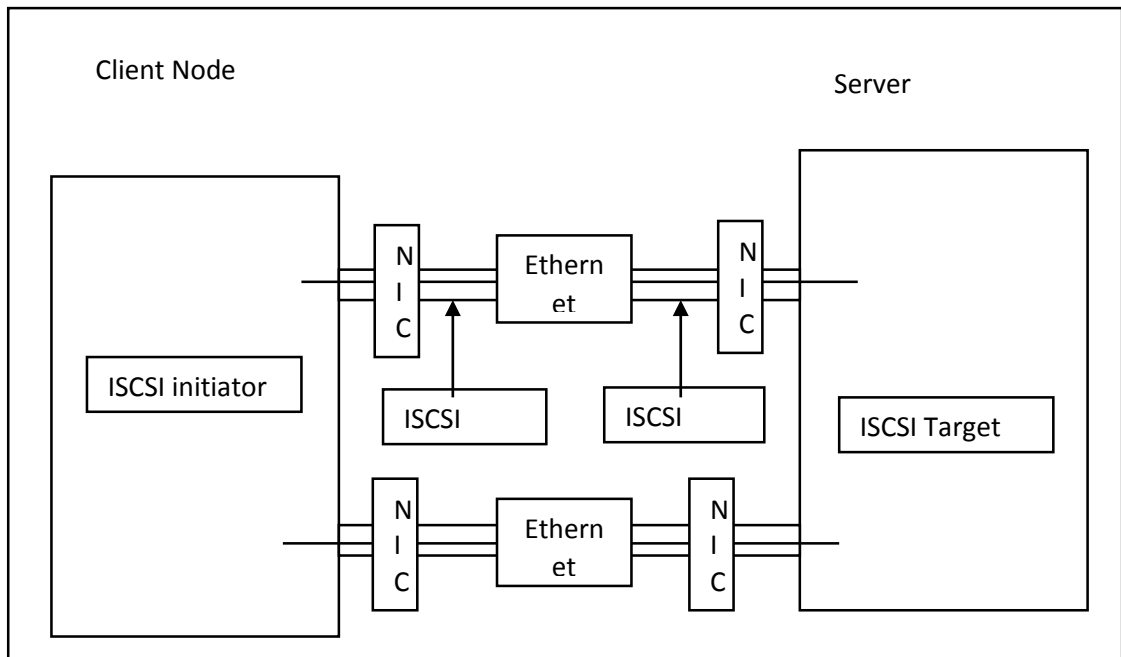
For an iSCSI initiator to connect to a target, it must first discover the targets available to it on the network. An iSCSI initiator can do the discovery in two ways that is by sending targets discovery or by internet storage name service(Li & Cao, 2017). When using the send targets discovery the initiator is configured with the targets network protocol which it uses to make discovery session with the target iSCSI service(Martins & Zucch, 2019). In this case the initiator makes send target command and the target responds by providing the names and addresses available to the initiator(Hemke et al., 2019). On the other hand the internet storage name service provides for the automatic discovery of devices in an IP SAN as all the devices are registered with the internet storage name service server. In this case to know the targets available to it, the iSCSI target just queries the internet storage name service server(Nleya & Mutsvangwa, 2018).

### **2.7.9 Internet Small Computer System Interface Session**

A session is created amongst a iSCSI initiator and a iSCSI target once an iSCSI initiator makes a logon or links to a target(Yakti & Salameh, 2019). During the session the initiator and the target are able to authenticate each other. Thereafter iSCSI facilitates the transmission of SCSI commands over TCP/IP(Dapeng et

al., 2010). The connection between an initiator and a target contains a group of TCP connections which creates a session. A session is composed of a session ID which includes the target part (TSID) and the initiator part (ISID). An iSCSI session is made up of two phases. That is the login session where the target and initiator authenticate each other(Hassan et al., 2017).

During the log in session the target and initiator negotiate parameter for the session. The login session is followed by a full phase session where ISCSI commands and data are sent(Bigang et al., 2006). In ISCSI, sessions can be classified as either an operational session or discovery session. The discovery session basically aims at establishing all available targets(Liang et al., 2019). The parameters negotiated during the login phase determines how many TCP connections can be established over the physical interfaces, data integrity checks and levels of error recovery. Each connection has a unique connection ID (CID), however new TCP connections may be created or existing ones may be removed from the session(Dapeng et al., 2010).



*Figure 2.4: ISCSI Session Error Handling*(Source:Nleya & Mutsvangwa, 2018).

In order for any process to recover from an error it requires to maintain enough state and data. Figure 2.4 illustrates ISCSI session error handling(Han et al., 2018). For ISCSI error recovery the ISCSI initiator needs to preserve the required command and data so as to be able to reconstruct new PDUs. On the other hand, the target needs to keep any unacknowledged data and its status response information(Ferrera & Niguidula, 2017).

Internet small computer system interface (ISCSI) uses retry and reassignment as mechanisms for handling errors. In retry the initiator may resend any missing commands or PDU data to the target(Jaichandra & Prasannakumar, 2019). The reassignment is used to recover errors where the TCP connection between the target and the initiator is lost(Ren et al., 2019). In a scenario where the TCP connection is lost the initiator creates a new connection and sends a task



management PDU informing the target to continue the connection using the new CID(Wang et al., 2015).

There are three levels of error detection and recovery provided by ISCSI protocol that is Level 0, Level 1 and Level 2(Hemke et al., 2019). In level 0, error recovery results in the dropping of the session and the session need to be restarted all over again by the application. Level 1 error recovery is done by retransmission for the corrupted PDU(Protocol Data Unit)(Martins & Zucch, 2019). This process is not associated with the SCSI layer. Level 2 is a robust error recovery which implements a complete connection recovery. In cases where there are many connections if one fails it is moved over to existing connections in a transparent manner(Wang & Wang, 2013). Level 1 and level 2 are more suited for mission critical where only undesirable sessions are dropped. Currently most ISCSI targets support Level 0 recovery which discovers and prevents data corruption. This is because users may rather deal with session loss than corrupted data(Ren et al., 2019).

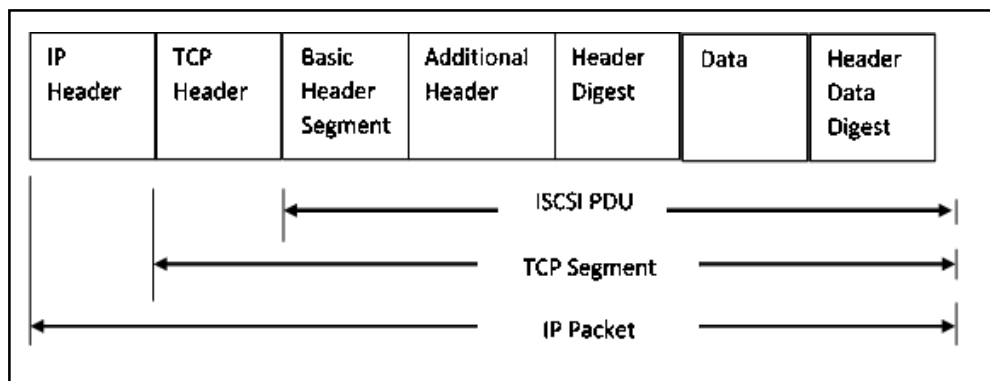
#### **2.7.10 ISCSI Session Logout and Shutdown**

The logout procedure is started by the initiator for and is used to close the session or connection(Murthy, 2015). However in cases where there are errors the target is the one that initiates a session by sending an asynchronous iSCSI command. In both cases it is the initiator that transmits a logout requests after which no more connections may be established. The targets responds with a log out message that the cleanup is complete and no more communications will be sent on the current session(Dapeng et al., 2010). Additionally the logout

message from the target includes the duration of time that target wait for information used for recovery purposes (Time2Retain) and the amount of time the initiator will have to wait before trying to create another connection(Time2Wait).Lastly the session and connections are closed by transmitting the TCP FINs(Sheltami, 2019).

### 2.7.11 ISCSI Protocol Data Unit

All communications between the initiators and targets is done using iSCSI Protocol Data Units (PDUs).This means that ISCSI uses PDUs as the basic unit for communication. ISCSI PDUs contain header segments and data segments. To facilitate their transport via IP network, PDUs are encapsulated in an IP packet for transport(Nleya & Mutsvangwa, 2018).



*Figure 2.5: ISCSI PDU Encapsulation in an IP Packet*(Source:Dapeng et al., 2010)

A PDU is made up of the segments shown in Figure 2.5 .The IP headers includes information necessary for routing the packet across the network while the TCP header includes information for ensuring guarantee delivery of packets to the target(Narale, 2019). The iSCSI header contains information on how the target

is supposed to extract data and SCSI commands(Sheltami, 2019). The header also contains an optional CRC also known as digest which is used to for data integrity and error correction. Examples of PDUs used in ISCSI include SNACK PDU, Ready to Transfer (R2T), Data In/Out, Login Request/Response and iSCSI Command/Response(Narale, 2019).

#### **2.7.12 ISCSI Read and Write Operations**

The most significant operations in an ISCSI SAN is the read and write operations. Distinct PDUs are used for the read and write operations(Liang et al., 2019). Once a session is started, the initiator is then able to transmit data-in PDU for the read operation and a data-out PDU for use during the write operation(Nleya & Mutsvangwa, 2018). For the transfer of data from the initiator to the target the command PDU is used. When the initiator desires to write data to the target it first issues an ISCSI write command(Liang et al., 2019). In return the targets issue a R2T PDU informing the initiator which volume of data needs to be transferred. After the initiator receives the R2T PDU from the target it responds with data out PDUs encapsulating the SCSI data.

Alternatively if the initiators requires to read from the target, it issues a read request inform of ISCSI data in PDUs(Li & Cao, 2017). After the data transfer is complete the target sends to the initiator a SCSI response PDU indicating successful completion of transfer of data and any error detected. Each SCSI command PDU corresponds to SCSI response PDU with zero or more data PDUs(Dapeng et al., 2010). The ISCSI data with their corresponding response PDUs are sent through the matching TCP connection through which their ISCSI command PDU was sent(Nleya & Mutsvangwa, 2018).

## **2.8 Topologies for iSCSI Connectivity**

There are two variants of topologies that can be used to create ISCSI SANS namely native and bridged topologies. The native topology allows communication only via IP and therefore no FC components are included(Hemke et al., 2019). In the case of native topology the initiators are connected directly to the targets or connected via IP switches and routers. On the other hand bridged topologies includes bridging between IP and FC. For instance the initiators can be in an IP network while the targets could be in an FC SAN(Martins & Zucch, 2019). Figure 2.6 illustrates the bridged and native connectivity.

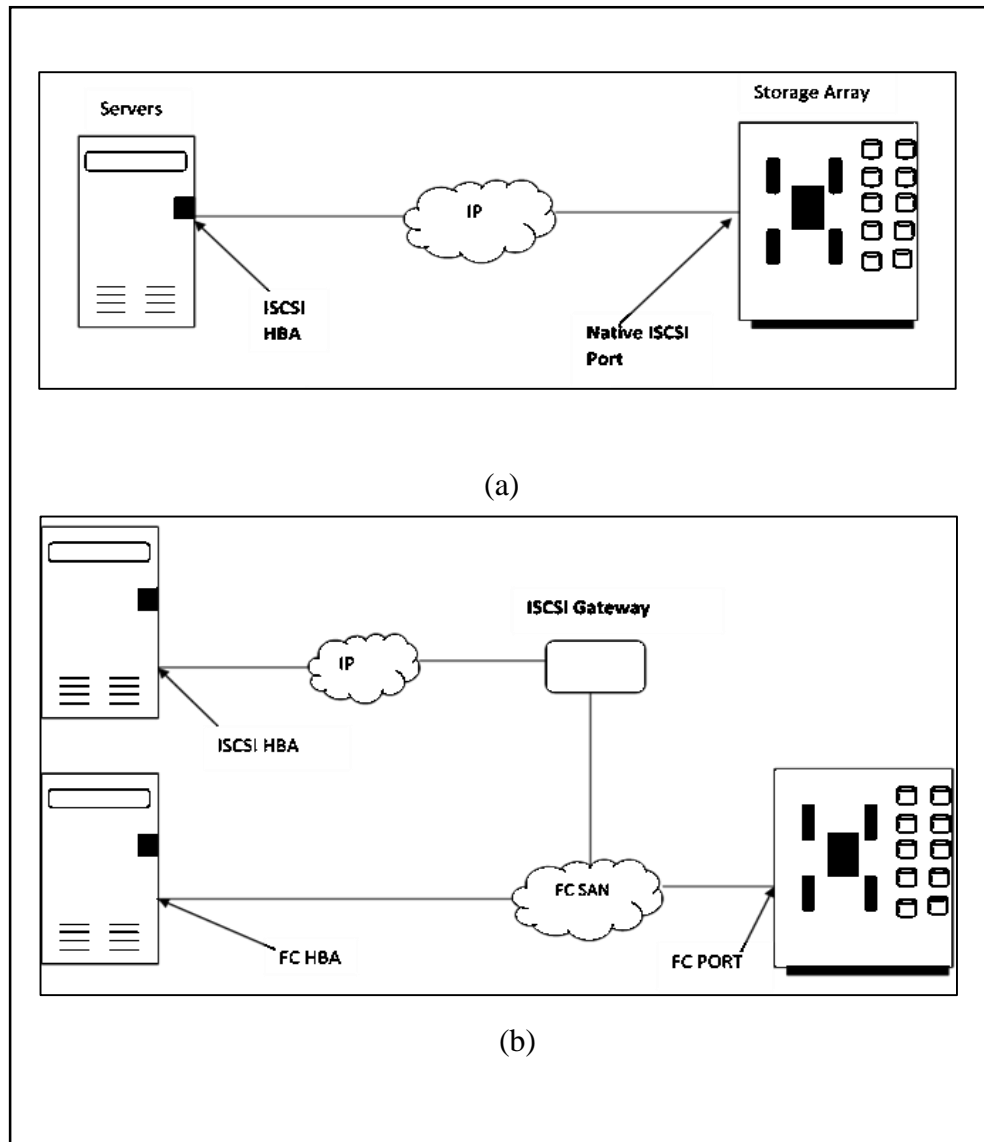


Figure:2.6:ISCSI Connectivity (a)Native ISCSI Connectivity (b)Bridged ISCSI Connectivity (Source:Lumb et al., 2003)

### 2.8.1 Native ISCSI Connectivity

For implementations where we have an ISCSI enabled array FC interconnection is not required. In the illustration presented in Figure 2.12 (a) there is an array of storage disks connected to an IP switch with an IP address and a listening port configured. Once the configuration of initiators is done, connection to the storage array is possible(Wang & Wang, 2013). The initiators immediately

requests for a list of logical unit numbers (LUNs) which are used to identify particular devices in the SAN(Meth & Satran, 2003).

Multiple initiators can be serviced by a single array port so long as the storage array can be able to handle the requests generated by the initiators(Martins & Zucch, 2019). To ensure availability many arrays are configured so as to have numerous targets configured on the initiators. Some NAS storage devices can be configured to function as ISCSI targets for file and block level access to SAN especially in situations where an ISCSI bridge is not available(He, 2019).

### **2.8.2 Bridged ISCSI Connectivity**

Bridged based ISCSI topology includes the FC components(Martins & Zucch, 2019).Figure 2.12(b) illustrates a bridged topology interconnected to a FC storage disks servicing a group of initiators interconnected using ISCSI(Toyoda et al., 2005). The storage disks array does not include Ethernet ports which makes it necessary to have a gateway to bridge the IP network with the FC SAN(Wang & Wang, 2013). In this topology the bridge device contains Ethernet ports for connection to the IP network and FC ports for connection to the ISCSI storage array. The ISCSI arrays is assigned the same IP addresses as the Ethernet ports. The initiator is assigned identical IP address as the bridge, the bridge is configured with one or more FC virtual initiators(He, 2019).

### **2.9 Introduction to Quality of Service**

Quality of service(QOS) in the context of this study is defined as the management of data transmission capabilities of a network in order to offer prioritization(Guo, 2019). In any network environment QOS is vital since it can

be used to optimize performance of a network in terms of jitter, packet loss, latency and throughput(Sheltami, 2019).

## **2.10 Levels of Quality of Service**

The levels of QOS include the modes used to provide QOS for flows in a network. These include best effort, differentiated service and Guaranteed Service(Haghighi & Heydari, 2018)

### **2.10.1 Best effort QOS**

In best effort service no guarantees on whether a packet is delivered or not. Best effort does not include QOS since no guarantees are provided in forwarding traffic( Wang, Member, Li, & Wu, 2018). File transfer protocol is able to work well with best effort however other applications require QOS in terms of bandwidth, delay and packet loss(Raschellà, Bouhafs, Seyedebrahimi, Mackay, & Shi, 2017).

### **2.10.2 Differentiated Service**

In the differentiated level, traffic is put into classes based on their requirements. Each class is serviced based on the configured QOS for the class(Samadi, Member, Fiorani, Shen, & Member, 2017). It is important to note that differentiated service does not provide QOS guarantees it only treats traffic differently based on their QOS requirements. Due to this fact differentiated service is known as soft QOS(Haghighi & Heydari, 2018).

### **2.10.3 Guaranteed Service**

In this level resources are preserved to meet a flows service requirement. This means it requires priori resource reservation(Zhao, 2018). For this reason it is

known as hard QOS since it offers rigid guarantees in the network. Resources reservations don't scale well in case there are thousands of flows at a particular time. To make this mechanism scalable aggregate reservations are used as a means of providing scalable guaranteed service(Datsika et al., 2018).

## **2.11 Quality of Service Functions**

This section looks at the QOS functions including packet classification and marking, traffic rate management and resource management/allocation.

### **2.11.1 Packet Classification and Marking**

The function of packet classification and marking is meant to establish packets belonging to a given class based on header information(Haghighi & Heydari, 2018). A marker is used to color the classes' traffic using the IP precedence or the differentiated service code point (DSCP)(Sun, Yu, & Fan, 2020). Packet classification involves the association of packets to a particular class based on header information fields. The identification for classification can range from simple to complex. The different classification types includes the flow of identification based on destination address, port number, source IP address, protocol, source port, port number and source IP address(Zhao, Ma, Zhou, & Zhang, 2018).

Another way is by use of differentiated service code point field or priority. In other cases packet length could be used to classify the packets(Guo, 2019). Packets could also be classified using the source and destination physical addresses also known as media access control (MAC) addresses. Classification of packets can also be achieved using any information residing in the router(



Zhu et al., 2019). Packet classification is also referred to as packet coloring or parked marking. Coloring is meant to identify a packet belonging to a particular class. Packets are colored by marking the Differentiated Service Code Point (DSCP) field or the IP precedence field. The IP precedence field is used to specify the priority with which a packet should be handled(Sheltami, 2019).

The IP precedence field consists of 3 bits and is contained in the type of service byte. Also included in the IP precedence bits is the type of service bits(Haghighi & Heydari, 2018). Type of service bits are meant to inform on how packets are to be handled in a network. However, the type of service bits are not much used in the real world(Khakurel & Musavian, 2018). DSCP field is a 6-bit field in the IP header and is similar to the IP precedence field hence it is configured in similar ways as the IP precedence(Khadir, Guermouche, Guittoum, & Monteil, 2022).

### **2.11.2 Traffic Rate Management**

The traffic rate management function is used to cater for traffic entering the network belonging to a particular class against the allocated resources. Traffic entering the network needs to be policed to ensure users consume within a given service limit and also avoid congestion(Guo, 2019).

Congestion degrades network performance which makes it impossible to provide QoS. Traffic policing and shaping are the two techniques used for traffic rate management(Zhu et al., 2019). The two techniques differ on how they treat traffic when the link capacity is exhausted(Ppallan et al., 2021). That is, policing drops packets when the link is exhausted while traffic shaping

delays packets and sends them later when capacity is available(Haghighi & Heydari, 2018). Another distinction is that the policing allows for bursts whereas traffic shaping send out packets at a constant rate(Zhao et al., 2018).

Traffic rate management uses a metering mechanism to measure the traffic. Token bucket algorithm is the common mechanism used by both policing and shaping to measure traffic(Sheltami, 2019). Token bucket algorithm determines whether a packet is conforming or non-conforming to the profile configured for it(Haghighi & Heydari, 2018). The token bucket only measure traffic and does not filter, alter or act on the traffic. Depending on whether the packet is conforming or not conforming the algorithm will either transmit or drop the packets(Sun et al., 2020).

### **2.11.3 Resource Allocation**

Resource allocation inside a router is done with the aid of a scheduling algorithm which determines the packet that leaves the queue(Liu, Lu, Xiao, Liu, & Xiong, 2021). How regularly a packet is served defines its resource allocation and bandwidth(Li, Wen, & Luo, 2018). The traditional scheduling algorithm is FIFO however it is not able to offer prioritization therefore not able to implement QOS(Zhu et al., 2019).

Packet dynamics and interconnection between networks of different bandwidth may lead to occasional or constant network congestion(Sheltami, 2019). In networks where there is no congestion any scheduling scheme can work however when there is congestion a scheduling mechanism is required in a

router to determine which packets in a queue are to be serviced(Haghighi & Heydari, 2018).

For a scheduling algorithm to be able to achieve QOS it requires to be able to differentiate packets based on their priority or service level(Dahan, Hindi, Ghoneim, & Als Salman, 2021). Scheduling algorithm should be able to prioritize traffic as well as allocate resources on per flow basis(Zhu et al., 2019). In addition a scheduling algorithm is required to provide performance isolation and fairness among flows. Other requirements that a scheduling algorithm should meet include ease of implementation and flow admission control(Guo, 2019).

## **2.12 Quality of Service Metrics**

Quality of service is a property of a network that enables the (Yan, Zhang, Zhong, Zhang, & Xin, 2022)provision of services to users based on their priority(De Rango & Fazio, 2022). QOS ensures that applications operate within their service level agreement. QOS can also be viewed as the ability of a network to provide performance guarantee in the network for different types of loads in a communication system(Backia, Baskaran, Raja, & Member, 2017). Metrics for measuring QOS include latency, packet loss and jitter. By Measuring jitter, packet loss and latency, it is possible to determine the level of quality of service provided to its users (Datsika et al., 2018). The following sections looks at the QOS metrics in detail.

### **2.12.1 Packet Loss**

Packets sent may fail to reach their intended destination. This situation is called packet loss(Dahan, 2021). Situations that may result in packet loss include

congestion which is caused by an increase in the number of users in the network. Another reason for packet loss is traffic policing. Traffic policing admit packets that conform and drop those that do not conform(Dahan, Binsaeedan, Altaf, Al-Asaly, & Hassan, 2021). Poorly formulated traffic policy rules might result in huge packet losses (Ding, Niu, & Wu, 2018). If by any chance the load of traffic that is generated in the network exceeds the bandwidth ability, policing mechanism will drop the excess traffic(Ezdiani, Nor, & Al-anbuky, 2019).

Packet loss can be calculated using equation 2.1 (Favraud, Chang, & Nikaein, 2018).

$$\text{Packet loss} = \frac{\text{packet sent} - \text{packets recieved}}{\text{packet sent}} \times 100 \quad 2.1$$

Source to destination packet loss is one of the most important QOS performance metrics for many applications such as storage because performance will drop dramatically if the packet loss exceeds a certain limit, and will be rendered unusable if the packet loss is very high(Barzegar & Fatehi, 2022). The standards for packet loss are as illustrated in Table 2.1.

**Table 2.1:Quality standards TiPhone TR 101 329 for Packet Loss**(Source:Favraud et al., 2018)

	<b>Category</b>	<b>Packet loss</b>
<b>Packet Loss standard</b>	Excellent	0%
	Good	3%
	Medium	15%
	Poor	25%

One way of eliminating packet lost is by ensuring there are no devices which are defective that could lead to packet loss( Fang, Qiu, Ding, & Ding, 2018).

### **2.12.2 Latency**

The time delay that a packet experiences as it moves from source to destination is known as latency(Lv, Yi, He, & Zeng, 2022). Variations in the latency can be caused by the quality of the network devices (cable / router /switch), serialization delay, routing and switching latencies, and queuing and buffer management(Hou, Chang, & Yang, 2017). Latency can be categorized as either packetization latency, queuing latency or propogasi all depending on how it is formed. Packetization delay is the delay that is as a consequence of time in the creation of a packet. That is the delay a packet experiences during formation.

Queuing latency is formed inside the router where there are queues for processing packets before routing(Nosheen & Khan, 2021). When a packet experiences delay routing due to queue lengths in the router, the situation is called queuing latency (Xiaoyan Huang, Yang, Member, Wu, & Leng, 2017).

Delay Propogasi is the delay caused by nature of the transmission media(Han, Li, Tang, Huang, & Zhao, 2018). The nature of network equipment's is a major influence to propagation delay. Propagation delay refers to the time a packet needs to move from source to destination at the speed of light( Hou et al., 2017).

In copper and Fibre optic transmission media, the speed of light is reduced(Ademaj & Bernhard, 2022). The reduction in speed associated by the nature of transmission media is referred to as velocity factor (VF).The velocity factor for copper and fiber optic cable are closely the same. Fiber optic cable speed is estimated at 70% of that of the speed of light while that of copper cable is between 40% to 80%(Hassan et al., 2017). Transmission Delay refers to the duration of time a packet takes to move from end to end in a medium. The quantity of data and the speed of the transmission media determines the transmission delay(Jalodia, Taneja, & Davy, 2021). Processing delay is the amount of time required by a router to establish the route for a packet. Table 2.2 outlines the standards for latency(Ding et al., 2018).

Equation 2.2 is used to calculate the value of the delay experienced by a packet(Ferrera & Niguidula, 2017).

$$\text{Packet delay} = \frac{\text{packet length}(\text{bit})}{\text{link bandwidth}(\text{bit/s})} \text{ seconds} \quad 2.2$$

**Table 2.2: Quality Standards ITU-T G.114 for Delay**(Source:Favraud et al., 2018)

<b>Delay (latency) standard</b>	<b>Category</b>	<b>Delay</b>
	Good	0-150 ms
	Medium	150-400ms
	Poor	>400ms

### 2.12.3 Jitter

The delay variations in packet delivery is known as jitter .Jitter can be caused by deviations in traffic flows and the quantity of collisions between packets (congestion) on the network (Ferrera & Niguidula, 2017). Congestion in a network happens when traffic exceeds the existing bandwidth so as to degrade the network performance. Small amounts of jitter do not affect network performance. Table 2.3 outlines the standards for jitter(Favraud et al., 2018). Calculation to find the value of jitter is done using equation 2.3.

$$\text{Jitter} = \frac{\Sigma \text{variation delay}}{\Sigma \text{packets recieved}} \text{seconds} \quad 2.3$$

**Table 2.3: Quality Standards ITU-T G.114 for Jitter**(Source:Favraud et al., 2018)

<b>Jitter standard</b>	<b>Category</b>	<b>Delay</b>
	Good	0 s/d ms
	Medium	20 s/d 50 ms
	Poor	>50 ms

To overcome the adverse effects of jitter a network administrator needs to implement bandwidth management which will ensure excess bandwidth is shared as needed or also increase the amount of bandwidth(Jia, Han, Zhang, Liu, & Shu, 2015).

#### 2.12.4 Throughput

Throughput is the definite bandwidth  $m$  measured in a precise time and in a certain network conditions that are used to transmit files of a certain size. Network throughput refers to summation of speeds of all the data transmitted to all nodes in a network. Throughput standards are as illustrated in Table 2.4 and equation 2.4 is used to calculate the amount of throughput(Xiaoyan Huang et al., 2017).

$$\text{Throughput} = \frac{\Sigma \text{sent data(bits)}}{\text{time data delivery(s)}} \quad 2.4$$

**Table 2.4: Quality Standards ITU-T G.114 for Throughput**  
(Source:Favraud et al., 2018)

<b>Throughput standard</b>	<b>Category</b>	<b>Delay</b>
	Excellent	100%
	Good	75%
	Medium	50%
	Poor	<25%



Throughput describes the real bandwidth at a particular time and on given conditions used to download a file of a certain size(Wang, Member, Li, & Wu, 2018). Factors that influence throughput include the count of network users, network topology ,weather, network devices and electric induction (Zhao, 2018).

To reduce the effects of low throughput network manager need to establish the capability of network equipment's to be used based on their ability to handle the network load(Raschellà et al., 2017). Bandwidth borrowing can also be employed to ensure sharing of bandwidth as may be required(Hassan et al., 2017). The network topology also can be reviewed to get the one that suits the current type of network(Mebarkia & Zsóka, 2019).

### **2.13 Quality of Service Architectures**

Quality of service architectures are schemes for providing quality of service in a network. They include; Integrated Services (IntServe) Architecture, Differentiated Services (DiffServ) Architecture and Multi-Protocol Label Switching(Li & Cao, 2017).

#### **2.13.1 Integrated Services (IntServe) Architecture**

IntServ is designed to provide QOS guarantees by reserving bandwidth before data is transmitted(Li & Cao, 2017). When a flow with a particular QOS requirement arrives at an ingress router, Intserv's Resource Reservation Protocol initiates a path establishment process by sending a path message to the destination router. The destination edge router tries to reserve bandwidth by

sending a reserve (RESV) message back to the ingress router(Samadi et al., 2017).

### 2.13.2 Differentiated Services (DiffServ) Architecture

The differentiated services approach provides a simpler and more scalable QOS by minimizing the amount of storage needed in a router by processing traffic flows in an aggregate manner, moving all the complex procedures from the core to the edge of the network(Ferrera & Niguidula, 2017). One of the main features of a diffserve architecture is the traffic conditioner. As illustrated in Figure 2.7, the traffic conditioner contains four main components that is the marker, dropper, meter and shaper. A meter establishes the traffic profiles to ensure that a certain flow does not exceed its allocated resources. A marker labels a packet in order to help in tracking the packet across the network (Ezdiani et al., 2019). A shaper is responsible for delaying the packets that exceed their allocated bandwidth as they wait for more bandwidth to be available. A dropper is responsible for dropping packets that violates their profiles(Datsika et al., 2018).

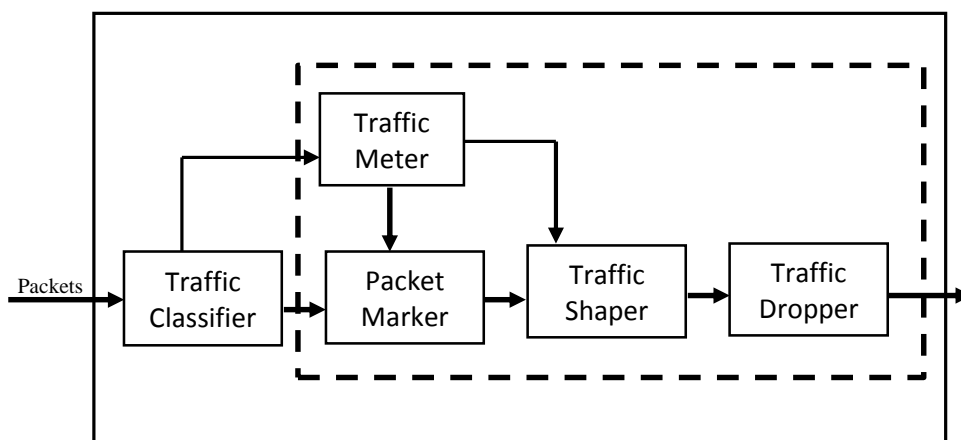


Figure 2.7: Overview of DiffServ operation(Source:Ezdiani et al., 2019)

### **2.13.3 Multi-Protocol Label Switching**

Multi-Protocol Label Switching (MPLS) is a packet forwarding scheme that uses fixed length labels to decide how packets are to be handled. As a packet enters the ingress router (known as a Label Edge Router (LER)) of an MPLS domain, a short fixed-length label is attached (Samadi et al., 2017). As the packet traverses the interior nodes of the MPLS domain, the label rather than the original headers is used to make forwarding decisions (Yang, Li, Liu, & Ma, 2018).

### **2.14 Techniques for Providing QOS in Internet Protocol Networks**

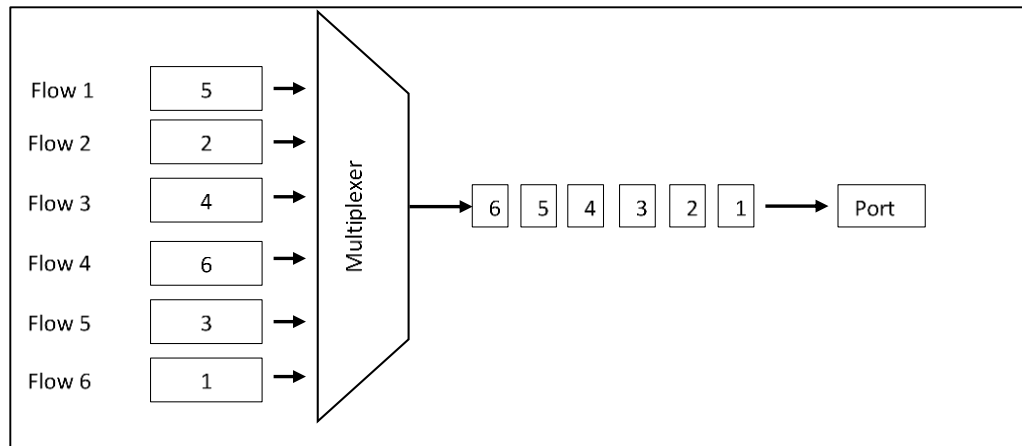
Provision of QOS in IP networks includes various techniques implemented using queuing and scheduling techniques, admission control techniques, and congestion avoidance techniques. The following sub-sections present different techniques whose goal is to ensure that high priority traffic is treated different from less important traffic using either scheduling techniques, bandwidth management techniques or burst handling techniques (Yang et al., 2018).

#### **2.14.1 First in First out Queuing (FIFO)**

First in first out (FIFO) is a very common used queuing technique due to its ease in configuration. Packets belonging to different flows pick up in the FIFO queue and processed in the order of arrival (Backia et al., 2017). FIFO belongs to the unconscious group which treat packets as they are. Packets from all input flows are queued into a memory stack after which they are dequeued in the order of arrival one by one onto the output link. Since FIFO does not perform any reorganization of the queue, there is no schedule overhead experienced by packets (Ding et al., 2018). This means turnaround time, waiting time and

response time for FIFO are low. However due to the absence of prioritization, systems using FIFO experience delays in meeting deadlines (Gulati & Ahmad, 2008). On the other hand lack of prioritization ensures that every process will eventually complete its transmission without the risk of starvation(Huang, 2013). All packets are placed in a single queue and are treated equally(Jiang, 2019).

In FIFO systems packets are allocated bandwidth in their order of arrival and as bandwidth becomes available. Since the queue buffer is finite any packets that cannot be accommodated are dropped. This phenomena is known as tail drop (Noorshams et al., 2013). FIFO works well in links that are not heavily congested. Since FIFO works on first come first serve basis, if a node initiates a large file transfer, it can consume all the bandwidth link to the disadvantage of other traffic(Paulraj & Kannigadevi, 2019). This phenomena is known as packet trains since the source sends a train of packets to its destination and packets from other hosts get caught behind the train(Mebarkia & Zsóka, 2019). To optimize utilization of network resources FIFO implements traffic shaping where traffic is delayed until resources are available avoiding the option of tail drop. In addition FIFO also uses AQM mechanisms to ensure fairness among flows(Raschellà et al., 2017). Figure 2.8 illustrates the functioning of FIFO queuing.



*Figure 2.8: First-In-First-Out (FIFO) queuing (Source: Favraud et al., 2018)*

### **2.14.2 Priority Queuing**

In priority queuing, packets are assigned to classes which are associated with certain priority value. Consequently, packets with high priority are processed first. Priority queue is able to differentiate traffic hence reducing delay of important traffic. On the other hand if there is continuous flow of high priority traffic, low priority will be starved (Fang et al., 2018). In basic implementations of priority queuing, it consists of four priority queues where packets are handled using FIFO. Packets belonging to the highest priority queue are serviced first (Datsika et al., 2018). A packet scheduler is used to check for existence of packets in the highest priority queue after the current packet is processed. Any packets that arrive in the high priority queue are processed immediately. The main benefit of using priority queuing is its ability to guarantee highest priority to storage area networks (especially for read requests) but on the other hand it causes delays to packets belonging to low priority class (Ferrera & Niguidula, 2017).

Priority queuing is most preferred in situations where mission critical traffic requires preference. To ensure smooth transition of high priority packets through the network priority queue uses intermediary network devices for processing packets(Samadi et al., 2017). To provide differentiated services, priority queuing classifies traffic with priority labels low, normal, medium and high. Packets which have not been attached to a class by default are assigned to the normal waiting queue. Data belonging to the high priority queue is handled first followed by that belonging to low priority queues. Due to its static configurations, priority queue is not able to adjust to the network which makes the technique poor in optimal utilization of resources. Figure 2.9 illustrates the functioning of priority queuing. For example if a certain traffic flow is not utilizing its share of bandwidth other flows should be able to borrow the idle bandwidth (Sugeng, Istiyanto, Mustofa, & Ashari, 2019). All incoming queues are assigned to a given network interface with each queue having a priority level(Samadi et al., 2017).

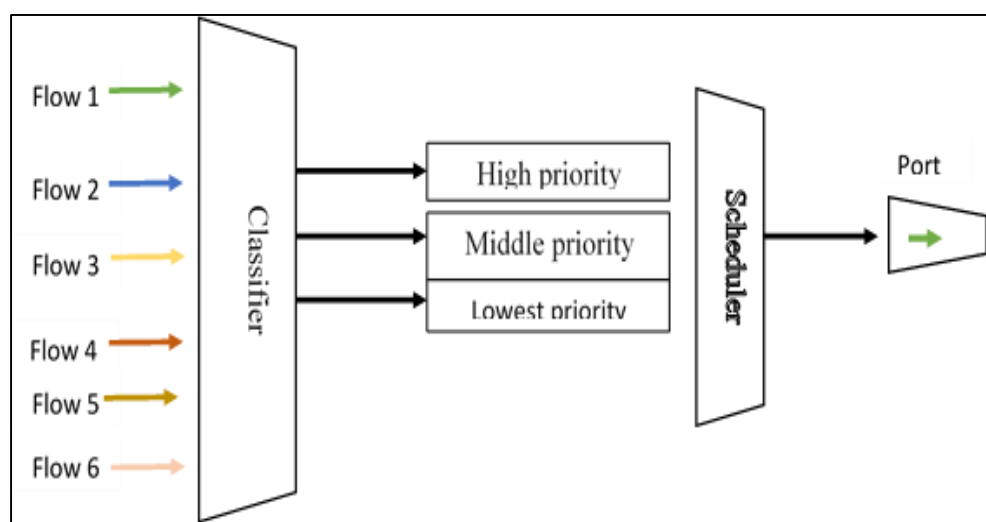


Figure 2.9:Priority Queuing (Source:Favraud et al., 2018)

Every time queues are being sent out of an interface the packets are scanned based on priority with high priority packets being at the head(Nleya & Mutsvangwa, 2018). However priority queuing has a weakness in that it does not automatically adapt to changing network requirements due to static configurations(Kailong et al., 2017). Although priority queuing is simple, in its implementation the low priority traffic may experience more delay and jitter(Zhao, 2018).

### **2.14.3 Class Based Queuing**

Class based queuing schedules packets based on a certain guaranteed transmission rate. If a particular class/queue has got no packets that needs forwarding, its bandwidth is shared among other queues(Backia et al., 2017). Class based queuing mechanism has the ability to cope with different bandwidth requirements since it allocates a specific percentage of the link each class which can be easily adjusted based on the availability of bandwidth(Ezdiani et al., 2019). The sharing of bandwidth based on availability ensures there is fairness in a class(Ferrera & Niguidula, 2017).

To ensure that not a class utilizes more than its fair share of bandwidth, class based queuing allocates a committed rate for each class which can only exceed when other classes are not using their bandwidth(Sugeng et al., 2019). The downside of class based queuing is that it only provides fair allocation if all packets are of the same size. If it happens that some queues contain larger packets, the packets end up being delayed(Samadi et al., 2017). This implies that classes with smaller packet sizes experience shorter delays. In addition class

based queuing does not provide strict priority to the deserving traffic such as mission critical applications(Mebarkia & Zsóka, 2019).

#### **2.14.4 Fair Queuing and Weighted Fair Queuing**

The fair queuing scheduling is a mechanism that classifies and forwards packets according to their configured service agreements. Fair queuing uses a round robin algorithm to allocate bandwidth where every flow has an equal chance(Zhao, 2018). The round robin algorithm ensures that flow from one class does not starve other classes off the bandwidth. The main advantage of priority queuing is that in a situation where there is congestion in a particular class, other classes are not affected and therefore the overall network performance is not affected. The downfall of priority queuing as a scheduling mechanism is that it does put into consideration the packet length(Mebarkia & Zsóka, 2019). This means if a particular class has big flows, then the class may use more bandwidth and therefore take longer to be served. However fair queuing is considered to be best suited in sharing bandwidth among different classes with the same bandwidth requirements(Yang et al., 2018).

In weighted fair queuing, inbound packets are clustered into classes and admitted to separate queues(Mebarkia & Zsóka, 2019). Then these queues are allocated priority based on their weights, with high weights corresponding to high priority. After admission packets are processed based on their weights in a round robin.

For instance if there are weights 1,2 and 3,this means that in the first queue only one packet will be processed, in the second queue two packets and in the third queue three packets. By any chance if the QOS mechanism has not



allocated weights to the classes all the case by default will assume equal weights. In this case we have fair queuing with priority(Samadi et al., 2017). All configurations in weighted fair queuing are automated with no room for tuning possibilities(Wang et al., 2018).

Weighted fair queuing is suitable for environments where there is a need to provide a constant rate of response to users or applications. In its implementations weighted fair queuing uses bitwise fairness where queues are served based on their byte sizes( Zhao, 2018). The performance of weighted fair queuing is better than that of TCP since to a significant extent it reduces the roundtrip time for slow connections by ensuring that the response time of a flow is reduced by a multiple factor (Nleya & Mutsvangwa, 2018). Bandwidth assignments are done based on the weights with each flow getting a maximum length limit(Mebarkia & Zsóka, 2019). Packets are sorted in order of arrival which determines the weights and transmission order. Weighted fair queuing supports variable packet sizes which do not determine the amount of bandwidth allocated to the traffic flows so as to ensure large traffic flows do not have more bandwidth than small traffic flows(Zhao, 2018). By regulating the weights automatically, weighted fair queuing is able to provide data rate guarantees by allocating each flow different bandwidth percentage hence preventing monopolization of the bandwidth by some flows. Figure 2.10 illustrates the functioning of WFQ(Samadi et al., 2017).

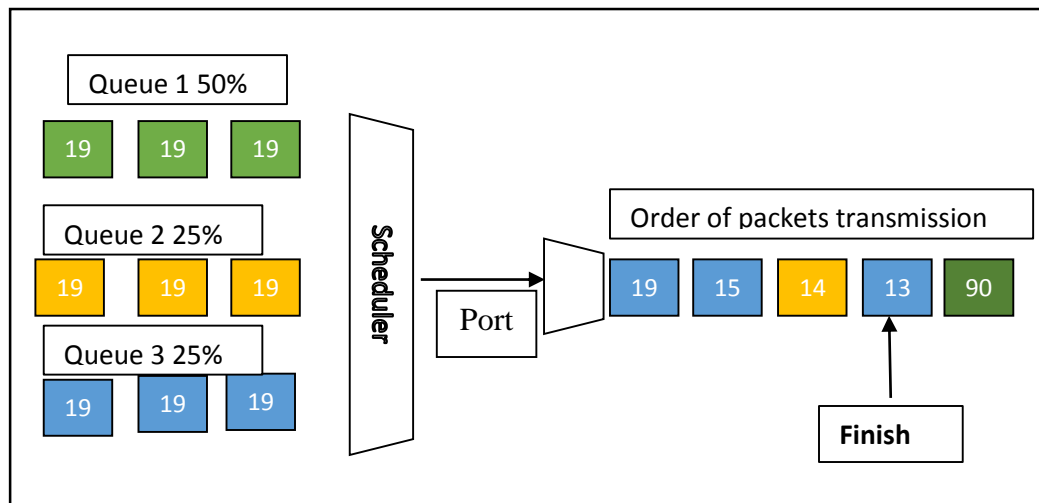


Figure 2.10: Weighted Fair Queuing (WFQ) (Source: Favraud et al., 2018)

#### 2.14.5 Class Based Weighted Fair Queuing

The class based weighted fair queuing scheme traffic is grouped in classes manually. Manual configuration provides flexibility of assigning bandwidth as well as an opportunity for the administrator configure customized classes. This ensures flexibility in allocating a minimum bandwidth amount on the fair queuing basis as well as on the basis of administrator defined classes (Datsika et al., 2018). Each class is allocated a guarantee amount of bandwidth and if there is a class that had no bandwidth allocated it makes use of the spare link bandwidth (Haghighi & Heydari, 2018). Class based weighted fair queuing may lead to situations where low priority flows could overrun the high priority flows, to mitigate this high priority traffic is differentiated in order to give it more preference (Ferrera & Niguidula, 2017). Figure 2.11 illustrates the functioning of the CBQ.

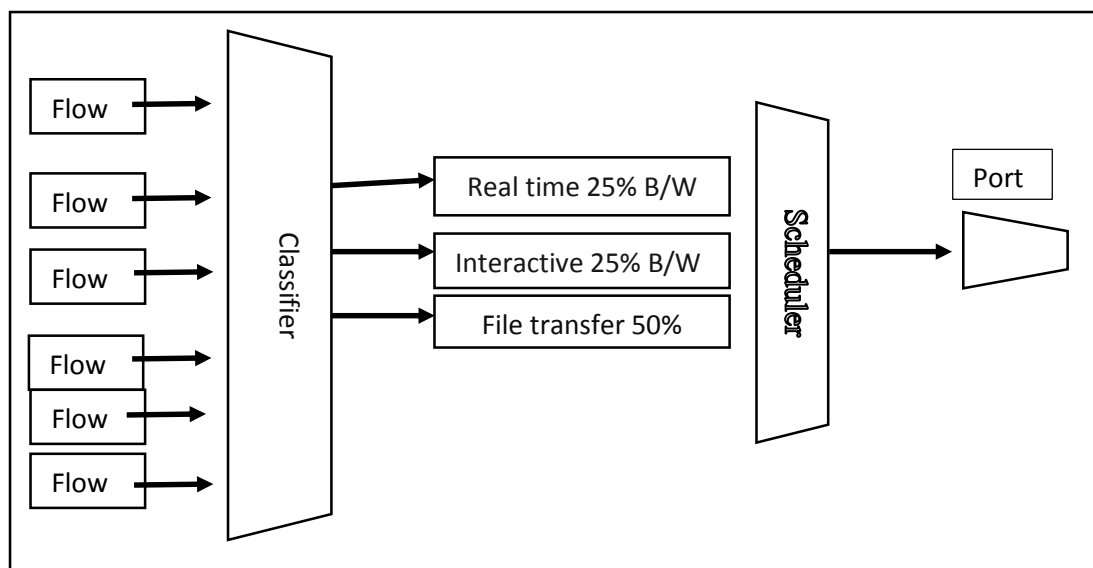


Figure 2.11: Class-based queuing (CBQ) (Source: Favraud et al., 2018)

#### 2.14.6 Custom Queuing (CQ)

In custom queuing flows are categorized into 16 FIFO queues with a defined buffer length. Each of the FIFO queues is then assigned a suitable percentage of the total bandwidth. Scheduling of the queues in the output interface is done in round robin (Ezdiani et al., 2019). However a fine tuning of row lengths can help to reach acceptable results. CQ provides guarantees for mission critical traffic while ensuring that other traffic in the network get predictable throughput (Wang et al., 2018). For instance in case where we have 16 queues, queue 0 by default is configured as a special queue for handling control and keep alive. For the rest of the queues 1 to 15 is used to transmit user traffic and therefore user traffic cannot be transmitted through queue 0 (Ezdiani et al., 2019).

Traffic classification is based on the access control lists for the input interface, packet sizes and the type of application utilizing it. After classification, queues are then served in a round robin manner until a limit threshold is met. Queues are then served in a round robin manner until a byte counter limit threshold is

met after which packet frames from the next queue are serviced (Hemke, Gawande, Gautum, & Email, 2019). Custom queue routers transmit a percentage of the configured traffic in each queue before servicing the next one (Hwang, 2019). During servicing of a particular queue packets are transmitted until a certain byte count limit is reached or until all the packets are transmitted and the queue is empty (Zhang, Lei, Zhang, Guan, & Li, 2019).

#### **2.14.7 Modified Weighted Round Robin and Deficit Weighted Round Robin**

Modified Weighted Round Robin (MWRR) uses a variable sized packets to decide which queue is to be processed (Mary & Jayapriya, 2019). Variable size is computed using the deficit counter based which is always initialized with the value of the queues weight. A packet is transmitted when the deficit counter assumes a value greater than zero (Kulkarni, 2015).

The number of packets,  $N$  to be transmitted can be calculated using the formula in equation 2.5. Where  $w$  is the total packets weight and  $mps$  is the mean packet size (Mary & Jayapriya, 2019).

$$N = \text{Normalize}\left(\frac{w}{mps}\right) \quad 2.5$$

High priority packets are permitted to jump to the front of the queue and the number of packets transmitted is equal to the ratio weight over mean packet size (Park, Kim, Jeong, Hong, & Kang, 2018). Packets are served from the head of the queue and if the modified counter is greater than the size of the packet (Li & Cao, 2017).

Deficit Weighted Round Robin (DWRR) was proposed by Shreedher and Varghese in 1995. DWRR services packets without the consideration for the mean size but the difference between packet size and packet length(Kailong et al., 2017). DWRR uses scholachastic fair queuing to assign data flows to queues(Nleya & Mutsvangwa, 2018). Queues are attended to in a round robin based on the quantum of service committed to each queue. If a queue is unable to transmit its packets due the size, the difference from the earlier quantum is added to the quantum for the next round(Simiscuka, 2017).

Since queues are serviced in a round robin, if a queue fails to get a turn in a given round it is recompensed in the following round. However once a flow is serviced it must wait for n-1 other flows to be serviced before it is serviced again. During each round, a flow transmits its entire quantum data once, as a result DRR has poor delay(Wang et al., 2018). Each queue of traffic is linked to a quantum and deficit counter. A deficit counter is initialized to zero and is used to store the credit of sending data for each queue and on the other hand the quantum represents the amount of data in bytes that each queue is able to transmit when its turn arrives(Ezdiani et al., 2019).

For packets to be served their deficit counter must be greater than zero. After a queue is served a deficit counter is reduced by a value equivalent to the size of packets sent until the counter is zero or negative after which the queue is no longer served (Yang et al., 2018). For each particular round of the deficit counter each non-empty queue is decreased(James & Shaikh, 2019).

In Modified Deficit Round Robin (MDRR), after a queue is served a specific amount of data is dequeued. Then the algorithm proceeds to service the next queue, if the amount transmitted exceeds the value allocated for a certain queue, then in the next round less data is transmitted for that queue (Guck, Bemten, Kellerer, & Member, 2019). As a result, MDRR is able to ensure that the average amount of data dequeued per queue nears the configured value (Ferrera & Niguidula, 2017).

#### **2.14.8 Hybrid Waiting Queues**

Combining different mechanisms enables the combined mechanism to have the positive qualities of all the aggregated techniques. However this combination also includes the weaknesses of individual mechanisms and overhead in memory when processing them. Overhead in memory is brought about by the fact that each memory for a given interface is associated with a given amount of latency for traffic that is transmitted through the interface (Ding et al., 2018). The more interfaces that traffic needs to go through the more the delay which is detrimental to applications such as storage area networks read requests (Zhao, 2018). To avert excessive delays, a compromise is made between the size, length and number of interfaces that data needs to travel through. Small buffer may cause data spillage while big buffer may cause huge delays a phenomena known as jitter effect (Favraud et al., 2018).

#### **2.14.9 Custom Class Based Weighted Fair Queuing and Priority Class Based Weighted Fair Queuing**

Custom Class Based Weighted Fair Queuing and Priority Class Based Weighted Fair Queuing is a combination of custom queuing and class based weighted

fair queuing where Custom Queuing(CQ) is responsible for bandwidth management to avoid congestion(Nleya & Mutsvangwa, 2018). After the bandwidth management the packets are sent out of the CQ interface to the class based weighted fair queuing input interface(Zhao, 2018). At the Class based weighted fair queuing packets are put into classes(Raschellà et al., 2017). With this method it is possible to reduce the delays within the network, which is not the case with ordinary CQ scheme(Mebarkia & Zsóka, 2019).

After the CQ and based weighted fair queuing, next the packets are assigned priority based on their service level agreements(Dong, Xie, Tang, Zhong, & Vasilakos, 2019). High priority packets that move out the priority queuing algorithm interface are served fast and since they were already classified and assigned bandwidth, there is no contention for bandwidth at this time therefore the high priority packets are transmitted faster independent of other flows(Zhao, 2018).

#### **2.14.10 Weighted Fair Queuing and Class Based Weighted Fair Queuing**

Weighted fair queuing is employed being the first to ensure fairness through restricting changes to the throughput for all applications(Mebarkia & Zsóka, 2019). After weighted fair queuing packets proceed to the class based weighted fair queuing algorithm where packets are assigned to classes based on network administrators specifications(Raschellà et al., 2017). In this way every high priority application gets its desired bandwidth and the remainder is shared among all active applications(Dong et al., 2019).

Weighted fair queuing is more effective when used in environments where priority is configured based on IP address(Samadi et al., 2017). Due to its capability to manage round trip delays, weighted fair queuing is combined with class based weighted fair queuing as the best solution to reduce Ethernet delays(Fang et al., 2018).

### **2.15 Admission Control for QOS**

Admission control is used to provide robust performance by limiting the number of sessions to join the network(Narale, 2019). The main purpose of admission control is to provide strong performance. This is to ensure that existing sessions are not degraded and new sessions are provided with QOS(Ramadan, 2017). If admission of new requests leads to poor performance, new sessions are rejected or the user is notified that the network cannot offer the configured reserved resources for a particular session(Narale, 2019). The decision to reject or admit can be made based on the existing resources such as bandwidth and quantity active transmissions. Admission control can be implemented either through explicit control or implicit control(Topalova, 2018). In explicit control resources are reserved unreservedly with applications sending requests to join the network via the resource reservation signaling mechanism. The admission control algorithm implementation depends on the network architecture. In networks where there are so many interconnected routers the algorithm can be located in each route.

Admission control helps ensure that classes are offered differentiated services according to the priority attached to the class and also that no particular class consumes more than its share of bandwidth. Admission control classifies traffic



based on a given service level agreement and the bandwidth allocated to that class of traffic(Mamman & Hanapi, 2017). To achieve differentiated service among classes, admission control uses capacity based admission control to control how classes of traffic are admitted into the network. Admission into the network is based on first in first serve criteria without any additional checks. In a case where all the bandwidth is exhausted no more flows are admitted until some bandwidth is available. To implement priority, admission control allows flows to preempt other flows based on their priority for some additional bandwidth. This feature ensures that low priority traffic is able to release its bandwidth when required by the high priority traffic ensuring quality of service for high priority flows (Shankaraiah & Venkataram, 2010).

However Kashihara and Tsurusawa (2010) believe that if flow admission of high priority traffic is not checked then low priority traffic will end up being starved. After traffic has been admitted into the network through admission control techniques, traffic shaping mechanisms are used to control the amount of traffic that circulates the local area network. This is accomplished by smoothing traffic based on the configured policy file(Wu, & Li, 2018).

In a network there are two categories of algorithms used to implement admission control namely Measurement Based Admission Control and Parameter Based Admission Control (Ojijo & Falowo, 2020)

### **2.15.1 Measurement Based Admission Control**

Measurement based admission control algorithms use the network statistics to make admission decisions( Zhang, Li, Li, & Zhao, 2019). They provide an opportunity to offer QOS to priori data flows. The measurement based

admission control algorithms provide for high link utilization since they are adaptive. The role of traffic grouping and characterization is shifted from the user to the network where traffic is characterized based on existing network conditions. Measurement based admission control has the following advantages. One it does not lead to overall allocation of resources as resources are assigned based on network statistics. Secondly QOS decisions are made based on aggregate behavior of flows instead individual of flows which is difficult to determine(Fang, Shen, Huang, & Feng, 2021).

However relying on measured quantities for making admission control decision raises a number of issues. These issues include estimation error, dynamics and separation of time scales and memory(Ojijo & Falowo, 2020). Firstly the reliant on estimation to make decisions creates a lot of uncertainty. Since inaccurate estimates may lead to bad admission decisions. Secondly since the flows of arrivals and departure vary with time, the effect of flow arrivals and departures effect on QOS arises. To improve on the accuracy of estimation there is need to know about flow history. In this case a big window memory is required which might remove dynamism from the algorithm. This results in a challenge of determining the appropriate memory size(Wang, Kang, Liu, Ma, & Li, 2020).

### **2.15.2 Parameter Based Admission Control**

Parameter based admission control algorithm makes bandwidth estimates based on worst case scenario that is where existing flows are sending at their peak rate(Alvarez et al., 2020). In this case there is low utilization of network bandwidth in cases where the flows are sending less than their peak rates.

Furthermore flows may be denied admission even though the current network condition allows it(Fang et al., 2021).

## **2.16 Admission Control Algorithms**

Admission control algorithms determine if a packet is to be admitted, delayed or dropped. These algorithms include simple sum algorithm, measured sum, acceptance region, equivalent bandwidth algorithm and end point admission control( Wang et al., 2020).

### **2.16.1 Simple Sum**

The simple sum algorithm is meant to make sure that the requested bandwidth does not exceed the available bandwidth. The simple sum algorithm is the simplest of the algorithm and therefore widely implemented in most switches and routers(Ojijo & Falowo, 2020). To reduce on the queuing delay, the weighted fair queuing (WFQ) algorithm is used to achieve performance isolation by grouping flows into queues and availing their reserved rate(Alvarez et al., 2020).

### **2.16.2 Measured Sum**

The measured sum tries to increase the network utilization by using the measured load of each flow and assigning bandwidth based on these measurements(Zhang et al., 2019). That is the reserved rates of flows are substituted with the measured rates. The only QOS metric used to determine the admission decision is bandwidth. The measured sum technique is bound to fail if delay variations are huge especially in high link utilization cases(Vincenzi, Lopez-Aguilera, & Garcia-Villegas, 2021).

### **2.16.3 Acceptance Region**

The acceptance region algorithm makes admission decision based on the information that either the system fall in the accepted region or rejected region. The calculation of acceptance region is made based on peak and mean rate. The acceptance region algorithms are simple however this simplicity results in simplification of network model which results in limitation of such algorithm(Wang et al., 2020).

### **2.16.4 Equivalent Bandwidth Algorithm**

Equivalent bandwidth is the least amount of bandwidth necessary for transmission of traffic generated by a source without QOS violations(Fang et al., 2021). Each source of flow is allocated an equivalent bandwidth and new flows are accepted if the sum of the allocate bandwidth are less than the available link capacity. The equivalence bandwidth algorithm is simple as it boils down to comparing the sum equivalents to the total available capacity. The reservation aspect of the equivalent bandwidth may lead to low network utilization since a traffic source to request more than it can utilize(Vincenzi et al., 2021).

### **2.16.5 End Point Admission Control**

In the end point admission control the end host sends packet probes on the data rate it would like to be reserved and notes the experienced packet loss(Ojijo & Falowo, 2020). Then flows are admitted if the packet loss is at a particular threshold. The end point admission control does not utilize information that is kept by router which does not keep per flow state or does not process reservation

requests(Fang et al., 2021). Probe packets can be treated as data packets or they can be assigned higher priority. The end point admission control suffers from the limitation that estimates may not be in line with what is observed(Zhang et al., 2019).

## **2.17 Congestion Avoidance Mechanisms for QOS**

Congestion avoidance refers to a group of mechanisms used to control the congestion and keep network load lower than the cap overall network capacity(Kotian, Shetty, & Begum, 2017). Congestion avoidance is required to regulate traffic injection into a network to avoid network saturation, which may lead to performance penalty(Topalova, 2018). In networks with QOS guarantees, congestion control mechanisms first attempt to regulate best-effort and misbehaving real-time traffic, and if required, then traffic from other service classes. The two main techniques used for congestion avoidance include Random Early Detection mechanisms and Weighted Random early detection mechanism(Baklizi & Ababneh, 2016).

### **2.17.1 Random Early Detection (RED)**

Random Early Detection (RED) uses TCP's based congestion avoidance mechanism is whereby if there is congestion in the network, RED algorithm drops packets and informs the source to stop transmitting(Misra, Oommen, Yanamandra, & Obaidat, 2019). In TCP environment the source reduces its rate of transmission until all packets reach their destination an indication that congestion is over. However, without RED which implements early detection all excess packets would be dropped phenomena known as tail drop due to the overflow of output buffers. Therefore the implementation of RED in any

network reduces the chances of tail drop by dropping packets selectively after any indication of congestion (Baklizi & Ababneh, 2016).

By dropping packets early RED eradicates dropping huge number of packets and decreases the likelihood of any global synchronization (Jamali, Alipasandi, & Alipasandi, 2019). Thus, RED ensures maximum utilization of the transmission line at all times. In addition RED drops packets from large users' sources than from small source users, therefore only those sources that transmit a lot of information are slowed down (Kalav & Gupta, 2019). RED has the advantage of managing congestion before it reaches a critical point as well as reducing delay by keeping the size of the queue for the packets not dropped small (Misra et al., 2019). Through TCP Synchronization avoidance mechanism RED reduces global instability in the network since many queues don't signal their source to decrease their window at the same time. In addition RED ensures fairness for both smooth and burst traffic since bursty traffic does not suffer extreme packet loss due to early detection (Sharma & Behera, 2017).

### **2.17.2 Weighted RED**

Weighted RED (WRED) uses priority to drop packets with high priority packets having a lower probability of being dropped (Topalova, 2018). However in order to achieve non-weighted behavior RED can be configured to ignore weights (Alkharasani, Othman, Abdullah, & Lun, 2017). WRED is best suited on any output interface where congestion is likely to occur for example in core routers (Jamali et al., 2019).

## **2.18 Packet Classification**

When there is a mixture of network traffic it is desirable to maintain performance isolation among network functions(Bangquan & Xiong, 2019). Performance isolation is the property of a network where a certain class of users should not impact the performance of others. In computer networks performance isolation is achieved through traffic classification(Chin, Xiong, & Hu, 2018).

Traffic classification is mainly used for two purposes that is the provision of quality of service as well as lawful interception(Zhigang Liu, 2019). In most networks, existing applications require different QOS and therefore it's important to offer differentiated quality of service for each type of application(Lopez-martin, Member, & Carro, 2017). There are several techniques available for traffic classification including use of IP address(Wang, Chen, Ye, & Sun, 2019). IP address traffic classification can be achieved through techniques such as by use of ports, deep packet inspection and classification based on statistical features(Kumar, Kim, & Suh, 2015).

### **2.18.1 Port-Based Approach**

Port based classification is one of the earliest technique of classifying traffic in a network. Port numbers can be used to identify the application transmitting traffic and are registered by the Internet Assigned Number Authority (IANA) (Lopez-martin et al., 2017). Although the use of port numbers for classifying traffic is simple and fast, its performance is poor. There are two types of ports

used in classification that is registered ports and dynamic or private (Sadiq et al., 2018).

Every TCP connection begins with a handshake and the port number associated with a certain packet is indicated in the header of the packet and during the entire period of communication the source and destination use the pair of ports indicated in the header (Shen, Xia, Zhang, & Jia, 2017). To match traffic to a certain port, a search from the lists of registered ports is done, the search creates an overhead which reduces performance (Wang et al., 2019).

Another disadvantage associated with port number classification is that applications using the same port number may require different QoS requirements which creates a challenge for using port numbers for QoS (Kumar et al., 2015). In addition encryption method used may bar the extraction of port numbers (Wang & Ye, 2018). The demerits associated with port based led to the development of classification technique based on deep packet inspection (Alkharasani et al., 2017).

### **2.18.2 Deep Packet Inspection**

To mitigate against the problems associated with port based classification, deep packet inspection is used. Deep packet inspection uses the session layer and application layer information to make its classification (Zeng & Gu, 2019). DPI utilizes more than just packet headers and port numbers to classify packets (Mamman & Hanapi, 2017). DPI uses a number of techniques to classify traffic including scanning for specific strings in the packets. DPI analyzes only few packets for each flow to make classification decision. However this



introduces processing overhead and it largely depends on the precision of the DPI systems in use(Wu et al., 2017). On the other hand this approach prevents a situation where all the bandwidth hungry applications are blocked of which some may be useful in the network(He et al., 2017).

The port based approach and deep packet inspection are used to classify peer to peer applications and for intrusion detection(Wang & Ye, 2018). Packets that have the same port and the same source and destination address are put on the same class. Deep packet inspection is unable to classify encrypted packets. However, it may leak the privacy of the data in some way(Wang et al., 2019).

### **2.18.3 Statistical Signature Based Classification**

In statistical signature based classification protocol, fingerprint values such as packet length and the duration between packet arrival time are used for traffic classification(Kumar, 2014). The packet length and inter arrival time in statistical signature based classification are used to establish the behavior protocols as compared to other classification techniques that are used to establish the behavior of application that generates the packets (Zeng & Gu, 2019).

### **2.18.4 IP Address Based Classification**

The network type influences the method used for differentiating between traffic classes and providing differentiated services for each class. In IP address based traffic classification, traffic is put into classes based on source and destination address (Fang et al., 2019). There are several parameters which can be configured by the administrator, however, based on integrated services

philosophy, the administrator simply defines, specifies and groups services into classes or levels which will have different treatment. Selected configurations are saved and retrieved to and from a database and scripts are generated and executed internally and transparently(Mary & Jayapriya, 2019). In each of the interfaces, traffic is classified based on the services and classes defined for that interface. For each defined service, an IP tables rule is introduced, to classify and to mark all traffic that follows the pattern identifying that service(Kulkarni, 2015).

All services grouped in the same class are marked with the same specific number. This number will be used by traffic conditioning module to give packets from each class different treatment(Hwang, 2019). Concerning traffic conditioning, a Class-Based-Queuing (Hierarchical Token Bucket queuing discipline) packet scheduler is used because its hierarchical approach is appropriate for setups where a static amount of bandwidth is shared among different users with the option of stipulating the amount of bandwidth that can be borrowed(Yang, Liu, Ranjan, Shih, & Lin, 2013).

## **2.19 QOS for Storage Area Networks**

The following sections have reviewed some of the solutions for implementing QOS in IP SANs. Included in the review is their strengths and weaknesses from which the features of the proposed solution are derived.

### **2.19.1 Stonehenge**

Quality of service is essential in mixed environment where various users with different levels of priorities and preferences are accessing the storage systems

simultaneously(Ghezzi et al., 2019). It is critical to guarantee that critical tasks get satisfying performance given limited resources. Lan (2005) developed Stonehenge to solve the issues of storage scalability, manageability and quality of service. Stonehenge is built on IP networks IDE hard drives, IDE controllers and off-the shelf low end personal computers( Han et al., 2019). To implement QOS Stonehenge dedicates a set of storage servers to manage disk arrays and single personal computers to perform the controlling functions such as storage reservation and run-time management (Lan, 2005).

### **2.19.2 PClock**

PClock was developed by Ajay, Arifmerchant and Peter(2013) which uses packets onset curves to indicate bandwidth and burst requirements of applications(Shen et al., 2017). When implemented PClock exhibited efficiency in performance isolation as well as burst handling. It also is able to allocate spare capacity to the applications needing it in order to speed up communications to the applications. When a request arrives the PClock algorithm performs three functions; updating the number of tokens, checking and adjusting tags and computing the tags(Yu, Guo, Liu, Zheng, & Zong, 2018).

The update number of tokens function updates the arrival upper bound function for the present arrival time while the check adjust tags is used to resynchronize flows to avoid starvation and the compute tags assigns start and finish tags. The PClock algorithm allows multiple workloads to share storage, with each workload receiving the level of service it requires(Fahad, Alharthi, Tari,

Almalawi, & Khalil, 2019). PClock allows each workload to specify its throughput, burst size and desired latency (Ajay, Arifmerchant & peter, 2013).

The PClock algorithm is as follows:

Packets arrival

1. Request arrival:
2. Let  $t$  be arrival time of request  $r$  from  $f_i$ ;
3. Update Numtokens();
4. CheckandAdjustTags();
5. ComputeTags();

Packets scheduling

1. Request scheduling:
2. Choose the request  $w$  with minimum finish tag  $f_j^w$  and dispatch to the server
3. Let the selected request belong to flow  $f_k$  with start tag  $s_k^w$ ;
4.  $\text{Mins}_k = s_k$ ; (Ajay, Arifmerchant & peter, 2013).

In order to assign tags the arrival upper bound function  $U_i^a()$  to the current time  $t$ . It maintains a variable numtokens for each flow  $f_i$ .

PClock guarantees that the well behaved flows are not missed and the requests of the background jobs are done in batches, which can lead to better disk utilization since many background jobs tend to be sequential (Shen et al., 2017).

The algorithm is able to redistribute spare capacity to workloads and background jobs that need it. The algorithm is also lightweight to implement and efficient to execute. However it does not offer control of how QOS mechanisms interact with storage devices (Ajay, Arifmerchant & peter, 2013).

### 2.19.3 Argon

The argon storage server ensures management of the inter service disk as well as caches to ensure efficiency (Xiaoyan Huang et al., 2017). The argon algorithm aims at providing each traffic flow with a fraction throughput associated with the flow when it has the server to itself. Argon implements routinely configured prefetch/write back sizes to protect streaming efficiency from disk seeks caused by competing workloads. Argon uses prefetching and write back aggregation as a tool for performance insulation (Matthew et al., 2007).

Argon adapts, extends and applies some existing mechanisms to provide performance isolation for pooled storage servers. Many operating systems such as eclipse operating system use time slicing of disk head time to achieve performance insulation. Argon goes beyond this approach by automatically determining the lengths of time slices required and by adding appropriate and automatically configured cache partitioning and prefetch/write back (Matthew et al., 2007).

Argon uses QOS aware disk scheduler in place of strict time slicing, for workloads whose access patterns would not interfere when combined. To implement fairness or weighted fair sharing between workloads argon uses amortization cache partitioning and quanta based scheduling (Zuberek & Strzeciwilk, 2018). Argon assumes that network bandwidth and CPU time has no effect on efficiency. To achieve complete isolation argon does not allow requests from different workloads to be mixed, instead it uses a strict quanta

based scheduling. This ensures that each client gets exclusive access to the disk during a scheduling quantum which avoids starvation because active client's quanta are scheduled in a round robin manner (Matthew et al., 2007).

Traditional disk and cache management allow interference among services access patterns to significantly reduce efficiency (Gémieux et al., 2018). Argon combines and automatically configures prefetch/write back cache partitioning and quanta based disk time scheduling to provide each service with a configurable fraction of efficiency it would receive without competition. This increases both efficiency and predictability when services share storage server (Matthew et al., 2007).

However as with all other storage specific solutions Argon runs on the storage device itself which requires multiple instances of it to be implemented in all the devices (Lumb et al., 2003). This increases overhead and CPU time. Again since there is no centralized management of QOS when the storage data is in transit from the source to destination QOS is not taken care of (Lichtblau & Streibelt, 2017). The argon design also assumes that bandwidth is not a factor in QOS however with IP SANs bandwidth management is very important since the storage data will be moving from source to destination via IP network (Bjorgeen & Haugerud, 2010).

#### **2.19.4 Facade**

Christopher, Arif and Guillermo (2003) developed Façade as a dynamic storage controller for controlling multiple input/output streams going to a shared storage device and to ensure that each of the input/output streams receives a

performance specified by its service level objective. Façade provides performance guarantees in highly volatile scenario. To achieve QOS Façade is implemented as a virtual store controller that is placed between hosts and storage devices in the network, and throttles individual input/output requests from multiple clients so that devices do not saturate. Figure 2.12 illustrates the structure of Facade(Zuberek & Strzeciwilk, 2018).

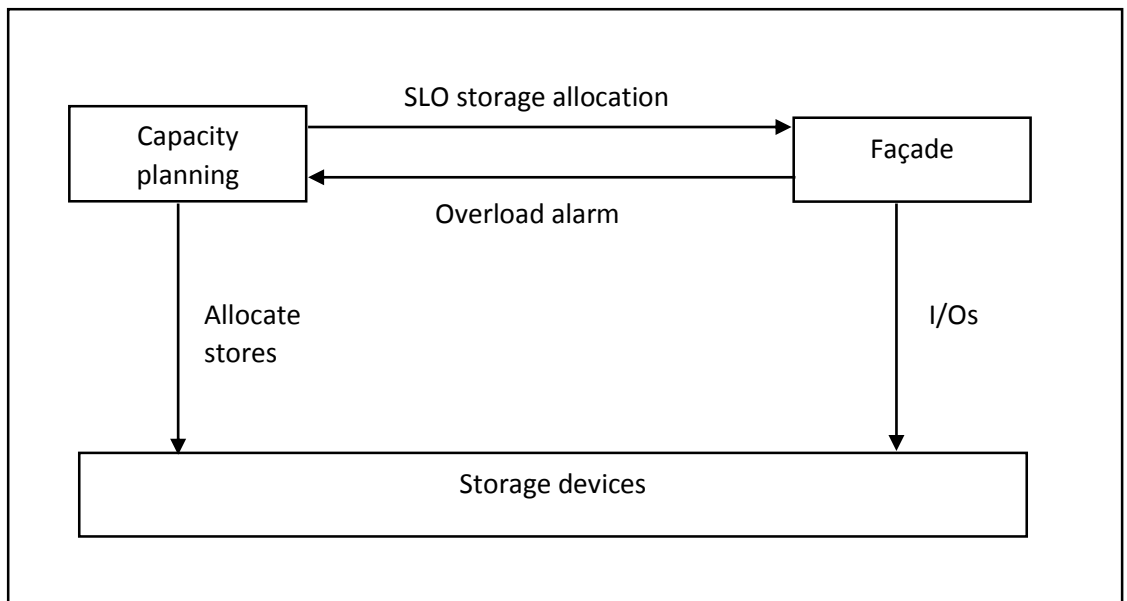


Figure 2.12: Facade Structure (Source: Christopher, Arif & Guillermo, 2003)

The capacity planner allocates storage for each workload on the storage device and ensures that the device has adequate capacity and bandwidth to meet the aggregate demands of the workloads assigned to it(Nam et al., 2004). The allocation is adjusted depending on the workload. Requests arriving at façade are queued in per workload input queues. To determine which requests are admitted to the storage devices façade relies on three components that is the I/O scheduler, statistics monitor and controller (Christopher, Arif & Guillermo, 2003).

The I/O scheduler maintains a target queue depth value and per workload latency target which it tries to meet using earliest deadline first (EDF) scheduling(Jamaluddin, 2019). The deadline for a request from a workload  $W_K$  is arrival Time ( $W_K$ ) + latencyTarget ( $W_k$ ), where arrival Time ( $W_K$ ) is its arrival Time and latency Target ( $W_K$ ) is a target supplied for  $W_K$  by the controller. Requests are admitted into the devices in two cases; if the device queue depth is now less than the current queue length target or if the deadline for any workload is already past( Lim et al., 2017). The intent of controlling queue depth is to allow workloads with low latency requirements to satisfy their SLOs (Christopher, Arif & Guillermo, 2003).

The Façade statistics monitor receives I/O arrivals and completions(Sheltami, 2019). It reports the completions to the I/O scheduler and also computes the average latency and read and write request arrival rates for active workloads every  $P$  seconds and reports them to the controller (Christopher, Arif & Guillermo, 2003).

The controller adjusts the target workload latencies and the target device queue length(Chin et al., 2018).Target workload latencies must be adjusted because the workload request rates vary and therefore it is necessary to give those requests a different latency based on the workload SLO. The device queue depth must also be adjusted to meet the varying workload requirements( Gu et al., 2018).The controller tries to keep the queue as full as possible to enhance device utilization. However this surges the latency. This means when any Workload demands a low latency, the controller reduces the target queue



depth(Xiaoyan Huang et al., 2017). The controller uses the I/O statistics it receives from the monitor every  $P$  seconds to compute a new latency target based on the SLO for each workload as follows;

Let the SLO for  $WK$  be  $((r, tr1, tw1), (r_2, tr2, tw2), \dots, (r_n, tr_n, tw_n))$  with a window  $w$  and the fraction of reads reported is as illustrated by equations 2.6 and 2.7.

$$\text{Let } r_0=0, r_{n+1}=\infty, tr_{n+1}=tw_{n+1}=\infty \quad 2.6$$

$$\text{Then latency Target } (W_K) = tr_i f_n + tw_i (1-f_r) \quad 2.7$$

If  $r_{i-1} \leq \text{read Rate } (W_K) + \text{write Rate } (W_K) < r_i$ .

Facade is able to efficiently utilize resources and balance the load among multiple backend devices while satisfying the performance requirement of many different client applications(Lumb et al., 2003). Facade is also able to adopt to workloads whose performance requirements change overtime. However façade cannot handle large workloads. This is because multiple instance of façade that are in every storage device cannot be able to cooperate in order to handle large workloads (Christopher, Arif & Guillermo, 2003).

### **2.19.5 Proportional Allocation of Resources for Distributed Storage**

#### **Access (PARDA)**

Proportional Allocation of Resources for Distributed Storage Access is a mechanism that ensures proportional share fairness between distributed hosts accessing a storage array without assuming any support from the array itself(Gulati & Waldspurger,2014). PARDA uses latency measurements to

detect overload and adjust issue queue lengths to provide fairness(Gulati & Waldspurger,2014). Many algorithms used for network QOS have been suggested, including many variations of the fair queuing technique(Lopez-martin et al., 2017). However these approaches are appropriate only in integrated setting where a one controller serves all requests for resources(Gulati & Waldspurger,2014).

The PARDA algorithm uses mean IO latency calculated over a defined period to detect overload and adapts the hosts issue queue (i.e. window size) length in response. Each host executes a different copy of the PARDA algorithm(Wang & Ye, 2018). The PARDA algorithm consists of two mechanisms that is latency estimation and window size calculation (Ajay, Irfan & Carl, 2014).

Latency estimation is derived from an exponentially-weighted moving average of IO latency at time denoted by  $L(t)$  which is maintained by each host.  $L(t)$  is used to smooth out short term variations. The weight is ascertained by smoothing parameter  $\alpha \in [0,1]$ . For a new latency observation equation 2.13 is used;

$$L(t) = (1 - \alpha) X l + \alpha X L(t - 1) \quad 2.8$$

The window size is calculated as in equation 2.14

$$w(t + 1) = (1 - \gamma)w(t - 1) + \gamma\left(\frac{\mathcal{L}}{L(t)} + \beta\right) \quad 2.9$$

From the above  $w(t)$  refers to the window size at time  $t$ ,  $\gamma \in [0,1]$  is the smoothing parameter, while  $\mathcal{L}$  denotes system wide latency threshold and  $\beta$  represents IO allocation shares per host(Alkharasani et al., 2017).

When the average latency  $L > \mathcal{L}$ , PARDA increases the window size. To prevent extreme behavior from the control algorithm  $w(t)$  is bounded by  $[W_{min}, W_{max}]$ .

The lower bound  $W_{min}$  ensures that no host experience starvation due to very few IO shares. The upper bound  $W_{max}$  reduces the chances of having very long queues reducing the latency experienced by hosts that begin issuing requests after a long period of inactivity(Martins & Zucch, 2019). A fair upper bound is derived based on queue length values as well as array configuration and number of hosts.

The latency threshold  $\mathcal{L}$  represents the acceptable response time and the control algorithm seeks to keep the overall cluster latency near to this value(Wang & Wang, 2013). Tests done by Ajay et al., 2014 confirmed that increasing the length of the queue beyond a certain point does not increase throughput. This means that  $\mathcal{L}$  can be configured to a value which is high enough to ensure that a high number of requests is maintained at the array(Nunome, 2014).

Alternatively administrators can specify  $\mathcal{L}$  based on requirements such as support for latency sensitive applications.

The parameter  $\beta$  is configured considering the IO shares associated with the host. Ajay et al., (2014) highlighted the two properties of the control equation based on formal model of proofs of FAST TCP. When the throughput equilibrium for host  $i$  is proportional to  $\frac{\beta i}{q_i}$ , where  $\beta i$  represents the per host share parameter and  $q_i$  is the queuing delay experienced by the host(Hwang, 2019). For a particular array with the capacity of  $C$  and latency threshold  $\mathcal{L}$  the window size at equilibrium is given by equation 2.15.

$$w_i = \beta_i + \beta_i \frac{c_i}{\sum_{j \in \mathcal{I}} \beta_j} \quad 2.10$$

To improve overall IO performance the scheduler implements borrowing of the IO shares from those which are not consuming their full allocation. In addition the scheduler does not switch the VMs after every IOs per VM as long as they demonstrate some spatial locality(Ou, Hwang, Chen, & Wang, 2015). Table 2.5 illustrates the summary of features contained in each of the discussed solutions as well as the features of the proposed solution.

**Table 2.5: Comparison of Storage Specific QOS Solutions**

<b>SOLUTION</b>	<b>Burst handling</b>	<b>Performance isolation</b>	<b>Bandwidth Sharing</b>	<b>Centralized Management of QOS.</b>
<b>STONEHENGE</b>	Absent	Present	Absent	Absent
<b>PCLOCK</b>	Present	Present	Present	Absent
<b>ARGON</b>	Absent	Present	Absent	Absent
<b>FACADE</b>	Absent	Present	Present	Absent
<b>PARDA</b>	Present	Present	Present	Absent
<b>PROPOSED</b>	Present	Present	Present	Present

## 2.20 QOS Optimization Theories

The following sections discusses the main theories used in the optimization design of performance isolation, bandwidth management and burst handling.

### 2.20.1 Theory of Effective Bandwidth

One of the main characteristic of IPSANs is that there is a mixture of traffic originating from different classes of users. Since most of the flows consist of variable bit rate (VBR), this means the bandwidth requirements fluctuates from some minimal level to a peak rate subject to the total bandwidth available(Hirose & Cappellaro, 2018). If bandwidth is assigned based on the

peak rate, then there is a chance of bandwidth wastage as a given class of users may not send flows equal to the peak rate always. On the other hand if bandwidth is assigned based on the mean rate, then the SLO of a given class of users may be violated as occasionally it will be sending at peak rate. The issue is to determine the effective amount of bandwidth a certain class of users gets without violating their SLO(Bassi et al., 2020).

The theory of effective bandwidth states that, the effective bandwidth of a time varying source is the minimum amount of bandwidth required to satisfy its QOS. The effective bandwidth theory answers the question of how much bandwidth of a given channel should be available for a given class of user to provide the required level of QOS(Berger & Whitt, 1998).

The problem of bandwidth management and burst handling optimization was addressed by the theory of effective bandwidth. When employing the effective bandwidth theory an appropriate bandwidth is allocated to a class of user and the class of user is treated as if it requires this effective bandwidth throughout the session. The feasibility of this effective bandwidth is determined the constraint that the sum of all effective bandwidths is less than or equal to the total bandwidth available(Rajan, Mesfin, & Sando, 2020).

Let  $x_i$  be the effective bandwidth assigned to a class  $i$ . Let  $I$  be the number of classes. Let  $B_{RW}^T$  be the total bandwidth available in the link. A bandwidth management scheme is said to be feasible if equation 1 holds

$$\sum_{i=1}^I x_i \leq B_{RW}^T$$

Given three classes of users the following pair constraints apply.

$$x_1 + x_2 + x_3 \leq B_{RW}^T \text{ and}$$

$$x_1, x_2, x_3 > 0$$

The first constraint denotes that all the allocations for all the classes should be less than or equal to the total bandwidth available. Whereas constraint two denotes that all the allocations of bandwidth for each class of user are positive. Since the needs of users in the network keep on changing, priority is introduced in order to determine an appropriate share of effective bandwidth. Theory of effective bandwidth was used for optimization design of bandwidth management and burst handling.

### 2.20.2 Computational Complexity Theory

The computational complexity theory states that; For a given deterministic Turing machine  $M$  and a given input  $x$  of length  $n$  the time  $T_M(x)$  on that input is the number of computations  $M$  makes on input  $x$  before its halts (Gómez, 2020).

$$T_M(n) = \max_{|x| \leq n} T_M(x) \quad 2.11$$

From equation 2.16, the efficiency of an algorithm can be captured by a function  $T$  from the set of natural numbers  $n$  to itself such that  $T(n)$  is equal to the maximum number of basic operations that the algorithm performs on inputs of length  $n$  in time  $T(x)$  (Rashelbach, Rottenstreich, & Silberstein, 2020).

Computational complexity theory is used when solving computational problems (Zheng et al., 2021). A computational problem is a mathematical expression that can be solved using an algorithm with finite number of steps (Akbar, Yektakhah, Xu, & Sarabandi, 2021). Computational complexity is

the cost of a computation in terms of time and space is dependent on the input size as well as the computational resources(Consolini, Locatelli, Minari, Nagy, & Vajk, 2019). This means that the input size and the computational resource demands defines the complexity of a problem(Jurkiewicz, Biernacka, Domzal, & Wojcik, 2021).

Packet classification is a core feature in IP networks. Devices such as routers use a set of rules to determine which action should be taken for a given packet. Metadata such as source address, destination address, source port, destination port are used as features of packet classification(Jurkiewicz et al., 2021).

Packet classification is the process of associating packets to a given class. Packet classification performance affect overall process of performance isolation(Alkharasani et al., 2017). This makes the performance of packet classification core to overall system performance when doing packet classification. This makes packet classification performance of interest to QOS implementation(Fang, Rao, Liu, & Zhao, 2021). It becomes difficult to scale the number of rules and the number of matching fields as well as reducing the time complexity(Nam et al., 2020).

The performance Isolation design used theory of computational complexity for the optimization design to significantly reduce the value  $\text{Max}_{|x| \leq n} T_M(x)$  when doing performance isolation.

## **2.21 Performance Isolation**

To implement performance isolation a classifier is used. Linux classifiers are used to allocate a packet to a given class of a *qdisc* (queuing discipline) during

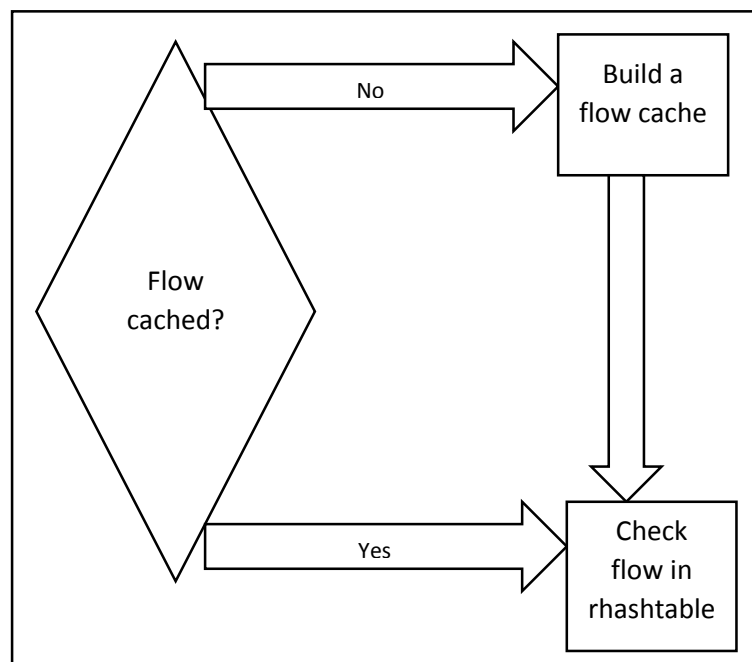
the queuing operation. A *qdisc* is a packet queuing algorithm that makes decisions on when and which packet to forward. A classful *qdisc* organizes traffic in the form of a tree like structure based on their classes. In Linux traffic classification can be done either using *iptables* or filters. When a packet arrives, its filters are matched to the classes until it reaches a leaf class then the packets are enqueued (Keller, 2006). When an incoming packet enters a root class it matches filters at the root class first. If filters match the ones at the root class it is assigned the root class otherwise the matching continues until the appropriate class is found(Sun et al., 2020). Packets filtering continues down a tree of classes and not upward. The *tc*(Traffic Control) feature of the Linux kernel provides many classifiers some of which are discussed in the following sections(Brown, 2006).

### **2.21.1 The Flower Classifier**

The *tc* flower classifier defines a mechanism for classifying packets using a flow key. The flow key is extracted using a Linux flow dissector and includes information extracted from the packet header or the packet meta data(Salim, 2015). After the flow key is populated it is compared with rules present in the classifier and if a match is found actions associated with the rule are executed. On the matching side the *fw* (Flower classifier) includes static matching on packet fields and Meta data while on the action side it provides actions supported which include either output or drop without any modification of packet fields and metadata. The Linux kernel implements *fw* filter both in software and hardware(Border, 2018).



When used in conjunction with the *tc* command the *fw* classifier is able to specify a rich collection of filters. In its inception the *fw* classifier was a 14 tuple packet classifier. However during its launching it was redesigned to use the kernel flow cache. The flow cache is built when the packet traverses from one layer to another of the network stack(Salim, 2015). As the packet traverses the network stack a cache is built and it can be reused by other layers of the network stack. For classification *fw* uses the following tuples that is the source and destination MAC address, the source and destination address and the source and destination port numbers. In addition the *fw* uses the netdev port for egress traffic classification(Salim, 2015).



*Figure 2.13: The Flower Classifier Operation(Salim, 2015)*

The filters are stored in the hash tables where they are used for look ups. As Figure 2.13 illustrates when a packet arrives in the *fw* classification system it establishes if the packet has the cache populated(Salim, 2015). If not the *fw*

create a cache by calling all the existing subsystems to populate their flow caches. On the other hand if the cache is already present the *fw* uses the cache fields as lookups and if a match is found the corresponding action is exercised(Almesberger & Ica, 1999).

### **2.21.2 Berkeley Packet Filter**

Berkeley Packet Filter (BPF) is the defacto filter in the UNIX variants. It incurs a lot of latency in handling both static and dynamic filtering tasks. This is because a filter update in BPF has to undergo the compilation phase, security checking phase and user kernel copying(Gulder & Déziel, 2017). In the compilation phase the human readable filter program is converted into BPF machine code program. In the kernel coping phase the BPF program is copied into the kernel. Lastly in the security checking phase the BPF program is scrutinized to ensure that it does not contain dangerous operations such as backward branches. All these phases induces a latency that may range from milliseconds to seconds depending on the number of filters(Salim, 2015).

Berkeley packet filter (BPF) was extended for use in Linux by replacing ports with sockets hooks and implementing branching. In its original form BPF included a binary choice of either if a packet matched certain filters it is either admitted or dropped, BPF is implemented in the Linux kernel to filter packets. When filtering packets BPF filters directly onto the memory space in the NIC therefore not requiring additional memory(Salim, 2015). As a result all the time spent by the BPF system is spent on interpreting the byte code for each filter. This results in performance degradation once the number of filters to be processed gets large. To prevent performance degradation a number of

techniques are employed, firstly is limiting the program size to 4096 instructions. Limiting the program size is good for speed however it end up limiting the functionality of the program. The second technique is to eliminate loops by ensuring that have positive offsets which ensures that the program will terminate. Lastly is the use of just in time compilation to reduce latency caused by interpreting byte code for each incoming packet(Brown, 2006). Figure 2.14 illustrates how the BPF is used for classification.

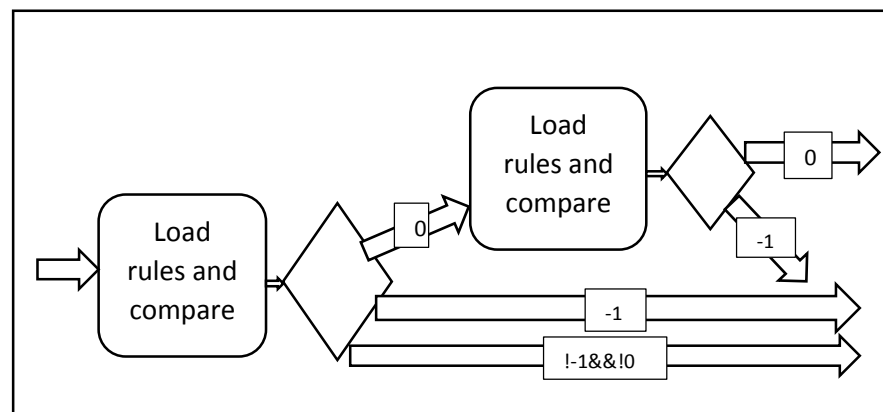


Figure 2.14: BPF usage overview(Salim, 2015).

### 2.21.3 Iptables Packet Filtering

Nefilter is a framework within the Linux kernel that provides a way for classification of packets using Iptables. Iptables is a kernel module which is part of netfilter which consists of a set of commands and tables containing rules that are used for classifying packets(Salim & Bates, 2016). Using netfilter a systems administrator is able to create Iptables hook functions that are used to filter packets as they pass through the networking stack. Iptables defines a table system on the user space where the system administrator defines chain of rules

for transforming and filtering packets. When classifying a packet the chain of rules are searched sequentially trying to match each packet.

Iptables has built in tables, namely NAT, Mangle and filter tables for storing chain of rules (Nedunchezian & Vijayakumar, 2016). The chain of rules are classified as;

- i) INPUT
- ii) PREROUTING
- iii) OUTPUT
- iv) FORWARD and
- v) POSTROUTING.

The filter table consists of FORWARD, INPUT AND OUTPUT chains which are used for filtering functions. All the rules in the Iptables consist of a set of matches and the corresponding actions. The chain of rules are processed in a sequential order that is the order in which they were added(Keller, 2006). If a given rule is satisfied then the corresponding action is returned and the search is stopped otherwise the process continues until a match is found. By any chance no match is found the default action is executed(Baidya, Chen, & Levorato, 2018).

Rules are deleted or inserted into the rules table using the *IPtool*. A rule is inserted as line of code that consists of the matching criteria as well as the associated action.

To use another table other than the default table, the specification should be done at the point at which the table command is specified (Keller, 2006). However, that may not in all cases be necessary since Iptables uses the default

filter table to implement all necessary commands. Commands for operations like insert, delete or add a rule are put at the end of the chain in which are used to instruct the program on what to do. The portion of the rule which is necessary for classification is sent to the Linux kernel. These portion of the rule includes details for matching the packet which may include port number, IP address, network interface, protocol and any other details necessary for classifying the packet(Baidya et al., 2018). The NAT table is used for Network Address Translation and all packets only pass through this table once. The first packet in a stream of packets is checked for matching rules and all the actions applied to the first packet in a stream is applied to all other packets that follow the first packet. This makes the NAT table not suitable for packet filtering(Salim, 2015).

The PREROUTING, OUTPUT and POSTROUTING chain are used for altering packets as they are processed by the firewall. The PREROUTING chain is used to alter all incoming packets as they enter the firewall. On the other hand the OUTPUT chain is used to alter those packets locally generated by the firewall. Finally the POSTROUTING chain is used to modify packets that are about to exit the firewall. The mangle table is used to modify/mangle header information of the packets. The content changed by the mangle table include the MARK or TOS and the TTL. Mangling of incoming packets is done by PREROUTING chain while mangling of outgoing packets after the routing decision is done by the POSTROUTING chain. OUTPUT chain is used to mangle the locally generated packets before routing decision is made. The INPUT chain is used to mangle packets before they reach the user space application(Balan, Potorac, & Graur, 2015).

After packets have gone through a series of routing decisions before the last routing decision the FORWARD chain is used to mangle the packets. After the packets goes through the firewall it is passed to the kernel space(Gulder & Déziel, 2017).

#### **2.21.4 RSVP & RSVP6 classifiers**

The Resource Reservation Protocol (RSVP) is used to classify IPV4 address based traffic while RSVP6 is used for IPv6. The RSVP classifies packet based on RSVP requests using source and destination IP addresses and port numbers. In speiciation of IP addresses the destination address must be specified as exact while the source can be optional(Border, 2018).

#### **2.21.5 Traffic Control Index Classifier**

The *Traffic Control index (tcindex) classifier* is used together with the dsmark qdisc. In order to classify traffic the dmask retrieves a value from the tcindex that is used to classify packets. The value obtained can be used in whole or part of it to find a filter that can be matched to a certain handle used to classify traffic(Brown, 2006).

The process of defining a key for matching a filter handle is as indicated in equation 2.6:

$$key = (skb_j > tc\_index \& mask) \gg shift \quad 2.12$$

The mask parameter indicates which bits of the of *skb->tc\_index* are to be used for matching. The shift indicates the number of bits returned by the bitwise AND should be shifted to the right(Keller, 2006). When a match is found the classifier returns the ID of the corresponding class defined by the *classid* parameter . In case where no match is found the *key* specified in the parameter

*fall\_through* is used as the class ID. Otherwise the process returns not found and the search proceeds to the subsequent classifier(Gulder & Déziel, 2017).

#### **2.21.6 Routing Table Based Classifier**

The routing based classifier is used to classify packets based on the routing table. The router based classifier is used in conjunction with the *tc* and the IP utility of the *iproute2*. The classifier uses a combination of either the mask and destination address or the source address and the mask in its definition for classification. In addition an IP realm must be defined(Keller, 2006).

For each realm a filter for destination or a source realm is specified. Each filter is designed to match to an ingress interface(Brown, 2006).

#### **2.21.7 U32 Classifier**

The U32 (Universal 32 bit) classifier filter is one of the most popular filter available in the current Linux implementation. The U32 filter is based on the hashing tables which gives it robustness when filtering rules are many. In its simplest implementation the U32 filter is composed of a list of records, each of which has an action and a selector(Salim & Bates, 2016). When a packet is being processed the IP packet is compared with the selectors configured until a match is found then the relevant action is performed. An example of an action is putting the packet into a predefined class. In the *tc* filter command line filters are configured using a filter specification, a selector and an action. Filters are specified as follows

```
tc filter add dev IF [ protocol PROTO ] [ (preference/priority) PRIO ] [ parent CBQ ] (Salim, 2015)
```

The protocol value defines the protocol to which the filter will be applied to. For this study it is the IP protocol(Gulder & Déziel, 2017). The preference or

protocol field specifies the priority of the currently configured filter. This is vital since there could be many filters with varied priorities(Almutairi, Stahl, & Bramer, 2021). The list of filters is passed in the order in which they were added however the filters are processed based on priority with the higher priority rules processed first(Salim, 2015).

The U32 selector contains the pattern to be matched to the packet. In particular it specifies the bits in the packet header that that are to be matched. With the U32 filter these bits may include those of IP address, Port number or the protocol. The following example illustrates how the configuration of selectors is done(Gulder & Déziel, 2017).

```
# tc filter add dev eth0 protocol ip parent 1:0 pref 10 u32 \match u32 00100000  
00ff0000 at 0 flowid 1:10
```

From the above example the selector line is the one that contains the match keyword. The example will match exactly the 00ff which is the match mask. In this case the 0xff will match exactly 0x10. The at keyword specifies where the matching criteria will start in this particular case at the beginning of the packet. In the U32 implementation it is possible to use either the general selectors or specific selectors (Salim, 2015).

General selectors consists of three parts that is the mask, pattern and offset .Using the general selector matches can be made to any single bit in the header of the packet. The general selector are more difficult to read and configure compared to the specific selectors(Brown, 2006). The syntax of the general selector is:



*match [u32 / u16 / u8 ] PATTERN MASK [ at OFFSET / nexthdr+OFFSET]*

The values *u32*, *u16* or *u8* specifies the length in bits that the pattern and mask should have. The offset defines is where the matching will begin.

However when the *nexthdr+* keyword is specified the matching will start at the upper layer header(Border, 2018).

The specific selectors can be found in the Linux *tc* (Traffic Control) program source code. The general selector makes code easy to understand and easy to read.

```
# tc filter add dev ppp0 parent 1:0 prio 10 u32 \match ip tos 0x10 0xff \flowid 1:4
```

The rule illustrated above will be able to match packets with a TOS filed as 0x10. With the U32 filter the specific rules are eventually translated to general ones whereby they are stored in the kernel memory. In addition it is important to specify the protocol since the UDP and TCP selectors are the same(Brown, 2006).

## **2.22 Limitations of Linear Search Based Classifiers**

In section 2.19, much of the classifiers employ linear search algorithm leading to various limitations. Table 2.6 illustrates a list of rules for a typical linear search based classifier policy and is followed by the probable limitations which are likely to occur.

**Table 2.6: Example of a Sequential Rule List Policy**

Rule	Destination IP address	Destination Port	Source IP address	Source port	Protocol	Action
R1	192.168.1.3	3260	192.168.2.4	ANY	ISCSI	Read
R2	192.168.1.4	3260	192.168.2.4	ANY	ISCSI	Read
R3	192.168.1.2	3260	192.168.2.4	ANY	ISCSI	Read
R4	192.168.1.5	3260	192.168.2.4	ANY	ISCSI	Read
R5	192.168.2.4	3260	192.168.1.3	ANY	ISCSI	Write
R6	192.168.1.3	3260	192.168.2.4	ANY	ISCSI	Read
R7	192.168.2.4	3260	192.168.1.5	ANY	ISCSI	Write
R8	192.168.1.1	3260	192.168.2.4	ANY	ISCSI	Read
R9	192.168.2.4	3260	192.168.1.1	ANY	ISCSI	Write
R10	192.168.1.2	3260	192.168.2.4	ANY	ISCSI	Read
R11	192.168.1.2	3260	192.168.2.4	ANY	ISCSI	Read
R12	192.168.1.3	3260	192.168.2.4	ANY	ISCSI	Read
R13	192.168.1.4	3260	192.168.2.4	ANY	ISCSI	Read
R14	192.168.1.2	3260	192.168.2.4	ANY	ISCSI	Write
R15	192.168.1.3	3260	192.168.2.4	ANY	ISCSI	Read
R16	192.168.1.5	3260	192.168.2.4	ANY	ISCSI	Read
R17	192.168.2.4	3260	192.168.1.4	ANY	ISCSI	Write
R18	192.168.1.5	3260	192.168.2.4	ANY	ISCSI	Read
.	.	.	.	.	.	.
.	.	.	.	.	.	.
R325	Any	Any	Any	ANY		Drop

**2.22.1 Shadowed Rule Limitation**

A shadowed is a rule that won't get to match since a rule preceding it will have matched all its packets(Cherian, 2016). For instance, in Table 2.6 R12 is

shadowed by R1. Shadowed rules may bring about speed problems as well as security issues. Security issue would arise if a rule implementing security would be shadowed by another rule(Dahan, Hindi, et al., 2021). For example if we are to deny entry to certain malicious packet by use of a particular rule, if this rule is shadowed by another rule above it, security would be breached (Nedunchezian & Vijayakumar, 2016).

In addition shadowed rules may result in reduced performance of a classifier since the shadowed rules waste the packet classifiers processing time(Paricio & Lopez-Carmona, 2021). Shadowed rules therefore can be deleted without changing the classification policy( He, Chomsiri, Nanda, & Tan, 2013b).

#### **2.22.2 Swapping Position between Rules Limitation**

Swapping rules can change the classification policy if the swapped rules result in putting packets in different classes and they can be able to match the same packet. Changing the packet action would alter the accuracy of the classifier(He, Chomsiri, Nanda, & Tan, 2013). In Table 2.6 swapping R325 with any other rule would result in a different action.

#### **2.22.3 Redundant Rules Limitation**

A redundant rule is one that has been implied by another one below it( He et al., 2013). In Table 2.6 R14 is redundant to R10.Redundant rules result in reduction in the speed of processing packets which waste classifiers processing time(Acharya et al., 2006).

#### **2.22.4 Bigger Rule Problem**

A big rule is one that matches all packets. In Table 2.6 R325 is the bigger rule. In other words the default rule. If bigger rule is placed before other rules it

shadows them(Pu Wang, Yan, Wang, & Zeng, 2022). This brings about a design problem since the rules position is of significant importance(Zhao, Inoue, & Yamamoto, 2004). This also brings about speed problems since a packet that can only match the bigger rule has to go through all other rules before it reaches the bigger rule which is usually the last(Suresh, 2016).

#### **2.22.5 Sequential Computation Limitation**

In a listed firewall the computation for packet classification is sequential which brings about speed problems when rules are many(Acharya, Member, Znati, & Member, 2008). The time required for packet classification will increase with an increase in the number of rules. For example the average number of rules that would be matched to a packet in a classifier of  $N$  rules is  $N/2$  and the time required is  $O(N)$ (Cherian, 2016).

#### **2.23 Performance Isolation Optimization**

Since the study is using packet classification optimization for performance isolation optimization, the research therefore reviews various approaches for packet classification optimization. Vasu and Ganesh(2014) proposed a technique for reordering packets based on the current network statistics. The technique further subdivides the packets into  $P$  partitions to reduce the search time(Shirvani Moghaddam & Moghaddam, 2022). The window size is used at store the history of the traffic pattern. A quantity match ratio is then calculated based on the values of the window size and it is this match ration that is used to reorder firewall rules. Through experiments this technique has been proven by Vasu and Ganesh (2014) to be effective in implementing firewalls rules. To

improve on this solution by Vasu and Ganesh (2014) the study partitioned the rules instead of the traffic to reduce the match time since the overhead incurred on segmenting packets which is more than that of segmenting rules. This follows from the fact that even in a medium sized organization the network could generate millions of packets unlike number of rules which could be far much less (Acharya et al., 2008).

Hamed and Al-shaer (2006) proposed a technique for optimizing firewall filtering rules by calculating the traffic statistics then using the results to dynamically reorder firewall rules. When implemented the solution proved to be simple and light weight. However its early rejection property may cause more packets to be dropped which would be detrimental to overall QOS especially for storage area networks(Sun & Cho, 2022).

El-Atawy, Samak, Al-Shaer, and Hong (2016) proposed two methods that is segment based tree search and segment based list search. The segment based tree search uses Huffman trees and traffic characteristics for each segment to reduce the search time. However the technique was proven to have a lot of overhead especially when it comes to maintaining the tree. To eliminate this overhead El-Atawy, Samak, Al-Shaer, and Hong (2016) used the segment based list search which included a most recently used list which is placed at the top of the classification rule list to reduce the search time. However Ganesh, Sudarsan, Vasu, Ramalingam, and Nadu(2014) observed that this method is more useful when the traffic is steady.

Trabelsi and Zeidan(2012) proposed a method which rejects packets early and also accepts packets as early as possible. The early acceptance is achieved with

the use of splay trees decisions which are updated using the history of the traffic characteristics. For rejection a multilevel approach for filtering packets is used before the decision for rejection is made. This is done in an attempt to ensure that users are not denied service (Ganesh et al., 2014).

Named and Al-Shaer(2006) used a branch and bound technique to resolve the optimal rule ordering problem by ensuring that the minimum number of rules are matched to the packets as well as maintaining the relationship among the rules. However Vasu and Ganesh(2014) observe that the proposed approach by Named and Al-Shaer(2006) has linear space complexity and the resulting time complexity was proven to be polynomial.

To enforce performance isolation traffic needs to be classified and resources bound to the classes of traffic. Classification refers to the association of packets to classes based on the packet header information(Nam, Choi, Yoo, Eom, & Son, 2020). However all the previous solutions discussed assume that the classification process does not affect overall performance of the storage area network during the performance isolation process. This is not true given that all the performance isolation techniques discussed use throttling techniques in order to achieve performance isolation. To apply throttling all packet flows need to be differentiated in terms of importance. In other words they need to be classified.

The study embarked on proving that the classification process if not optimized could lead to performance degradation of a storage area network in attempts to achieve performances isolation. The study used throttling of workloads from initiators based on the resources required. For throttling to happen it is necessary

to have classified traffic. Since experiments are done on the Linux platform the study uses classifiers available in the Linux platform. The *tc* command of the Linux kernel already contains a classifier action subsystem. These includes the flower classifier, Berkeley packet filter, IPtables, RSVP and RSVP6 classifier, Traffic control index classifier, routing table classifier and Universal 32 bit classifier (Almesberger & Ica, 1999).

Amongst all the classifiers available in Linux the study settled on the U32 classifier for two main reasons. One is that the U32 classifier can use any bit patterns in the packet header for classification (Salim, 2015). Secondly in Linux classifier performance benchmarking experiment performed by Salim and Bates (2016) proved that the U32 is the best in terms of performance.

However the traditional implementation of U32 packet classifier does look up for rules in a sequential manner until a match is found. Rules are distinct entries in a classifier for putting packets into classes (Gulder & Déziel, 2017). This implies that the delay incurred in finding a match for a specific rule is proportional to the size of the rule list. While this may not be a big deal when the rule list is small, when there are many rule lists it may cause performance degradation of the system due to increased delays (Barzegaran, Cervin, & Pop, 2020). Again since packet filtering entails more processing load than routing, the filtering process becomes more complex as rules increases (Almesberger & Ica, 1999). In addition classifiers must be able to handle more packets as transmission media speeds increases otherwise the classifier may result in delays during packet processing (Baidya et al., 2018). Other Problems associated with sequential rule lists include shadowed rules, possibility of swapping of rules,

redundant rules and the design complexity brought about by the bigger rule problem(Chomsiri, He, & Nanda, 2012). A shadowed rule is one that won't get to match any packets because the rules above it have already matched the packets it should have matched(Blenk, Kellerer, & Schmid, 2019). Shadowed rules results in speed and security problems. Swapping rules involves changing the rules position. In some cases swapping of rules may result in changes in the classification policy. A bigger rule is a rule that shadows all the rules in the classification policy. It takes a lot of experience from system administrators to determine the position of bigger rules(Suresh, 2016).

In order to optimize performance when implementing isolation the study optimized the classification technique employed. Therefore the focus of this chapter is to optimize the U32 classifier for optimization of performance during isolation flows belonging to particular classes. The study uses the techniques of reordering classification rules, splitting the rules and building a tree rule structure for the classifier optimal performance. To achieve performances isolation the proposed approach binds resources to the classes generated and then using experiments demonstrates that the proposed performance isolation solution with an optimized classifier performs better than that without an optimized classifier. Metrics used to measure optimization of proposed systems include latency, throughput and accuracy (Chomsiri et al., 2012).

#### **2.24 Bandwidth Management for QOS**

Bandwidth in computer networking is defined as the data rate supported by a network connection and is expressed in bits per second (Alkharasani et al., 2017). The term is derived from electrical engineering where it represents the



distance between the highest and lowest signals on the communication channel(Wang, & Member, 2017).

In a network, there is a combination different categories of traffic that have different requirements based on the application they are using (Celik, Radaydeh, & Member, 2017). Although in the recent years the available bandwidth has increased it is still one of the major causes of bottlenecks in communication networks if not managed( He et al., 2017). Therefore making the utilization of bandwidth to be efficient is still one of the key aspects that promote network performance. The management of bandwidth begins at the network planning and design stage and in the long run through a variety of techniques based on the various layers of the TCP/IP model.

The TCP/IP protocol suite defines the manner in which nodes communicate over the internet. The TCP part is responsible for segmentation and reassembly of packets while the IP part is responsible for transmitting packets(Ravali, 2019). The TCP/IP protocol suite is made up five layers namely physical layer, data link layer, network layer, transport layer and application layer (Bora, Singh, & Arsalan, 2019). Each upper layer is supported by lower layer protocols. TCP/IP is designed to be flexible and can be extended to meet various requirements as long as service interfaces to the layers remain intact(Chinmay, 2019). It offers a networking model and offers a generic means to separate computer networking functions such as bandwidth management into multiple layers(Ravali, 2015). Some methods for managing bandwidth based on the layers of TCP/IP model are addressed in the following sections.

### **2.25 Layer One and Two Bandwidth Management**

At layer one of the TCP/IP model bandwidth management can be achieved by installing the appropriate transmission media and networking equipment (Lim et al., 2017). To achieve higher bandwidth additional hardware is installed (Wu et al., 2017). The layer two of the TCP/IP model is responsible for delivery of frames. In this layer frames contend for access to shared link and therefore bandwidth management at this layer aims at managing contention for the available link (Alkharasani et al., 2017). At layer two there are several techniques used for reducing contention for bandwidth include network segmentation, employing full duplex links and prioritizing frames (Celik et al., 2017).

Network segmentation is division of the network into smaller components. Network segmentation implemented using bridges, where the network is divided into many segments joined by bridges. This has the cost implications of buying new hubs and bridges (Lim et al., 2017). Network segmentation can also be done using the network interface card. This is where each segment on the network is connected to a different NIC on the server. This means the servers will be required to route between NICs (Song, 2018).

### **2.26 Layer Three Bandwidth Management**

At layer three the main focus of bandwidth management is to eliminate congestion by use of routers which control rate at which packets are sent into the network. Congestion occurs when the traffic traversing the network exceeds the capacity of the network, and needs to be managed to prevent degradation of QOS (Xu, 2018). If not managed congestion wastes whatever bandwidth is

available and degrades the QOS in a network. The TCP/IP model network layer is responsible for assignment of IP address to hosts in the network. Bandwidth management at the network layer occurs in real time as the packets arrive( Wu et al., 2017). And there after applying static or dynamic bandwidth allocations(Randrianantenaina & Member, 2017). Techniques for managing bandwidth at layer three are addressed in the following sections.

### **2.26.1 Upgrading Cables and Ethernet Hubs**

To increase the bandwidth, the shielded twisted pair can easily be replaced by fibre optic cable for faster speeds(Lim et al., 2017).On the other hand Replacing 100 Mbits/sec Ethernet network interface cards (NICs) with gigabit Ethernet increases bandwidth considerably(Mamman & Hanapi, 2017). This solution entails the purchase and installation of cabling. Besides the cost, other key areas of improvement include ,managerial overhead where external connections require managerial time and effort(Randrianantenaina & Member, 2017).

### **2.26.2 Network Segmentation and Full Duplex Ethernet**

Network segmentation using switches is done by replacing hubs with switches. Since each port in a switch represents a collision domain, adding a switch breaks the network into small collision domains(Wu et al., 2017). The creation of small collision using switches is known as micro segmentation. Micro segmentation comes with many benefits including low latency, support for virtual Local area networks (VLANs) and prioritization. In addition it is less costly as compared to layer one cost of installing cables(Xu, 2018).

Full duplex Ethernet technique is able to increase the bandwidth of a connection up to twice its capacity(Wang, Sun, & Cao, 2018). When a dedicated full duplex

connection is used it further eliminates contention for the link. However to achieve full duplex Ethernet NICs must be able to support duplex mode. Therefore full duplex Ethernet requires NICs replacement and reconfiguration of the client and server software's(Wang, 2018).

### **2.26.3 Bandwidth Allocation, Sharing and Reservation**

A network is made up of the users who need bandwidth to meet their service demands(Paul, Tachibana, & Hasegawa, 2016). However, in a realistic network, bandwidth is limited and some methods of allocating it are needed when total demand is greater than the resource limit. Therefore bandwidth management is required to ensure proper utilization of the existing bandwidth. Bandwidth management includes the techniques of bandwidth allocation, bandwidth sharing and bandwidth reservation

Bandwidth allocation is about efficiently allocating the network bandwidth among the sources(Randrianantenaina & Member, 2017). Static bandwidth allocation technique assigns a maximum amount of bandwidth to each class and implements traffic shaping to control the data traffic(Garg & Dixit, 2021). Classes are not restricted to use less than their bandwidth allocations however a class is limited to use more than its allocated bandwidth(Song, 2018). Dynamic bandwidth allocation techniques is an alternative that enables network resources to be adjusted in order to improve network utilization(Ji & Member, 2018). When a certain class of users require more bandwidth than that it is reserved for, a review is initiated to ask for more. If the allocated bandwidth to a class is more than enough, some of the bandwidth can be shared(Mamman & Hanapi, 2017). In this way bandwidth usage can be improved significantly. In dynamic

bandwidth allocation various classes of algorithms are used. One such class of algorithm is based on parameter measures. In this case a parameter is calculated up to the current period and in future bandwidth is allocated based on the previous history of usage(Paul et al., 2016).

In a bandwidth sharing method, traffic is divided into classes and each class is allocated a percentage of the bandwidth(Xu, 2018). When given classes reach its limit, no more data belonging to that class can be forwarded. However if all other classes are not utilizing their whole share, a class can borrow bandwidth for a short while and send its traffic(Randrianantenaina & Member, 2017).

Bandwidth reservation is a technique where a certain data flow is allocated a specific amount of bandwidth for guaranteed QOS. The reservation protocol enables one to reserve special QOS for their data(Wang et al., 2018). When an application receives data packets for which it requires a certain QOS it sends a RSVP request back to the sending application(Sboui et al., 2019). As the data traverses the network, the QOS is negotiated with the routers and other network devices. Those network equipment's that do not contain the RSVP functionality simply ignores the RSVP traffic and do not participate in the negotiation(Song, 2018).

#### **2.26.4 Load Shedding and Buffer Allocation**

Load shedding also referred to as packet dropping is a function performed by the router when it cannot handle all incoming packets(Mamman & Hanapi, 2017). On this case the packets may be dropped based on a certain priority. However priority schemes are challenging to implement owing to the fact

negotiation from all users is required(Salomo, Pratama, Choi, & Member, 2018).

The dropping of excess packets has been proven not to be effective since it would require differentiation between data packets and acknowledgment packets. For example dropping TCP acknowledgment packets would delay the allocation of buffers and hence cause congestion(He et al., 2017).

In buffer allocation technique buffers are allocated in routers. If by any chance congestion occurs, the packets can be temporarily stored in the pre allocated buffers to be transmitted later(Kourtessis, Lim, Merayo, Yang, & Senior, 2019).

In this way congestion is reduced because the stored packets are no longer in transit(Marir, Wang, Li, & Jia, 2018).

#### **2.26.5 Flow Control Using Choke Packets and VLANs**

Flow control using choke packets reduces congestion by ensuring that the router is not overwhelmed by many packets by transmitting a choke packet to the source. A choke packet is a packet sent to the transmitter by the receiver indicating there is congestion and the receiver should reduce the rate of sending packets. In response the source reduces the amount of packets sent(Xiaohong Huang, Yuan, & Ma, 2018). To increase the speed of transmission for choke packets, they are sent to intermediate routers in which case the response becomes fast. The internet control message protocol is responsible for sending choke packets( Lim et al., 2017).

Flow control targets at shaping traffic from source to destination while in congestion control aims at regulating the traffic flow in the network(Mamman & Hanapi, 2017). Flow control reduces network throughput due to increased

packet transit time which makes it an effective bandwidth management technique. Flow control is only effective if it manages to decrease the quantity of traffic in a link at critical points and time(Randrianantenaina & Member, 2017).

Virtual LANS are defined in IEEE standard 802.1Q. The IEEE 802.1Q is a standard that implements VLANs in an Ethernet network. IEEE 802.1q defines a system for tagging Ethernet frames and the mechanism to be used by networking devices to handle VLAN frames. VLANS allows for users to be grouped together irrespective of their physical locations. VLANS create a smaller broadcast domains which in turn reduces bandwidth consumed by broadcasts. In this way more bandwidth is availed to users(Ji & Member, 2018).

### **2.27 Layer Four Bandwidth Management**

At layer four bandwidth is managed by not only controlling the amount of connections, but also regulating the packets flow amongst the hosts. At layer four bandwidth management can be achieved through limiting the amount of end to end connections and controlling the flow of packets amongst two hosts(Lim et al., 2017). In this case bandwidth management is achieved using Transmission Control Protocol Rate Control and rate control resource reservation protocol( Liu, Li, Xu, & Li, 2021).

Sliding window is a TCP features that is used to reduce congestion by regulating the amount to packets that can be transmitted by a given host. Acknowledgements are used to communicate to the transmitting host to continue transmitting otherwise the transmitting host stops transmitting(Sovandara, April, & Penh, 2015). TCP sliding window can be

adjusted dynamically in responses to any flow timeouts in the network by shaping traffic, TCP rate control makes it smoother and more predictable(Alkharasani et al., 2017).

The resource reservation protocol (RSVP) method is used to optimize bandwidth by monitoring and controlling bandwidth along each link between sender and receiver(Gémieux et al., 2018). The RSVP reserves bandwidth between a sender and receiver monitoring each connection and route(Xu, 2018). Before information is sent, the receiver establishes whether each device along the route has spare bandwidth available, if not the transmitting device is informed(Jamaluddin, 2019).

### **2.28 Layer Five Bandwidth Management (Application layer)**

The emergence of many web applications has led to the demand for application level QOS in many network setups. The high contention for bandwidth may lead to bandwidth sensitive applications not working properly(Kulkarni, 2015). At the application layer bandwidth management is achieved using variety of QOS tools. This tools are used to provide priority to traffic based on application type so as to ensure bandwidth intensive applications do not crowd the network(Chang et al., 2017). This solution provides a predefined classification of protocols based on applications and an all-inclusive policies for traffic control such as rate shaping and priority marking(Yong et al., 2015).This makes it possible for network administrators to distinguish between desirable and undesirable traffic flows within the same protocol. Application layer bandwidth management is supported for all application matches, custom application rules and file transfer types(Cos, 2012).



Traffic generated by applications in a network is classified as either constant bit rate (CBR) or variable bit rate (VBR)(Mahajan & Mahajan, 2015). The CBR applications produce traffic flows at a constant rate and therefore can be allocated a given amount of bandwidth to achieve intended QOS. This implies that allocating more bandwidth does not improve user satisfaction(Nahrstedt, Arefin, & Rivas, 2011). VBR applications produce varied traffic flows and are able to utilize all the available bandwidth. For VBR traffic the more the bandwidth the better the QOS(Randrianantenaina & Member, 2017).

At the application layer port blocking and bandwidth caps are the two most popular techniques for implementing bandwidth management. Blocking can be done for only ports associated with P2P applications in an attempt to improve network performance(Huang, Yuan, & Ephremides, 2019). However blocking ports as a method of managing bandwidth has a number of drawbacks. First is that P2P applications such as bit torrents allow users to choose the port before the start of a download(Xu, 2018). Using Bandwidth Caps technique is applied to discourage users from consuming huge amounts of bandwidth(Jamaluddin, 2019). Bandwidth caps is effective in bandwidth savings but cannot effectively manage congestion during peak hours(Ali & Chen, 2019). Additionally, this technique lacks the granularity to differentiate traffic. This technique can be improved by applying caps to certain applications at certain times of the day(Eramo, 2019).

### **2.29 Dynamic Bandwidth Management Algorithms**

Dynamic bandwidth management involves the assignment of bandwidth based on network changes. The two main algorithms used in dynamic bandwidth

management include Hierarchical token bucket and per connection queue (PCQ). PCQ does not offer prioritization and therefore not much has been studied about it in this study, in contrast HTB offers prioritization and implementation in Linux traffic control (Siregar, Fadli, & Hizriadi, 2020a) .

HTB falls into the category of class based queuing disciplines (Iswadi, Adriman, & Munadi, 2019). A queuing discipline is a mechanism for queuing and dequeuing packets under the influence of an algorithms( Mathews, Kramer, & Gotzhein, 2018). HTB operates between the IP layer and the mac layer(Iswadi, 2019). In HTB flows are structured in a hierarchy of classes namely root, inner and leaf classes(Bosk, Gaji, Schwarzmann, Lange, & Zinner, 2021). All traffic goes through the root classes which is situated at the top(Qian et al., 2017). Inner classes are below the root classes with child classes as leaf classes. The leaf classes have no child classes however they have parent classes(Iswadi, Adriman, & Munadi, 2019). Flows control in each class is achieved by an internal token bucket which is populated with tokens limited by the rate a particular class is permitted to transmit(Sarmah, 2019).When a packet is transmitted belonging to a particular class its bucket is subtracted with the number equal to the rate(Lee & Kim, 2013).

Each class is configured with two rates that is rate bucket with tokens and a ceil bucket which contains ctokens (ceil tokens)(Ren, Feng, & Dou, 2017). Tokens and ctokens is a measure of the amount of time a class occupies the scheduler output line. During transmission a class could either be in green, yellow or red states(Siregar, Fadli, & Hizriadi, 2020). In the green state the class has sent less

data than its allocated rate and therefore it can send more (Aljoby, Wang, Fu, & Ma, 2018). In the yellow state the class has exceeded its guaranteed rate but not ceiling rate. In the red state the class has sent more than the ceiling and cannot send any data.

HTB uses DRR (Deficit Round Robin) algorithm for scheduling in which the class deficit is decremented based on the size of the packet (Aljoby et al., 2018).

Ctokens decrease by a ratio equal to  $\frac{\text{packet length}}{\text{rate}}$ . This is the amount of time a packet is in the scheduler queue. Ctokens is added to the time elapsed after transmission to take into account the time that elapsed since the last transmission in the same queue ctokens. The following example is used to explain this concept further. Let  $t_2$  be the current time and  $t_1$  be the last time since the last transmission (Garroppo et al., 2019).

$$ctokens(t_2) = ctokens(t_1) + (t_2 - t_1) - \frac{\text{packet length}}{\text{rate}} \quad 2.13$$

Given that  $C$  is the capacity of the network in bps, any rate assigned to class

$$r < C. \text{ Therefore } \frac{\text{packet length}}{\text{rate}} > \frac{\text{packet length}}{C}. \quad 2.14$$

Equation 2.7 and 2.8 shows that when there is consecutive transmission from the same class the tokens constantly decrease (Ren et al., 2017). This is because the transmission is done at rate  $r$  therefore the value of  $t_2 - t_1$  is added to the  $C$  pool which is equal to  $\text{packet length} / r$  which is less than  $\text{packet length} / \text{rate}$  (Garroppo et al., 2019).

If the expiration of the deficit for the current green class expires the scheduler might switch to the next green class. This is the case due to the working of DRR algorithm which is used in HTB as a scheduling algorithm. Scheduling algorithms are algorithms that determine the order in which packets are processed (Isyadi et al., 2019). The DRR scheduling algorithm decrements the deficit after every transmission and in some cases it becomes zero or negative. In the mentioned cases  $1/10$  of rate is added to the deficit by default and then the scheduler can switch to the next green class if any. If there are no green classes the current one will continue to send until it is red or other become green (Lee & Kim, 2013)

Another case is when there is a bucket underflow (Ren, 2017). Bucket underflow is when ctokens bucket becomes empty which is an indication to the scheduler that the class is exceeding its ceil and therefore should switch to the next class. Ctokens takes the values in the interval  $[-cburst, cburst]$  where cburst is the peak rate (Siregar et al., 2020).

When cburst is negative an underflow happens and the class status becomes red. On the other hand if ctokens goes above cburst the excess ctokens are discarded. Since underflow has got a high priority the deficit expiration if the underflow occurs the class stops sending data without putting into consideration the deficit. However if the deficit expires and other classes are red, the current transmitting class continues to send by adding a quantum value to deficit. It is important to configure a high cburst to ensure all the classes are green so as to allow transmission of all bytes from the current class before switching to the next

one(Ren, 2017). When a class has reached its ceiling rate it queues packets until new tokens are available in a process known as policing. The working of the HTB is summarized in Figure 2.15.

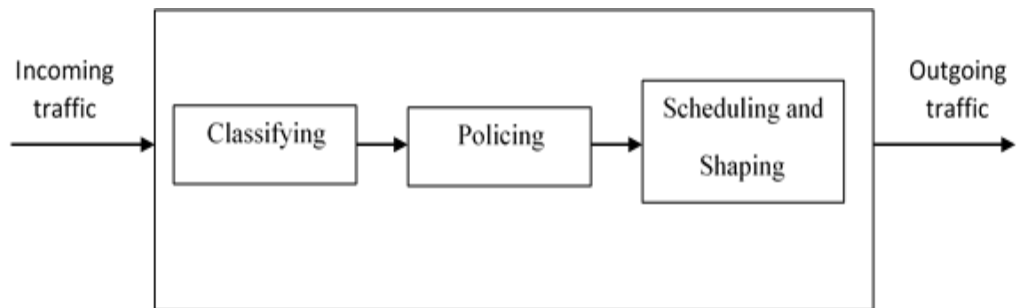


Figure 2.15: Functioning of HTB (Source: Bosk et al., 2021).

The key strength of HTB is bandwidth borrowing which ensures maximum utilization of the available bandwidth. Configurations for bandwidth borrowing is based on priority, high priority classes can borrow more bandwidth(Sarmah, 2019). In HTB each class is configured with allowed rate( $R$ ), burst rate( $BR$ ), Guaranteed rate( $GR$ ) and rate that the class can borrow( $BW$ ). Therefore for any class  $i$ , in HTB a definition of its allowed rate( $R$ ) can be calculated as indicated in equation 2.9(Lee & Kim, 2013).

$$R_i = \min(BR_i, GR_i + BW_i) \quad 2.15$$

Each class is configured with priority  $p$  and a quantum. Leaf classes borrow bandwidth from their parents. If a leaf class has no parent then  $BW=0$ . For any class  $i$  with parent  $p$  and quantum  $i$  and priority  $p$  then the following equation holds(Iswadi et al., 2019).

$$BW_i = \left\{ \frac{Q_i R_p}{\sum_{j \in D} Q_i \text{ where } p_j = p_i} \text{ if } \min_{j \in D(p)} p_i \geq p_i \right\} \quad 2.16$$

With equation 2.10 it is possible to ascertain that rate is borrowed from parent and decided among all descendants levels based on priority according to quantum  $Q_i$ (Lee & Kim, 2013)

HTB cannot alone provide fairness and utilization, since it relies on prediction of output capacity of a link. We therefore need to include the current network statistics. Commercial routers do not provide optimization of bandwidth sharing for QOS by dynamically assigning bandwidth based on priority and network conditions (Ren et al., 2017). We propose the traffic aware HTB for QOS provisioning based on priority(Lee & Kim, 2013). The proposed solution has been analyzed with a series of systematic experiments. The experiments we have verified that the proposed QHTB offers optimized bandwidth utilization and low latencies.

### **2.30 Burst Handling for QOS**

One of the objectives of QOS is to regulate traffic injected into a (Wang, Xu, Chen, Sun, & Zhang, 2018). Traffic is said to be bursty if the data flow rate deviates sharply in short periods of time. The rates may jump from a peak of 12Mb/s for example to an average rate of 2 Mb/s and vice versa as illustrated in Figure 2.16(a). Bursty traffic is very difficult to handle since its behavior is unpredictable due to the differences between peak rate and average rate.

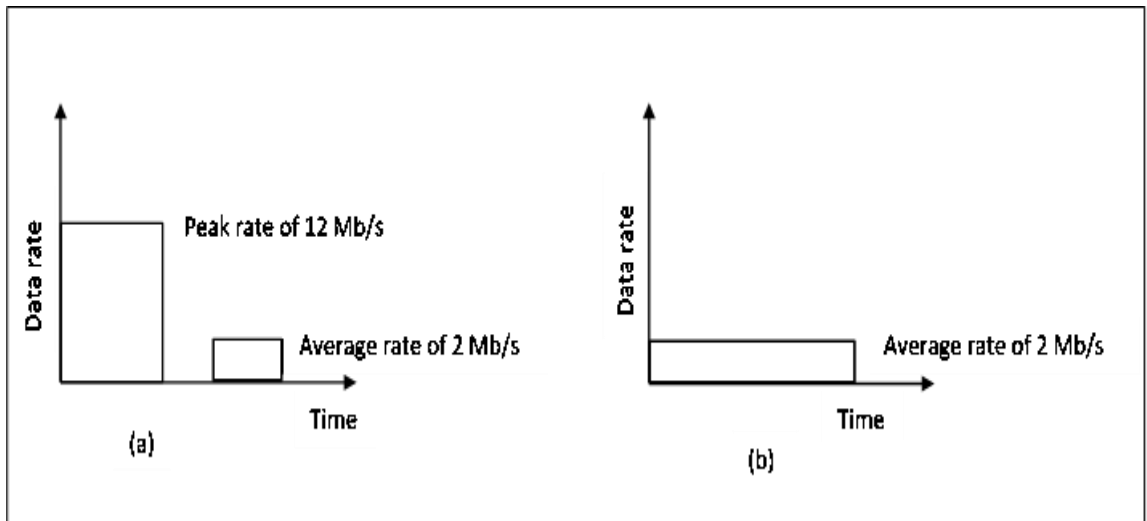


Figure 2.16: Bursty traffic handling for QoS with (a) Showing Bursty Traffic, and (b) Shaped Traffic (Source: Lichtblau & Streibelt, 2017)

To take care bursty of traffic, traffic shaping is required to regulate access to the available bandwidth. Traffic shaping is meant to avoid congestion and reduce delay that may arise due to contention for the available bandwidth (Aduragbemi, 2018). Traffic shaping is meant to regulate traffic at a constant rate as illustrated in Figure 2.16(b). Leaky bucket and token bucket algorithms are most popular traffic shaping solutions (Zuberek & Strzeciwilk, 2018).

### 2.30.1 Leaky-Bucket Traffic Shaping

The leaky bucket algorithm shapes traffic by transforming a turbulent traffic flow into a smooth traffic flow by regulating the amount of traffic that exits an interface where the algorithm is configured as illustrated in Figure 2.17 (Khalid & Hashim, 2014). A leaky-bucket interface is placed between the source of traffic and the network (Chang, Wu, & Lin, 2018). Traffic is regulated to flow into the network in the same manner flow of water would in a leaking bucket. Leaky bucket has the advantage of being easy to implement (Amjad, 2019).

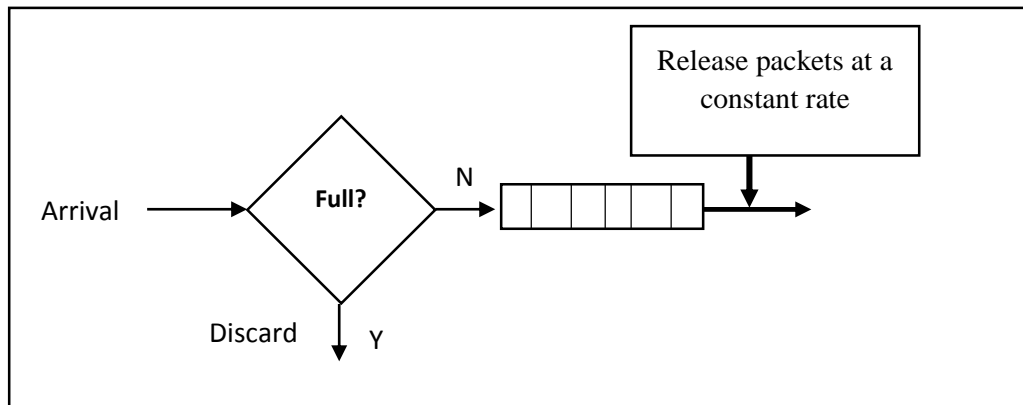


Figure 2.17: Leaky Bucket Algorithm (Source: Zuberek & Strzeciwiłk, 2018).

At the core of the leaky bucket algorithm is a finite queue to which packets are admitted depending on the capacity of the interface buffer. Packets that find the buffer full are discarded (Zuberek & Strzeciwiłk, 2018). The amount of traffic that flows from the interface demonstrates how much traffic a given interface can handle and this is communicated to the source to determine how much traffic the source should transmit (Zhou, Yan, Berger, & Ruepp, 2018). In this way leaky bucket algorithm regulates the amount of traffic that enters a given network (Salomo et al., 2018).

In the Leaky bucket algorithm implementations packets are classified as either fixed sized packets or variable size packets. Variable size packets are transmitted at each for each clock tick while for fixed size packets, a fixed amount of packets are transmitted at a particular time (Wang et al., 2018). The Leaky bucket algorithm is suitable for networks where we have variable number of packets as well as fixed number of packets (Amjad, 2019). However the leaky bucket algorithm does not put into account idle time for example if a host is not



sending its bucket remains empty and by any chance a host experiences congestion only its average rate is transmitted(Zuberek & Strzeciwiłk, 2018).

### 2.30.2 Token Bucket Algorithm

In the token bucket algorithm tokens are used to regulate traffic. A token refers to the permission to transmit one bit of data by a host in the network(Khalid & Hashim, 2014). Tokens are spawned at a frequency of one token per unit time and stored in a queue of finite size(Xiaoge Huang, Cao, Li, & Chen, 2020). When the token pool is full, any additional token generated are discarded. The token bucket algorithm takes into account idle time where any host that has experienced idle time accumulates its share of tokens which can be used during congestion(Truong-huu, Member, Gurusamy, & Member, 2017). Token bucket algorithm is efficient in environments where the amount of packets to be transmitted is equal to the amount of tokens. Functioning of token bucket algorithm is as illustrated in Figure 2.18(Lichtblau & Streibelt, 2017).

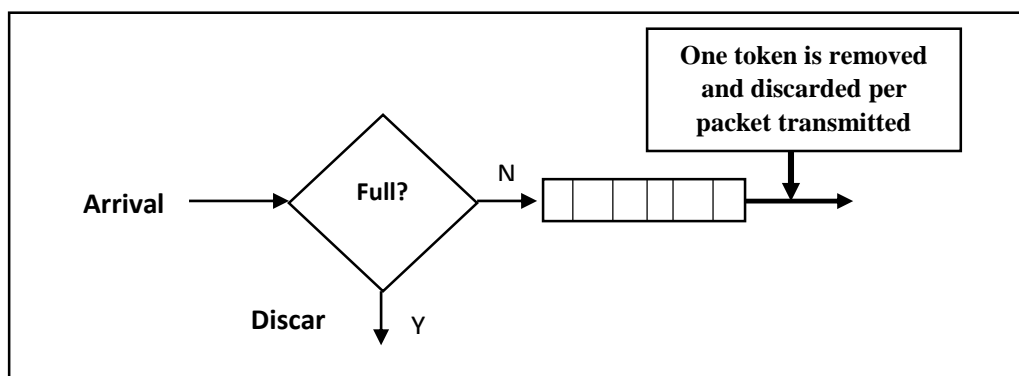


Figure 2.18:Token Bucket Algorithm(Source: Zuberek & Strzeciwiłk, 2018).

### **2.30.3 Combining Token Bucket and Leaky Bucket**

The Token bucket and leaky bucket can be combined in order to ensure idle hosts are compensated for their idle periods as well as regulate traffic(Khan, Member, & Alam, 2022). In the combined implementations, the leaky bucket is implemented as the first and token bucket follows as the second. For optimal performance the number of tokens out of the bucket need to be greater than the number of tokens entering the bucket(Lichtblau & Streibelt, 2017).

### **2.31 Optimization of Bandwidth Management and Burst Handling**

Guo(2005) found that QOS in SANS has been researched for years such as façade, chameleon, triage and Stonehenge. Facade uses the technique of throttling I/O requests to the storage to achieve the required SLO. However façade earliest deadline is not effective when we have burst workloads(Kim, Kwon, Lee, & Song, 2021). Chameleon leaky bucket is not efficient since it is not work conserving because it reserves bandwidth to support each client's storage QOS requirements sharing of resources proportionally. Solutions such as YFQ and cello balance user requirements. Stonehenge uses a disc scheduler to guarantee bandwidth between the storage server and the client(Ramaswamy, 2008).

Lu et al., (2005) looked at the integration of storage QOS and network QOS. Other solutions mentioned above looked at storage QOS and network QOS separately. They also proposed a priority based greedy algorithm for allocating storage server link network bandwidth to clients. Formulated mathematical models to calculate the required bandwidth. Solution implemented on object based storage system. Object based storage does not use file system instead it

uses object attribute mechanism. The authors implemented a solution to calculate the needed network bandwidth for clients based on their SLO. Then they designed a priority based greedy bandwidth allocation to allocate the link network bandwidth. (Lu et al., 2005).

(Chambliss, Alvarez, Pandey and Jadav, 2003) developed SLED (Service Level Enforcement Discipline for Storage) which is able to throttle very busy workloads responsible for performance degradation. SLED is decentralized and therefore can be used to manage large storage systems. SLED main aim is to ensure effectiveness of storage systems by directing resources to those flows that do not have. However this approach may cause poor performance in high priority classes. In addition SLED is implemented on an FC SAN. Peng and Varman, 2020) developed pTrans a framework for reservation guarantees based on directed acyclic graphs. However pTrans was found not to give accurate estimates for resource demand and available resources during run time which is crucial for dynamic resource allocation. Peng also developed Bqueue which is framework for providing reservations and limits on storage systems. However Bqueue uses a simple round robin scheduler which has an advantage of low overhead but as determined in literature simple round robin end up causing delay especially in environments where there are packets of varied sizes and priorities. Peng and Varman (2018) developed pShift which is a framework for providing I/O reservations and limits. Pshift uses estimates to provide optimal token distribution however it was found to be less scalable.

Motivated by the above discussion the study implements a scheduler shaper that achieves better bandwidth utilization and low latencies better than the conventional solutions available(Micha & Shah, 2020).The study formulates NUM mathematical model for the optimal utilization of network bandwidth. We solve this model using the Lagrange multiplier to find the optimal allocation value for each class of user. The study demonstrates through simulation that the proposed solution is efficient in the utilization bandwidth and reducing latency. The study implements the proposed solution on a router positioned between the initiator and the target where the algorithm runs to avoid multiple copies of the same algorithm running in the network. This is expected to reduce overhead of processing multiple copies of the algorithm and eventually increase network performance.

A major component of providing QOS in a network is the scheduler(Jin, Xia, & Guan, 2020). Packet schedulers are necessary in providing or ensuring bounded delay guaranteed bit rate and fair service allocation to all flows(Sanyoto, Perdana, & Biso, 2019). This can be achieved by solving the contention problem of a given resource and deciding on the sequence in which packets are transmitted from the node(Qian et al., 2017).

The router requires scheduling mechanisms to output packets arriving and ensure differentiated QOS(Siregar et al., 2020). The selection of an appropriate scheduling algorithm is key to providing QOS(Huang et al., 2020). A good scheduling mechanism should avoid unfairness between packets( Mathews & Glandevadhas, 2020). Low priority packets should not be starved. In addition a

good scheduler should provide good utilization constantly adjust the laws of their operation based on network statistics(Iswadi et al., 2019)

Packet schedulers are classified as either time stamped or frame based(Chin, 2021). Time stamped include the weighted fair queuing, worst case fair queuing, virtual lock and self-clocked fair queuing(Dong et al., 2019). The advantage of time stamped scheduling algorithm is that they provide tight latency bounds and provide good fairness. However they have high complexity(Siregar et al., 2020).

Frame based schedulers operate by rounds. Where each flow is served in a given round( Lin, Che, Jiang, & Wei, 2019).Weighted round robin, deficit round robin and elastic round robin are frame based schedulers .These schedulers are easy to implement, however they have high latencies(Salomo et al., 2018). This study focuses on DRR which is implemented in HTB.

DRR services flows in a round robin and succeeds in eliminating the unfairness of pure packet based round robin. However DRR latency become high when we have two flows with higher rate than the other. A good scheduling algorithm should have low computation cost, easy to implement, efficient and good fairness. DRR has a computation cost of  $O(1)$  though it does not have optimal fairness. This is because a flow continuously sends packets up to an amount of its deficit quantum which increases delay for smaller packets. Based on the deficiency of the DRR the study has put forward Hierarchical Priority Dynamic Deficit Round Robin scheduling algorithm (HPDDRR) technique that integrates traffic shaping and scheduling. HPDDRR uses a dynamic deficit counter that is generated based on the current network statistics for a given round. By using a

quantum for the highest rate high priority queue ensures high priority traffic is given preference hence achieving reduced delays. The hierarchy further ensures that flows are grouped based on classes which prevents interference.

### **2.32 Integration of QOS Technique**

TCP provides best effort which is unsatisfactory for providing QOS to storage traffic. Providing QOS guarantee require a number of functions to be performed such as performance isolation, bandwidth management and traffic shaping. There has been many proposed solutions for providing QOS in IP SANs.

Jaiman et al.( 2019) developed Heron which is an algorithm that is aimed at reducing tail latency when dealing with heterogeneous workloads. Heron does this by predicting which workloads will require large execution time. To reduce latency heron ensures that requests requiring small execution time are not scheduled behind those that require large execution. However this technique relies on predictions. If the predictions are wrong then resources may be wasted.

Peng et al.(2019) Developed fair-EDF to provide latency guarantees for storage servers. Results obtained showed that fair-EDF is able to provide fairness for heterogeneous workloads. However this mechanism was found not to be scalable. In addition fair-EDF lacks the mechanism to separate workloads with large execution time from those with small execution time.

Peng et al.(2019) also developed pShift which is a token allocation algorithm for balancing resources between storage servers. However it was found to have scalability problems. Peng & Varman (2018) came up with Bqueue which is a scheduling system that provides QOS reservations and limits. To handle

dynamic workloads bandwidth is computed at regular intervals. The problem with Bqueue was found to be that it uses tokens allocation as the only control measure for QOS.

Cui et al.,(2019) developed tail cutter as a mechanism for reducing latency in cloud storage systems. Tailcutter uses parallel request to cloud datacenters to reduce latency. However it only uses latency as a QOS control measure for storage. Peng & Varman (2020) Developed pTrans which is a framework for reservation allocation based on direct graph model. However pTrans is not able to give accurate estimates for resources available at run time which leads to inaccurate allocation of resources and wastage. In addition pTrans was found to increase with workload.

Techniques like PARDA(Gulati & Waldspurger, 2009), Argon(Wachs et al., 2007) use queue length management and disk time reservation for implementing proportional throughput fairness as a means for implementing QOS in storage area networks. Technique such as PriorityMeister(Zhu, Tumanov, Kozuch, & Ganger, 2017) and Triage(Karlsson, Karamanolis, & Zhu, 2005) use throughput performance isolation among competing workloads by use I/O throttling techniques such as Leaky bucket algorithm, deficit round robin and start-time fair queuing(SFQ) to manage how much throughput competing workloads receive. Other techniques like mClock (Gulati & Varman, 2007) and Pisces(Shue, Freedman, & Shaikh, 2012) support throughput QOS using maximum minimum fairness.

On latency there are two categories of techniques used, that is those that enforce average latency and those that aim at decreasing tail latency. Techniques that use mean latency SLO for QOS in storage systems include Facade (Lumb et al., 2003), Triage (Karlsson et al., 2005) and PClock (Gulati et al., 2007). In contrast the proposed system binds latency to each host based on its SLO however it dynamically adjusts latency based on the priority of workloads calculated from the network statistics hit ratios. This ensures optimal SLO compliance by each class of users.

Other studies like those done in cosTLO (Wu, Yu, & Madhyastha, 2015) and C3 (Suresh et al., 2015), use redundancy to reduce coverage and tail latencies. C3 (Suresh et al., 2015) reduces tail latency through dynamic redundancy and distributed rate control. Other techniques such as PriorityMeister (Zhu et al., 2014) and cake (Vulimiri et al., 2013) cut tail latency with the use of scheduling. PriorityMeister (Zhu et al., 2014) uses priority base I/O throttling as well as priority based scheduling for busy workloads.

Previous works reviewed in this research includes techniques only either for latency support and only those for throughput support. In contrast this research implements an integration of three techniques in an attempt to support throughput, latency, and jitter for users in IP SANs. To improve on the previous work this study integrates the three functions of performance isolation, bandwidth management and burst handling otherwise used separately in the previous studies. These integration is aimed at increasing throughput, reducing latency and reducing jitter



In addition most of the techniques reviewed in chapter one are either predictive, static or do not take into consideration the network statistics. Static techniques are not adaptive and therefore there is a high probability of network congestion. Predictive techniques have the downside that if the predictions are wrong it leads to low utilization of network resources. On the other hand decentralized techniques result in computation overhead where copies of the same algorithm run in multiple locations.

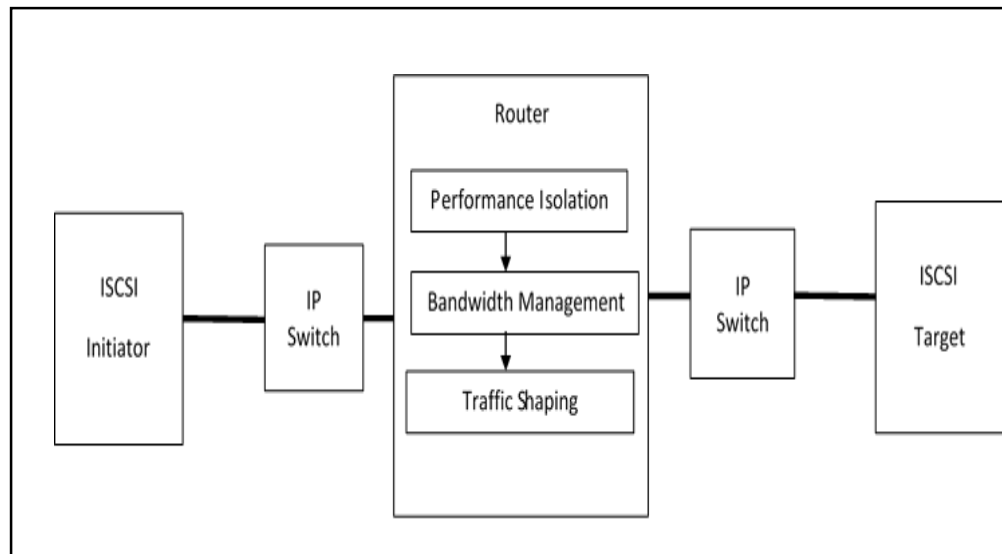
To further improve on the previous work this study incorporates the features of dynamism, use of network statistics to prioritize traffic and finally the use of a centralized mechanism to reduce the overhead experienced when multiple copies of the same algorithm are run. To measure the performance of the proposed system in providing QOS the metrics of throughput, latency, and jitter are used.

### **2.32.1 IQMIS**

The aim of this study was to develop an algorithm integrating and optimizing bandwidth management, performance isolation and burst handling features. The main idea was to achieve guaranteed QOS for storage traffic in IP SANs. The study first looked at performance isolation whose objective was to classify traffic and allocate dynamically network resources to various classes of traffic to reduce resource contention, thereby providing guaranteed QOS for storage traffic. The U32 classifier was used for the purposes of classifying traffic in order to achieve performance isolation(Hassan et al., 2017).The study came up

with enhanced list based packer classifier for performance isolation in IP SANs(ELPCIS) for the optimization of performance isolation

Bandwidth management and burst handling was implemented using HPDDRR algorithm an improvement on the existing DRR used in HTB. In order to have adequate control of the users' resource consumption, both read and write requests are throttled (Jiwu & Weimin, 2005). The target outbound commands translates to the initiators reads. On the other hand the targets incoming data translates to the initiators writes(Sheltami, 2019). The proposed algorithm was implemented in a central router that will lie between the initiator and the target (see Figure 2.19) thus providing centralized management of QOS that is absent in all other reviewed techniques. Centralized management of QOS is expected to eliminate any overhead that would have resulted from any attempt to coordinate separate algorithms running on storage devices as in the case with the existing solutions. In addition the centralized algorithm was able to maintain information about network resources and their users and dynamically adapt regulating values to rapidly changing workloads.



*Figure 2.19: IQMIS Architecture*

### **2.33 IP Networks Validation**

Validation is the process of determining the degree to which a simulation model and its associated data are accurate representation of the real world from the perspective of the intended use of the model(Saunders, Lewis, & Thornhill, 2007). In IP networks generally, validation helps in predicting network performance and also determine if the network works correctly. There are different scopes of validation. That is the timing of the validation and the approach of validation(Mikac & Horvatić, 2019). The scope of validation determines the level to which the validation is done. With scope validation there are three possibilities, which includes; unit testing, functional testing and verification(De Sio, Azimi, & Sterpone, 2020). Unit testing is used to test the correctness of network devices configuration for example Domain Name service (DNS) configuration. While unit testing is easy to implement it does not indicate the end to end behavior which is crucial for determining network

performance. On the other hand functional testing is used to check end to end behavior for example if packet reaches its destination. This means that unit testing can guarantee network behavior(Hashemian, Carlsson, Krishnamurthy, & Arlitt, 2020). However functional testing does not offer completeness due to the fact that even though a single packet does not reach the destination the other packets may reach the destination(Scazzariello, Ariemma, & Caiazzi, 2020). This is where verification comes in. Verification ensures correctness of all possible scenarios in the defined scope. It mainly takes the formal mathematical approach. Since verification offers strong guarantees it offers network managers and researchers a strong confidence to the performance of a network(Jose-Ignacio, Serrano-Martinez, & Monica, 2019).

When it comes to timing, validation can be done either post deployment or pre deployment. In post deployment, validation checks whether the network set up has the intended impact(Bonati et al., 2021). In the context of this research the manipulation of bandwidth and block size was done to check if they have the intended impact on the results. On the other hand pre deployment validation is meant to shield the network system from errors that may occur during the deployment. It therefore provides a higher degree of protection from errors as compared to post deployment validation(Xian Zhang & Peng, 2019).

### **2.33.1 IP Network Validation Approaches**

There are four main approaches to IP network validation. That is text analysis, emulation, operational state analysis and model based analysis. The four

approaches as used based on intended purpose. However an integration of two or more approaches can be used to achieve better results(De Sio et al., 2020).

Text analysis validation is done by scanning network configuration without the consideration of the semantics. For instance, text analysis can check if the network device has IP address configured(Hewage, Ahmad, Mallikarachchi, Barman, & Martini, 2022). Text analysis is not able to determine a network behavior and therefore considered to be inadequate when considering network performance. It is mainly used when the other methods are not available(Parks & Peters, 2022).

Emulation validation employs a test bed with real physical device or virtual devices(Bonati et al., 2021). In an emulation validation, a network manager or researcher can deploy the intended configuration to determine the resulting network performance(De Sio et al., 2020). Emulations are used when it is difficult or inconveniencing to build a full replica of the network due to limited resources or just for research purposes(Scazzariello et al., 2020).

In operational state analysis a network is validated in its production. The key advantage to this is that validation is done on the actual network. However it has some disadvantages(Amid et al., 2020). One is that errors are transferred to the actual network since its post deployment. Secondly it can't be used to test for large scale failure scenarios as it would disrupt the entire network(Parks & Peters, 2022).

Model based analysis validation builds a working model for a working network behavior(Binder, 2018). It checks the behavior of a network using a range of

scenarios. It employs the strategies of simulations and mathematical models(Cedillo, Insfran, Abrahao, & Vanderdonckt, 2021). Model based analysis is the only one that can implement verification since it employs evaluation of more than one scenario(Saha et al., 2020).

The study adopted and integration of text based analysis, emulation and model based analysis. Text based analysis was used to check for the correct configurations done during experiment setup. This is important so as to sure all the initiators read the same block size and that experiments stated and run for the same period of time. An emulation test best was then setup using virtual machines as well as physical machines. Virtual machines were used to save on costs of buying physical machines. Model based analysis was used to test the IP SANs on different scenarios that is using the block sizes of 4KB, 64KB and 1MB. In addition two scenarios were considered that when using IQMIS and when using best effort.

## **CHAPTER THREE: METHODOLOGY**

### **3.0 Overview of the Chapter**

This chapter presents the research methodology used in this study. The key activities and their impact on the research are highlighted. The research philosophies, strategy, approaches and techniques, data collection techniques, quality control and ethical issues are presented.

### **3.1 Research Philosophy**

This study adopted positivism philosophy. Saunders, Lewis and Thornhill (2009) defined research philosophy as what the researcher is doing when carrying out the research and developing knowledge in a particular field in relation to how data should be gathered, analyzed and used. The philosophy used in a particular research informs on the vital assumptions on how the researcher views the world. The philosophy used in a research is determined by the practical concerns and the view of the relationship between the existing knowledge and the procedure of developing new knowledge (Bryman et al., 2008). Lewis and Thornhill(2009) hold that there are four categories of philosophies that can be adopted in a research; positivism, interpretivism, realism and pragmatism. Positivism explores the causal relationship between variables. It uses lab experiments, mathematical modelling and survey methods as research designs.

In this research a number of reasons informed the decision to adopt the positivism research philosophy. First is that the research was bent on the optimization the techniques of performance isolation, bandwidth management

and traffic shaping. In performance isolation the research hypothesized that without optimization of the classification process it would lead to the performance degradation of the system. Experiments were setup using the proposed solution and without the proposed solution and the quantitative results compared. In bandwidth management and burst handling mathematical models were formulated and used to estimate the optimal bandwidth for each class of user based on the hit ratio. The same mathematical models were used to estimate the optimal quantum for packet scheduling for burst handling. Finally in the integration of the previously mentioned QOS techniques, the research looked at the causal relationship between the techniques of performance isolation, bandwidth management and burst handling on IP SANs QOS. Furthermore the researcher sees himself as a neutral observer and it is expected that researchers using the same instrument should reach the same conclusion.

## **3.2 Research Design**

Greener(2008) defines research design as a structured approach of investigation applied to obtain reliable answers to research questions with regards to research problem. This section presents the research strategy adopted in the study as well as the data collection techniques used.

### **3.2.1 Research Strategy**

In this study, an experimental research strategy was adopted. Experimental research is characterized by more control over the research environment where variables are manipulated to observe their effect on other variables. Experiment is a research strategy that involves finding causal relationships between



variables through the effect of manipulating one variable on another (Saunders et al., 2009). It is suitable for phenomenon with known variables or initial hypothesis that aimed at testing or manipulating a theory (Greener, 2008). It is also used to test and answer ‘How’ and ‘why’ research questions and lies in the positivism philosophy domain (Winterton, 2008).

The experimental strategy was used because it provides greater control over the research environment where bandwidth management, burst handling and performance isolation was manipulated to observe their effect on IP-SANs QOS (Sekaran, 2003). The goal of experimental research design is to explain effects and determine a causal relation between variables. In this study experiments were set up to determine the effect of bandwidth management, burst handling and performance isolation on IP-SANs QOS. The purpose of the experiment was to study causal links between bandwidth management, burst handling and performance isolation on IP-SANs QOS.

### **3.2.2 Data Collection and Analysis**

To characterize system performance in the course of experiments, a variety of different tools were be utilized. The most important task is monitoring QOS performance metrics about the I/O requests transmitted through the system (Valenzuela, Monleon, Esteban, Portoles, & Sallent, 2003). Parkdale was used monitor throughput and latency while wireshark was used to monitor hit ratio. Parkdale is a disk benchmarking tool used to test the performance of storage device. The decision to use Parkdale as a tool for traffic generation is due to the fact that it allows for block level access sued in SANs.

Wireshark is free open source packet analyzer that was developed by Gerald Combs in 1998. It is mainly used for network troubleshooting as well as for research. This study settled on wireshark as a packet analyzer since results can be refined using a display filter. This was important since the study looked at the ISCSI protocol which was used as filter. In addition, Wireshark is able to capture packets from simulation which was the source of data used in the study. Appendix B depicts the wireshark interface showing the packets generated using the ISCSI protocol. Traffic configuration Linux command `tc -s qdisc dev eth 0` was used to show class statistics with information under each class. The data collection was started, stopped, post-processed and collected using Parkdale and wireshark and `tc -s qdisc dev eth 0`.

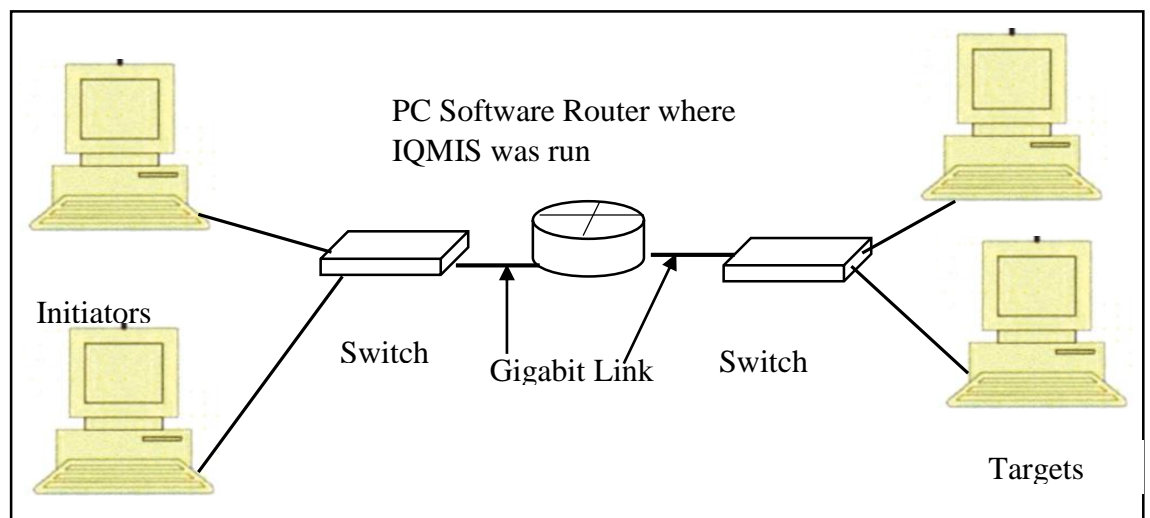
All experiments were run for a period of 200 seconds. Experiments were run three times and the averages recorded. For performance isolation the number of rules were varied from 18 rules (best case) and 1152 rules (worst case). For all the experiments the IO size was varied from 4KB (no congestion), 64KB and 1MB (with congestion). The rules and file sizes were chosen based on what was found used in literature reviewed so as to provide ground for comparison purposes between IQMIS and existing solutions. All experiments were setup with the configurations with the IQMIS and without the proposed solution (Best Effort).

Since the study generated quantitative data descriptive statistics were used to analyze data. Using Microsoft excel spreadsheet quantitative data was

aggregated and analyzed using descriptive statistics. The results are then presented in tables accompanied by explanations and discussions.

### 3.3 Experimental Setup

This section outlines the experimental setup employed during experimental design. Areas covered include a description of the how network traffic was generated, performance isolation optimization, bandwidth management optimization burst handling and the integration of the techniques into an optimization technique for QOS management on IPSANs. The architectural layout of the IQMIS is as presented in Figure 3.1.



*Figure 3.1: Experimental Setup for IQMIS*

However, for purposes of possible replication and clarity of this study, Table 3.1 shows the hardware and software specifications for the experiment set up.

**Table 3.1: Hardware and Software specifications to be used for the Experiment**

<b>Equipment</b>	<b>Specification of the Hardware and Software</b>	<b>Quantity</b>
OS(initiator)	Windows 7	4
OS(target)	Windows 7	2
OS(router)	Ubuntu version 16.0	1
Memory(initiator)	2GB	1
Memory (target)	2GB	1
Memory (router)	2GB	1
Hard disk (initiator)	500 GB	1
Hard disk (target)	500 GB	1
Hard disk (router)	500GB	1

### **3.3.1 Traffic Generation**

Parkdale was used to simulate read and writes to the storage devices .Attempt to read or write varied IO sizes by the initiators was used to generate concurrent workload of a wide-ranging variety. A generated workload can be reproduced using the same IO size (Paulraj & Kannigadevi, 2019). While the key task of the Parkdale tool is to produce workloads, it can indicate the speed in which the read or write operations are being carried out for every I/O request. This is an additional advantage. In the real implementation of IP-SANs, I/O requests are generated by applications running on the iSCSI initiator machines (Han et al., 2019). The initiator is a server that utilizes the storage resource. Appendix C depicts part of Parkdale interface with the configuration of reading or writing a file size of 30MB using 64KB block size. From appendix C it is clear that allows

for a lot of flexibility choosing the block size as well as the file size. This allowed for the researcher to be able to manipulate block size so as to make the intended point. Manipulation of variables is a key strength of experimental research design which was adopted in this study and choosing of Parkdale as a tool enabled the exploitation of this key feature of variable manipulation in experimental research design.

### **3.3.2 Optimization of Performance Isolation**

Classification was done using the U32 classifier available in the Linux kernel due to its robustness. The study settled on U32 classifier as it is the most robust in terms of performance compared to the others available. However during the classification process , U32 classifier searches for rules in a linear order as it looks for matches for a specific packet(Yakti & Salameh, 2019). However linear search does not perform well especially when the number of rules increases. To optimize the performance of the U32 classifier a rearrangement of the rules based on the hit ration was sought. This sorting was to ensure that those rules that have a high hit ratio will fall at the top. This is expected to reduce the number of matches required for packet classification to be done, while maintaining the reliability of the original classification policy. The reliability is maintained if the reorganized and original rules constantly produce the same results. To optimize the classifier further the go statement was used to jump to rules hence splitting them to sub rules which reduces the time complexity of the search from  $O(N)$  to  $O(\frac{N}{p})$  where  $p$  is the number of partitions(Hamed & Al-shaer, 2018). Use of the go to statement was to split the rules to form a

hierarchical tree like structure. Using a tree structure to classify traffic further reduced the number of matches hence improving on the performance. of the proposed system(Acharya, Znati, & Member, 2008).For performance isolation optimization design, the study came up with an Enhanced List Based Packet Classifier for Performance Isolation in Internet Protocol Storage Area Networks(ELPCIS).

### **3.3.3 Optimization of Bandwidth Management and Burst Handling**

The Hierarchical Token Bucket (HTB) qdisc was used to set up a tree hierarchy of classes and their bandwidth(Salmani, 2015). Since HTB uses DRR scheduling mechanism which is known to have high latency and also leads to low bandwidth utilization of resources especially when there are flows in the same queue with different rates (Sarmah & Sarma, 2019). The optimization of bandwidth management and burst handling was meant to eliminate the weakness of DRR by adding dynamism to the quantum selection based on network statistics and use of hierarchy of queues instead of FIFO queues which do not offer prioritization.

The study adopted scheduler/shaper named hierarchical priority based dynamic deficit round robin (HPDDRR) for optimization of bandwidth management and burst handling. HPDDRR employs the technique of hierarchy structure of flows to reduce the number classes' queues and reduce the processing delay of the queues. Secondly the study implemented the use of priority calculated from hit ratio of flows to calculate the deficit quantum which ensures that the quantum is dynamic based on network statistics. The hierarchical structure allows for isolation of traffic between flows. In addition so as to retain the complexity of

O (1), the hierarchical structure was configured to have one level. The property of dynamic counter was meant to ensure packets get transmitted as much as possible in every round robin during DRR as the deficit will be calculated based on highest rate of the highest priority queue. This improves on best effort static bandwidth allocation since a class will be allocated bandwidth based on the current network requirements. The feature of traffic classification further improves on latency experienced by DRR as packets of similar rates are put in the same queue which reduces the waiting time which might be high for low rate packets when mixed with high rate packets.

### **3.3.4 Integration of Performance Isolation, Bandwidth Management and Traffic Shaping**

In the integration of performance isolation, bandwidth management and burst handling the study came up with an Integrated QOS Management Technique for Internet protocol Storage Area Networks (IQMIS). The IQMIS algorithm incorporating the features of performance isolation, bandwidth management and burst handling was developed and is as presented in algorithm 6.

The input for the IQMIS is packets and rules for packet forwarding. The proposed algorithm takes packets as input, puts the packets into classes then allocate bandwidth to the specified classes. In order to implement traffic shaping, bandwidth borrowing was configured to ensure bursty packets are able to use extra bandwidth(Yakti & Salameh, 2019). If no unused bandwidth is available a class will transmit its committed rate waiting for any unused bandwidth. By ensuring no particular class uses more than its share of resources provides QOS guarantees (Hemke et al., 2019). The algorithm was run in a

central software router which sits at the core of the network. Centralizing the management of QOS was implemented to remove the overhead of having individual algorithms running in all the storage devices as observed in the reviewed solutions, therefore centralization of the management of QOS did contribute to the improvement of the overall system performance(Inumula, 2015).

### **3.3.5 Validation of the IQMIS**

In order to facilitate validation IQMIS its features were prototyped. The IQMIS was designed and implemented as a shell script. Simulation experiments were set up to evaluate the designed algorithm on QOS metrics as defined by the ITU (see tables 2.2, 2.3 and 2.4). That is the metrics of latency, throughput and jitter. The experiment setup equipment's was as illustrated in Figure 3.1. The router was put at the middle where the IQMIS was run. Simulation of packets from the initiator to the target was done to test the performance of the algorithm against the QOS metrics. The major benefits of a simulation-based evaluation are that it is scalable, repeatable, and the network conditions can be controlled(Saunders et al., 2009). Experimental tests were carried out incorporating the IQMIS and in other instances not incorporating the IQMIS that is best effort. This was done to determine the corresponding changes in the IP SAN performance when IQMIS was used and when not used.

### **3.4 Quality Control**

The following sections look at how quality control was ensured. It includes a discussion of how validation of tools used was done. It further outlines how the reliability test was carried out.



### **3.4.1 Validity**

According to Kothari C.R, (2004) validity is the most important criteria for indicating the degree to which a measuring instrument measures what it is supposed to measure.in other words it is the extent to which a test or experiment provides an accurate representation of its real equivalent(Surucu & Maslakci, 2020). A valid simulation is the precise depiction of the simulated task within the perspective of research objective. There are two types of validity that are used in simulation design experiments. That is face validity and construct validity(Guido-Sanz et al., 2022).

Face validity is the personal view of how realistic a particular simulation is. Face validity is dependent on the observed features of the simulation as well as the functional and structural aspects(Thi & Nha, 2021). Consequently the technical design of the simulation is a key determinant of face validity. In subjecting the tools to validation, the process started by discussion with the supervisors of the study who scrutinized all the tools to assess their appropriateness in addressing critical issues in the study (Greener, 2008).

A simulation can have a face validity at the same time irrelevant as it could have no correlation with actual performance of an experiment. Face validity is assessed by the user's feedback about how good is it a representation of the real task. Consequently face validity is not clearly tested and may not be a vital contributor to the experimental success(Hehman, Calanchini, Flake, & Leitner, 2019).

Construct validity is more objective than face validity as it determines the degree to which a simulation provides a precise representation of the real world environment. A simulation with good construct validity should be sensitive to variations in performance as the variables are manipulated (Coleman, 2022). Predictive validity which is related to construct validity is concerned with how the simulation accurately predicts prospective real world performance. In the study construct validity was used as it looks at how close the proposed system achieves close results from what is in theory.

Validity was established through testing IQMIS and Best Effort where the experimental results were checked against the objectives.

### **3.4.2 Reliability**

Test-retest technique of reliability testing was employed whereby experiments were done repeatedly to allow for reliability testing (Sekaran, 2003). To ensure reliability of the data collection tools, each measurement was conducted at least thrice and where the variance was large, the results were nullified until a consistent result was obtained.

### **3.5 Review of Objectives**

Table 3.2 presents a summary of the objectives of the study and how they were addressed in the study.

**Table 3.2: Summary of Objectives**

<b>Objective</b>	<b>Question</b>	<b>Data Source</b>	<b>Method</b>	<b>Analysis</b>
i. To analyze techniques for providing QOS for IP networks.	i. How are the techniques used to provide QOS in IP networks?	Literature	Systematic structured literature review approach	Content Analysis
ii. To Optimize QOS techniques for performance isolation, bandwidth management and burst handling for QOS in IP SANS.	ii. How can the performance isolation, bandwidth management and burst handling QOS techniques be optimized for providing QOS management for IP SANS?	Simulation Experiments	Simulation Experiments	Metric Analysis
iii. To develop an integrated QOS management technique for IP-SANS.	iii. How can integrated QOS management be developed using performance isolation, bandwidth management and burst handling?	Simulation Experiments	Prototyping	Metric Analysis
iv. To Validate the integrated technique for providing QOS management in IP-SANS.	i. How valid is the developed technique for managing QOS in IP SANS	Simulation Experiments	Simulation Experiments	Metric analysis

### **3.6 Ethical Considerations**

The experiments was set up in the Meru University computer laboratories therefore permission to carry out the research was granted by the university. Since the study did not entail dealing with sensitive human data, no special authorizations was required. However, the National Commission for Science, Technology and Innovation (NACOSTI) research permit was obtained. Appendix A depicts the research permit. In addition approval from the directorate of post graduate studies of Meru University of Science and Technology.

### **3.7 Summary**

This chapter has described the methodological procedures that were used to answer the research question posed in chapter one of this study. Justification has been offered explaining the reasons for selecting the research philosophy, research design. Data collection methods are discussed and relevant issues relating to the study's reliability and validity were presented. Similarly, a review of the objectives of the study were presented in this chapter. The remaining chapters of this thesis will thus seek to present the detailed results that met each of the objectives presented, chapter four presents performance isolation optimization, chapter five presents optimization of bandwidth management and burst handling and chapter six which presents the overall integrated technique for QOS management in IPSANs. The thesis is concluded in Chapter Seven where the recommendations and future work are also detailed.

## CHAPTER FOUR: PERFORMANCE ISOLATION OPTIMIZATION

### 4.1 Chapter Overview

This chapter presents an overview of approaches used to implement performance isolation in SANs, the problem definition, proposed solution, optimization of performance isolation and finally a summary of the chapter.

### 4.2 Problem Definition

For a policy consisting of a list of  $n$  unsorted rules  $r_1, r_2, \dots, r_n$ . A packet  $d_i$  is said to match rule  $r_i$  if the fields of rule  $r_i$  match the header field of packet  $d_i$ . A packet  $d_i$  may match any of the rules  $r_i, I = 0, 1, 2, \dots, n-1$ . If the matching rule is found on the  $i$ th position then  $i+1$  comparisons will have been made. Thus the average number of comparisons for a successful search denoted as  $C$  is

$$C = \frac{1}{n} \sum_{i=0}^{n-1} (i + 1) \quad 4.1$$

$$\text{Which translates to } \frac{n(n+1)}{2} \div n = \frac{n+1}{2} \quad 4.2$$

From equation 4.1 it is clear that the time complexity is  $O(N)$ .

The optimization problem is thus to arrive at a legitimate rule order that results in the optimal cost  $C$ .

To partially achieve the second objective of this study in regard to performance isolation, the following sub-objectives were formulated and ELPCIS presented in the next section;

- i. Optimize the U32 Linux classifier using the technique of sorting rules, partitioning rules and linear tree rule structure.
- ii. Evaluate the optimization of the classifier using the metrics of time complexity, latency, throughput and accuracy.

- iii. Use the classification technique to bind resources to users' class to implement performance isolation.
- iv. Evaluate performance isolation of the classifier in the IP SAN environment using response time and throughput metrics.

### **4.3 Proposed Solution**

Since this study used the U32 classifier which is linear search based and a classifier performance is affected by the speed of matching rules to packets. It is important to optimize linear search to arrive at the minimum number of packet matches required for classification decision using linear search(Balogun, 2019). For this purpose the study uses three techniques for linear search optimization namely re-ordering the rules, splitting and then structuring the rules in a sequential tree like structure to remove the anomalies and further reduce the matching time(Bhaumik, Saha, & Das, 2016). To further optimize the search, the study uses jump search to move from one linear tree rule structure to another. The rules are reordered to ensure those rules with high frequency appear at the top. The splitting of rules is meant to facilitate the use of jump search where the search process can jump to a given list perceived to have more frequently used rules. The structuring of the rules in a tree like structure is meant to eliminate the anomalies of rule positioning, redundant rules and shadowed rules as well as increase the search speed. The root structure can contain any number of root nodes and any number of child nodes(Lin & Masa, 2019). Classification action is performed at the leaf nodes. The applied optimization techniques was found to reduce the number of matches required for a packet to be classified(Border, 2018). To solve the optimization problem of performance isolation the study

came up with an enhanced list based packet classifier for performance isolation in IP SANs(ELPCIS).The working of ELPCIS is as illustrated in Figure 4.1

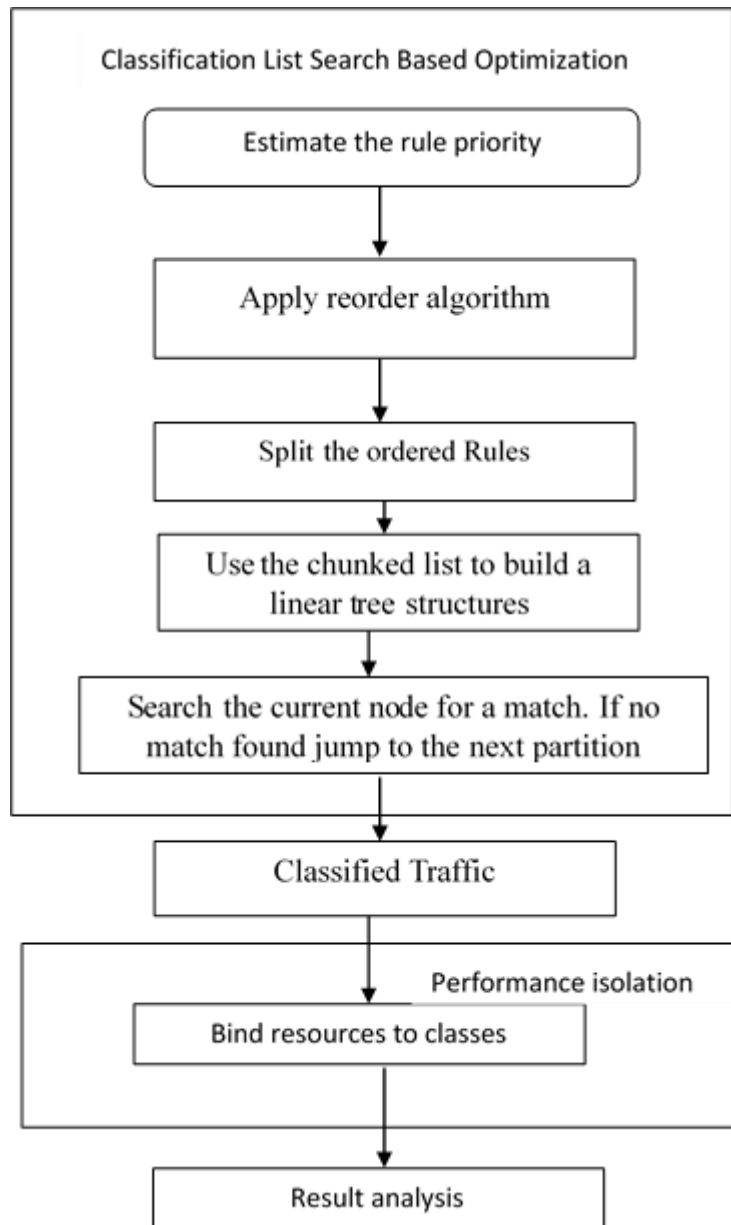


Figure.4.1: ELPCIS Methodology

Figure 4.1 illustrates the process of performance isolation by ELPCIS where it begins with estimation of the rules priority. Rules priority is important since it indicates the placement of rules in the classifier. Next ELPCIS sorts the rules based on priority with the high priority rules appearing at the top. After rules

reordering ELPCIS partitions the rules into chunks. Packets are matched in the chunks created with the algorithm jumping from one partition to another. After the packets are put into their corresponding classes resources are assigned to them which ensures they don't use more than allocated unless there is excess. This ensures greedy classes of users do not affect the performance of well behaving classes hence implementing performance isolation.

#### **4.3.1 Packets Feature Extraction and Selection**

The selection of features for classification is a critical step since it is vital to only select relevant features. If many features are used for classification, it creates classification overhead due to many look up required for the features(Zhi Liu, Sun, Zhu, Gao, & Li, 2017). Packets in an IP network can be identified using header information. Feature selection algorithms like filter model and wrapper model are used to extract features for classification when the features themselves are not clear. However for this study which used U32 classifier for classification, the decision of features to use was easy. This is because U32 supports only the features of source IP address, destination IP address, source port, destination port and transport protocol as features for classification. To improve previous solutions for example the solution that was proposed by Zhao, Shimaie, & Nagamochi (2004) , this study used the five features of packets classification instead of the three used by Zhao, Shimaie, & Nagamochi (2004). The adoption of five features was to increase the granularity of the classifier. In addition research has shown that there is no perfect classification technique. Traffic classification in modern links require tradeoffs between accuracy, performance and cost(Dainotti & Claffy, 2012). The main aim was to



optimize the classifier performance to greatly reduce any delays that might be caused by the classification process and interfere with quality of service for users. Packet features used for classification and their description are as illustrated in Table 4.1.

**Table 4.1: Packet Features Used for Classification**

<b>Feature</b>	<b>Description</b>
Source IP	IP Address of the initiator
Destination IP	IP Address of the target
Source port	Port of the initiator
Destination port	Port of the target
Protocol	Transport protocol(ISCSI)

#### **4.3.2 User Classes and Operational Metrics**

The information technology industry classifies storage users as either task users, knowledge users or power users. The task users are employees in an organization performing repetitive tasks within a small set of applications, which are usually not CPU and memory-intensive. Knowledge users are employees in an organization whose tasks include accessing the internet, using email, and creating complex documents, such as spreadsheets. Power Users are employees who run CPU and graphic intensive applications. All these users require a certain level of operational resources (Liveoptics et al., 2019). Table 4.2 illustrates the operational resources required per user for the corresponding class.

**Table 4.2: Estimated Operational Resource Per User**

<b>Class of user</b>	<b>Memory</b>	<b>Disk space</b>	<b>IOPS</b>
Task user	1GB Memory	25 GB Disk space,	5 IOPS
Knowledge user	2GB Memory	40 GB Disk space,	10-20 IOPS
Power user	4GB Memory	40 GB Disk space	25 IOPS

Table 4.2 shows the estimated operational resources per user. The table further shows that power users require more operational resources in terms of IOPS, memory and disk space. Storage level agreement (SLO) is a quality of service aspect that can be used for measuring performance of a storage system or storage service provider. A SLO is a combination of one or more QOS metrics with their corresponding values(Storage Performance council, 2019). Metrics for measuring storage performance include IOPS, latency, response time and throughput. The following are the explanation for concepts of storage performance metrics(Storage Performance Council, 2019).

Block size is a unit of data that is read during an I/O operation. It is a payload size of a single unit. Comparing it to a highway it is the size of vehicles in a highway some are small like the cars while others are big like trucks(Liveoptics et al., 2019). The block size impacts throughput. For example a 24KB block has 6 times the amount of data as the 4KB block. Block sizes are dictated by the operating system and the type of application. Block sizes impact storage performance regardless of the type of storage system in place whether it is n FC SAN or an IP SAN(Storage Performance Council, 2019).

However in reality most applications draw a unique mix of block sizes at any given time depending on the activity (Liveoptics et al., 2019). For comparative purposes the study used block sizes as used by other researchers. Other considerations include; since the windows operating system uses a default block size of 64KB it was included as the experiments were run on windows operating system. Again the block sizes were chosen based on the options available in the Parkdale tool used for traffic generation. Therefore for the purpose of this research the study block sizes of 4KB, 64KB and 1MB were for the simulation of reads and writes. IOPs (input/output operations per second) is the standard unit of measurement for the maximum number of reads and writes to non-contiguous storage locations. Throughput is a product of IOPs and block size. Their relationship is as illustrated in equation 4.3 (Liveoptics et al., 2019).

$$\textit{Throughput} = \textit{IOPs} * \textit{Block Size in bytes} \quad 4.3$$

Queue depth is the number of I/O commands that can be queued at a time on a storage controller at the initiator side or at the target side. If the storage controller queue depth is reached, the storage controller rejects incoming commands by returning a QFULL response.

In a configuration with multiple initiators all hosts should have similar queue depths. This prevents hosts with small queue depths from being starved by hosts with large queue depths. For small midsize storage area networks a queue depth of 32 is recommended. Equation 4.4 is used to calculate the queue depth (Liveoptics et al., 2019).

$$\textit{Queue depth} = \textit{IOPs} * \textit{Response time} \quad 4.4$$

Table 4.2, Equation 4.2 and Equation 4.3 were used to calculate the SLO for the system that was emulated using the experiments. The SLO that was derived is as depicted in Table 4.3.

**Table 4.3: SLO for Classes of Storage Users**

<b>Class of user</b>	<b>IOPS</b>	<b>Throughput for Block size 4KB</b>	<b>Throughput Block size 64KB</b>	<b>Throughput Block size 1MB</b>	<b>Response time for QD32</b>
Task user	5 IOPS	20kb/s	320kb/s	5000kb/s	6.4 ms
Knowledge user	10-20 IOPS	40-80kbs	640-1280kbs	10240-20480kb/s	3.2 ms
Power user	25 IOPS	100kb/s	1600kb/s	25000kb/s	1.3 ms

Therefore from Table 4.3 the SLO for various classes of users was derived based on the IOPs and block size. The values for the SLO are throughput in kb/s followed by IOPS and latency. For a block size of 4KB the SLO for task, knowledge and power users is as follows; task users(20kb/s,5IOPS,6.4MS),Knowledge users(60kb/s,15IOPS,1.6-3.2ms) and power users(100kb/s,25IOPS,1.3ms).The same case applies for 64kb and 1 Mb block sizes.

To generate a rule list in the study used random Source IP address, Destination Port, Destination IP Address, Source port, protocol and actions to generate Table 4.4.

**Table 4.4: Sample Classifier Policy with 325 Rules.**

<b>Rule</b>	<b>Dest IP add</b>	<b>Dest Port</b>	<b>Src IP add</b>	<b>Src port</b>	<b>Protocol</b>	<b>Action (Assign class )</b>
R1	192.168.2.4	3260	192.168.1.1	ANY	ISCSI	Power user
R2	192.168.2.4	3260	192.168.1.2	ANY	ISCSI	Knowledge user
R3	192.168.2.4	3260	192.168.1.3	ANY	ISCSI	task user
R4	192.168.2.4	3260	192.168.2.4	ANY	ISCSI	Power user
R5	192.168.2.4	3260	192.168.1.3	ANY	ISCSI	Task user
R6	192.168.2.4	3260	192.168.1.1	ANY	ISCSI	Power user
R7	192.168.2.4	3260	192.168.1.2	ANY	ISCSI	Knowledge user
R8	192.168.2.4	3260	192.168.1.3	ANY	ISCSI	Task user
R9	192.168.2.4	3260	192.168.1.1	ANY	ISCSI	Power user
R10	192.168.1.1	3260	192.168.2.4	ANY	ISCSI	power user
R11	192.168.1.2	3260	192.168.2.4	ANY	ISCSI	Knowledge user
R12	192.168.1.1	3260	192.168.2.4	ANY	ISCSI	power user
R13	192.168.1.1	3260	192.168.2.4	ANY	ISCSI	Power user
R14	192.168.1.1	3260	192.168.2.4	ANY	ISCSI	Power user
R15	192.168.1.3	3260	192.168.2.4	ANY	ISCSI	Task user
R16	192.168.1.1	3260	192.168.2.4	ANY	ISCSI	Knowledge user
R17	192.168.1.3	3260	192.168.2.4	ANY	ISCSI	Task user
R18	192.168.1.1	3260	192.168.2.4	ANY	ISCSI	Power user
.	.	.	.	.	.	.
R325	Any	Any	Any	ANY		Drop

From table 4.4 it depicts some of the problems associated with linear based classifier. Firstly it is noted there is redundancy caused by R1 and R18, R2 and R7, R12 and R18. This increases the search time for the classifier. In addition R325 is placed inappropriately could shadow all the rules in the classifier.

#### **4.4 Performance Isolation Optimization Techniques**

The following sections look at the various techniques used for optimization of the classification process in order to reduce the time required for matching the rules to packets.

##### **4.4.1 Rules Priority Estimation and Sorting**

For optimal performance rules with the greatest hits are placed at the top and those with fewer hits follow (Danielsson, Seceleanu, Jagemar, Behnam, & Sjodin, 2019). An experiment was performed to determine the rule hits ratio. Results of the hit ratio experiment is as illustrated in Figure 4.2

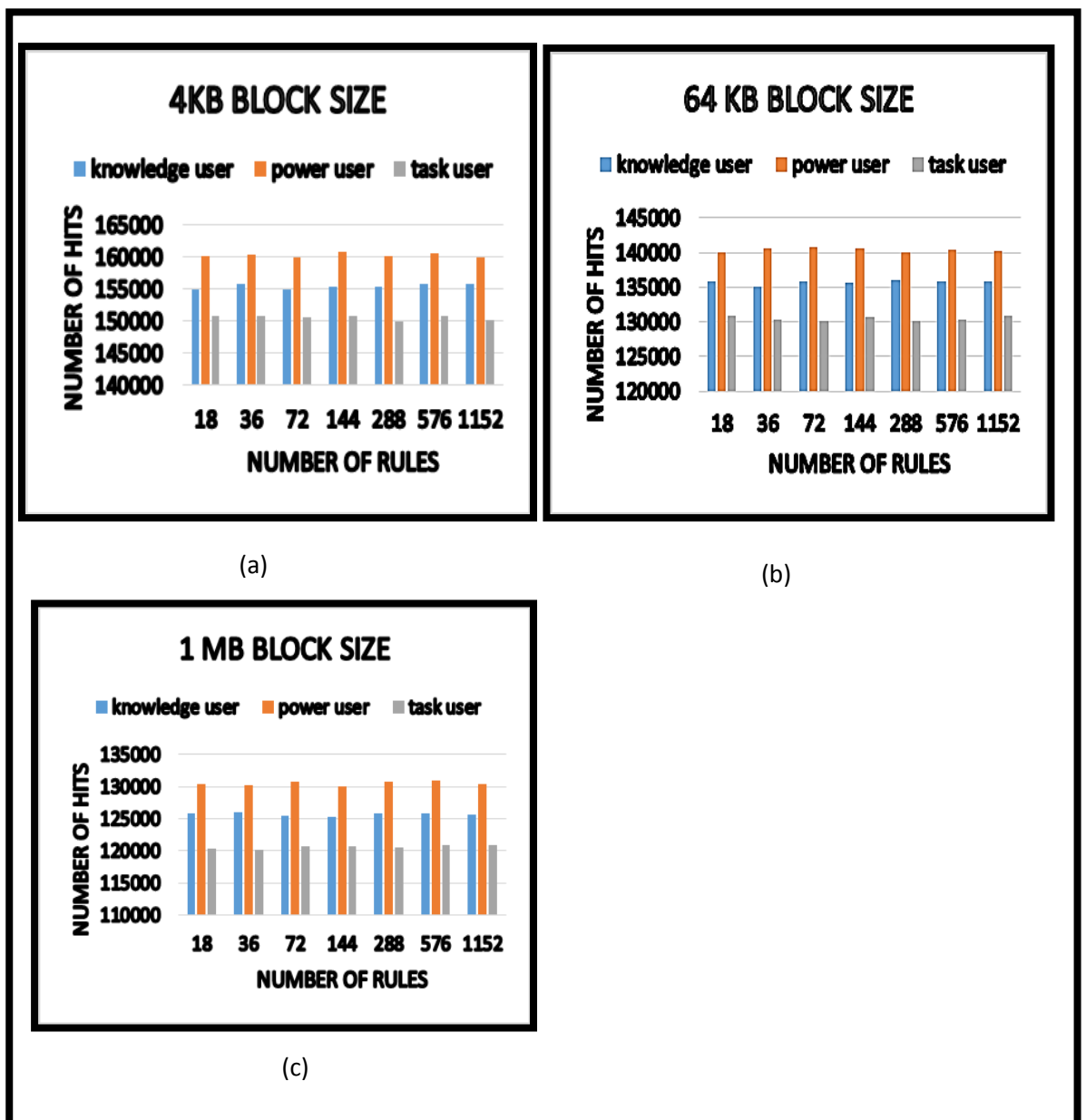


Figure 4.2: Rule Hits Distribution over Varied Block Sizes

Figures 4.2 shows the distribution of the hit for rules over an attempt to read/write files of size 4GB with blocks of size 4KB, 64KB and 1MB. The results show that the smaller the block size the larger the number of hits. These phenomena is due to the fact that the smaller the payload the larger the number

of operations for the read and writes hence the higher the hit rates. The second observation is that heavy hit rules are experienced from the power users followed by the knowledge users and lastly the task users. This suggests that, in order to reduce the operational cost of the classifier, heavy hit rules, should be placed at the top of the rule list. This will ensure more common rules are matched very fast therefore increasing the performance of the classifier. To establish the severity of the arrangement of rules experiments were set up with and without the reordering. Those experiments that include the proposed solution have put into consideration the arrangement based on rule hits while the others have not. The last observation is that there is no correlation between the number of rules and the hits ratio(Lin & Masa, 2019).

**Algorithm1: Sorting Algorithm**

**Input: An Array of Rules and their Priorities**

**Output: Sorted Array of Rules and their Priorities**

1. **For**(  $i = 1; i < n; i++$ )
2.     **If** (  $p_i > p_{i+1} \text{ AND } i \cap i+1 = \emptyset$  **then**
3.          $temp = r_i$
4.          $r_i = r_{i+1}$
5.          $r_{i+1} = temp$
6.     **Endif**
7. **End for**

Algorithm 1 accepts rules and their priority which then it sorts them based on the priority. High priority rules are placed at the top while low priority rules are place at the bottom of the list. The placement of high priority rules at the top ensures they are matched first since they hit regularly which ensures the reduction of search time.



#### 4.4.2 Partitioning the Rule List

Next the jump searching algorithm is used to split the rules into partition of size  $m$ . In its simplest implementation the jump search algorithm operates with jumps the size of the square root of the number of items (Nam et al., 2020). This fact results into equation 4.5.

$$m = \sqrt{N}. \quad 4.5$$

From equation 4.5,  $m$  is the jump search for a list of size  $N$ .

#### Proof 1

For any jump search for a list of size  $N$ , given that the number of rules in every jump size is  $\frac{N}{m}$ . For  $m$  elements in a block the, and beginning at 0 algorithm will perform  $m-1$  searches . Therefore the time complexity of jump search is

$$O\left(\frac{N}{m} + m-1\right) \quad 4.6$$

which is less than  $O(N)$  since it is divided by the number of partitions.

The average number of comparisons required to find a match is

$$C(N)_{avg} = \frac{1N}{2m} + \frac{1}{2}(m-1) < \frac{1}{2}(n+2) + 1 \quad 4.7$$

for a sorted list.

Taking the derivative of equation on the right hand side results to 0 and this results in:

$$\frac{d}{dm} \left( \frac{1N}{2m} + \frac{1}{2}(m-1) \right) = 0 \quad 4.8$$

$$\frac{-N}{m^2} + 1 = 0 \quad 4.9$$

$$m = \sqrt{N} \quad 4.10$$

For Table 4.4 there are 324 rules which follows that  $m=18$  to arrive at Table 4.5 on page 164.

**Algorithm 2: Splitting the rules**

Input: N,

Output: Partitioned Array

1.  $N = array.Length();$
2.  $m = \sqrt{N}$
3. Function Portion (array, size) {
4.  $m\ dividearray = [];$
5. For ( $i = 0; i < n; i++$ ) {
6.  $m\ tail = dividearray[dividearray.length - 1];$
7. if ( $!tail || last.length == size$ )
8.  $dividearray.push([array[i]]);$
9. } else {
10.  $tail.push(array[i]); //Else add the current element into the chunk }$
11. return dividearray;}
12. End if
13. End for

**Algorithm 3: Jump Search**

N-Total number of rules.

m-is the number of Partitions

D- A list of packets.  $D = \{d_1, d_2, \dots, d_n\}.$

R-A list of rules.  $R = \{r_1, r_2, \dots, r_n\}$

F-A set of packet header fields/column fields.  $F = \{f_1, f_2, f_3, f_4, f_5\}.$

A- A set of packet classes  $A = \{a_1, a_2, \dots, a_n\}$

Input: R, D

Output: A

1.  $m = \sqrt{N}$
2. If  $d_i.f_i == r_i.f_i$
3. Perform action  $i //output\ class\ associated\ with\ action\ i$
4. Else
5. If  $d_i.f_i \neq r_i.f_i$  then
6.  $m++ // go\ to\ the\ next\ partition$
7. End if End if
8. go to step 2

Algorithm 2 will take input as an array of list of rules and their priorities. The algorithm will then proceed to split the array into chunks of length size. The algorithm then returns a nested array with chunks of arrays.

Algorithm 3 will take in list of rules and packets as input. It splits the rules in portions of size  $m$ . It searches for a class associated with a particular packet after which it returns the class associated with a given packet based on if the rule matches the packet. If no match is found in the current partition it moves to search next partition.

With the partitioned rule list the study uses IP ranges and port ranges to reduce the number of lines to conserve memory. After removing the anomalies observed in Table 4.4 and sorting the rule list results in Table 4.5 which is used to build a sequential tree rule structure shown in Figure 4.4. It is a sequential tree since there is no branching.

**Table 4.5: Partitioned Rule List**

<b>Rule</b>	<b>Destination IP address</b>	<b>Destination Port</b>	<b>Source IP address</b>	<b>Source port</b>	<b>Protocol</b>	<b>Action(assignment class)</b>
R1	192.168.1.3	3260	192.168.2.4	ANY	ISCSI	Power user
R2	192.168.1.4	3260	192.168.2.4	ANY	ISCSI	Knowledge user
R3	192.168.1.2	3260	192.168.2.4	ANY	ISCSI	Task user
R4	192.168.1.5	3260	192.168.2.4	ANY	ISCSI	Power user
R5	192.168.2.4	3260	192.168.1.3	ANY	ISCSI	Knowledge user
R6	192.168.1.3	3260	192.168.2.4	ANY	ISCSI	Task user
R7	192.168.2.4	3260	192.168.1.5	ANY	ISCSI	Power user
R8	192.168.1.1	3260	192.168.2.4	ANY	ISCSI	Power user
R9	192.168.2.4	3260	192.168.1.1	ANY	ISCSI	Knowledge user
R10	192.168.1.2	3260	192.168.2.4	ANY	ISCSI	Power user
R11	192.168.1.2	3260	192.168.2.4	ANY	ISCSI	Power user
R12	192.168.1.3	3260	192.168.2.4	ANY	ISCSI	Power user
R13	192.168.1.4	3260	192.168.2.4	ANY	ISCSI	Task user
R14	192.168.1.2	3260	192.168.2.4	ANY	ISCSI	Knowledge user
R15	192.168.1.3	3260	192.168.2.4	ANY	ISCSI	Task user
R16	192.168.1.5	3260	192.168.2.4	ANY	ISCSI	Knowledge user
R17	192.168.2.4	3260	192.168.1.4	ANY	ISCSI	Task user
R18	192.168.1.5	3260	192.168.2.4	ANY	ISCSI	Power user
.	.	.	.	.	.	.
.	.	.	.	.	.	.
R19	Any	Any	Any	ANY		Default

Table 4.5 shows the portioned rules list with all the redundancies removed. The table shows that there are more hits from rules associated with power users indicating that the power users have more priority. The action associated with each rule is assignment of a class otherwise the packets are put in default class using rule number 19. The rule lists were used to build a linear tree rule.

#### **4.4.3 Linear Tree Rule Structure Design**

The listed classification rules in Table 4.4 have been theoretically analyzed in chapter two and proven to have conflicts and redundant rules. To eliminate these problems the study proposed a sequential tree rule classifier. The proposed tree rule classifier structure is able to curb some of the limitations of listed rule classifiers. First it is able to avoid conflict which may result due to shadowed rules and redundant rules. Shadowed and redundant rules are eliminated by having distinct values for a particular field. Zhao et al., (2011) proved that removing shadowed rules and redundant rules do not affect classifier policy. By use of tree rule structure the study arrived at a design that has a single path from the root node to the terminal hence eliminating the dangers of the bigger rule problem and swapping of rules(Cherian, 2016).

In the first step of building the tree rules, a partitioned list is taken and one attribute from the list is used as the root node. Other attributes are then ordered in a parent to child design until all of them are exhausted. Only unique values of each field are taken to eliminate the problems of rule positioning, redundant rules and shadowed rules as well as increasing the search speed, Figure 4.3 illustrates the process of tree building.

To further improve on the design the study used range matches instead of exact matches as the root node contains as many lines as there are users in the network. Since the study focus is on allocation of resources the source IP address was used as the root node to guarantee resources to packets as they traverse the network from source to destination( He et al., 2013).

The tree rule structure was arrived in two steps that is sorting step and the partitioning step.

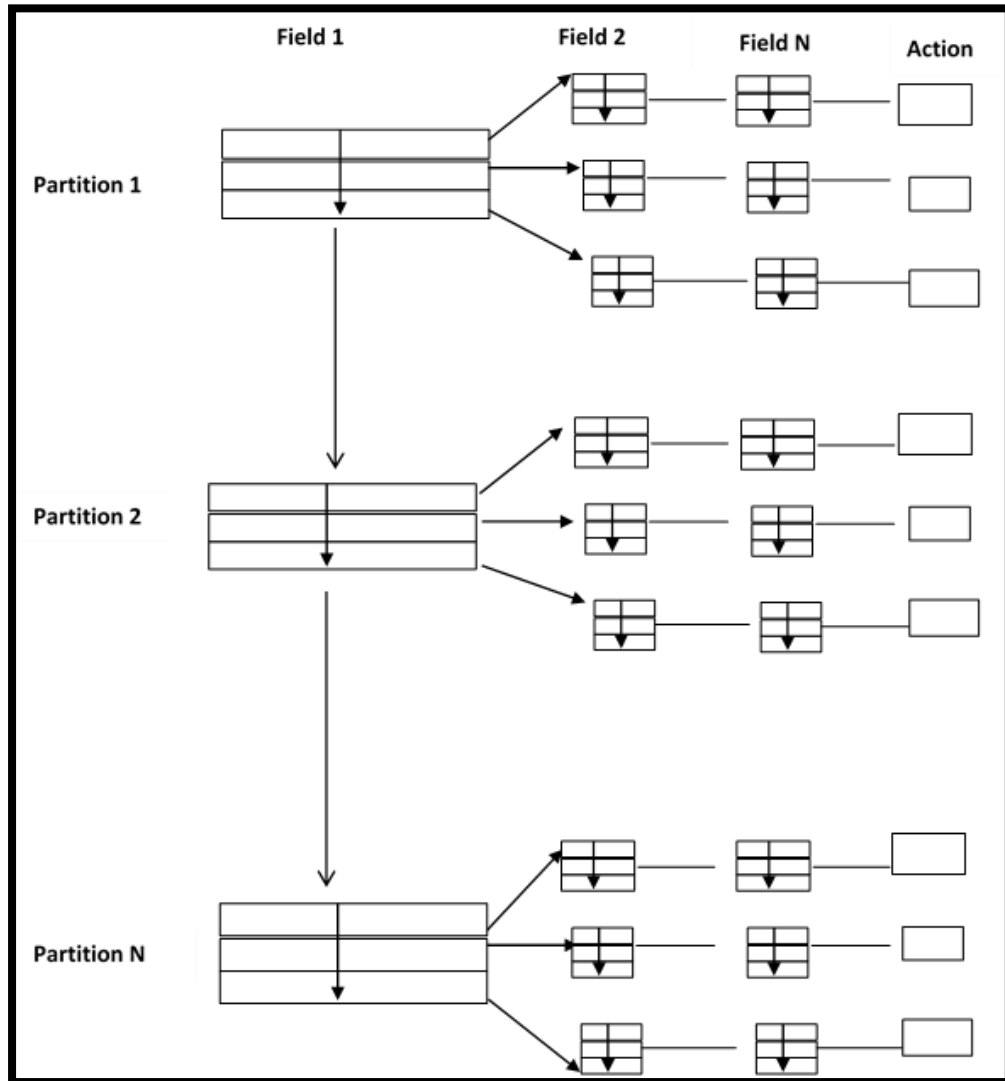


Figure 4.3: Linear Tree Rule Structure Building

Figure 4.3 shows how the linear tree rule was built. Partition 1 depicts the first partition in the many partitions of size  $\sqrt{N}$ . The first partition is composed of high priority rules since it sits at the top. Field 1 to Field N depicts the packet header information used for classification. Rules in a partition are traversed from top to bottom. If a match is found on partition 1 then the fields nodes are traversed horizontally upto the action node. At the action node the associated action is

performed and the algorithm stops. However if no match is found in partion 1 the alforithim proceeds searching all the partions untill partion N. If no match is found in aprtion N then the bigger rule which is the last rule in partion N is used to place packets in the deafult class.

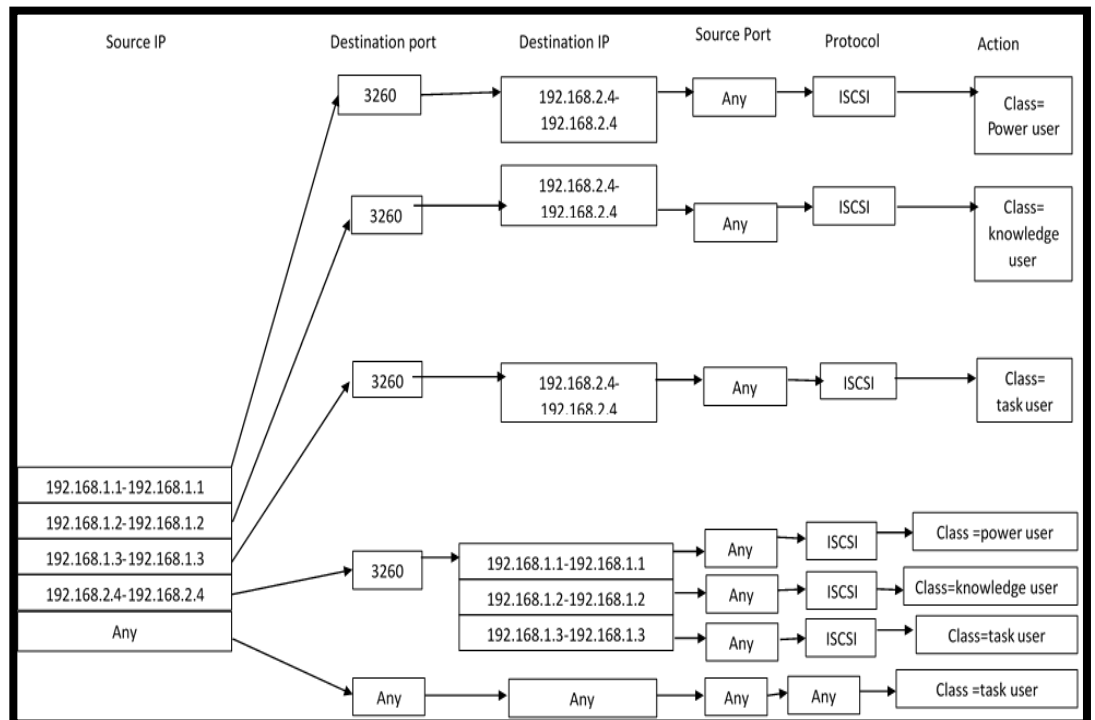


Figure 4.4: Sequential Tree Rule Structure Based on Table 4.5

Figure 4.4 illustrates the result of building the tree rule. From Figure 4.4 it is evident that rules that are associated with power users are at the top. This can be explained by the fact that power users have higher priority. The figure further shows that the lines to be searched were decreased from 19 rules to just 5 rules by the use of IP address range.

**Definition 1**

A field  $f$  of rule  $R_i$  is said to be equal to its corresponding field in packet  $d_j$  iff the values corresponding to  $f$  in both the packet field and the rule field are equal.

$$d_j[f] = R_i[f] \quad \text{iff } f_j = f_i.$$

4.11

#### Algorithm 4: ELPCIS Algorithm

The following are definition of variables used in the algorithm

$N$ -Total number of rules.

$m = \sqrt{N}$ ;  $m$  is the number of the partitions.

$n$  = Number of rules in each partition.

$R$ -A list of rules.  $R = \{r_1, r_2, \dots, r_n\}$

$D$ - A list of packets.  $D = \{d_1, d_2, \dots, d_n\}$ .

$F$ -A set of packet header fields/column fields.  $F = \{f_1, f_2, f_3, f_4, f_5\}$ .

$P$ -Partitions of rules.  $P = \{p_1, p_2, \dots, p_n\}$

$A$ -A set of packet classes  $A = \{a_1, a_2, \dots, a_n\}$

$W$ -default class.

Update()-a function that keeps track of recent traffic history.

Reorder () - a function that reorders rules based on traffic characteristics.

Resplit()-a function that re-splits the reordered rules into partitions.

1. INPUT; R, A, W
2. OUTPUT; Packet Classes
3. For(  $m = 0; m < n; m++$ )
4. For(  $i = 0; i < n; i++$ )
5. If  $d_i.f_i = r_i.f_i.p_i$  then
6. Output  $a_i$
7. Else
8. Output  $w$
9. Endif,Endfor ,End for
10. If(  $m == n$  )
11. Update();
12. Reorder();
13. Resplit(); end if

Algorithm 4 takes input as incoming packet  $d_i$ . The fields of the packet  $d_i$  is compared to the fields of rule  $r_i$ . If they match then the corresponding action is performed. If none of the rules match in the current block of rules then the counter  $m$  is incremented to move to the next block. If none of the rules match



then the packets are placed in the default class  $W$ . When all the portioned blocks are searched then the update function is called for capturing current network changes in terms of hit ratio. Then again the rules are reordered based on the new hits statistics using *reorder()* function. After reordering the *resplit()* function is called which splits the rules into partitions which are used to build linear tree rule.

#### 4.4.4 Time Complexity Analysis

From Table 4.4 , since the chances for a match is the same for each rule, then the average number of matches when using List based packet classifier is  $(19/2) \times 5 \times C = 48C$  where  $C$  is the time is required to compare one field of a packet to one field of a rule and five is the number of fields in the table that require comparison.

With the tree rule in Figure 4.4 implemented in ELPCIS, there are on average 5 lines in the source IP field to get  $(1 + \log_{10} 5)C$ , one line in the destination port field to get  $(1 + \log_{10} 1)C$ , three lines in the destination IP field to get  $(1 + \log_{10} 3)C$ , one line in the source port field  $(1 + \log_{10} 1)C$  and one line in the in the protocol field  $(1 + \log_{10} 1)C$ . The additional overhead of 1 is due to the range match used in the design( He et al., 2013).

Consequently when using the tree rule classifier the average number of comparison required to make a match is  $\{(1 + \log_{10} 5)C + (1 + \log_{10} 1)C + (1 + \log_{10} 3)C + (1 + \log_{10} 1)C + (1 + \log_{10} 1)C\} = 6C$ . Where  $C$  the time is required to compare one field of a packet to one field of a rule. By dividing the obtained cost  $C$  by one hundred and then multiplying the result by one

hundred gives cost C in percentage. Then by subtracting the cost C obtained for List based packet classifier from cost C obtained when using ELPCIS the mathematical simulation results show that ELPCIS cost C is reduced by 42% compared to the List Based Packet classifier.

#### **4.4.5 Performance Evaluation**

The experiment was bent on evaluating the performance of the proposed solution. The evaluation consisted of experiments that examined the following questions;

1. How does average throughput vary with changes in the number of rules and the block size?
2. How does average response time vary with changes in the number of rules and the block size?

For this experiments the study used three computers (Intel 2.8 GHZ CPU with 2GB of RAM and 500 GB hard disk). Their roles were that of target, router and initiator. The router contains two Ethernet cards of 100 Mbps and it was directly connected with the target and the initiator as illustrated in Figure 4.5. The router is the computer sitting at the middle. The study used the rule lists in Table 4.5 for best case scenario.

For all the experiments a file size of 4GB which is the maximum file size possible with the Parkdale simulator was simulated for reads and writes. This file size was also chosen due to that fact that, the bigger the file size the more the traffic needed to test ELPCIS for the implemented functionalities. All experiments have two configuration that is none isolated (List based

performance isolation) and isolated (ELPCIS). Duration from each experiment is 200 seconds.

#### 4.4.6 Throughput.

Experiments to measure throughput was carried out with and without the proposed solution.

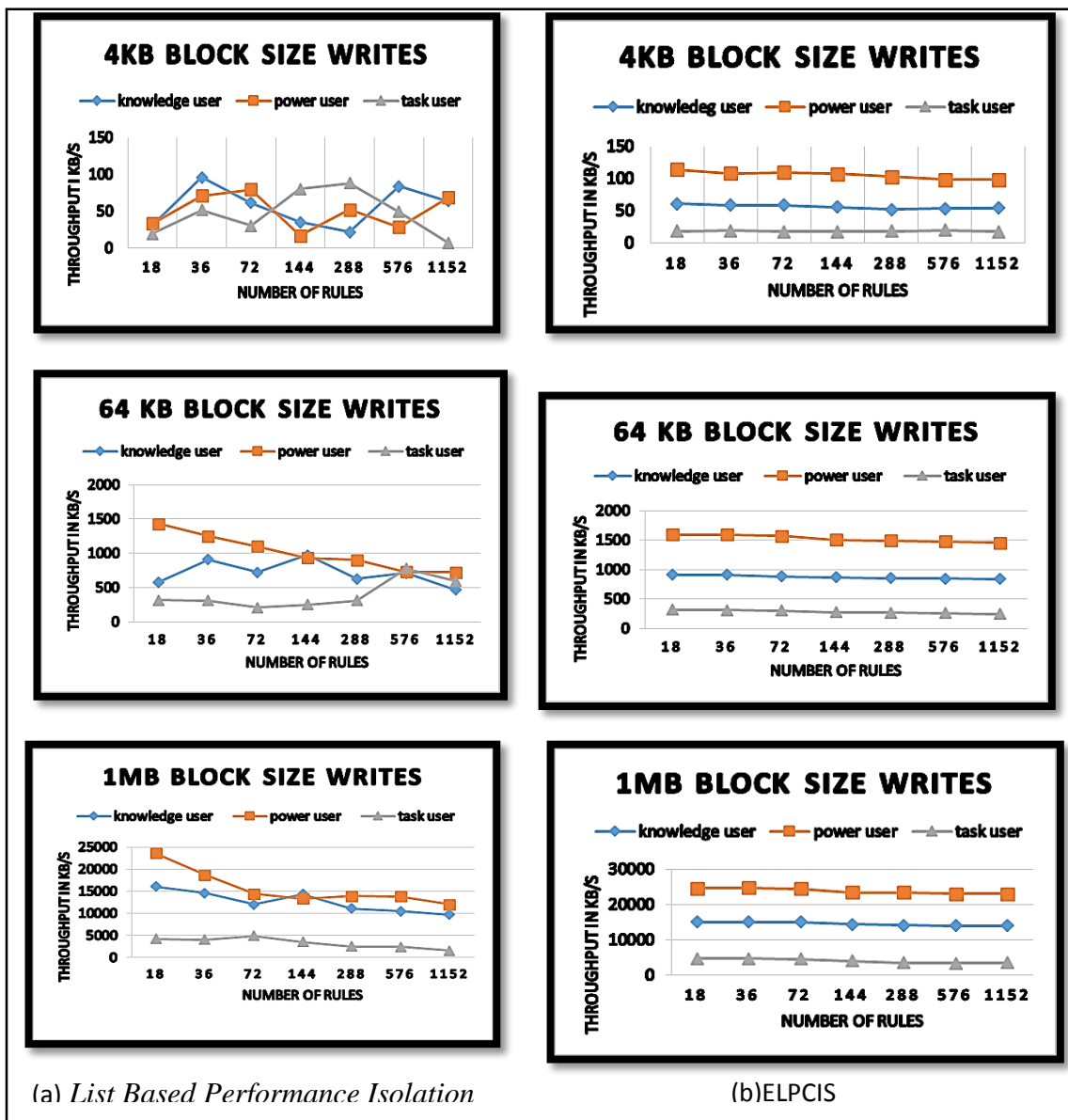


Figure 4.5 Writes throughput comparison (a) List Based Performance Isolation and (b) ELPCIS for varied block sizes.

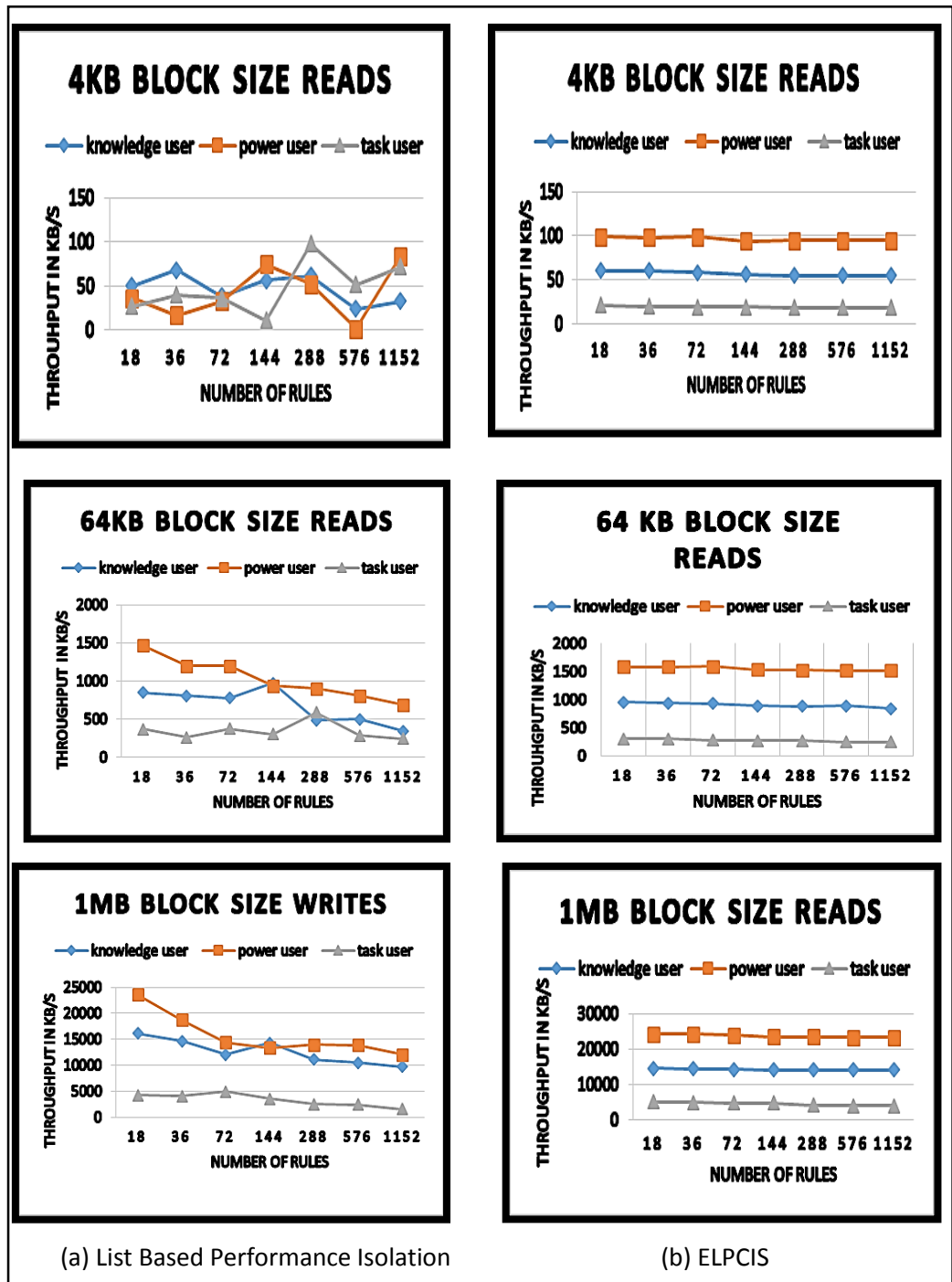


Figure 4.6 Reads throughput comparison (a) List Based Performance Isolation and (b) ELPCIS for varied block sizes.

In Figure 4.5(a) and Figure 4.6(a) shows that there is a steady throughput degradation for the List Based Performance Isolation due to an increase in the

number of rules. Initially when numbers of rules are less, the throughput of the List Based Performance Isolation is similar to that of the ELPCIS. As the number of rules increases the performance of the List Based Performance Isolation deteriorates while that of the ELPCIS stabilizes. Another observation is that since the throughput is a product of block size the higher the block size the higher the throughput.

Figure 4.5(a) and Figure 4.6(a) further shows that the storage users are not able to achieve their SLO with the list based classifier, this is because the List Based Performance Isolation causes delays due to the increase in the number of rules and therefore results in reduced performance as the rules increase.

However with the ELPCIS as illustrated in Figure 4.5(b) and Figure 4.6(b), rule search time is reduced during the classification process and therefore all the classes of users are able to achieve an SLO close to the system being emulated. This is intuitively consistent with what is expected that the users should meet SLOs close to the system modelled irrespective of the number of rules( He et al., 2013).

In addition these results are consistent with what is expected and also with research done in(He et al., 2013) where experiments were performed to compare the performance of a IPTables which is a list based firewall versus the optimized list based firewall. The results showed that the performance of the optimized list based firewall is not drastically affected by the increase in the number of rules unlike that of list based firewall(Danielsson et al., 2019).

#### 4.4.7 Latency

To evaluate for Latency experiments were set up for reads and writes with the ELPCIS and with the List Based Performance Isolation. The results are as illustrated in Figure 4.7 and 4.8

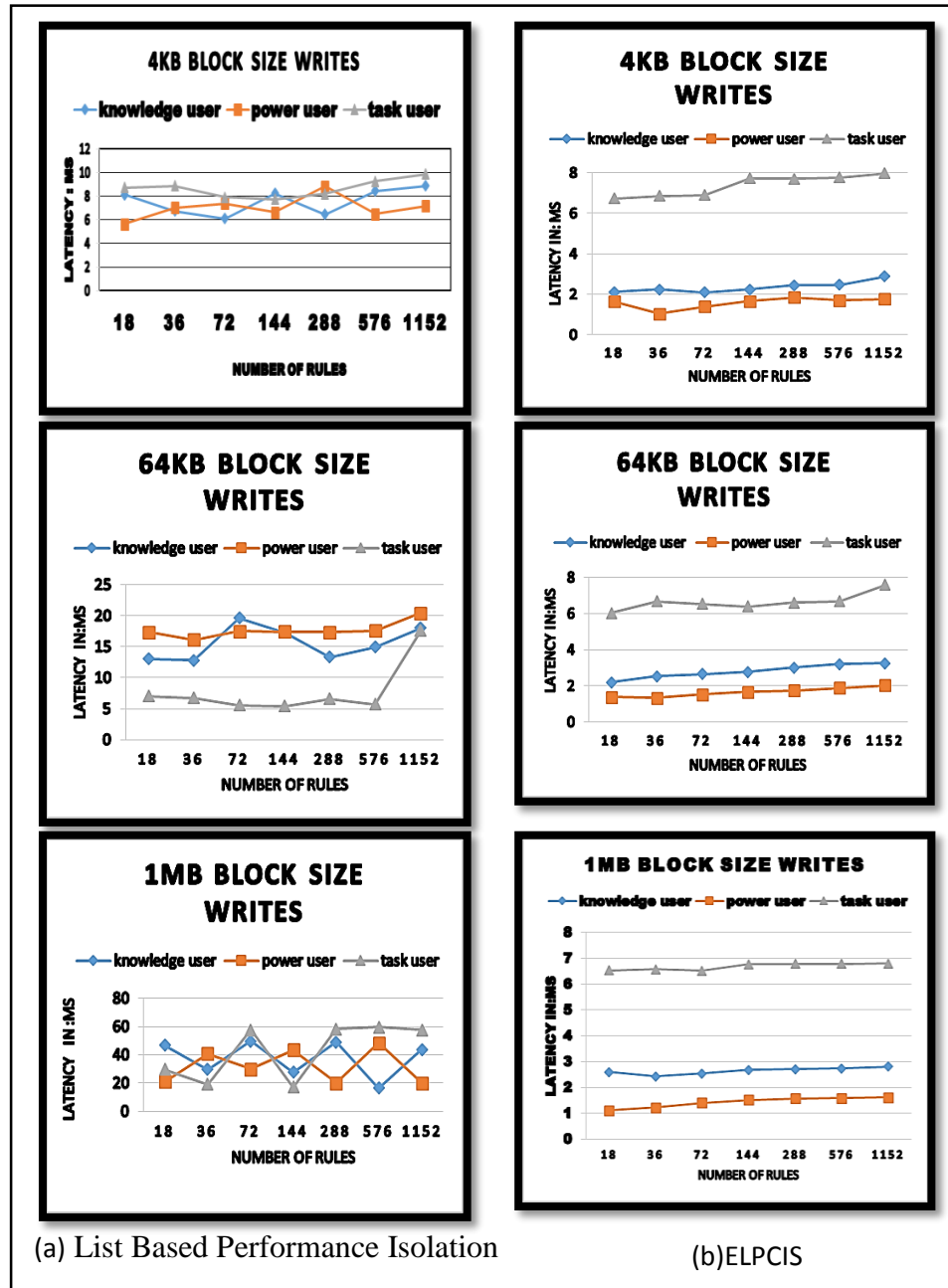
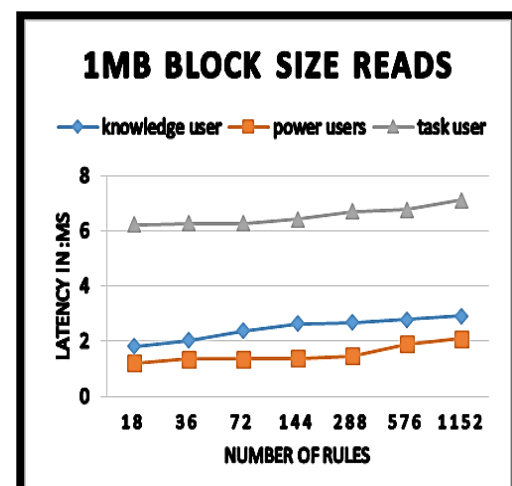
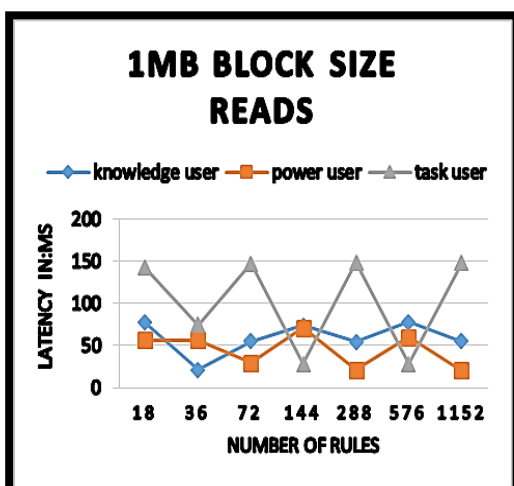
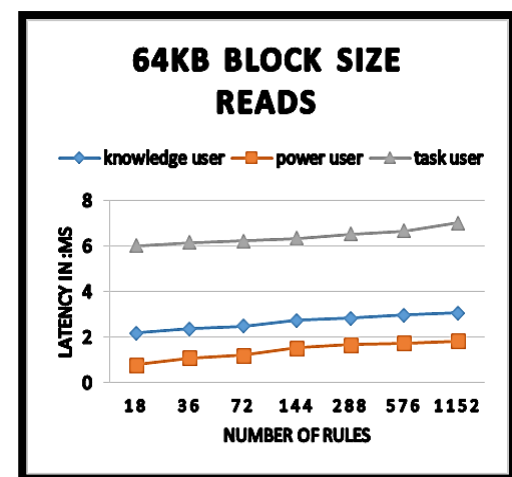
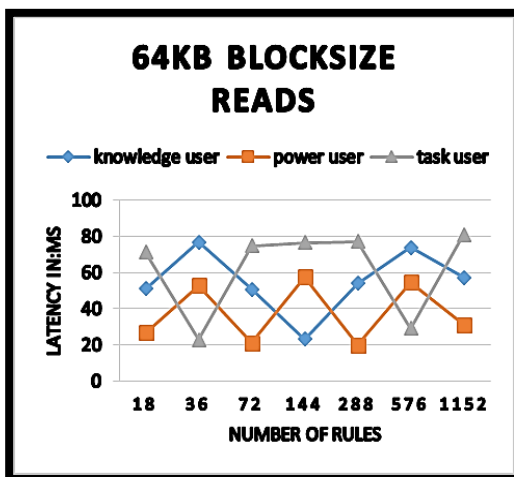
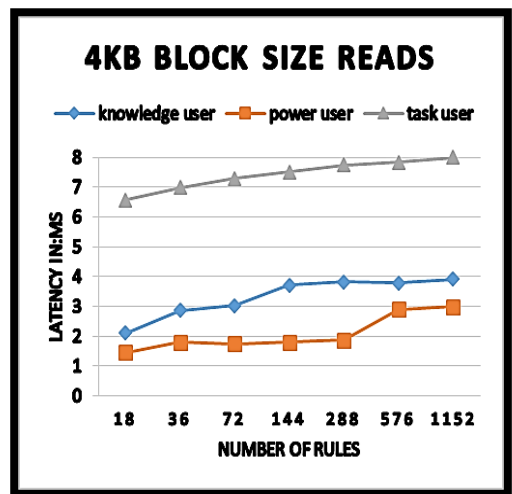
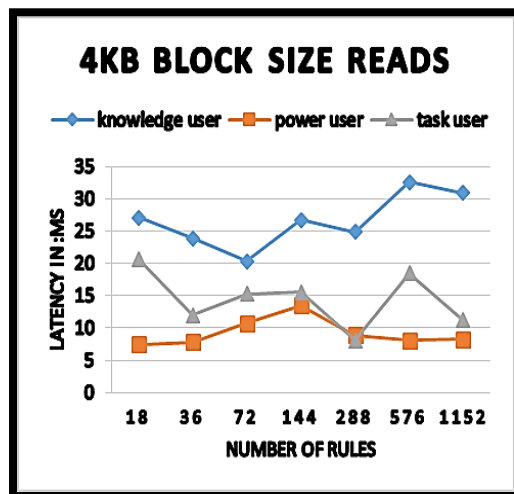


Figure 4.7 Latency comparison for writes (a) List Based Performance Isolation (b)

ELPCIS



(a) List Based Performance Isolation

(a)ELPCIS

Figure 4.8 Latency comparison for writes (a) List Based Performance Isolation and (b) ELPCIS.

From Figure 4.7 (a) and Figure 4.8(a) it is observed that the latency for the List Based Performance Isolation steadily increase with the number of rules. On the other hand the latency of the ELPCIS slightly increases then stabilizes( He et al., 2013b). Indicating that the performance of the ELPCIS is not adversely affected by the number of rules(Danielsson et al., 2019).

From Figure 4.7(a) and Figure 4.8(a) it further observed that the emulated classes of users are not able to achieve their SLO with List Based Performance Isolation. However with the ELPCIS as illustrated in Figure 4.7(b) and Figure 4.8(b) all the classes of users are able to achieve an SLO close to the system being emulated(Pan, Huang, Tang, & You, 2018). These results are consistent with those in (Gulati & Waldspurger, 2009) where the authors varied the number of input output operations(Pan et al., 2018). Figures 4.7 shows that the latency for List Based Performance Isolation solution increased by a factor 2X for 4KB, a factor of 3X for 64KB and a factor of 10X for 64KB compared to that ELPCIS. Figure 4.8 further shows that the increase in latency for List Based Performance Isolation solution by a factor 4X for 4KB, a factor of 10X for 64KB and a factor of 20X for 64KB more compared to that ELPCIS. More latency was experienced for reads more than writes due to the additional overhead of seek and rotational latency experienced when doing reads.

#### **4.5 Classifier Accuracy**

To evaluate for accuracy a file of 4GB was simulated for reads and writes with a queue depth of 32 and a block size of 64KB. The decision to use a block size of 64kb due to the reason that it's because it's the default block size for windows. For queue depth of 32 as it's the default in Parkdale and 4GB is the



maximum file size possible for simulation while using Parkdale. After the reads and writes were completed the command *tc-s qdisc ls dev etho* was run on the router to generate the total packets generated and the classification per class. Table 4.6 shows the number of packets classified correctly and total number of packets generated for the experiment for all the classes of users form ELPCIS and List Based Packet classifier. The result shows that ELPCIS had almost double the amount of packets classified correctly compared to List based packet classifier proving it is more accurate. The table further shows that there were more packets generated by the power user's class than the other classes indicating that the power users had more hits translating to higher priority.

**Table 4.6: Statistics of Packet Classification**

Class	List Based Performance Isolation	ELPCIS
	Number of packets	Number of packets
Power user	4,871,229	4,973,264
Knowledge user	4,813,052	4,935,052
Task user	4,812,035	4,922,035
Total number of packets	14,496,316	14,830,351
Total number of packets classified correctly	8,117,936	13,199,012

By dividing the total number of packets classified correctly by the total number of packets generated by all the classes of users. Results obtained showed that

the List Based Performance Isolation achieved an accuracy is 56% as compared to that of ELPCIS of 89%. This indicates an improvement of 33% in classification accuracy when using ELPCIS.

#### **4.6 Summary**

In this chapter the research embarked on the problem of performance isolation. To achieve performance isolation packets from initiators needs to be classified for them to be offered differentiated treatment. However most Linux based classifiers including the u32 classifier filter traffic according to a certain classification policy which traditionally consist of a list of rules. Arriving packets are sequentially compared against a list of rule until a match is found. Due to increase in network speeds and mix of traffic in IP SANs it's important for packet classifiers to inspect packets as fast as possible.

In the chapter the study has discussed in details the problems of linear search and techniques for optimizing linear search. The chapter presents a solution named ELPCIS for optimizing packet classification process for achieving performance isolation through throttling of flows.

The ELPCIS was tested and compared with List Based Performance Isolation and it was found that ELPCIS gives better performance in terms of throughput and response time when implementing performance isolation. ELPCIS was tested on an IPSAN and was found to be more suitable than the traditional List Based Performance Isolation.

## **CHAPTER FIVE: OPTIMIZATION OF BANDWIDTH MANAGEMENT AND BURST HANDLING**

### **5.1 Chapter Overview**

This chapter looks at the various techniques used in the optimization of bandwidth management and burst handling. Techniques for bandwidth management used include those of bandwidth borrowing and bandwidth allocation. That for burst handling include traffic shaping. The chapter implements an optimized technique for bandwidth management and traffic shaping. Finally the proposed technique was evaluated using the throughput and latency metrics.

### **5.2 Problem Definition**

Let  $I$  be a set of users to be allocated bandwidth. Three QOS attributes that comprise the SLO for each class of user are defined to include IOsize, IOPs and response time which can be defined as;

1. IOPs: The Input/output (I/O) commands per second
2. Response time as the time it takes for a request to receive a response and constitutes total latency.
3. Queue depth: the number of I/O commands that can be queued at a time on a storage controller at the initiator side or at the target side.
4. IO size as the amount of data read/written at a given instance.

Let  $S_i$  denote the SLO associated with a particular class of users where  $i$  is a set of the three QOS metrics as indicated in Equation 5.1

$$S_i = \{ (IOsize_i, IOPS_i, Latency_i) : \forall i \in I \} \quad 5.1$$

Let  $rsz_i$  denote the I/O request size of class  $i$ ,  $IOP_i$  represent IOPs for class  $i$  and  $rt_i$  represent response time for class  $i$ . Further, let  $B_R^i$  be the total request bandwidth by class  $i$  based on the SLO.

Consequently, the QOS attribute  $rsz_i$ ,  $IOP_i$  and  $rt_i$  have got the relationship expressed in equations 5.2 and 5.3

$$B_{TR}^i = IOP_i * rsz_i \quad 5.2$$

$$IOP_i = \frac{queudepth}{rt_i} \quad 5.3$$

Therefore, the total bandwidth,  $B_{TR}$  required by all the classes can be described as 5.4;

$$B_{TR} = \sum_{i=1}^n (IOP_i \times rsz_i) \quad \forall i \in I \quad 5.4$$

Let  $B_{RW}^i$  represent the amount of bandwidth that is configured for the class  $i$  to borrow. The total bandwidth to be borrowed  $B_{RW}^T$  can be described as in equation 5.5

$$B_{RW}^T = \sum_{i=1}^n B_{RW}^i \quad \forall i \in I \quad 5.5$$

Describing the total bandwidth capacity of the network as  $B_C$ , then

$$B_C = B_{TR} + B_{RW}^T \quad 5.6$$

Now, let  $x_i$  be the rate assigned to class  $i$ . Then the utility rate of class  $i$  can be expressed as  $U_i(x_i)$  which is a concave differentiable function. This means that an increased allocation to a given class increases the total bandwidth

allocated but it has no effect to the one class that has more resources already.

This characteristic makes the utility function to be logarithmic in nature.

Assuming that the network has a fixed capacity and therefore the goal is to maximize the collective utility of users in the network subject to network capacity constraints. From this narrative, the maximization problem is formulated as follows;

$$\max \sum_i^n U_i(x_i) \quad 5.7$$

$$\text{Subject to } \sum_i^n x_i \leq B_{RW}^T \quad 5.8$$

$$x_i > 0, \forall i \in I \quad 5.9$$

In the above equations  $U_i(x_i)$  is the utility function of class  $i$  at rate  $x_i$ .  $I$  is the set of classes of users in the network. User  $i$  is identified with utilization  $x_i$ .  $B_{RW}^T$  is the total excess bandwidth available. This study thus sought to maximize the concave objective subject to linear constraints.

Based on proportional fairness the utility function

$$U_i(x_i) = \text{Log } x_i \quad 5.10$$

$$\text{Let } P \text{ be a set of priority that is } P = \{p_i, i \in I\}. \quad 5.11$$

By introducing priority  $p_i$  we have

$$U_i(x_i) = p_i \text{Log } x_i \quad 5.12$$

Let  $x_i^*$  be the optimal rate and  $x_i$  be the minimal rate.

Then for any allocation vector  $x_i = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$  there is an allocation equation as

shown in equation 5.13 based on fairness utility.

$$\sum_i \frac{x_i - x_i^*}{x_i^*} \leq 0 \quad 5.13$$

From equation 5.13 it is apparent that for any allocation the sum of changes in the utilities will be less than zero (Guo, Langrené, Loeper, & Ning, 2021). That is if the rate of a given class  $i$  increases there is some rate of another class of users that decreases (Vigneri, Paschos, & Mertikopoulos, 2019).

If excess bandwidth is assigned based on priority proportional fairness results in the corresponding inequality as shown in equation 5.14.

$$\sum_i p_i \frac{x_i - x_i^*}{x_i^*} \leq 0 \quad 5.14$$

The study uses the fairness to investigate the different fairness criteria of max-min, minimum delay fairness and proportional fairness. The parameter  $\alpha$  takes values in the interval  $(0, \infty)$  (Yitu Wang, Wang, Cui, Shin, & Zhang, 2018).

The study defines  $\alpha$  as the fair utility function as shown in equation 5.15.

$$U_i(x_i) = \frac{p_i x_i^{1-\alpha}}{1-\alpha} \quad \text{Where } \alpha \geq 0, \alpha \neq 1 \quad 5.15$$

Different values of  $\alpha_i$  yield different fairness criteria. Case one of fairness we have  $\alpha \rightarrow 1$  (Zhang, Deng, & Liang, 2018).

The utility function for this case after introducing priority is as shown in equation 5.16.

$$U_i(x_i) = p_i \text{Log } x_i \quad 5.16$$

Case two of  $\alpha$  fairness we have delay fairness where  $\alpha=2$ (Zhang et al., 2018).

Therefore the utility function for our case is as illustrated by equation 5.17.

$$U_i(x_i) = \frac{p_i}{x_i} \quad 5.17$$

Equation 5.18 means that if a class  $i$  is trying to transmit a file of size  $rsz_i$  and the rate allocated to this class is  $x_i$ , then results in  $\frac{rsz_i}{x_i}$  as the time taken to transfer the file.

Case three is that of  $\alpha$  fairness is that of minimum maximum fairness where  $\alpha \rightarrow \infty$ (Gu et al., 2019).

From the three cases of  $\alpha$  fairness discussed above can be summarized as indicated in equation 5.18.

$$U_i(x_i) = \begin{cases} p_i \frac{x_i^{1-\alpha}}{1-\alpha}, & \alpha > 0, \alpha \neq 1 \\ p_i \text{Log } x_i, & \alpha = 1 \end{cases} \quad 5.18$$

Equation 5.19 represents the priority proportional fairness. From this the study modeled the solution for priority based fairness utility maximization as depicted in equation 5.19 subject to constraints in equation 5.20.

$$\text{Max } p_i \log x_i + p_1 \log x_1 + p_2 \log x_2 + p_3 \log x_3 + \quad 5.19$$

Subject to

$$x_1 + x_2 + x_3 \leq B_{RW}^T, x_1, x_2, x_3 > 0 \quad 5.20$$

In order to solve the optimization problem, the study needs to find the optimal allocations  $x_1^*, x_2^*, x_3^*$ . To get the optimal allocations the study applied the Lagrange Multiplier on equation 5.19. Again since the theory of convex optimization holds if the complementary slackness is satisfied, this means the Lagrange multipliers to be used has to be positive. The Lagrangian multiplier for the problem is as illustrated in equation 5.21.

$$L(x, \lambda) = p_1 \log x_1 + p_2 \log x_2 + p_3 \log x_3 + \lambda(B_{RW}^T - x_1 - x_2 - x_3) \quad 5.21$$

Where  $\lambda$  is the rate at which the optimal value changes as the input increases.

Applying partial derivative,  $\frac{\partial L}{\partial x_i}$  results in equation 5.23.

$$x_1 = \frac{p_1}{\lambda}, x_2 = \frac{p_2}{\lambda}, \text{ and } x_3 = \frac{p_3}{\lambda} \quad 5.22$$

Using the constraint in equation 5.23 results in equation 5.24 and 5.25.

$$x_1 + x_2 + x_3 \leq B_{RW}^T \quad 5.23$$

$$B_{RW}^T = \frac{p_1}{\lambda} + \frac{p_2}{\lambda} + \frac{p_3}{\lambda} \quad 5.24$$

$$\lambda = \frac{p_1 + p_2 + p_3}{B_{RW}^T} \quad 5.25$$

By substituting  $\lambda$  back in equation 5.25 results in equations 5.26, 5.27 and 5.28.

$$x_1^* = \frac{p_1 B_{RW}^T}{p_1 + p_2 + p_3} \quad 5.26$$

$$x_2^* = \frac{p_2 B_{RW}^T}{p_1 + p_2 + p_3} \quad 5.27$$

$$x_3^* = \frac{p_3 B_{RW}^T}{p_1 + p_2 + p_3} \quad 5.28$$



Generally, the optimal rate is thus shown in equation 5.29.

$$x_i^* = \frac{p_i \sum_{i=1}^n B_{RW}^T}{\sum_i p_i} \quad 5.29$$

Where  $x_i^*$  is the optimal allocation for any class  $i$  and with priority  $p_i$ .

### 5.3 Proposed Solution

In this section the proposed HPDDRR (Hierarchical Priority Dynamic Deficit Round Robin) which is a scheduler shaper is described to improve on latency and bandwidth utilization for flows. HPDDRR is a two stage mechanism which employs a single level hierarchy to aggregate flows into classes with similar priority and packet size. The key idea that enables the HPDDRR to alleviate the latency problem of DRR is the grouping of flows into classes with similar priority and almost similar packet sizes. This is a feature important since DRR is optimal when it acts with flows with similar packet sizes. The grouping of flows is so as to balance packet size per flow which will solve the problem of delays caused by large packets to small packets. The proposed algorithm begins by calculating the hit ratio for each class of flows which is used to determine the priority of the flows. The priority of the classes is established using the equation  $p_i = \frac{h_i}{N}$ . Where  $h_i$  is the hit count of class  $i$  and  $N$  is the total number of hits.

Use of hit ratio is meant to ensure optimal utilization of bandwidth since the flows are allocated bandwidth proportional to their priority which is derived from their need. This reduces the chances of idle bandwidth or under allocation. Classification is done based on priority with flows of the same priority being

put in the same class. From the classification the flows proceed to the shaper where packets that do conform to rates allocated are forwarded to the scheduler while those that do not conform are queued as they await bandwidth to be available.

During shaping, a flow is accepted if and only if the flow capacity is less than the guaranteed rate plus borrowing rate. Each class/flow can be in one of the following states. First, it can borrow since the bandwidth is sufficient and the number of packets sent is less than rate. Alternatively, it may borrow even though there are no tokens but it can be borrowed from parent class and the number of packets sent is greater than rate and less than ceil. Finally, it may be ~can't borrow state where bandwidth available for borrowing is less than the capacity of packets to be sent. Packets are classified using the u32 filter putting them into corresponding leaf classes. Bandwidth allocation is done using the HTB algorithm. HTB starts from the bottom of the class tree to find the class in the *can send state* until the class of the *can send state* is found. If there are many flows in the can send state the algorithm will select the high priority classes. Each class sends its own quantum bytes by the means of poling until it's in the may borrow state. When the leaf classes is in May-borrow state it will borrow tokens from its parent's class until it is in can't send state.

To ensure the drop rates are low when bandwidth to be borrowed is not enough the lower priority classes releases some bandwidth at the same time ensuring that the users that releases the bandwidth their allocations do not fall below acceptable levels. Low priority classes are the ones that release bandwidth to

ensure the high priority classes do not suffer from quality degradation. When there is enough free bandwidth available the proposed scheme gives the amount close to the maxima  $x_i^*$  otherwise if the available bandwidth is lower  $B_{RW}^T$  then the bandwidth allocation adjustments will be performed and the allocations for some low priority classes will be adjusted downwards and allocated the bandwidth of  $X_i$ . Flows are put into priority grades based on the SLO. When there is a need free the excess bandwidth the algorithm looks up at the low priority classes and checks the one that has bandwidth greater than the minima. If it finds that the current low priority class bandwidth is greater than the maxima the look up stops and the flow releases bandwidth to the high priority needy flow. If all the low priority flows cannot release enough bandwidth to satisfy the new flow, the high priority flows are queued.

A node with priority is assigned a bandwidth  $\frac{p_i \sum_{i=1}^n B_{RW}^T}{\sum_i^n p_i}$  where  $B_{RW}^T$  is the total available bandwidth. The higher the priority the more the bandwidth a flow receives.

At the scheduler the quantum for each round is calculated based on the rates of the highest priority class. Figure 5.1 illustrates the schematic architectural representation of HPDDRR scheduler shaper. HPDDRR begins by grouping traffic, then shaping traffic and then the flows are allocated bandwidth and are sent to the IPSAN. This ensures that packets have a service tag associated with them as they traverse the network. This ensures that resources are allocated dynamically based on need.

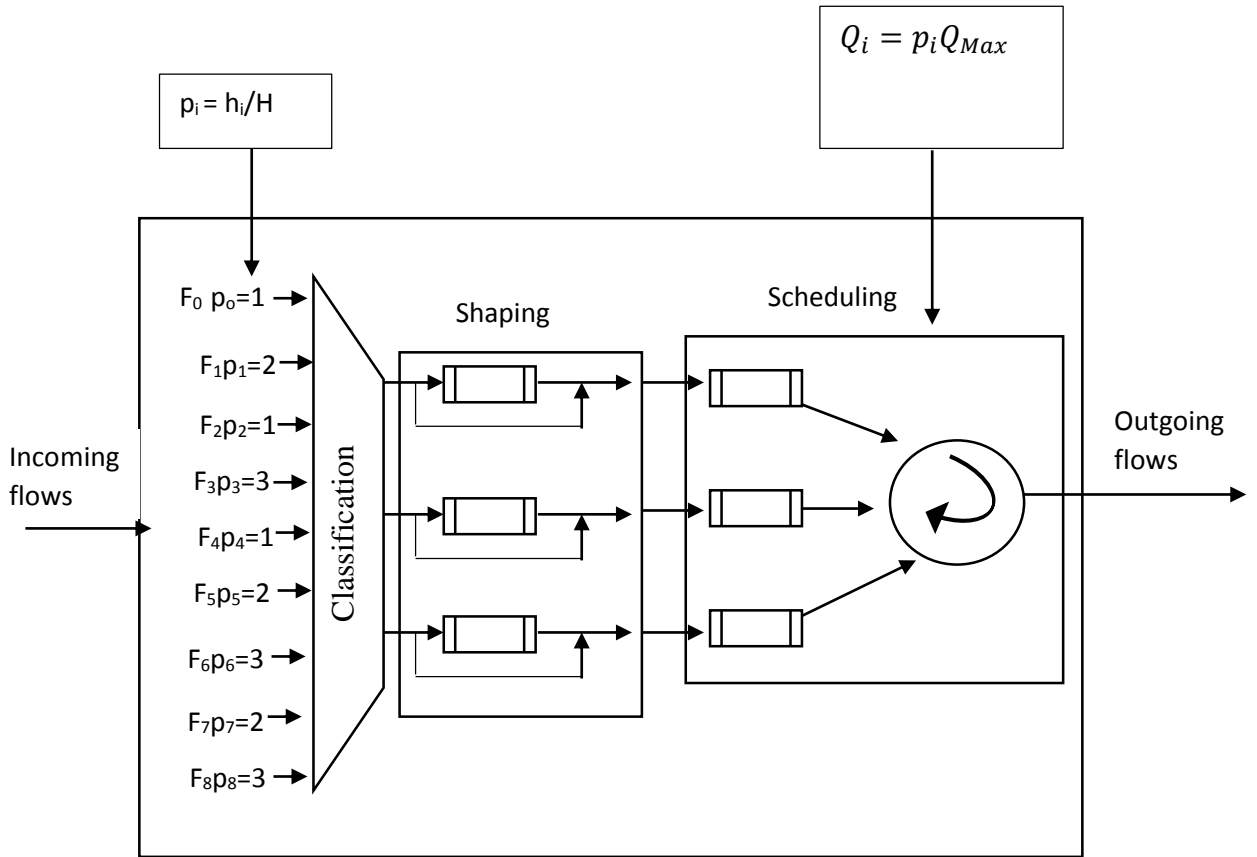


Figure 5.1: Architecture of the HPDDRR.

Algorithm 5 presents the step by step execution of HPDDRR for the optimization of bandwidth management and burst handling. To best represent the algorithm different parameters are defined as follows;

**Algorithm5: HPDDRR**

$Q_{max}$ : The biggest quantum size Possible. (Constant integer-1500bytes)

$Q_i$ : Quantum the ideal rate a flow should receive in each round service (integer)

$B_{RW}^T$ : Total available bandwidth

$DC_i$ -Deficit from the previous round (integer)

$P_i$ : Priority of class  $i$

$pkt_i$ : Packet belonging to queue  $i$

$B_{TR}^i$  : is total bandwidth allowed to class  $i$

$X_i^*$  : is the maximum rate that a class can request

$H$ : Total hits

$h_i$ : Hits from class  $i$

$NQI$ -New quantum  $i$

**INPUT:**  $h_i, H, B_{RW}^T, B_{RW}^i$

**OUTPUT:**  $pkt_i$

Step1: Calculate the priority

$p_{i=\frac{h_i}{H}}$  //  $h_i$  total hits for class  $i$ ,  $H$  is the total number of hits

Step2: Aggregate traffic into queues based on size and priority

//shaping

Step 3: Shape traffic

$$B_{RW}^T = p_i \sum_{i=1}^n B_{RW}^i$$

If  $X_i^* \leq B_{TR}^i$  Then

Forward packets for scheduling

Else

Queue packets (delay packets)

End if

//scheduling

Step4: Calculate the deficit counter based on priority

$DC_i=0$ ;

$Q_i = p_i Q_{Max}$

While  $Q_i > 0$  and queue  $i$  is not empty do

*Packet size = size (head (queue<sub>i</sub>))*

*If packet size ≤ Q<sub>i</sub> then*

*Send (dequeue (queue i))*

*NQ<sub>i</sub> = Q<sub>i</sub> - packet size*

*Else If packet size ≤ NQ<sub>i</sub> then*

*Transmit packet and set NQ<sub>i</sub> = NQ<sub>i</sub> - packet size*

*Else*

*DC<sub>i</sub> = NQ<sub>i</sub>*

*Endif Endif*

*Queue<sub>i</sub> ++*

*End while*

Step 5: *If (empty (queue<sub>i</sub>)) then*

*DC<sub>i</sub> = 0;*

*Repeat*

*Endif*

Algorithm 6 works by shaping traffic then allocating bandwidth. Algorithm 6 starts by shaping traffic. The maximum rate  $X_i^*$  is the maximum allowed rate for class  $i$ .  $B_{RW}^i$  is the total bandwidth allocated to class  $i$ . If the class rate is less than or equal to the available bandwidth the flows are forwarded to the scheduler otherwise they are delayed.

Next the packets arrive at the scheduler. In the scheduler there are  $n$  queues running from  $1$  to  $n$  that are served in a round robin fashion. Queue  $i$  belongs to class  $i$ . Deficit counter  $DC_i$  stores bytes that a queue belonging to class  $i$  did not

use in the previous round. At the beginning the  $DC_i$  is set to zero. Quantum  $Q_i$  represents the amount of capacity that each queue can use at each round of service. Each queue  $i$  belonging to class  $i$  has a different QOS requirement. For each queue  $i$  there is an associated priority. The requirements of flows belonging to a class  $i$  are established by calculating the priority  $i$ . The priority is used as the performance measure. Based on the priority which is dynamic, the quantum  $Q_i$  is calculated using formula  $Q_i = \frac{h_i}{H} Q_{Max}$  and allocated to each queue based on network statistics.  $Q_{max}$  is the maximum packet size that for any packet in an Ethernet network.

If the quantum size  $Q_i \geq \text{packet size}$ , then the packet is transmitted, else the algorithm moves to the next queue. Once a packet is transmitted its size in bytes is subtracted from the quantum  $Q_i$  to form  $NQ_i$ . If the  $NQ_i$  is not sufficient to transmit the packet in the head of the queue then the  $NQ_i$  is stored in  $DC_i$  to be used in the next round. Then the algorithm moves to the next round. In the end the total bandwidth received by a queue  $i$  is the total quantum's received by the queue. That is

$$B_{RW}^i = \sum_{i=1}^n Q_i \quad 5.31$$

The difference between HPDDRR and DRR is that in HPDDRR the quantum is dynamic whereas in DRR the quantum is static.

#### **5.4 Bandwidth Management Optimization**

This section presents the results for bandwidth management. Bandwidth management was implemented using the techniques of bandwidth allocation and bandwidth borrowing.

### 5.4.1 Bandwidth Allocation

Bandwidth allocation experiment was performed to establish if HPDDRR is able to enforce proportional bandwidth allocation. An essential feature is that HPDDRR should allocate each class of users bandwidth proportional to their share in the range  $[X_i, X_i^*]$ . Three hosts running Parkdale and generating 64KB read/writes IO sizes were used. In addition DRRR was used for the host level scheduler. The proposed solution was run in the router with hosts' priority given allocations based on priority  $p_i$  set according to shares 1:2:3 for Hosts 1 to 3. Tabel 5.1 illustrates bandwidth utilization and latencies achieved when HPDDRR implements strict resource allocation. Figure 5.2 further depicts these results.

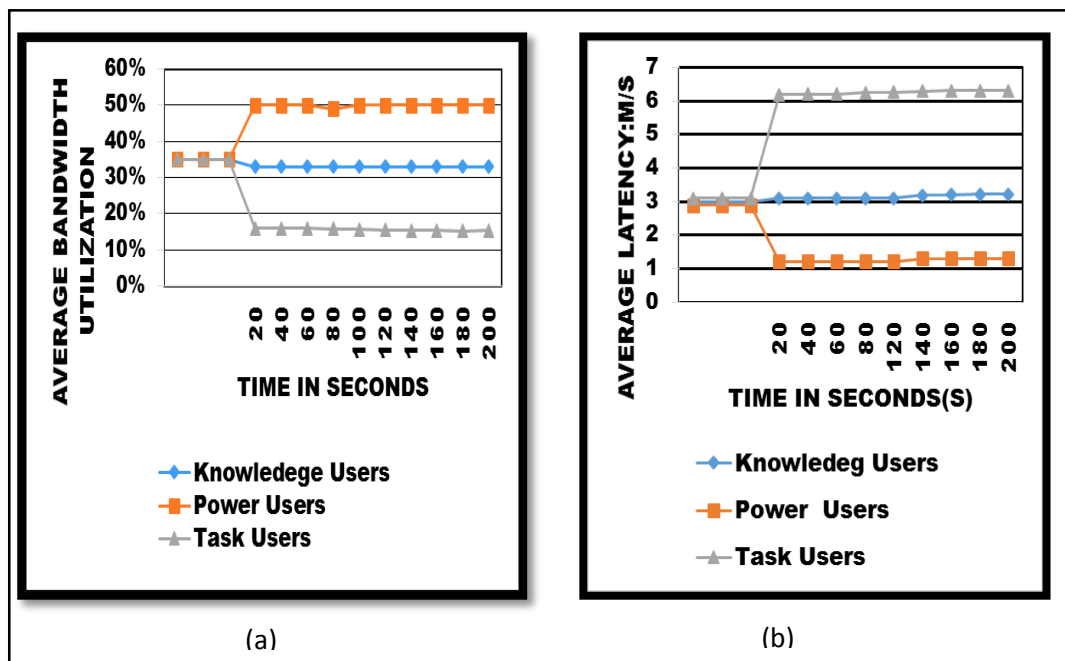


Figure 5.2 Bandwidth Allocation (a) Bandwidth Utilization and (b) Latency received for the three Classes of users with 1:2:3 share ratio.

From Figure 5.2 (a) it is observed that between time  $t=0$  to  $t=20$  all the classes of users seem to have equal utilization of bandwidth when HPDDRR is not activated. At  $t=20$  HPDDRR is activated and the results shows that it takes 10



seconds for the system to converge to each class of users SLO. This convergence time is better than that of PARDA(Gulati, Shanmuganathan, Zhang, & Varman, 2019) and mClock(Hao et al., 2017) of 30 seconds for the same configurations of IO size and queue depth. It is with activation of HPDDRR that bandwidth utilized by each class of users is proportional to the overall  $P_i$  values in proportion to the share ratio. Power users received a percentage ratio of 50% utilization, Knowledge users received an average percentage ratio of 33% and task users attained an average percentage ratio of 16%, each matching their 3:2:1 ratio. From this results it is evident that HPDDRR is able to maintain bandwidth allocation in proportion to the allocations based on their priority. Secondly it is observed that latencies achieved are consistent with the expected relationship between bandwidth allocation and latency. Higher bandwidth allocation results in smaller latency(Cui et al., 2019).

Figure 5.2 and Table 5.1 confirms the effectiveness of HPDDRR in bandwidth allocation where bandwidth is distributed based on priority. These results are similar to those obtained in Solutions like Stonehenge(Gulati et al., 2019), Argon(Wachs et al., 2007) and Aqua(Wu & Brandt, 2005) support proportional allocation where users get a disk time share proportional to their weights.

**Table 5.1: Bandwidth Utilization and Latency observed when Strict Priority allocation is used.**

<b>Class of User</b>	<b>% Utilization</b>	<b>Average Latency(MS)</b>
Task users	16	6.2
Knowledge users	33	3.1
Power users	50	1.2

Table 5.1 shows bandwidth utilization and latency observed. The results show high utilization was experienced by power users while the least utilization was experienced by task users reflects each class priority. Also it is observed from the table that the more the utilization the less the latency.

#### **5.4.2 Bandwidth Borrowing**

This section outlines results obtained from HPDDRR in attempts to optimize bandwidth borrowing.

**Table 5.2: Bandwidth Utilization and Latency observed when Bandwidth Sharing is implemented.**

<b>Class of User</b>	<b>% Utilization)</b>	<b>Average Latency(MS)</b>
Task users	17	5.8
Knowledge users	37	1.8
Power users	38	0.8

Table 5.2 illustrates an increase in average bandwidth utilization for task users and knowledge users as power users host was stopped at  $t=100$  seconds. The table shows that other classes of users increase their utilization ad HPDDRR

adopts to the changes in the network. From table 5.2 is also observed that an increase in utilization reduces the latency.

In this case the experiment intended to test the HPDDRR ability to implement bandwidth borrowing. It is expected that the proposed algorithm needs to be aware of changing bandwidth requirements and adopt accordingly based on priority. Experiments were carried out using a 1: 2: 3 share ratio. The three Hosts were used each generating a work load corresponding to the classes of task users, knowledge user and power users. Each host run a 64KB random read/write IO size. All the Hosts are started at the same time with the host corresponding to power users stopped at between times  $t=100$  to  $t=120$  seconds.

Between times  $t=0$  and  $t=20$  HPDDRR is not activated and all users seem to utilize equal share of bandwidth as well as experience the same latencies. Fig

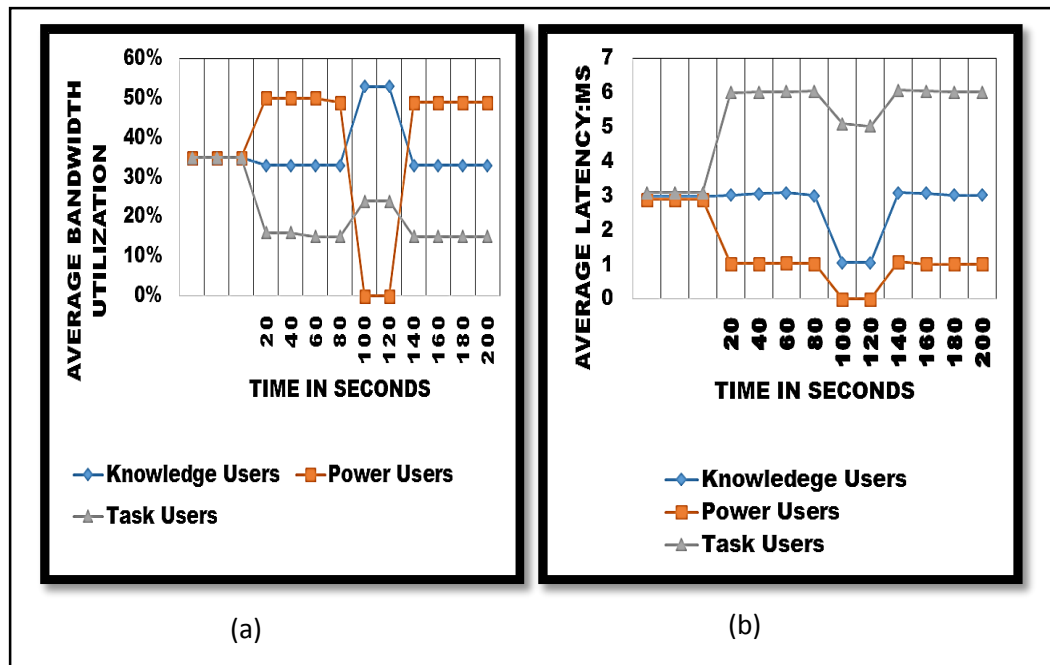


Figure 5.3: Bandwidth Borrowing (a) Bandwidth utilization adaptation (b) Latency adaptation based on the share ratio 1:2:3

However at  $t=20$  HPDDRR is started. Figure 5.3 plots the bandwidth and latency observed by the three classes of users for a period of 200 seconds.

Note that in Figure 5.3, all flows get utilization proportional to their priority from  $t=20$  to  $t=100$ . Note that when the host for power users was stopped at  $t=100$  seconds, the now available capacity is distributed in a proportional manner. Note that the power users did not receive any extra share when restarted at  $t=140$  seconds since its arrival rate is the same to its SLO rate. These results are similar to those achieved in ( Li & Feng, 2020) where they were able to optimize throughput and latencies for consolidated hosts under SLO constraints. In addition there is a clear reduction in latency for knowledge users and task users when the power users host was stopped. This affirms the claim by(Peng, 2019) that when throughput increases latency reduces. The results by Li and Feng(2020),Peng and Varman(2018) demonstrated the same pattern where an increase in throughput caused a corresponding decrease in latency. Throughput optimization attained by Li and Feng(2020) in their research was also achieved through bandwidth borrowing so that when particular host is not using its share, the excess bandwidth is distributed to those hosts that need it. In the study this has been achieved by determining maximum bandwidth distribution based on demands. Optimization of bandwidth usage increases the throughput as it reduces the latency. Similar patterns were observed in results obtained in (Peng & Varman, 2018)

In conclusion of this section it is noted that latency can be reduced by managing bandwidth for each class of users, an observation supported by results obtained by cited authors in the previous section. It also observed that with HPDDRR it takes 10 seconds to converge after power users host was stopped and then started. The convergence time is better than that obtained in PARDA(Gulati &

Waldspurger, 2007) where the convergence time was 30 seconds for the same configurations for IO size and Queue depth. The results obtained in this section demonstrate the HPDDRR algorithm capability of supporting bandwidth borrowing as a feature of supporting bandwidth management in IP SANs.

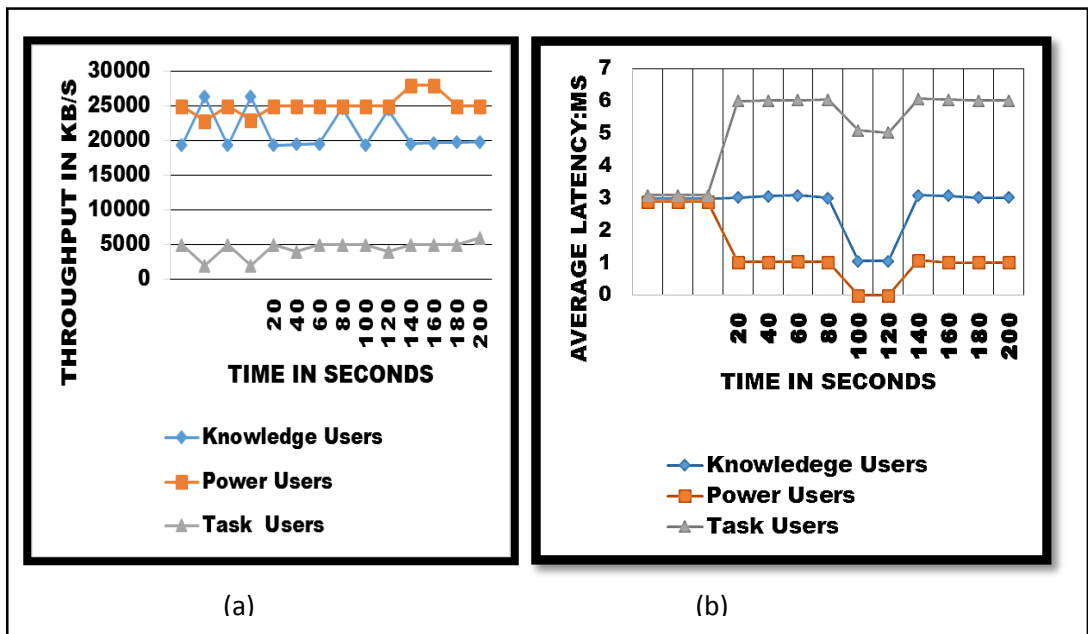
### **5.5 Handling Bursts**

As mentioned earlier, storage traffic is bursty in nature due to application characteristics among other factors discussed in chapter one of this thesis. This bursty nature of IO workload makes it difficult to implement proportionate bandwidth allocation as well as low latency. Experiments were run to establish how HPDDRR behaves when there are large bursts. Burst arrival scenario is simulated by having a class of users transmit flows whose rate is more than its allocated rate. A bursty flow is most likely to miss deadline due to high delays. It is expected that the algorithm should be able to absorb bursts for other flows that send bursts equal or less than the allowed value. Solutions like PARDA(Gulati et al., 2019) and mClock(Hao et al., 2017) use the idle credits to handle bursts. The flow with the greatest idle value is given the preference. Contrary to that, HPDDRR uses priority  $P_i$  to allocate idle bandwidth for handling bursts. This ensures the high priority traffic always gets best of service. Experiments were set up where, three Hosts each running windows server 2016 configured with a 26GB data disk were used. Each host run a 1MB read/write workload. A 1MB IO size was used so as to generate more traffic compared to 64KB.

**Table 5.3: Throughput and Latency observed when Priority based Burst Handling is Implemented.**

Class of User	Average Throughput(KB/s)	Average Latency(MS)
Task users	4500	6.1
Knowledge users	21223	4.0
Power users	25135	1.3

Table 5.3 results demonstrates how the system adapts when HPDDRR is enabled to adhere to each class of users SLO.it shows that HPDDRR uses excess bandwidth to handle burst. Knowledge users send burst continuously which causes an increase in latency when no excess bandwidth is available. Knowledge users experience violation of its SLO latency with an increase in latency from 3.2m/s to 4.0 m/s.



*Figure 5.4: Burst handling (a) Knowledge users sends burst more than allowed values Power users and Task users send bursts equivalent to allowed value (b) Knowledge users don't meet the deadline whereas power and task users meet the deadlines.*

To test how the system handles bursts, the following SLO parameters were used; <IMB,25000KB/s,1.4MS>,<IMB,20000KB/s,2.4MS>,<IMB,5000KB/s,6.4MS>,for power users, knowledge users and super users respectively. Figure 5.4 plots the results for 200 seconds. Table 5.3 illustrates the bandwidth and latencies attained when doing burst handling.

Figure 5.4 demonstrates how the system behaves before HPDDRR is enabled and after it is enabled. From Figure 5.4 it is noted that for the first 20 seconds knowledge users send bursts of 1200 KB every 5 seconds. This is seen to reduce the throughput of task users and powers users as well as increase their latency. In this case all the class of users SLO is violated. At  $t=20$  HPDDRR is enabled and takes 10 seconds to converge to the SLO. This convergence time is better than that of PARDA and mClock(Gulati et al., 2019) of 30 seconds for the same configuration of IO size and queue depth. At  $t=60$  the knowledge users send again spikes of 1200KB/s each 5 seconds, however this time other users are not affected. This can be attributed to the capability of HPDDRR to shape traffic. At  $t=140$  both power users and knowledge users send spikes of 2000KB every five seconds. However it is evident that the throughput for power user's increases but the latency does not unlike for knowledge users. This is due to the fact that HPDDRR uses priority to assign extra bandwidth for handling bursts unlike the knowledge users whose latency increase significantly due to lack of extra bandwidth for handling bursts which has been allocated to power users who have higher priority. This phenomena proves that HPDDRR uses priority

to handled bursts. High priority flows will be given priority when it comes to allocation of spare bandwidth required for transmitting bursts of traffic.

From Figure 5.4 it is further evident that HPDDRR is able to absorb burst if the burst value is not higher than the burst size parameter and therefore able to handle burst for well behaving flows. These results are similar to those achieved in researches by Peng et al., (2019) and Peng and Varman(2020) where the authors were able to guarantee latencies based on SLO by shaping workloads. This was made possible by ensuring bursty and non-busy flows are smoothed in order to avoid head of line congestion.

## **5.6 Summary**

In this chapter the problem of bandwidth management and burst handling was studied. The study proposes HPDDRR which uses hierarchical structure and a dynamic quantum to increase bandwidth utilization as well as reduce latencies experienced by flows in IP SANs.

Evaluation done on HPDDRR shows that it is able to provide proportional allocation of bandwidth to classes of users based on priority and adopt the utilization experienced by traffic classes of users based on network conditions. HPDDRR has also been proven through experiments that it is able to absorb bursts from classes of user's flows.

A hierarchical shaper can support more precise scheduling for the high rate traffic, this can significantly reduce latency and jitter relative to existing approaches. With the hierarchical structure the sorting granularity for



connection is reduced due to grouping. This reduces the implementation overhead and interference between competing connections.

## **CHAPTER SIX: INTEGRATION OF QOS TECHNIQUES AND VALIDATION**

### **6.1 Chapter Overview**

This chapter looks at the Integration and evaluation of the QOS techniques of performance isolation, bandwidth management and burst handling.

### **6.2 Integrated QOS Management Technique**

The implementation of IQMIS was done on a central router so as to ensure centralized management of QOS which is expected to reduce overhead of having the IQMIS running on multiple locations unlike in solutions such as PARDA(Gulati & Waldspurger, 2009) and Argon(Wachs et al., 2007) which cannot support centralized management of QOS.

Figure 6.1 illustrates the four functional modules of IQMIS namely; priority estimation module, performance isolation module, bandwidth management module and burst handling module. The figure further shows that IQMIS begins by estimating the priority of particular flows of traffic. This priority is used in the performing performance isolation. After performance isolation burst handling is done then bandwidth management. The traffic is then sent to the IPSAN. The Figure further illustrates that IQMIS gets its input from IPSAN for the adjustment of resources allocation based on the network statistics.

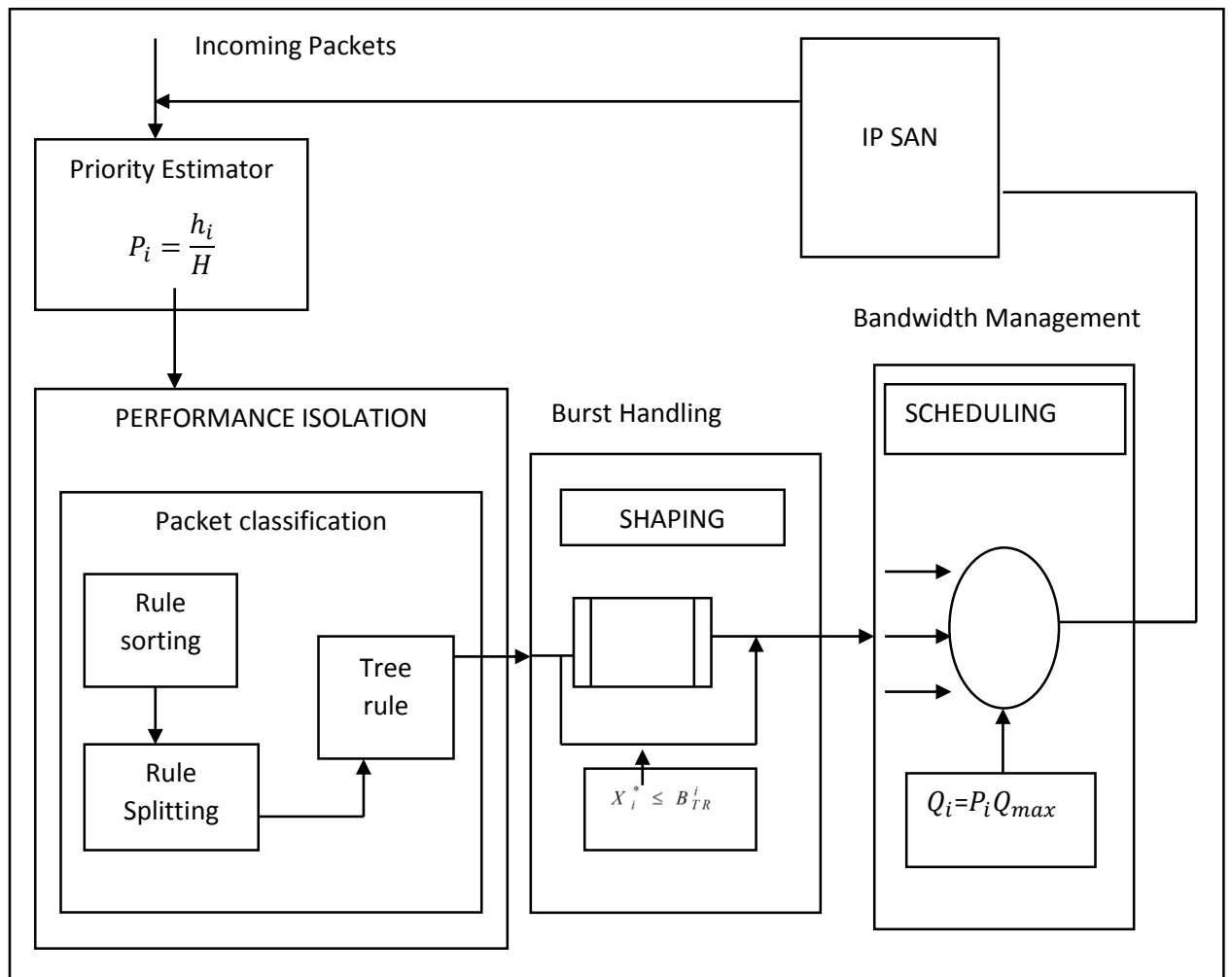


Figure 6.1: IQMIS Architecture.

### 6.2.1 Priority Estimation Module

The priority estimation module is designed to capture the current network statistics and calculate the priority of each flow  $i$ . The rule that experiences more hits and the class associated with this rule are given high priority. Whereas the rule that experiences less hits and its associated class are considered to be of lower priority. The priority is meant to be used to forecast on the amount of

resources a certain class of users requires. For each flow  $i$  the priority estimation module calculates its priority using equation  $p_i = \frac{h_i}{H}$ . The  $p_i = \frac{h_i}{H}$  indicates that the value  $H$  can be used to adjust the  $p_i$  for allocating SLO constraints. The larger the  $H$  the smaller the  $p_i$ . The smaller the  $p_i$ , the smaller the amount of resource particular flow will get. The values of  $p_i$  and those of latency and jitter are inversely related. That is when  $p_i$  increases latency and jitter reduces. On the other hand throughput and  $p_i$  are linearly related that is the bigger the value of  $p_i$  the larger the value of throughput.

### 6.2.2 Performance Isolation Module

In IP SANs environment there are mixed workloads which compete for bandwidth. To ensure that users operating within their SLOs do not get interference from users not operating within their SLOs, the performance isolation module classifies packets based on classes and enforces a strict isolation of resources to different classes of users. Since the proposed solution was implemented in a Linux environment, the researchers utilized the U32 as the classifier of choice as discussed in chapter three of this thesis.

A packet  $d_i$  is classified based on the header information that is source IP, destination IP, Source port, destination port and protocol. The same fields are components of rule  $r_i$ . A packet  $d_i$  is said to match rule  $r_i$  if and only if  $d_i.f_i = r_i.f_i$ . Based on the rule that match the associated action is performed. The rules are assigned priority based the number of hits then are sorted based on descending order of priorities with high priority rules sitting at the top. To further ease the search the rules are split into  $m$  partitions where  $m = \sqrt{N}t$  is the

number of partitions while  $N$  is the total number of rules. The partitions enables the use of jump search while the number of jumps will be equal to  $\frac{N}{m}$ . This reduces the time complexity from  $O(N)$  to  $O(\frac{N}{m} + m - 1)$ . Once the rules are partitioned a linear tree rule is built to place packets in their respective classes. After classification packets are forwarded to the shaper

### 6.2.3 Burst Handling Module

The burst handling module is meant to delay packets so that they form a constant flow. Burst handling is implemented using traffic shaping. The proposed traffic shaping algorithm takes in various QOS classes  $i$  ( $i=1..n$ ) and uses a dynamic time interval  $t_s$  to send traffic in burst. Since the egress interface is a timed interface it is not possible to reduce the rate at which it transmits packets, to achieve a rate lower than the interface rate, packets are sent then stopped at regular interval  $t_s$ . This eventually makes the average rate lower than interface rate. The time interval  $t_s$  is meant to ensure that on average a committed rate  $B_{TR}^i$  is sent for each class.

Each session consist of  $n$  queues  $q_i$  each containing flows belonging to the same class and priority. High priority queue are placed at the top .At any time if the queue is not full and the time  $t_s$  has not elapsed the incoming flows are not delayed. Otherwise the packets are sent and the  $t_s$  is reset to zero. The two events that trigger the sending of packets is when  $X_i^* \leq B_{TR}^i$  and  $t_s$  has expired.

### 6.2.4 Bandwidth Manager

The bandwidth management algorithm begins by establishing the quantum  $Q_i$  which is the amount bits that can be transmitted in each round from queue

$i$  based on priority  $p_i = \frac{h_i}{H}$ . The value  $Q_i$  represents  $P_i \times Q_{Max}$ . Where  $Q_{Max}$  is the maximum possible size of any packet that can exist in the network. To ensure service differentiation queues are arranged hierarchically in one level instead of one FIFO queue found in best effort. The one level arrangement ensures that the complexity of  $O(L)$  is obtained where  $L$  is the number of hierarchy levels. Again to retain the complexity of  $O(1)$ ,  $Q_i$  is always greater than any packet size. This means that for each round, queue  $i$  is able to transmit at least one packet. However if all the packets cannot be transmitted in the first round the remainder of  $Q_i$  is stored in counter  $DC_i$ . Otherwise if all packets are transmitted  $DC_i$  is set to zero. To ensure fairness the  $DC_i$  is to  $Q_i$  in the next round.

**Algorithm 6: IQMIS**

N: Total number of rules

D: Packets

m: Rule partitions

H: Total number of hits for rules

F: Fields of a rule or packet

Q: Quantum size

$B_{TR}^i$ : Total bandwidth allocated to class  $i$

W: Default action

Input: R, list of rules  $r \in R$

D: Packets  $d \in D$

Output: Flow of packets

//Count the hits

1. If  $current\ rule_i = incoming\ rule$  then
2.      $Count_i = count_i + 1$
3. Else

```

4.    $Count_i = 1$ 
5. EndIf
   // calculate priority
6.  $p_i = \frac{count_i}{H}$ 
   //Sort rules
7. For ( $i = 1; i < n; i++$ )
8.   If ( $p_i > p_{i+1} \wedge D_i \cap i+1 = \emptyset$ ) then
9.      $temp = r_i$ 
10.     $r_i = r_{i+1}$ 
11.     $r_{i+1} = temp$ 
12.   Endif
13. End for
   // Partition rules
14.  $m = \sqrt{N}$ 
   // Classify packets
15. For ( $m = 0; m < n; m++$ )
16.   For ( $i = 0; i < n; i++$ )
17.     If  $d_i \cdot f_i = r_i \cdot f_i \cdot p_i$  then
18.       Output  $a_i$ 
19.     Else
20.       Output  $w$ 
21.   End if End for End for
   // Burst handling
22. If  $X_i^* \leq B_{TR}^i$  Then
23.   Forward packets for scheduling
24. Else
25.   Queue packets (delay packets)
26. EndIf
   // Scheduling
27.  $Q_i = P_i Q_{max}$ 
28. If  $packet\ size \leq Q_i$  Then
29.   Dequeue packet
30.    $NQ_i = Q_i - Packet\ size$ 
31.   Forward packets
32. Else
33.   Queue++
34. End if
35. Stop

```

### 6.3 Validation of IQMIS

The following sections discuss how QIMIS was validated. Details of each validation approach are discussed therein.

### 6.3.1 Validation Metrics

Experiments were set up to establish the performance of the proposed system based on the QOS metrics of throughput, jitter and latency. Reads were simulated to mimic the real IP SAN environment.

Equations 6.1 and 6.2 were used to calculate the percentage throughput deviation and latency deviation respectively

$$\%throughput\ deviation = \frac{attained\ throughput - expected\ throughput}{expected\ throughput} \times 100 \quad 6.1$$

$$Latency\ deviation = observed\ latency - expected \quad 6.2$$

The following sections presents the results of throughput, latency and jitter obtained by using I/O sizes of 4KB, 64KB and 1MB for a period of 200 seconds. For all the experiments three scenarios were considered corresponding to IO sizes of 4KB, 64KB and 1MB. These choice of IO size was influenced by the fact that in the literature other researchers use the same IO sizes as used in this study. The choice was influenced by the need to compare the performance of other researches with the results of the study.

### 6.3.2 User QOS Mapping

Different users have varied QOS requirements which should be matched to corresponding QOS requirements. Users flows mapped to the same SLO are put on the same queue. Through this mapping the router can be able to provide differentiated treatment of flows. Based on the user's requirements in delay and throughput we map users to three QOS levels. The mapping relations are shown



in Table 4.3. Power users and knowledge users are sensitive to delay as illustrated by the low latency/response time. The task users are less sensitive to delay however they require bandwidth guarantee.

### **6.3.3 Validation Setup**

The validation experiment is as illustrated in Figure 3.1. Specifications for initiators and targets are as illustrated in Table 3.2. Parkdale disk benchmarking tool was used to simulate the reads and writes. The initiators were setup with initiator mode ISCSI driver while the target storage were configured target mode. Experiments were used to validate the proposed systems based on latency, throughput and jitter. In all the experiments a File size of 50MB was used unless otherwise stated. All experiments were run three times for a period 200 seconds and averages recorded.

### **6.4 Validation Results**

Table 4.7 shows a summary of all the packets generated per each class of user based on IO size.

**Table 6.1 Total Number of Packets Generated**

<b>IO Size</b>	<b>Total Number of Packets Generated Per Class for ISCSI Protocol.</b>			
	<b>Knowledge user</b>	<b>Task users</b>	<b>Power users</b>	<b>Default class</b>
<b>4KB</b>	60,645,186	56,842,884	60,091,227	2,736,342
<b>64KB</b>	50,278,725	47,894,490	52,016,415	2,194,290
<b>1MB</b>	38,649,555	38,345,076	45,360,000	1,556,415
<b>Grand Total</b>	149573466	143082450	157,467,642	6487047

Table 6.1 shows that power users generated more packets than other classes of users. This means that rules associated with the class experiences more hits therefore making the class high priority class.

#### **6.4.1 Throughput versus IO Size**

Throughput is the measure of the number of packets delivered to the destination successfully(Nam et al., 2020). For good QOS the value of throughput should be high. We expect correlation between throughput and IO size. To verify this experiments were configured with three hosts across a 26 GB volume.

Figure 6.2 shows the throughput of task users, knowledge users and power in the best effort case and using the proposed solution scenarios. In best effort scenario in as illustrated in Figure 6.2(a), the lack of QOS management scheme causes hosts to have unstable throughput and a lot of unfairness. Ideally power users would perform better than other users. In the proposed solution scenario as illustrated by Figure 6.2(b), the unfairness is corrected by isolating users by

decoupling the throughput of the three classes of users and lets them process packets at their own rates.

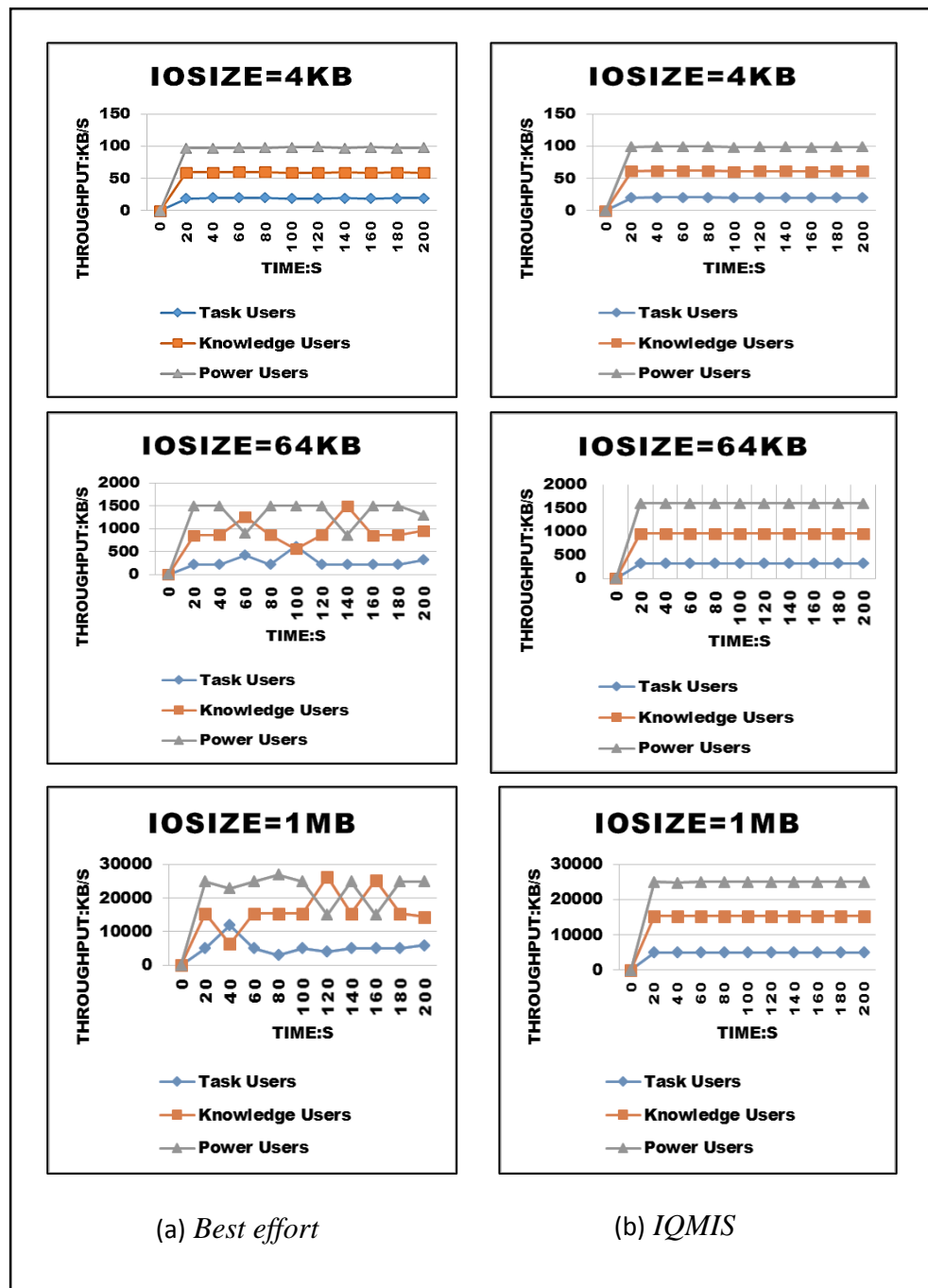


Figure 6.2: Throughput for 200 seconds (a) Best effort (b)IQMIS.

Generally from Figure 6.2(b) it is observed that at  $t=0$  there is the lowest throughput which increase up to  $t=20$  where it stabilizes. The stability is brought about by the proposed solution being able to optimize bandwidth usage as well as isolate performance of one flow from the other. Table 6.2 further illustrates the results of scenario 1.

**Table 6.2: Scenario 1 with IO size of 4KB**

Class of User	Expected SLO Throughput	IQMIS Throughput	Best Effort	Deviation 1	Deviation 2
Task users	20	19.4	19	-0.6	-1
Knowledge users	60	59.5	59	-0.5	-1
Power Users	100	99.8	99	-0.2	-1

Scenario1 represents the situation when using an IO size of 4KB for both using IQMIS and best effort. Table 6.2 shows that all users receive a throughput close to the SLO with a negative percentage deviation from the SLO of 3%, 0.8% and 0.2 Kb/s for task users, knowledge users and power users respectively when using the IQMIS. The same is observed when using the best effort. That is the task users, knowledge users, and power users attained a negative percentage deviation of 5%, 1.6% and 1%. These results show that when using best effort the reduction in throughput increases by a factor 2X compared to when using

IQMIS. These indicates that IQMIS is able to make the network operate very close to the SLO compared to best effort. However the deviation are small and this can be explained by the fact that since with IO size of 4KB the congestion is low and therefore all the users are able to meet their SLO. This is consistent with results obtained by authors Jaiman et al.,(2018) showing that a large IO size produce more traffic that would congest the network. Therefore the smaller the IO size the less the congestion. Table 6.3 depicts the results of scenario 2.

**Table 6.3: Scenario 2 with IO size of 64KB**

Class of user	Expected SLO Throughput	IQMIS Throughput	Best Effort	Deviation 1 (IQMIS)	Deviation 2 (Best Effort)
Task users	320	300.16	263.13	-19.84	-56.87
Knowledge users	960	919.55	858.09	-40.45	-101.91
Power Users	1600	1540.03	1282.22	-59.97	-317.78

In scenario 2 when using an IO size of 64KB the following observation were made. Results in Table 6.3 indicate that with the IQMIS a negative percentage deviation from the SLO of 6.2%, 4.2% and 3.7% for task users, knowledge users and power users respectively was attained. On the other hand when using best effort a negative percentage deviation from the SLO of 17%, 10.6% and 19.86% for task users, knowledge users and power users respectively.

This can be explained by the fact that an increase in IO size results in a corresponding increase in traffic which causes congestion(Jaiman et al., 2018). However for the proposed solution since it implements performance isolation, bandwidth management and traffic shaping the deviation is minimal compared to that of best effort. Table 6.4 represents the results of scenario 3.

**Table 6.4: Scenario 3 with IO size of 1MB**

<b>Class of user</b>	<b>Expected SLO Throughput</b>	<b>IQMIS Throughput</b>	<b>Best Effort</b>	<b>Deviation 1 (IQMIS)</b>	<b>Deviation 2 (Best Effort)</b>
Task users	5000	4012.3	3543.30	987.7	1456.7
Knowledge users	15000	12750	11762	2250	3238
Power Users	25000	22890	20896	2110	4104

In scenario 3 an IO size of 1MB was used and the results are as indicated in table 6.4. Table 6.4 shows that IQMIS all users experienced a negative percentage deviation from the SLO of 19.8%, 15% and 8% for task users, knowledge users and power users respectively. On the other hand when using best effort a negative percentage deviation from the SLO of 29.1%, 22% and 16% for task users, knowledge users and power users respectively.

This can be explained by the fact that an increase in IO size results in a corresponding increase in traffic which causes congestion(Jaiman et al., 2018).

However for the proposed solution since it implements performance isolation, bandwidth management and traffic shaping the deviation is minimal compared to that of best effort.

It is further observed that when using best effort the task user's experiences the greatest deviation for scenario 2 and scenario 3 and the lowest is experienced by knowledge users. This is contrary to what is expected given that power users have got higher priority and therefore should have a smaller percentage deviation. This can be explained by the fact that best effort technique lacks the mechanism of prioritization present in IQMIS. This is consistent with results obtained in (Gulati & Varman, 2010) where it was found that resource reservations and controls are able to provide predictable performance. In the results obtained the expectations were that high priority users should be provided with predictable service. The results obtained were consistent with the expectations and those obtained by Billaud and Gulati(2013),Gulati and Waldspurger(2009) and Peng(2019) proving that IQMIS is work conserving

#### **6.4.2 Latency and IO size**

Latency is the time it takes for a packet to reach its destination. Latency has a lot of effect on network performance degradation and effects the user QOS. High latency is caused by congestion which results in poor QOS. In this case the study considered end to end delay that is the time taken from source to destination(Jaiman et al., 2019). Figure 6.3 analyzes the latency experienced by the three classes of users against time for best effort and proposed solution. Latency was measured for three scenarios for IO sizes of 4KB, 64KB and 1MB.

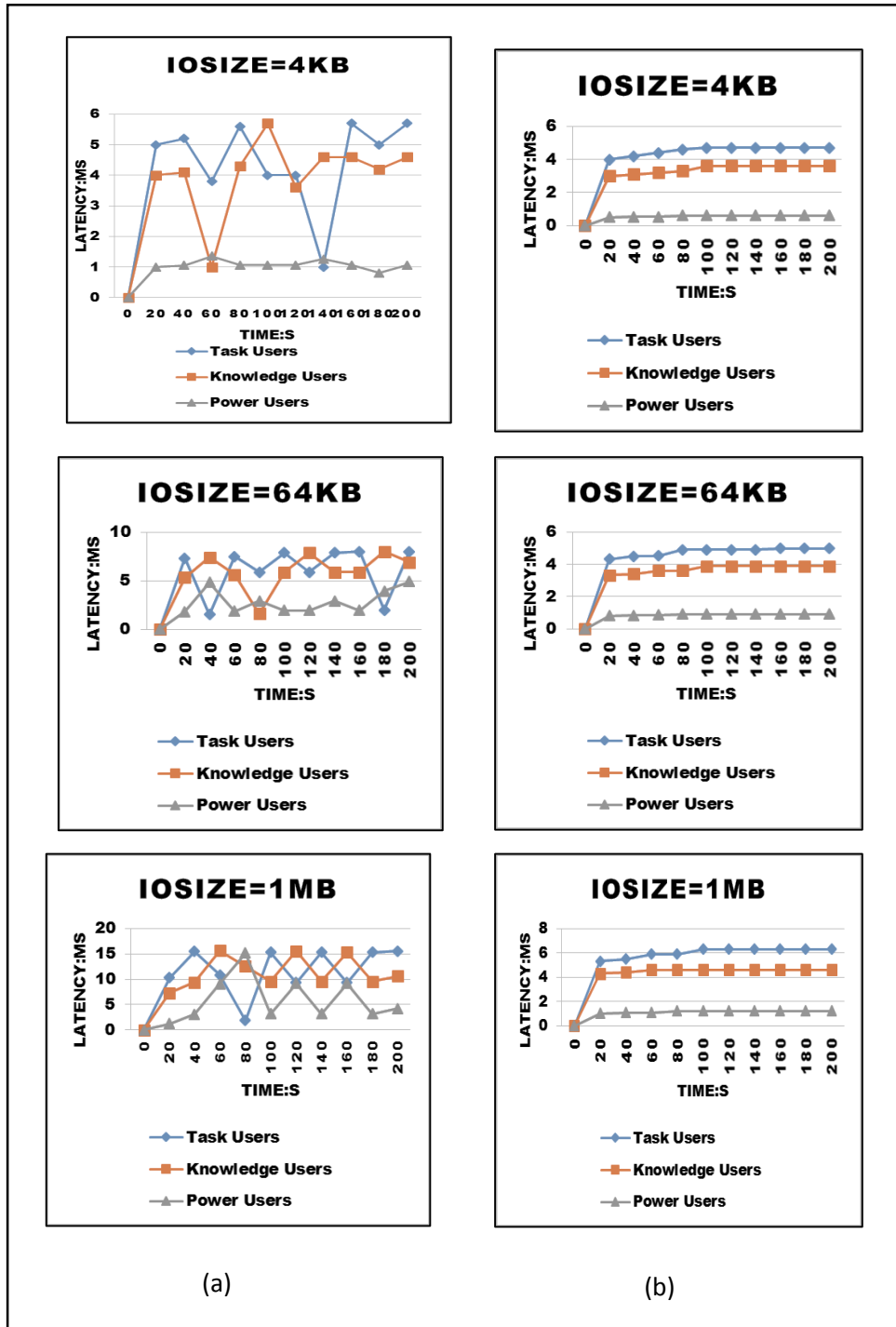


Figure 6.3: Latency for 200 seconds (a) Best effort, (b) IQMIS

For scenario 1 where an IO size of 4KB was used all users experienced a latency lower than that expected for both the IQMIS and the best effort. Even though



the best effort has no QOS mechanisms implemented here in IQMIS, all the users still meet their deadlines. This can be explained by the fact that when a small IO size is small there is low congestion since they occupy the network for a short time(Jaiman et al., 2019) which does not lead to resource competition and therefore does not require any management. Table 6.5 depicts scenario 1.

**Table 6.5: Scenario 1 with IO size of 4KB**

<b>Class of User</b>	<b>Expected SLO Latency</b>	<b>IQMIS Latency</b>	<b>Best Effort Latency</b>	<b>Deviation 1 (IQMIS)</b>	<b>Deviation 2 (Best Effort)</b>
Task users	6.4	5.6	6.0	-0.8	-0.4
Knowledge users	3.2	2.4	2.9	-0.8	-0.3
Power Users	1.3	0.6	1	-0.7	-0.3

**Table 6.6: Scenario 2 with IO size of 64KB**

<b>Class of user</b>	<b>Expected SLO Latency</b>	<b>IQMIS Latency</b>	<b>Best Effort Latency</b>	<b>Deviation 1 (IQMIS)</b>	<b>Deviation 2 (Best Effort)</b>
Task users	6.4	5.8	7.7	-0.6	+1.3
Knowledge users	3.2	2.8	5.4	-0.4	+2.2
Power Users	1.3	1.1	2.6	-0.2	+1.3

**Table 6.7: Scenario 3 with IO size of 1MB**

<b>Class of user</b>	<b>Expected SLO Latency</b>	<b>IQMIS Latency</b>	<b>Best Effort Latency</b>	<b>Deviation 1 (IQMIS)</b>	<b>Deviation 2 (Best Effort)</b>
Task users	6.4	6.0	10.79	-0.4	+4.39
Knowledge users	3.2	2.8	10.47	-0.4	+7.27
Power Users	1.3	1.1	5.52	-0.2	+4.22

In scenario two and three an IO size of 64KB and 1MB are used respectively. An increase in IO size resulted in an increase in the traffic which leads to competition of bandwidth(Jaiman et al., 2019). Tables 6.5, 6.6 and 6.7 shows

that IQMIS achieves a negative deviation for all users. A negative deviation means that users were able to achieve a latency lower than expected which means all users were able to meet their deadlines. Conversely with best effort, it was observed that all users attained a positive deviation which means that user surpassed the latency threshold that was expected and none met their deadlines. This phenomena can be explained by the fact that best effort lacks the QOS techniques of performance isolation, bandwidth management and burst handling implemented in the proposed solution.

Absence of these mentioned techniques results in free for all competition for bandwidth due to lack of prioritization mechanisms users are able to interfere with each other resulting in an ununiformed latency(Gulati & Varman, 2007). In addition FIFO queues used in best effort do not provide a way for isolating traffic. An increase in latency can also can be attributed to the use of DRR in best effort. When using DRR for scheduling, big packets lead to an increase in head of line latency which delays smaller maybe more important packets. Results obtained by Wang et al., (2012) also showed similar pattern where it was found that achieving low latency requires smaller queues. Lack of optimized scheduling algorithm results in larger queues which results in increased latency as witnessed in best effort. Similarly mixing of big and small packets results in headline delay causing more latency for smaller packets as witnessed when best effort is used.

It is further noted that for all the scenarios all users experience a low latency between time  $t=0$  and time  $t=20$ . This is because before the 20 seconds traffic

has not reached saturation therefore there is less latency. In summary it is important to note that when using IQMIS, each class of user the latency goals are satisfied contrary to when using best effort. When using best effort latency increases by a factor of 2X. This demonstrates the inability of conventional scheduling techniques in providing acceptable latencies in presence of huge traffic loads.

### 6.4.3 Jitter and IO size

Jitter is the variation in delay experienced by packets reaching a destination thus its presence is unwanted but unavoidable. Therefore there is always a small amount of jitter. From the experiments jitter observed under IQMIS and best effort was recorded as shown in Table 6.8 and Figure 6.5.

**Table 6.8: Average Jitter in Milliseconds**

		IQMIS			BEST EFFORT		
Storage user	Time	4KB	64KB	1MB	4KB	64KB	1MB
Task users	200s	4.2	5.2	5.6	6.0	6.5	9.9
Knowledge users	200s	3.2	3.5	3.8	4.0	4.7	7.6
Power users	200s	1.0	1.1	1.2	1.3	2.3	4.4

Table 6.8 illustrates average jitter in milliseconds with IQMIS experiencing smaller jitter compared to best effort for all IO sizes. It is further observed that the jitter increases with an increase in IO size.

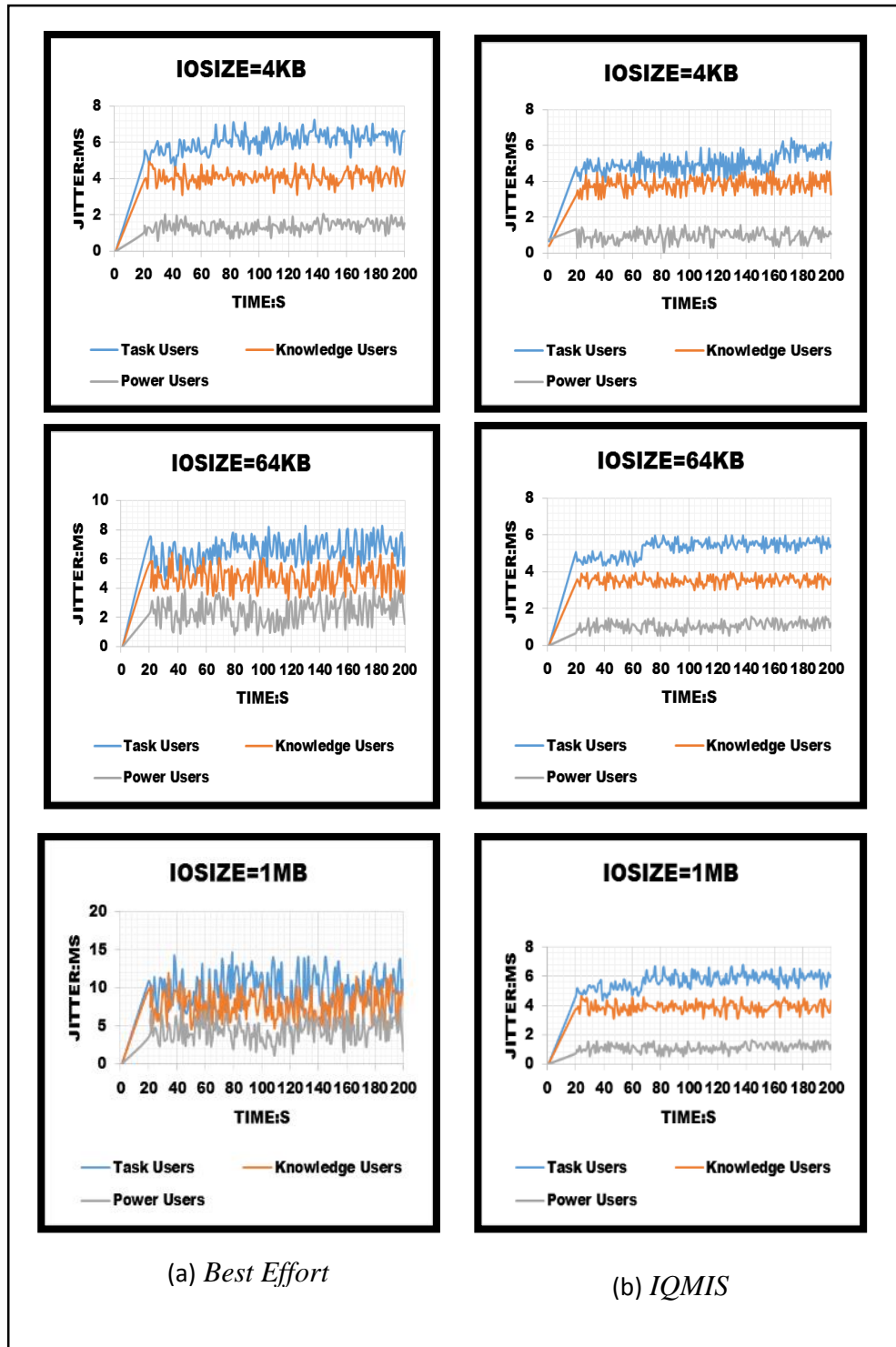


Figure 6.4: Jitter for 200 seconds (a) Best Effort (b) IQMIS

Figure 6.4 analyzes the jitter experienced for 200 seconds for IO sizes of 4kb, 64KB and 1MB. At  $t=0$  a jitter of 0 was observed and therefore the system experiences the best performance at  $t=0$ .

From Figure 6.4 it is observed that when an IO size of 4kbyte is used there is no congestion and therefore jitter of the three classes of users is small. This is consistent with results obtained by Peng and Varman(2020) and Jaiman et al.,(2019) that the larger the IO size the more time the traffic occupies the network and therefore the more the jitter.

For an IO size of 64KB the best effort technique a jitter of 6.5 for task users, 4.7 for knowledge users and 2.3 for power users was observed. While for the IQMIS jitter of 5.2, 3.5 and 1.1 were observed for task users, knowledge users and power users respectively. A reduction of 20%, 25% and 52% on average. This clearly shows that the IQMIS outperforms best effort. The same trend of an increase in jitter is observed for 1MB. The reason why the proposed technique outperforms the best effort is that the proposed technique uses a hierarchy of levels for flows which isolates traffic and avoids interference between flows as opposed at the best effort where all flows are using single FIFO queues. Minimum jitter was observed for 4kbyte, while maximum jitter was observed for 1MB IO size. From the results it is also observed that task users have the highest jitter for the given configuration. This can be attributed to their low priority.

## **6.5 Summary**

In this chapter the integrated approach has been implemented that includes the QOS techniques of performance isolation, bandwidth management and traffic shaping. The performance isolation module ensures that flows don't interfere with each other performance. The bandwidth management module ensures that each flow/class of user gets a share proportion to its current need. This is achieved through regular computation of priority. The IQMIS is implemented in Linux router and causes little delay. Through experimentation it has been verified that the IQMIS works as intended.

## **CHAPTER SEVEN: CONCLUSION, RECOMMENDATIONS AND FUTURE WORK**

### **7.0 Chapter Overview**

This chapter discusses a summary of the study as well as the main contributions of the study. Then the thesis looks at future work and possible extensions to the proposed solution

### **7.1 Conclusions**

Contributions of this thesis were drawn from meeting the set out objectivist contributions include: A comprehensive literature review on QOS in IP Networks. Optimization of QOS techniques of performance isolation (ELPCIS), bandwidth management and burst handling (HPDDRR). Development of IQMIS based on the optimized techniques of performance isolation, bandwidth management and burst handling. Demonstration of improved QOS provision by new IQMIS. Briefly, each of these is described and the sections of coverage highlighted for ease of reference

**Analyze the QOS techniques used in IP networks:** To address the first objective, the thesis first looked at technologies for the support of IP storage area networks were reviewed in section 2.7 these include FCIP, IFCIP and ISCSI. ISCSI was found to be able to run on any existing IP network unlike FCIP and IFCIP which require some fiber channel aspect. ISCSI was also found to be cheap and easy to implement.

Secondly in section 2.12 the thesis looked at the various QOS metrics that relates to storage area networks which were to be used to evaluate the performance of the ultimately proposed solution. Consequently, the various



QOS techniques namely the Intserve, Diffserve and MLP were identified and analyzed as presented in section 2.13 of this thesis. It was established that the Diffserve was more simpler and scalable QOS and therefore suitable for the IP SANs.

**Optimization of Performance isolation:** To solve the problem of performance isolation optimization, the study came up with ELPCIS which is a classifier that groups traffic based on classes to provide limits and reserves in order to achieve performance isolation between the groups of users. Through the use of limits and reserves ELPCIS is able to achieve performance isolation which allows classes of users to achieve throughput and latencies within their SLO independent of the behavior of other users. Empirical results shows that ELPCIS is light weight and efficient with a low implementation cost of 6% compared to that of List based packet classifier of 48% which is a reduction in terms of cost by 42%. Further the results show that ELPCIS has classification accuracy of 89% compared to that of Lists based packet classifier of 56% which is an improvement in terms of accuracy by 33%. ELPCIS provides a scalable performance isolation algorithm that uses network statistics to allocate resources and achieves elastic network utilization.

**Optimization of bandwidth management and burst handling:** Chapter five of this thesis combined the objectives of optimization of bandwidth management and burst handling. To address the problem of optimization of bandwidth management and burst handling the study developed HPDDRR which is a Scheduler shaper that dynamically and adaptively applies priority to

attain optimization goal. HPDDRR enforces hierarchical reservation, limit and proportional shares of bandwidth based on priority. Specifically HPDDRR maximizes the throughput and reduces latency under the stated constraints. Through experiments the study has showed that HPDDRR is work conserving by implementing bandwidth sharing and is also able to efficiently enforce controls for diverse workloads. In addition the results shows that well behaving flows do not miss deadlines. Moreover HPDDRR out performs earlier solutions in terms of adopting quickly to the network changes by incurring a small convergence time of 10 seconds compared to convergence of 30 seconds incurred in best effort and earlier solutions. A reduction of convergence time by 20 seconds. Furthermore to network administrators and researchers, HPDDRR offers a solution that automatically uses network statistics to determine priority of flows and allocates network resources appropriately to flows based on their priority requirements without requiring manual interventions.

**Integration and validation of the integrated technique:** QOS is a vital issue in environment of mixed works like IP SANS. To solve the problem of offering complete QOS in IP SANS the study came up with the design and validation of IQMIS and presented the results. Evaluation of IQMIS shows that it is able to provide fair access to storage, control latency close to the SLO and provide high throughput than best effort. Empirical results show that when using IQMIS Latency is reduced by a factor of 2X as compared to best effort. Throughput is improved by 2000kb/s for high priority users. Experiments further show an improvement in convergence time to 10 seconds as compared to the best effort and existing solutions 30 seconds. IQMIS was found to provide strong

performance isolation, superior latency, throughput and jitter compared to Best Effort. Although prioritization would result in starvation, however the dynamic nature of IQMIS and reservations to users is able to prevent starvation. Moreover IQMIS is able to provide end to end prioritization of users in an environment of variable workloads like IPSANS. IQMIS can be used to provide end to end QOS control solution in IPSANS. Ultimately IQMIS can be used as a catalyst for developing frameworks and techniques for providing end to end QOS in IPSANS.

## **7.2 Future Work**

This study opens several interesting avenues for future research that we would recommend. In future the researchers would also like to explore techniques of using performance isolation in providing availability and reliability guarantees. In addition the authors of this thesis would like to recommend the use of the proposed techniques of performance isolation and HPDDRR technique Non storage systems. Finally the thesis would like to recommend the use of the integrated QOS management technique in providing building blocks to implement application based QOS.

## **7.3 Publications**

### **7.3.1 First Original Research Article Publication Titled**

“An Enhanced List Based Packet Classifier for Performance Isolation in Internet Protocol Storage Area Networks” as shown in Appendix D

### **7.3.1 Second Original Research Article Publication Titled**

“HPDDRR: Optimized Scheduler Shaper for Bandwidth Management and Traffic Shaping in Internet Protocol Storage Area Networks” as shown in Appendix E.

## REFERENCES

- Acharya, S., Member, S., Znati, T., & Member, S. (2008). Architectures and Algorithms to aid Firewall Optimization. *International Journal of Computer Applications*, 6(1), 1–16.
- Acharya, S., Wang, J., Ge, Z., Znati, T., & Greenberg, A. (2006, April). Simulation study of firewalls to aid improved performance. In *39th Annual Simulation Symposium (ANSS'06)* (pp. 8-pp). IEEE.
- Ademaj, F., & Bernhard, H. P. (2022). Quality-of-Service-Based Minimal Latency Routing for Wireless Networks. *IEEE Transactions on Industrial Informatics*, 18(3), 1811–1822. <https://doi.org/10.1109/TII.2021.3071596>
- Akbar, F., Yektakhah, B., Xu, H., & Sarabandi, K. (2021). A Low-Complexity Time-Domain Method for a Fast and Accurate Measurement of Q-Factor and Resonant Frequency of RF and Microwave Resonators. *IEEE Access*, 9, 96478–96486. <https://doi.org/10.1109/ACCESS.2021.3094409>
- Ali, B. S., & Chen, K. (2019). Towards Efficient , Work-Conserving , and Fair Bandwidth Guarantee in Cloud Datacenters. *IEEE Access*, 7, 109134–109150. <https://doi.org/10.1109/ACCESS.2019.2930888>
- Aljoby, W., Wang, X., Fu, T. Z., & Ma, R. T. (2019). On SDN-enabled online and dynamic bandwidth allocation for stream analytics. *IEEE Journal on Selected Areas in Communications*, 37(8), 1688-1702.
- Alkharasani, A. M., Othman, M., Abdullah, A., & Lun, K. Y. (2017). An Improved Quality-of-Service Performance Using RED's Active Queue Management Flow Control in Classifying Networks. *IEEE Access*, 5, 24467-24478.
- Almesberger, W. (1999). *Linux Network Traffic Control Implementation Overview*. Available at EPFL ICA <http://diffserv.sourceforge.net>.
- Almutairi, M., Stahl, F., & Bramer, M. (2021). ReG-Rules: An Explainable Rule-Based Ensemble Learner for Classification. *IEEE Access*, 9, 52015–52035. <https://doi.org/10.1109/ACCESS.2021.3062763>
- Alvarez, I., Moutinho, L., Pedreiras, P., Bujosa, D., Proenza, J., & Almeida, L. (2020). Comparing Admission Control Architectures for Real-Time Ethernet. *IEEE Access*, 8(ii), 105521–105534. <https://doi.org/10.1109/ACCESS.2020.2999817>
- Amid, A., Biancolin, D., Gonzalez, A., Grubb, D., Karandikar, S., Liew, H., ... Nikolic, B. (2020). Chipyard: Integrated Design, Simulation, and Implementation Framework for Custom SoCs. *IEEE Micro*, 40(4), 10–21. <https://doi.org/10.1109/MM.2020.2996616>
- Amjad, A. (2019). Analysis of QoS in Different Application by using Opnet Workbench. *International Journal of Advancements in Technology*, 10(1), 1–6. <https://doi.org/10.4172/0976-4860.1000221>

- Azadegan, M., & Beheshti, M. T. H. (2014, December). PID-type congestion controller design for TCP networks. In *2014 IEEE Conference on Systems, Process and Control (ICSPC 2014)* (pp. 7-12). IEEE.
- Baskaran, S. B. M., Raja, G., Bashir, A. K., & Murata, M. (2017). QoS-aware frequency-based 4G+ relative authentication model for next generation LTE and its dependent public safety networks. *IEEE Access*, *5*, 21977-21991.
- Baidya, S., Chen, Y., & Levorato, M. (2018, April). eBPF-based content and computation-aware communication for real-time edge computing. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (pp. 865-870). IEEE.
- Baklizi, M., & Ababneh, J. (2016). Performance Evaluation of the Proposed Enhanced Adaptive Gentle Random Early Detection Algorithm in Congestion Situations. *International Journal of Current Engineering and Technology*, *6*(5), 1658-1664.
- Balan, D. G., Potorac, A. D., & Graur, A. (2015). IPV6 EXTENSION OF HTB-TOOLS FOR LINUX TRAFFIC CONTROL BASED ON HTB. *Acta Technica Napocensis*, *56*(3), 48.
- Balogun, G. B. (2019). A Modified Linear Search Algorithm. *African Journal of Computing & ICT*, *12*(2), 43-54.
- Bangquan, X. I. E., & Xiong, W. X. (2019). Real-Time Embedded Traffic Sign Recognition Using Efficient Convolutional Neural Network. *IEEE Access*, *7*, 53330–53346. <https://doi.org/10.1109/ACCESS.2019.2912311>
- Barzegar, B., & Fatehi, S. (2022). An Efficient Hybrid Ranking Method for Cloud Computing Services Based on User Requirements. *IEEE Access*, *10*(June), 72988–73004. <https://doi.org/10.1109/ACCESS.2022.3189172>
- Bassi, M. A., Lopez, M. A., Confalone, L., Gaudio, R. M., Lombardo, L., & Lauritano, D. (2020). Enhanced Reader.pdf. *Nature*, Vol. 388, pp. 539–547.
- Berger, A. W., & Whitt, W. (1998). Effective bandwidths with priorities. *IEEE/ACM Transactions on Networking*, *6*(4), 447–460. <https://doi.org/10.1109/90.720887>
- Bhaumik, M., Saha, S., & Das, S. (2016). A new modified linear search. *International Journal of Computer Applications* .2,85-86.<https://doi.org/10.13140/RG.2.2.12687.38560>

- Bigang, L. I., Jiwu, S. H. U., & Weimin, Z. (2006). *SCSI Target Simulator Based on FC and IP Protocols in TH-MSNS Tsinghua Science and Technology*, 11(5), 589-596.
- Billaud, J. P., & Gulati, A. (2013, April). hClock: Hierarchical QoS for packet scheduling in a hypervisor. In *Proceedings of the 8th ACM European Conference on Computer Systems* (pp. 309-322).
- Binder, R. V. (2018). Optimal scheduling for combinatorial software testing and design of experiments. *Proceedings - 2018 IEEE 11th International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2018*, 295–301. <https://doi.org/10.1109/ICSTW.2018.00063>
- Biswas, S. (2010). Network storage and its future. *International Journal of Computer Science and Information Technologies*, 1(4), 235–239.
- Blenk, A., Kellerer, W., & Schmid, S. (2019). On The Impact of the Network Hypervisor on Virtual Network Performance. *2019 IFIP Networking Conference (IFIP Networking)*, 1–9.
- Bonati, L., Johari, P., Polese, M., D’Oro, S., Mohanti, S., Tehrani-Moayyed, M., ... Melodia, T. (2021). Colosseum: Large-Scale Wireless Experimentation Through Hardware-in-The-Loop Network Emulation. *2021 IEEE International Symposium on Dynamic Spectrum Access Networks, DySPAN 2021*, 105–113. <https://doi.org/10.1109/DySPAN53946.2021.9677430>
- Bora, G., Bora, S., Singh, S., & Arsalan, S. M. (2014). OSI reference model: An overview. *International Journal of Computer Trends and Technology (IJCTT)*, 7(4), 214-218.
- Border, D. (2018, June). Performance of a Linux-based Network Router. In *2018 ASEE Annual Conference & Exposition*.
- Bosk, M., Gajić, M., Schwarzmann, S., Lange, S., & Zinner, T. (2021). HTBQueue: A Hierarchical Token Bucket Implementation for the OMNeT++/INET Framework. *arXiv e-prints*, arXiv-2109.
- Brahneborg, D., Duvignau, R., Afzal, W., Mubeen, S., & Member, S. (2022). GeoRep — Resilient Storage for Wide Area Networks. *IEEE Access*, 10(April), 75772–75788. <https://doi.org/10.1109/ACCESS.2022.3191686>
- Brown, M. A. (2006). *Traffic Control HOWTO*. <http://linux-ip.net/articles/Traffic-Control-HOWTO/>.
- Bryman, A., Becker, S., Sempik, J., Bryman, A., Becker, S., & Sempik, J. (2008). *Qualitative and Mixed Methods Research: A View from Social Policy Quality Criteria for Quantitative, Qualitative and Mixed Methods Research: A View from Social Policy*. (July 2013), 37–41. <https://doi.org/10.1080/13645570701401644>
- Cedillo, P., Insfran, E., Abrahao, S., & Vanderdonckt, J. (2021). Empirical Evaluation of a Method for Monitoring Cloud Services Based on Models

- at Runtime. *IEEE Access*, 9, 55898–55919. <https://doi.org/10.1109/ACCESS.2021.3071417>
- Celik, A., Radaydeh, R. M., Al-Qahtani, F. S., & Alouini, M. S. (2017). Resource allocation and interference management for D2D-enabled DL/UL decoupled Het-Nets. *IEEE Access*, 5, 22735-22749.
- Chambliss, D. D., Alvarez, G. A., Pandey, P., Jadav, D., Xu, J., Menon, R., & Lee, T. P. (2003, October). Performance virtualization for large-scale storage systems. In *22nd International Symposium on Reliable Distributed Systems, 2003. Proceedings.* (pp. 109-118). IEEE.
- Chang, W., Wu, C., & Lin, Y. (2018). Efficient Time-Slot Adjustment and Packet-Scheduling Algorithm for Full-Duplex Multi-Hop Relay-Assisted mmWave Networks. *IEEE Access*, 6, 39273–39286. <https://doi.org/10.1109/ACCESS.2018.2856867>
- Cherian, M., & Chatterjee, M. (2016). Optimized Firewall with Traffic Awareness. *Int. J. Comput. Networks Appl*, 3(2), 32-37.
- Chin, K. (2021). Joint Trajectory and Link Scheduling Optimization in UAV Networks. *IEEE Access*, 9, 84756–84772. <https://doi.org/10.1109/ACCESS.2021.3086954>
- Chin, T., Xiong, K., & Hu, C. (2018). Phishlimiter : A Phishing Detection and Mitigation Approach Using Software-Defined Networking. *IEEE Access*, 6, 42516–42531. <https://doi.org/10.1109/ACCESS.2018.2837889>
- Chinmay, V., & Rishabh, G. (2015). A review paper on OSI Model-a seven layered architecture of OSI Model. *International Journal of Innovative Research in Technology IJIRT*, 1(12).
- Chiu, C. H., Singh, D. K., Wang, Q., Lee, K., & Park, S. J. (2017, June). Minimal coflow routing and scheduling in openflow-based cloud storage area networks. In *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)* (pp. 222-229). IEEE.
- Chomsiri, T., He, X., & Nanda, P. (2012, November). Limitation of listed-rule firewall and the design of tree-rule firewall. In *International Conference on Internet and Distributed Computing Systems* (pp. 275-287). Springer, Berlin, Heidelberg.
- Coleman, P. (2022). Validity and Reliability within Qualitative Research for. *International Journal of Caring Sciences*, 14(3), 2041–2045.
- Consolini, L., Locatelli, M., Minari, A., Nagy, A., & Vajk, I. (2019). Optimal Time-Complexity Speed Planning for Robot Manipulators. *IEEE Transactions on Robotics*, 35(3), 790–797. <https://doi.org/10.1109/TRO.2019.2899212>
- Cos, W. (2012). *CoS and QoS - Managing Bandwidth , Complexity , and Cost.*



- PKE Consulting. <http://www.pkeconsulting.com/pkecqos.pdf>.
- Council, S. P., & City, R. (2019). *Policies and Guidelines of the Storage Performance Council*. Storage Performance Council.
- Cui, Y., Dai, N., Lai, Z., Li, M., Li, Z., Hu, Y., ... Chen, Y. (2019). TailCutter: Wisely cutting tail latency in cloud CDNs under cost constraints. *IEEE/ACM Transactions on Networking*, 27(4), 1612–1628. <https://doi.org/10.1109/TNET.2019.2926142>
- Dahan, F. (2021). An Effective Multi-Agent Ant Colony Optimization Algorithm for QoS-Aware Cloud Service Composition. *IEEE Access*, 9, 17196–17207. <https://doi.org/10.1109/ACCESS.2021.3052907>
- Dahan, F., Binsaeedan, W., Altaf, M., Al-Asaly, M. S., & Hassan, M. M. (2021). An Efficient Hybrid Metaheuristic Algorithm for QoS-Aware Cloud Service Composition Problem. *IEEE Access*, 9, 95208–95217. <https://doi.org/10.1109/ACCESS.2021.3092288>
- Dahan, F., Hindi, K. El, Ghoneim, A., & Alsalman, H. (2021). An Enhanced Ant Colony Optimization Based Algorithm to Solve QoS-Aware Web Service Composition. *IEEE Access*, 9(3), 34098–34111. <https://doi.org/10.1109/ACCESS.2021.3061738>
- Dainotti, A., Pescapé, A., & Claffy, K. C. (2012). Issues and future directions in traffic classification. *IEEE network*, 26(1), 35-40.
- Danielsson, J., Seceleanu, T., Jagemar, M., Behnam, M., & Sjodin, M. (2019). Testing Performance-Isolation in Multi-Core Systems. *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, 1, 604–609. <https://doi.org/10.1109/COMPSAC.2019.00092>
- Dapeng, J. U., Chuanyi, L. I. U., & Dongsheng, W. (2010). Performance Comparison of IP-Networked Storage. *Tsinghua Science and Technology*, 14(1), 29-40.
- Datsika, E., Kartsakli, E., Vardakas, J. S., Antonopoulos, A., Kalfas, G., & Informatika, S. L. (2018). QoS-aware resource management for converged Fiber Wireless 5G Fronthaul networks. *2018 IEEE Global Communications Conference (GLOBECOM)*, 1–5.
- De Rango, F., & Fazio, P. (2022). A Stochastic Approach for Resource Prediction Error and Bandwidth Wastage Evaluation in Advanced Dynamic Reservation Strategies. *IEEE Transactions on Mobile Computing*, 1–1. <https://doi.org/10.1109/tmc.2022.3176046>
- De Sio, C., Azimi, S., & Sterpone, L. (2020). An Emulation Platform for Evaluating the Reliability of Deep Neural Networks. *33rd IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, DFT 2020*, 2020–2023. <https://doi.org/10.1109/DFT50435.2020.9250872>
- Dhabake, K. A. (2016). Study On-Comparison between IP SAN and FC SAN.

*International Journal of Computer Science Trends and Technology (IJCTST)*, 4(4), 157–160.

- Ding, W., Niu, Y., & Wu, H. A. O. (2018). QoS-Aware Full-Duplex Concurrent Scheduling for Millimeter Wave Wireless Backhaul Networks. *IEEE Access*, 6, 25313–25322. <https://doi.org/10.1109/ACCESS.2018.2828852>
- Dong, P., Xie, J., Tang, W., Zhong, H. U. A., & Vasilakos, A. V. (2019). Performance Evaluation of Multipath TCP Scheduling Algorithms. *IEEE Access*, 7, 29818–29825. <https://doi.org/10.1109/ACCESS.2019.2898110>
- El-Atawy, A., Samak, T., Al-Shaer, E., & Hong, L. (2007). Using online traffic statistical matching for optimizing packet filtering performance. *Proceedings - IEEE INFOCOM*, 866–874. <https://doi.org/10.1109/INFCOM.2007.106>
- Eramo, V. (2019). Optimizing the Cloud Resources , Bandwidth and Deployment Costs in Multi-Providers Network Function Virtualization Environment. *IEEE Access*, 7(Vm), 46898–46916. <https://doi.org/10.1109/ACCESS.2019.2908990>
- Ezdiani, S., & Al-Anbuky, A. (2015). Modelling the integrated QoS for wireless sensor networks with heterogeneous data traffic. *Open Journal of Internet Of Things (OJIOT)*, 1(1), 1-15.
- Fahad, A., Alharthi, K., Tari, Z., Almalawi, A., & Khalil, I. (2014). CluClas : Hybrid Clustering-Classification Approach for Accurate and Efficient Network Classification. *39th Annual IEEE Conference on Local Computer Networks*, 168–176. <https://doi.org/10.1109/LCN.2014.6925769>
- Fang, C. H., Shen, L. H., Huang, T. P., & Feng, K. Ten. (2021). Delay-Aware Admission Control and Beam Allocation for 5G Functional Split Enhanced Millimeter Wave Wireless Fronthaul Networks. *IEEE Transactions on Wireless Communications*, PP, 1. <https://doi.org/10.1109/TWC.2021.3112299>
- Fang, J., Rao, Y., Liu, P., & Zhao, X. (2021). Sequence Transfer-Based Particle Swarm Optimization Algorithm for Irregular Packing Problems. *IEEE Access*, 9, 131223–131235. <https://doi.org/10.1109/ACCESS.2021.3114331>
- Fang, X., Chen, J., Ye, F., Feng, D., & Li, J. (2015, August). Introduction of metadata-request queue with immediate response for I/O path optimizations on iSCSI-based storage subsystem. In *2015 IEEE International Conference on Networking, Architecture and Storage (NAS)* (pp. 100-105). IEEE.
- Fang, Z., Qiu, Q., Ding, Y., & Ding, L. (2018). A QoS Guarantee Based Hybrid Media Access Control Protocol of Aeronautical Ad Hoc Network. *IEEE Access*, 6, 5954–5961. <https://doi.org/10.1109/ACCESS.2017.2775342>
- Favraud, R., Chang, C. Y., & Nikaiein, N. (2018). Autonomous Self-Backhauled

- LTE Mesh Network with QoS Guarantee. *IEEE Access*, 6, 4083–4117. <https://doi.org/10.1109/ACCESS.2018.2794333>
- Ferrera, L. E., & Niguidula, J. (2017). An analysis of latency, jitter and packet loss in a network with telepresence system. *International Journal of Advanced Computational Engineering and Networking*, (6), 33–37.
- Ganesh, A., Sudarsan, A., Vasu, A. K., & Ramalingam, D. (2014). Improving Firewall performance by using a cache table. *International Journal of Advances in Engineering & Technology*, 7(5), 1.
- Gaonkar, P. E., Bojewar, S., & Das, J. A. (2013). A survey: data storage technologies. *Int. J. Eng. Sci. Innov. Technol*, 2, 547-554.
- Garg, S., & Dixit, A. (2021). Bin-Packing-Based Online Dynamic Bandwidth and Wavelength Allocation Algorithm in Super-PON. *IEEE Access*, 9, 139379–139392. <https://doi.org/10.1109/ACCESS.2021.3118461>
- Garroppo, R. G., Giordano, S., Lucetti, S., & Valori, E. (2005, June). The wireless hierarchical token bucket: a channel aware scheduler for 802.11 networks. In *Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks* (pp. 231-239). IEEE.
- Gauger, C. M., Kohn, M., Gunreben, S., Sass, D., & Perez, S. G. (2005, October). Modeling and performance evaluation of iSCSI storage area networks over TCP/IP-based MAN and WAN networks. In *2nd International Conference on Broadband Networks, 2005*. (pp. 850-858). IEEE.
- Gemieux, M., Li, M., Savaria, Y., David, J. P., & Zhu, G. (2018). A Hybrid Architecture With Low Latency Interfaces Enabling Dynamic Cache Management. *IEEE Access*, 6, 62826-62839.
- Ghazal, S., Ben Othman, J., & Claudé, J. P. (2012). Traffic management based on token bucket mechanism for WiMAX networks. *Cluster Computing*, 15(4), 391-400.
- Ghezzi, A., Milano, P., Agiatzidou, E., Johanses, F. T., Milano, P., & Milano, P. (2014). *Internet Interconnection Techno-Economics : A Proposal for Assured Quality Services and Business Models*. <https://doi.org/10.1109/HICSS.2014.95>
- Gode, N., Kashalkar, R., Kale, D., & Bhingarkar, S. (2014). Feature-based Comparison of iSCSI Target Implementations. *International Journal of Computer Applications*, 85(16), 13–16. <https://doi.org/10.5120/14923-3438>
- Gómez, C. (2020). Complexity and time. *Physical Review D*, 101(6), 1–8. <https://doi.org/10.1103/PhysRevD.101.065016>
- Greener, S. (2008). Business Research Methods.[e-book] Dr. In Sue Greener and Ventus Publishing ApS. Available through:< [http://www. bookbon.com](http://www.bookbon.com)>[Accessed 9 May 2019]. Retrieved from

[http://gent.uab.cat/diego\\_prior/sites/gent.uab.cat.diego\\_prior/files/02\\_e\\_01\\_introduction-to-research-methods.pdf](http://gent.uab.cat/diego_prior/sites/gent.uab.cat.diego_prior/files/02_e_01_introduction-to-research-methods.pdf)

- Gu, L., Zeng, D., Tao, S., Guo, S., Jin, H., Zomaya, A. Y., & Zhuang, W. (2019). Fairness-Aware Dynamic Rate Control and Flow Scheduling for Network Utility Maximization in Network Service Chain. *IEEE Journal on Selected Areas in Communications*, 37(5), 1059–1071. <https://doi.org/10.1109/JSAC.2019.2906746>
- Gu, Y. U., Cui, Q., Chen, Y. U., Ni, W. E. I., Tao, X., & Zhang, P. (2018). Effective Capacity Analysis in Ultra-Dense Wireless Networks With Random Interference. 6, 19499–19508. <https://doi.org/10.1109/ACCESS.2018.2820901>
- Guck, J. W., Van Bempten, A., & Kellerer, W. (2017). DetServ: Network models for real-time QoS provisioning in SDN-based industrial environments. *IEEE Transactions on Network and Service Management*, 14(4), 1003-1017.
- Guido-Sanz, F., Anderson, M., Talbert, S., Diaz, D. A., Welch, G., & Tanaka, A. (2022). Using Simulation to Test Validity and Reliability of I-BIDS: A New Handoff Tool. *Simulation and Gaming*, 53(4), 353–368. <https://doi.org/10.1177/10468781221098567>
- Gulati, A., Merchant, A., & Varman, P. J. (2007). pClock: an arrival curve based approach for QoS guarantees in shared storage systems. *ACM SIGMETRICS Performance Evaluation Review*, 35(1), 13-24.
- Gulati, A., Shanmuganathan, G., Zhang, X., & Varman, P. (2019). Demand based hierarchical QoS using storage resource pools. *Proceedings of the 2012 USENIX Annual Technical Conference, USENIX ATC 2012*, 1–13.
- Gulati, A., Merchant, A., & Varman, P. J. (2010). mClock: Handling Throughput Variability for Hypervisor IO Scheduling. In *9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 10)*.
- Gulati, A., Ahmad, I., & Waldspurger, C. A. (2009, February). PARDA: Proportional Allocation of Resources for Distributed Storage Access. In *FAST* (Vol. 9, pp. 85-98).
- Gulder, S., & Déziel, M. (2003, November). Quality of Service Mechanism for MANET using Linux. In *Proc. INSC Symposium, NATO C3 Agency*.
- Guo, H. B., & Kuo, G. S. (2005, May). A dynamic and adaptive bandwidth management scheme for QoS support in wireless multimedia networks. In *2005 IEEE 61st Vehicular Technology Conference* (Vol. 3, pp. 2081-2085). IEEE.
- Guo, I., Langrené, N., Loeper, G., & Ning, W. (2021). Robust utility maximization under model uncertainty via a penalization approach. *Mathematics and Financial Economics*, (2013), 1–33. <https://doi.org/10.1007/s11579-021-00301-5>

- Guo, L. (2019). Personalized QoS Prediction for Service Recommendation With a Service- Oriented Tensor Model. *IEEE Access*, 7, 55721–55731. <https://doi.org/10.1109/ACCESS.2019.2912505>
- Haghighi, A. L. I. A., & Heydari, S. S. (2018). Dynamic QoS-Aware Resource Assignment in Cloud-Based Content-Delivery Networks. *IEEE Access*, 6, 2298–2309. <https://doi.org/10.1109/ACCESS.2017.2782776>
- Hamed, H., & Al-Shaer, E. (2006, March). Dynamic rule-ordering optimization for high-speed firewall filtering. In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security* (pp. 332-342).
- Han, K. A. I., Li, S., Tang, S., Huang, H., & Zhao, S. (2018). Application-Driven End-to-End Slicing: When Wireless Network Virtualization Orchestrates With NFV-Based Mobile Edge Computing. *IEEE Access*, 6, 26567–26577. <https://doi.org/10.1109/ACCESS.2018.2834623>
- Han, W., Xue, J., & Yan, H. (2019). Detecting anomalous traffic in the controlled network based on cross entropy and support vector machine. *IET Information Security*, 13(2), 109-116. <https://doi.org/10.1049/iet-ifs.2018.5186>
- Hao, M., Li, H., Tong, M. H., Pakha, C., Suminto, R. O., Stuardo, C. A., ... & Gunawi, H. S. (2017, October). MittOS: Supporting millisecond tail tolerance with fast rejecting SLO-aware OS interface. In *Proceedings of the 26th Symposium on Operating Systems Principles* (pp. 168-183).
- Hashemian, R., Carlsson, N., Krishnamurthy, Di., & Arlitt, M. (2020). Contention aware web of things emulation testbed. *ICPE 2020 - Proceedings of the ACM/SPEC International Conference on Performance Engineering*, 246–256. <https://doi.org/10.1145/3358960.3379140>
- Hassan, M. M., Albakr, H., Al-Dossari, H., & Mohamed, A. (2017). Resource provisioning for cloud-assisted body area network in a smart home environment. *IEEE Access*, 5, 13213-13224.
- He, L., Member, S., Wang, J., & Member, S. (2017). Bandwidth Efficiency Maximization for Single-Cell Massive Spatial Modulation MIMO: An Adaptive Power Allocation Perspective. *IEEE Access*, 5, 1482–1495. <https://doi.org/10.1109/ACCESS.2017.2668420>
- He, X., Chomsiri, T., Nanda, P., & Tan, Z. (2013). Improving cloud network security using the Tree-Rule firewall. *Future Generation Computer Systems*. <https://doi.org/10.1016/j.future.2013.06.024>
- He, X., Chomsiri, T., Nanda, P., & Tan, Z. (2014). Improving cloud network security using the Tree-Rule firewall. *Future generation computer systems*, 30, 116-126. <https://doi.org/10.1016/j.future.2013.06.024>
- He, Y., Tang, L., & Ren, Y. (2019). Maximizing sleeping capacity based on QoS provision for information-centric Internet of Things. *IEEE Access*, 7,

111084-111094.

- Hehman, E., Calanchini, J., Flake, J. K., & Leitner, J. B. (2019). Establishing construct validity evidence for regional measures of explicit and implicit racial bias. *Journal of Experimental Psychology: General*, *148*(6), 1022–1040. <https://doi.org/10.1037/xge0000623>
- Hemke, M. S. V, Gawande, A. D., Gautum, P. L. K., & Email, W. S. (2013). ISCSI-The Future of the Storage Network. *IEEE Access*, *2*(4), 235–240. <https://doi.org/10.1109/access.2022.3180491>
- Hewage, C. T. E. R., Ahmad, A., Mallikarachchi, T., Barman, N., & Martini, M. G. (2022). Measuring, Modeling and Integrating Time-Varying Video Quality in End-to-End Multimedia Service Delivery: A Review and Open Challenges. *IEEE Access*, *10*, 60267–60293. <https://doi.org/10.1109/access.2022.3180491>
- Hirose, M., & Cappellaro, P. (2018). Time-optimal control with finite bandwidth. *Quantum Information Processing*, *17*(4), 1–8. <https://doi.org/10.1007/s11128-018-1845-6>
- Hou, R., Chang, Y., & Yang, L. (2017). Multi-constrained QoS routing based on PSO for named data networking. *IET Communications*, *11*(8), 1251–1255.
- Huang, L., Chen, H. S., & Hu, T. T. (2013). Survey on Resource Allocation Policy and Job Scheduling Algorithms of Cloud Computing1. *J. Softw.*, *8*(2), 480–487.
- Huang, S., Yuan, D., & Ephremides, A. (2019). Bandwidth Partition and Allocation for Efficient Spectrum Utilization in Cognitive Communications. *Journal of Communications and Networks*, *21*(4), 353–364. <https://doi.org/10.1109/JCN.2019.000031>
- Huang, Xiaoge, Cao, C., Li, Y., & Chen, Q. (2020). Opportunistic Resource Scheduling for LTE-Unlicensed With Hybrid Communications Modes. *IEEE Access*, *6*, 47857–47869. <https://doi.org/10.1109/ACCESS.2018.2867427>
- Huang, Xiaohong, Yuan, T., & Ma, M. (2018). Utility-Optimized Flow-Level Bandwidth Allocation in Hybrid SDNs. *IEEE Access*, *6*, 20279–20290. <https://doi.org/10.1109/ACCESS.2018.2820682>
- Huang, X., Yang, K., Wu, F., & Leng, S. (2017). Power control for full-duplex relay-enhanced cellular networks with QoS guarantees. *IEEE Access*, *5*, 4859–4869.
- Hwang, Z. H. (2014). *Benchmarking of IP-based Network Storage Systems* (Master's thesis). Aalto University.
- Inumula, K. M. (2015). Exploring causal relations in data mining by using directed acyclic graphs ( dag ). *International Journal of Advance*

*Computational Engineering and Networking (IJACEN)*.(5), 49–55.

- Iswadi, D., Adriman, R., & Munadi, R. (2019, August). Adaptive switching PCQ-HTB algorithms for bandwidth management in routerOS. In *2019 IEEE international conference on cybernetics and computational intelligence (CyberneticsCom)* (pp. 61-65). IEEE.
- Iswadi, D., Adriman, R., & Munadi, R. (2019a). Adaptive Switching PCQ-HTB Algorithms for Bandwidth Management in RouterOS. *Proceedings: CYBERNETICSCOM 2019 - 2019 IEEE International Conference on Cybernetics and Computational Intelligence: Towards a Smart and Human-Centered Cyber World*, 61–65. <https://doi.org/10.1109/CYBERNETICSCOM.2019.8875679>
- Jacob, E., & Jaswal, S. (2017, May). Optimized utilization of disks in storage area network by storage tiering. In *2017 International Conference on Computing, Communication and Automation (ICCCA)* (pp. 637-640). IEEE.
- Jaichandra, S., & Prasannakumar, K. R. (2015). A Case Study on Different SAN Technologies – FC SAN and IP SAN. *International Journal of Science, Engineering and Technology Research (IJSETR)*. 4(6), 2211–2214.
- Jaiman, V., Mokhtar, S. Ben, Quéma, V., Chen, L., Jaiman, V., Mokhtar, S. Ben, ... Tam-, E. R. H. (2018). Héron : Taming Tail Latencies in Key-Value Stores under Heterogeneous Workloads To cite this version : HAL Id : hal-01896686 Héron : Taming Tail Latencies in Key-Value Stores under Heterogeneous Workloads.
- Jaiman, V., Mokhtar, S. Ben, Quéma, V., Chen, L. Y., & Rivière, E. (2019). Héron: Taming tail latencies in key-value stores under heterogeneous workloads. *Proceedings of the IEEE Symposium on Reliable Distributed Systems, 2019-October*, 191–200. <https://doi.org/10.1109/SRDS.2018.00030>
- Jalodia, N., Taneja, M., & Davy, A. (2021). A Deep Neural Network-Based Multi-Label Classifier for SLA Violation Prediction in a Latency Sensitive NFV Application. *IEEE Open Journal of the Communications Society*, 2(November), 2469–2493. <https://doi.org/10.1109/OJCOMS.2021.3122844>
- Jamali, S., Alipasandi, N., & Alipasandi, B. (2014). An improvement over random early detection algorithm: a self-tuning approach. *Journal of Electrical and Computer Engineering Innovations (JECEI)*, 2(2), 57-61.
- Jamaluddin, M. H. (2019). Singly-Fed Rectangular Dielectric Resonator Antenna With a Wide Circular Polarization Bandwidth and Beamwidth for WiMAX / Satellite Applications. *IEEE Access*, 7, 66206-66214.
- James, A., & Shaikh, S. (2019). A New Architecture for Network Intrusion Detection and Prevention. *IEEE Access*, 7, 18558–18573. <https://doi.org/10.1109/ACCESS.2019.2895898>

- Ji, Y., & Member, S. (2018). Reconfigurable Optical OFDM Datacenter Datacenter Networks. *IEEE Photonics Journal*, 10(5), 1–16. <https://doi.org/10.1109/JPHOT.2018.2859275>
- Jia, G., Han, G., Zhang, D., Liu, L., & Shu, L. (2015). An Adaptive Framework for Improving Quality of Service in Industrial Systems. *IEEE Access*, 3, 2129–2139. <https://doi.org/10.1109/ACCESS.2015.2496959>
- Jiang, J. W. (2012). *Wide-Area Traffic Management for Cloud Services* (Doctoral dissertation). Princeton University.
- Jin, X. I., Xia, C., & Guan, N. A. N. (2020). Real-Time Scheduling of Massive Data in Time Sensitive Networks With a Limited Number of Schedule Entries. *IEEE Access*, 8, 6751–6767. <https://doi.org/10.1109/ACCESS.2020.2964690>
- Jose-Ignacio, C. V., Serrano-Martinez, D. J., & Monica, H. (2019). Management Emulation for Advanced Networks Interconnection in all America: 2019 topology. *2019 IEEE 39th Central America and Panama Convention, CONCAPAN 2019, 2019-Novem.* <https://doi.org/10.1109/CONCAPANXXXIX47272.2019.8976946>
- Jurkiewicz, P., Biernacka, E., Domzal, J., & Wojcik, R. (2021). Empirical Time Complexity of Generic Dijkstra Algorithm. *Proceedings of the IM 2021 - 2021 IFIP/IEEE International Symposium on Integrated Network Management, (Im)*, 594–598.
- Kailong, Z., Min, W., Hang, S., Ansheng, Y., Fortelle, A. D. La, & Kejian, M. (2017). QoS-CITS : A Simulator for Service-oriented Cooperative ITS of Intelligent Vehicles \*. *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, 751–756. <https://doi.org/10.1109/ICIS.2017.7960093>
- Kalav, D., & Gupta, S. (2012). Congestion control in communication network using RED, SFQ and REM algorithm. *International Refereed Journal of Engineering and Science*, 1(2), 41-45.
- Karlsson, M., Karamanolis, C., & Zhu, X. (2005). Triage: Performance Differentiation for Storage Systems Using Adaptive Control. *ACM Transactions on Storage*, 1(4), 457–480. <https://doi.org/10.1145/1111609.1111612>
- Keller A. (2006). *Manual TC Packet Filtering and netem - tcn.hypert.net*. Retrieved September 10, 2022, from <http://tcn.hypert.net/tcmanual.pdf>
- Khadir, K., Guermouche, N., Guittoum, A., & Monteil, T. (2022). A Genetic Algorithm-Based Approach for Fluctuating QoS Aware Selection of IoT Services. *IEEE Access*, 10, 17946–17965. <https://doi.org/10.1109/ACCESS.2022.3145853>
- Khakurel, S., & Musavian, L. (2018). QoS-Aware Utility-Based Resource



- Allocation in Mixed-Traffic Multi-User OFDM Systems. *IEEE Access*, 6, 21646–21657. <https://doi.org/10.1109/ACCESS.2018.2823004>
- Khalid, P. S., & Hashim, W. (2014). System Performance of Adaptive Bandwidth Traffic Shaping Mechanism for Residential Safety System. *Lecture Notes on Information Theory Vol*, 2(2) <https://doi.org/10.12720/lnit.2.2.141-145>
- Khan, R., Member, S., & Alam, M. M. (2022). Throughput and Channel Aware MAC Scheduling for SmartBAN Standard. *IEEE Access*, 7, 63133–63145. <https://doi.org/10.1109/ACCESS.2019.2916159>
- Kim, T., Kwon, T., Lee, J. U. N., & Song, J. (2021). F / Wvis : Hierarchical Visual Approach for Effective Optimization of Firewall Policy. *IEEE Access*, 9, 105989–106004. <https://doi.org/10.1109/ACCESS.2021.3100141>
- Kothari C.R. (2004). *Research Methodology Methods and Techniques*. 2nd revised Edition. New Age International Limited Publishers.
- Kotian, P. P., Shetty, V. K., & Begum, S. (2017). Study on Different Mechanism for Congestion Control in Real Time Traffic for MANETS. *International Research Journal of Engineering and Technology (IRJET)*, 4(11), 1627-1631.
- Kourtessis, P., Lim, W., Merayo, N., Yang, Y., & Senior, J. M. (2019). Efficient T-CONT-Agnostic Bandwidth and Wavelength Allocation for NG-PON2. *IEEE/OSA Journal of Optical Communications and Networking*, 11(7), 383–396. <https://doi.org/10.1364/JOCN.11.000383>
- Kozhedub, I., & Air, N. (2018). Mandatory Resource Access Control based on a Reachability Matrix in Storage Area Networks. *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, 539–543. <https://doi.org/10.1109/DESSERT.2018.8409191>
- Kulkarni, S. (2015). Implementation of iSCSI for Mobile Platforms and its Performance Optimization on Mobile Networks. *CUCT 2006 international Conference, Jeju Island, Korea (pp. 3-5)*. Springer
- Kumar, A., Kim, J., & Suh, S. C. (2015). Incorporating Multiple Cluster Models for Network Traffic Classification. *2015 IEEE 40th Conference on Local Computer Networks (LCN)*, 185–188. <https://doi.org/10.1109/LCN.2015.7366302>
- Lee, C. H., & Kim, Y. T. (2013). QoS-aware hierarchical token bucket (QHTB) queuing disciplines for QoS-guaranteed Diffserv provisioning with optimized bandwidth utilization and priority-based preemption. *International Conference on Information Networking*, 351–358. <https://doi.org/10.1109/ICOIN.2013.6496403>
- Lewis, P., & Thornhill, A. (2019). *Research Methods for Business Students*. 8th

Edition, Pearson, New York

- Li, F., Cao, J., Wang, X., & Sun, Y. (2017). A QoS guaranteed technique for cloud applications based on software defined networking. *IEEE access*, 5, 21229-21241.
- Li, G., Member, S., Wu, J. U. N., & Li, J. (2018). SLA-Aware Fine-Grained QoS Provisioning for Multi-Tenant Software-Defined Networks. *IEEE Access*, 6, 159–170. <https://doi.org/10.1109/ACCESS.2017.2761553>
- Li, N., Jiang, H., Feng, D., & Shi, Z. (2016, April). Pslo: Enforcing the xth percentile latency and throughput slos for consolidated vm storage. In *Proceedings of the Eleventh European Conference on Computer Systems* (pp. 1-14).
- Li, S., Wen, J., & Luo, F. (2018). Time-Aware QoS Prediction for Cloud Service Recommendation Based on Matrix Factorization. *IEEE Access*, 6, 77716–77724. <https://doi.org/10.1109/ACCESS.2018.2883939>
- Liang, J., Long, Y., Mei, Y., & Wang, T. (2019). A Distributed Intelligent Hungarian Algorithm for Workload Balance in Sensor-Cloud Systems Based on Urban Fog Computing. *IEEE Access*, 7, 77649–77658. <https://doi.org/10.1109/ACCESS.2019.2922322>
- Lichtblau, F., Streibelt, F., Krüger, T., Richter, P., & Feldmann, A. (2017, November). Detection, classification, and analysis of inter-domain traffic with spoofed source IP addresses. In *Proceedings of the 2017 Internet Measurement Conference* (pp. 86-99).
- Lim, H., & Choi, S. (2005). Design and implementation of iSCSI-based virtual storage system for mobile health care. *Proceedings of the 7th International Workshop on Enterprise Networking and Computing in Healthcare Industry, HEALTHCOM 2005*, 37–40. <https://doi.org/10.1109/HEALTH.2005.1500383>
- Lim, W., Kourtessis, P., Senior, J. M., Na, Y., Allawi, Y., Jeon, S. B., & Chung, H. (2017). Dynamic bandwidth allocation for OFDMA-PONs using hidden Markov model. *IEEE Access*, 5, 21016-21019.
- Lin, S., Che, N., Jiang, S., & Wei, M. (2019). Toward Delay-Based Utility Maximization: Modeling and Implementation in an SDWN Platform. *IEEE Access*, 7, 185086–185098. <https://doi.org/10.1109/ACCESS.2019.2960772>
- Lin, Z., & Masa, V. G. (2019). A New Partition-Based Random Search Method for Deterministic Optimization Problems. *2019 Winter Simulation Conference (WSC)*, 3504–3515.
- Liu, L., Lu, C., Xiao, F., Liu, R., & Xiong, N. N. (2021). A Practical, Integrated Multi-Criteria Decision-Making Scheme for Choosing Cloud Services in Cloud Systems. *IEEE Access*, 9, 88391–88404. <https://doi.org/10.1109/ACCESS.2021.3089991>

- Liu, X., Li, Z., Xu, P., & Li, J. (2021). Joint Optimization for Bandwidth Utilization and Delay Based on Particle Swarm Optimization. *IEEE Access*, 9, 92125–92133. <https://doi.org/10.1109/ACCESS.2021.3091693>
- Liu, Z., Sun, S., Zhu, H., Gao, J., & Li, J. (2017). BitCuts: A fast packet classification algorithm using bit-level cutting. *Computer Communications*, 109, 38-52.
- Liu, Zhigang. (2019). MR-CNN : A Multi-Scale Region-Based Convolutional Neural Network for Small Traffic Sign Recognition. *IEEE Access*, 7, 57120–57128. <https://doi.org/10.1109/ACCESS.2019.2913882>
- Liveoptics, F. C., For, P., Lasky, R., By, P., Lasky, R., Optics, L., & Technology, D. (2019). Performance overview. <https://www.liveoptics.com/>
- Lopez-martin, M., Member, S., & Carro, B. (2017). Network Traffic Classifier With Convolutional and Recurrent Neural Networks for Internet of Things. *IEEE Access*, 5, 18042–18050. <https://doi.org/10.1109/ACCESS.2017.2747560>
- Lu, Y., Du, D. H. C., & Ruwart, T. (2005). QoS provisioning framework for an OSD-based storage system. *Proceedings - Twenty -Second IEEE/Thirteenth NASA Goddard Conference on Mass Storage Systems and Technologies*, 28–35. <https://doi.org/10.1109/msst.2005.27>
- Lumb, C. R. (2003). Façade: Virtual Storage Devices with Performance Guarantees. In *2nd USENIX Conference on File and Storage Technologies (FAST 03)*.
- Lv, S., Yi, F., He, P., & Zeng, C. (2022). QoS Prediction of Web Services Based on a Two-Level Heterogeneous Graph Attention Network. *IEEE Access*, 10, 1871–1880. <https://doi.org/10.1109/ACCESS.2021.3138127>
- Mahajan, N., & Mahajan, S. (2015). *AN EFFICIENT TOKEN BUCKET ALGORITHM INCREASING WIRELESS*. *International Journal of Engineering Research and General Science*, V3(3),2091-2730
- Malviya, P. (2016). A Study Paper on Storage Area Network Problem-Solving Issues. *International Journal of Computer Science Trends and Technology (IJ CST)*, 4(4), 151–156.
- Mamman, M., & Hanapi, Z. M. (2017). An Adaptive Call Admission Control With Bandwidth Reservation for Downlink LTE Networks. *IEEE Access*, 5, 10986–10994. <https://doi.org/10.1109/ACCESS.2017.2713451>
- Marir, N., Wang, H., Li, B., & Jia, M. (2018). Distributed Abnormal Behavior Detection Approach Based on Deep Belief Network and Ensemble SVM Using Spark. *IEEE Access*, 6, 59657–59671. <https://doi.org/10.1109/ACCESS.2018.2875045>
- Martins, M. B. P., & Zucchi, W. L. (2015). FCoE and iSCSI Performance Analysis in Tape Virtualization Systems. *IEEE Latin America*

*Transactions*, 13(7), 2372-2378.

- Mary, N. A. B., & Jayapriya, K. (2014). An extensive survey on QoS in cloud computing. *International Journal of Computer Science and Information Technologies*, 5(1), 1-5.
- Mathews, A. B., & Glandevadhas, G. (2020). Improved Proportional Fair Algorithm for Transportation of 5G Signals in Internet of Medical Things. *International Journal of Innovative Technology and Exploring Engineering*, 9(2), 1810–1814. <https://doi.org/10.35940/ijitee.b7471.129219>
- Mathews, K., Kramer, C., & Gotzhein, R. (2018). Token bucket based traffic shaping and monitoring for WLAN-based control systems. *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC, 2017-Octob*, 1–7. <https://doi.org/10.1109/PIMRC.2017.8292201>
- Mebarkia, K., & Zsóka, Z. (2019). Service traffic engineering: Avoiding link overloads in service chains. *Journal of Communications and Networks*, 21(1), 69-80.
- Meth, K. Z., & Satran, J. (2003). Features of the iSCSI protocol. *IEEE Communications Magazine*, 41(8), 72–75. <https://doi.org/10.1109/MCOM.2003.1222720>
- Micha, E., & Shah, N. (2020). Proportionally fair clustering revisited. *Leibniz International Proceedings in Informatics, LIPIcs*, 168. <https://doi.org/10.4230/LIPIcs.ICALP.2020.85>
- Mikac, M., & Horvatić, M. (2019). an Approach for Teaching and Understanding Computer Networks Using Realistic Emulation Tool. *ICERI2019 Proceedings*, 1(November), 1209–1219. <https://doi.org/10.21125/iceri.2019.0371>
- Misra, S., Oommen, B. J., Yanamandra, S., & Obaidat, M. S. (2009). Random early detection for congestion avoidance in wired networks: a discretized pursuit learning-automata-like solution. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 40(1), 66-76.
- Mistry, S., Prajapati, J., Patel, M., & Saxena, M. S. S. (2020). NAS ( Network Attached Storage ). *Journal of Communications and Networks*, 65(6), 6571–6575.
- Murthy, S. K. D. K. N. N. (2015). A Survey of Optimizing the Performance of iSCSI. *International Journal of Science and Research (IJSR)*, 4(3), 1356–1359. Retrieved from <https://www.ijsr.net/archive/v4i3/SUB152302.pdf>
- Nahrstedt, K., Arefin, A., Rivas, R., Agarwal, P., Huang, Z., Wu, W., & Yang, Z. (2011). QoS and resource management in distributed interactive multimedia environments. *Multimedia Tools and Applications*, 51(1), 99-132. <https://doi.org/10.1007/s11042-010-0627-7>

- Nam, Y., Choi, Y., Yoo, B., Eom, H., & Son, Y. (2020, May). EdgeIso: Effective Performance Isolation for Edge Devices. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (pp. 295-305). IEEE.
- Nam, Y. J., Ryu, J., Park, C., & Ahn, J. S. (2004). A network bandwidth computation technique for IP storage with QoS guarantees. In *IFIP International Conference on Network and Parallel Computing* (pp. 473-480). Springer, Berlin, Heidelberg.
- Named, H., & Al-Shaer, E. (2006). Dynamic rule-ordering optimization for high-speed firewall filtering. *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security, ASIACCS '06, 2006*, 332–342. <https://doi.org/10.1145/1128817.1128867>
- Narale, S. A. (2015). Cloud computing techniques helps to meet QOS : *International Journal of Advanced Computational Engineering and Networking*,(12), 87–90.
- Nedunchezian, R., & Vijayakumar, V. (2016). Classifier with Temporal characteristics. *International Journal of Innovative Research in Science, Engineering and Technology*,6(2),19–25.
- Neto, A. J. R. and N. L. S. Da Fonseca. (2007). “Um estudo comparativo do desempenho dos protocolos iSCSI e Fibre Channel.” *IEEE Latin America Transactions*, 5(3), 151–157.
- Neto, A.J.R., & Da Fonseca, N.L.S. (2007). Um estudo comparativo do desempenho dos protocolos iSCSI e Fibre Channel [A study of the performance of the iSCSI and fiber channel protocols]. *IEEE Latin America Transactions*, 5(3), 151-157. <https://doi.org/10.1109/TLA.2007.4378498>
- Nleya, B., & Mutsvangwa, A. (2018). Enhanced congestion management for minimizing network performance degradation in OBS networks. *SAIEE Africa Research Journal*, 109(1), 48-57.
- Noertjahyana, A., Palit, H. N., Chandra, R., Andjarwirawan, J., Puspa, L., Chandra, D., ... Puspa, L. (2020). ScienceDirect ScienceDirect Comparative Analysis of NFS and iSCSI Protocol Performance on Comparative Analysis of NFS Cinder and iSCSI Protocol Performance on OpenStack Technology OpenStack Cinder Technology. *Procedia Computer Science*, 171(2019), 1498–1506. <https://doi.org/10.1016/j.procs.2020.04.160>
- Noorshams, Q., Kounev, S., & Reussner, R. (2012). Experimental evaluation of the performance-influencing factors of virtualized storage systems. In *Computer Performance Engineering* (pp. 63-79). Springer, Berlin, Heidelberg.
- Nosheen, S., & Khan, J. Y. (2021). Quality of Service-and Fairness-Aware Resource Allocation Techniques for IEEE802.11ac WLAN. *IEEE Access*,

9, 25579–25593. <https://doi.org/10.1109/ACCESS.2021.3051983>

- Nunome, A., Hirata, H., & Shibayama, K. (2014, August). A distributed storage system with dynamic tiering for iSCSI environment. In *2014 IIAI 3rd International Conference on Advanced Applied Informatics* (pp. 644-649). IEEE.
- Aduragbemi, O. (2018). *Traffic Shaping for Congestion Control*. Retrieved from <https://1library.net/document/ydxjjo1z-traffic-shaping-for-congestion-control.html>.
- Ojijo, M. O., & Falowo, O. E. (2020). a Survey on Slice admission Control Strategies and Optimization Schemes in 5G Network. *IEEE Access*, 8, 14977–14990. <https://doi.org/10.1109/aACCESS.2020.2967626>
- Osama, S.(2011).Storage area network implementation on an educational institute network computer networking and communication. *World Comput Sci Inform Tech J* 1(7):292–296
- Ou, Z., Hwang, Z. H., Chen, F., Wang, R., & Ylä-Jääski, A. (2015). Is cloud storage ready? A comprehensive study of IP-based storage systems. In *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)* (pp. 1-10). IEEE.
- Pan, Q., Huang, K., Tang, H., & You, W. (2018). A network slicing deployment method for guaranteeing service performance. *2018 IEEE 4th International Conference on Computer and Communications, ICC 2018*, 579–584. <https://doi.org/10.1109/CompComm.2018.8781076>
- Paricio, A., & Lopez-Carmona, M. A. (2021). Application of Traffic Weighted Multi-Map Optimization Strategies to Traffic Assignment. *IEEE Access*, 9, 28999–29019. <https://doi.org/10.1109/ACCESS.2021.3058508>
- Park, S., Kim, Y., Jeong, S., Hong, C., & Kang, T. (2015). A Case Study on Effective Technique of Distributed Data Storage for Big Data Processing in the Wireless Internet Environment. *Wireless Personal Communications*. <https://doi.org/10.1007/s11277-015-2794-3>
- Parks, L., & Peters, W. (2022). Natural Language Processing in Mixed-methods Text Analysis: A Workflow Approach. *International Journal of Social Research Methodology*, 00(00), 1–13. <https://doi.org/10.1080/13645579.2021.2018905>
- Paul, A. K., Tachibana, A., & Hasegawa, T. (2016). An Enhanced Available Bandwidth Estimation Technique for an End-to-End Network Path. *IEEE Transactions on Network and Service Management*, 13(4), 768–781. <https://doi.org/10.1109/TNSM.2016.2572212>
- Paulraj, J. P. I., & Kannigadevi, R. (2014). Efficient Resource Provisioning Using Virtualization Technology in Cloud Environment. *International Journal of Innovative Research in Science, Engineering and Technology*, 3(3), 2200–2205.

- Peng, Y., Liu, Q., & Varman, P. (2019, August). Latency Fairness Scheduling for Shared Storage Systems. In *2019 IEEE International Conference on Networking, Architecture and Storage (NAS)* (pp. 1-8). IEEE.
- Peng, Y., Liu, Q., & Varman, P. (2019). Scalable QoS for Distributed Storage Clusters using Dynamic Token Allocation. *IEEE Symposium on Mass Storage Systems and Technologies, 2019-May*, 14–27. <https://doi.org/10.1109/MSST.2019.00-19>
- Peng, Y., & Varman, P. (2018). BQueue: A coarse-grained bucket QoS scheduler. *Proceedings - 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID 2018*, 93–102. <https://doi.org/10.1109/CCGRID.2018.00024>
- Peng, Y., & Varman, P. (2020). PTrans: A Scalable Algorithm for Reservation Guarantees in Distributed Systems. *Annual ACM Symposium on Parallelism in Algorithms and Architectures*, 441–452. <https://doi.org/10.1145/3350755.3400273>
- Ppallan, J. M., Arunachalam, K., Gantha, S. S., Jaiswal, S., Song, S., & Nigam, A. (2021). A Method for Enabling Context-Awareness at Transport Layer for Improved Quality-of-Service Control. *IEEE Access*, 9, 123987–123998. <https://doi.org/10.1109/ACCESS.2021.3110266>
- Preethi, B. (2017). An Approach to Guarantee Quality of Service & Future Enhancement of Storage Area Network. *International Journal of Computer Science and Technology*, 8, 0976-8491.
- Puters, E. C. O. M. (2012). The History of Storage Systems. *Proceedings of the IEEE*, 100, 1433–1440. <https://doi.org/10.1109/JPROC.2012.2189787>
- Qian, Y., Li, X., Ihara, S., Zeng, L., Kaiser, J., Süß, T., & Brinkmann, A. (2017). A configurable rule based classful token bucket filter network request scheduler for the lustre file system. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2017*. <https://doi.org/10.1145/3126908.3126932>
- Rajan, M. S., Mesfin, A., & Sando, S. (2020). An Effective and Active Bandwidth Distribution in Networked Control Systems. *International Journal of Engineering and Advanced Technology*, 9(4), 2150–2155. <https://doi.org/10.35940/ijeat.d8983.049420>
- Ramadan, H. H., & Kashyap, D. (2017). Quality of service (QoS) in cloud computing. *International Journal of Computer Science and Information Technologies (IJCSIT)*, 8(3), 318-320.
- Ramaswamy, P. (2008). *Provisioning task based symmetric QoS in iSCSI SAN* (Doctoral dissertation, Wichita State University).
- Randrianantenaina, I., Dahrouj, H., Elsayy, H., & Alouini, M. S. (2017). Interference management in full-duplex cellular networks with partial spectrum overlap. *IEEE Access*, 5, 7567-7583.

- Raschellà, A., Bouhafs, F., Seyedehbrahimi, M., Mackay, M., & Shi, Q. (2017). Quality of Service Oriented Access Point Selection Framework for Large Wi-Fi Networks. *IEEE Transactions on Network and Service Management*, 14(2), 441–455. <https://doi.org/10.1109/TNSM.2017.2678021>
- Rashelbach, A., Rottenstreich, O., & Silberstein, M. (2020). A Computational Approach to Packet Classification. *SIGCOMM 2020 - Proceedings of the 2020 Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, 542–556. <https://doi.org/10.1145/3387514.3405886>
- Ravali, P. (2013). A comparative evaluation of OSI and TCP/IP models. *International Journal of Science and Research*, 4(7), 514-521.
- Ren, S. (2017). A Service Curve of Hierarchical Token Bucket Queue Discipline on Soft-Ware Defined Networks Based on Deterministic Network Calculus: An Analysis and Simulation. *Journal of Advances in Computer Networks*, 5(1), 8–12. <https://doi.org/10.18178/jacn.2017.5.1.232>
- Ren, S., Feng, Q., & Dou, W. (2017). An end-to-end qos routing on software defined network based on hierarchical token bucket queuing discipline. *ACM International Conference Proceeding Series, Part F1287*, 0–4. <https://doi.org/10.1145/3089871.3089883>
- Ren, Y., Li, T., Yu, D., Jin, S., & Robertazzi, T. (2014). Design, implementation, and evaluation of a NUMA-aware cache for iSCSI storage servers. *IEEE Transactions on Parallel and Distributed Systems*, 26(2), 413-422.
- Riabov, V. V. (2004). Storage Area Networks (SANs). *The Internet Encyclopedia*. Vol. 3, Wiley & Sons, pp. 329-339.
- Romli, M. A., Prayudi, Y., & Sugiantoro, B. (2019). Storage Area Network Architecture to support the Flexibility of Digital Evidence Storage. *International Journal of Computer Applications*, 975, 8887.
- Sadiq, A. L. I. S., Alkazemi, B., Mirjalili, S., Ahmed, N., Khan, S., & Ali, I. (2018). An Efficient IDS Using Hybrid Magnetic Swarm Optimization in WANETs. *IEEE Access*, 6, 29041–29053. <https://doi.org/10.1109/ACCESS.2018.2835166>
- Saha, A. K., Si, M., Ni, K., Datta, S., Ye, P. D., & Gupta, S. K. (2020). Ferroelectric thickness dependent domain interactions in FEFETs for memory and logic: A phase-field model based analysis. *Technical Digest - International Electron Devices Meeting, IEDM, 2020-Decem*, 4.3.1-4.3.4. <https://doi.org/10.1109/IEDM13553.2020.9372099>
- Salim, J. H. (2015). Linux Traffic Control Classifier-Action Subsystem Architecture. *Proceedings of Netdev 0.1*.



- Salim, J. H., & Bates, L. (2016). *The CLASHoFIRES: Who's Got Your Back? Proceedings of netdev 1.1*, Feb 10-12, 2016, Seville, Spain
- Salmani, V., & Shin, S. W. (2015, September). An empirical evaluation methodology for iscsi storage networking. In *2015 IEEE 14th International Symposium on Network Computing and Applications* (pp. 216-225). IEEE.
- Salomo, Y., Pratama, M., Choi, K. A. E. W. O. N., & Member, S. (2018). Bandwidth Aggregation Protocol and Throughput-Optimal Scheduler for Hybrid RF and Visible Light Communication Systems. *IEEE Access*, 6, 32173–32187. <https://doi.org/10.1109/ACCESS.2018.2844874>
- Samadi, P., Member, S., Fiorani, M., Shen, Y., & Member, S. (2017). Flexible Architecture and Autonomous Control Plane for Metro-Scale Geographically Distributed Data Centers. *Journal of Lightwave Technology*, 35(6), 1188–1196. <https://doi.org/10.1109/JLT.2017.2652480>
- Sanyoto, A. N., Perdana, D., & Bisono, G. (2019). Performance Evaluation of Round Robin and Proportional Fair Scheduling Algorithm on 5G Millimeter Wave Network for Node Density Scenarios. *International Journal of Simulation: Systems, Science & Technology*, 1–6. <https://doi.org/10.5013/ijssst.a.20.02.17>
- Sarmah, S., & Sarma, S. K. (2019, March). A novel approach to prioritized bandwidth management in 802.11 e WLAN. In *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)* (pp. 1-5). IEEE.
- Sarmah, S., & Sarma, S. K. (2019). A Novel Approach to Prioritized Bandwidth Management in 802.11e WLAN. *2019 IEEE 5th International Conference for Convergence in Technology, I2CT 2019*, 1–5. <https://doi.org/10.1109/I2CT45611.2019.9033871>
- Saunders, M., Lewis, P. and Thornhill, A. (2009). *Research Methods for Business Students fifth edition*. Pearson, New York.
- Saunders, M., Lewis, P., & Thornhill, A. (2007). *Research methods for Business Students 4th edition*. Pearson Education Limited, England.
- Sboui, L., Member, S., & Ghazzai, H. (2016). On Green Cognitive Radio Cellular Networks : Dynamic Spectrum and Operation Management. *IEEE Access*, 4, 4046–4057. <https://doi.org/10.1109/ACCESS.2016.2592178>
- Scazzariello, M., Ariemma, L., & Caiazzi, T. (2020). Kathará: A Lightweight Network Emulation System. *Proceedings of IEEE/IFIP Network Operations and Management Symposium 2020: Management in the Age of Softwarization and Artificial Intelligence, NOMS 2020*, 10–11. <https://doi.org/10.1109/NOMS47738.2020.9110351>
- Sekaran, U. (2003) *Research Methods for Business: A Skill-Building Approach*.

4th Edition, John Wiley & Sons, New York.

- Sharma, A. K., & Behera, A. K. (2017). Modified random early detection algorithm to enhance the performance of bursty network traffic. *Research Journal of Computer and Information Technology Sciences*. ISSN, 2320, 6527.
- Sheltami, T. R. (2019). A Survey on Autonomic Provisioning and Management of QoS in SDN Networks. *IEEE Access*, 7, 73384–73435. <https://doi.org/10.1109/ACCESS.2019.2919957>
- Shen, J., Xia, J., Zhang, X., & Jia, W. (2017). Sliding block-based hybrid feature subset selection in network traffic. *IEEE Access*, 5, 18179-18186.
- Shimano, S., Nunome, A., Hirata, H., & Shibayama, K. (2015, July). An information propagation scheme for an autonomous distributed storage system in iSCSI environment. In *2015 3rd International Conference on Applied Computing and Information Technology/2nd International Conference on Computational Science and Intelligence* (pp. 142-147). IEEE.
- Shirvani Moghaddam, S., & Moghaddam, K. S. (2022). A General Framework for Sorting Large Data Sets Using Independent Subarrays of Approximately Equal Length. *IEEE Access*, 10, 11584–11607. <https://doi.org/10.1109/ACCESS.2022.3145981>
- Shue, D., Freedman, M. J., & Shaikh, A. (2012). Performance Isolation and Fairness for {Multi-Tenant} Cloud Storage. In *10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)* (pp. 349-362).
- Simiscuka, A. A., Bezbradica, M., & Muntean, G. M. (2017, June). Performance analysis of the quality of service-aware networking scheme for smart internet of things gateways. In *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)* (pp. 1370-1374). IEEE.
- Siregar, B., Fadli, A., & Hizriadi, A. (2020). Controlling of Quality of Service in Campus Area Network Using OpenDaylight with Hierarchical Token Bucket Method. *7th International Conference on ICT for Smart Society: AIoT for Smart Society, ICISS 2020 - Proceeding*, 1–5. <https://doi.org/10.1109/ICISS50791.2020.9307599>
- Song, M. (2018). Minimizing Power Consumption in Video Servers by the Combined Use of Solid-State Disks and Multi-Speed Disks. *IEEE Access*, 6, 25737–25746. <https://doi.org/10.1109/ACCESS.2018.2832221>
- Sugeng, W., Istiyanto, J. E., Mustofa, K., & Ashari, A. (2015). The impact of QoS changes towards network performance. *Int. J. Comput. Networks Commun. Secur*, 3(2), 48-53.
- Sun, H., Yu, H., & Fan, G. (2020). QoS-Aware Task Placement With Fault-

- Tolerance in the Edge-Cloud. *IEEE Access*, 8, 77987–78003. <https://doi.org/10.1109/ACCESS.2020.2977089>
- Sun, J., & Cho, H. (2022). A Lightweight Optimal Scheduling Algorithm for Energy-Efficient and Real-Time Cloud Services. *IEEE Access*, 10, 5697–5714. <https://doi.org/10.1109/ACCESS.2022.3141086>
- Sun, Z., Pedretti, G., Mannocci, P., Ambrosi, E., Bricalli, A., & Ielmini, D. (2020). Time Complexity of In-Memory Solution of Linear Systems. *IEEE Transactions on Electron Devices*, 67(7), 2945–2951. <https://doi.org/10.1109/TED.2020.2992435>
- Suresh, L., Canini, M., Schmid, S., & Feldmann, A. (2015). C3: Cutting tail latency in cloud data stores via adaptive replica selection. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)* (pp. 513-527).
- Suresh, N., & Bai, B. M. (2016). Predictive Modelling of Tree Rule Firewall for the Efficient Packet Filtering. *International Journal of Computer Science and Information Security*, 14(10), 189.
- Surucu, L., & Maslakci, A. (2020). View of VALIDITY AND RELIABILITY IN QUANTITATIVE RESEARCH | Business & Management Studies: An International Journal. *Business & Management Studies: An International Journal*, pp. 2694–2726. Retrieved from <https://www.bmij.org/index.php/1/article/view/1540/1365>
- Thi, V., & Nha, T. (2021). Understanding Validity and Reliability From Qualitative and Quantitative Research Traditions. *Vnu Journal of Foreign Studies*, 37(3), 1–10. Retrieved from <https://doi.org/10.25073/2525-2445/vnufs.4672>
- Topalova, I., & Radoyska, P. (2018). Control of traffic congestion with weighted random early detection and neural network implementation. *ICAS 2018*, 16.
- Toyoda, M., Yamaguchi, S., & Oguchi, M. (2005). TCP congestion window controlling algorithms on iSCSI sequential read access. *Proceedings - International Workshop on Biomedical Data Engineering, BMDE2005, 2005*, 1271–1274. <https://doi.org/10.1109/ICDE.2005.292>
- Trabelsi, Z., & Zeidan, S. (2012). Multilevel early packet filtering technique based on traffic statistics and splay trees for firewall performance improvement. *IEEE International Conference on Communications*, 1074–1078. <https://doi.org/10.1109/ICC.2012.6364218>
- Truong-huu, T., Member, S., Gurusamy, M., & Member, S. (2017). Dynamic Flow Scheduling With Uncertain Flow Duration in Optical Data Centers. *IEEE Access*, 5, 11200–11214. <https://doi.org/10.1109/ACCESS.2017.2716345>
- Valenzuela, J. L., Monleon, A., San Esteban, I., Portoles, M., & Sallent, O.

- (2004, September). A hierarchical token bucket algorithm to enhance QoS in IEEE 802.11: proposal, implementation and evaluation. In *IEEE 60th Vehicular Technology Conference, 2004. VTC2004-Fall. 2004* (Vol. 4, pp. 2659-2662). IEEE.
- Vasu, A. K., & Ganesh, A. (2014). Improving Firewall Performance by Eliminating Redundancies In Access Control Lists. *Priya Ayyappan & Anirudhan Sudarsan International Journal of Computer Networks (IJCN)*, (6), 92.
- Vigneri, L., Paschos, G., & Mertikopoulos, P. (2019). Large-Scale Network Utility Maximization: Countering Exponential Growth with Exponentiated Gradients. *Proceedings - IEEE INFOCOM, 2019-April*, 1630–1638. <https://doi.org/10.1109/INFOCOM.2019.8737600>
- Vincenzi, M., Lopez-Aguilera, E., & Garcia-Villegas, E. (2021). Timely Admission Control for Network Slicing in 5G with Machine Learning. *IEEE Access*, 9, 127595–127610. <https://doi.org/10.1109/ACCESS.2021.3111143>
- Vishvanath, R., & Nasreen, A. Survey on Recent Technology of Storage Area Network and Network Attached Storage Protocols. *Paper, For International Journal Of Innovative Research In Electrical, Electronics, Instrumentation And Control Engineering*, 2(8).
- Vulimiri, A., Godfrey, P. B., Mittal, R., Sherry, J., Ratnasamy, S., & Shenker, S. (2013). Low latency via redundancy. *CoNEXT 2013 - Proceedings of the 2013 ACM International Conference on Emerging Networking Experiments and Technologies*, 283–294. <https://doi.org/10.1145/2535372.2535392>
- Wachs, M., Abd-El-Malek, M., Thereska, E., & Ganger, G. R. (2007, February). Argon: Performance Insulation for Shared Storage Servers. In *FAST* (Vol. 7, pp. 5-5).
- Wang, A., Venkataraman, S., Alspaugh, S., Katz, R., & Stoica, I. (2012, October). Cake: enabling high-level SLOs on shared storage systems. In *Proceedings of the Third ACM Symposium on Cloud Computing* (pp. 1-14).
- Wang, B., Sun, Y., & Cao, Q. I. (2018). Bandwidth Slicing for Socially-Aware D2D Caching in SDN-Enabled Networks. *IEEE Access*, 6, 50910–50926. <https://doi.org/10.1109/ACCESS.2018.2867542>
- Wang, C., Wu, Y., Wang, Z., & Xu, T. (2013, December). ISCSI-based data protection system for virtual machine. In *Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC)* (pp. 2085-2089). IEEE.
- Wang, J., Zhao, Z., Xu, Z., Zhang, H., Li, L., & Guo, Y. (2015). I-sieve: An inline high performance deduplication system used in cloud storage. *Tsinghua Science and Technology*, 20(1), 17-27.

- Wang, K., Member, S., Li, J., & Wu, J. U. N. (2018). QoS-Predicted Energy Efficient Routing for Information-Centric Smart Grid: A Network Calculus Approach. *IEEE Access*, 6, 52867–52876. <https://doi.org/10.1109/ACCESS.2018.2870929>
- Wang, P. A. N., Chen, X., Ye, F., & Sun, Z. (2019). A Survey of Techniques for Mobile Service Encrypted Traffic Classification Using Deep Learning. *IEEE Access*, 7, 54024–54033. <https://doi.org/10.1109/ACCESS.2019.2912896>
- Wang, P. A. N., & Ye, F. (2018). Datanet : Deep Learning Based Encrypted Network Traffic Classification in SDN Home Gateway. *IEEE Access*, 6, 55380–55391. <https://doi.org/10.1109/ACCESS.2018.2872430>
- Wang, P., Gilligan, R. E., Green, H., & Raubitschek, J. (2003, April). IP SAN- from iSCSI to IP-addressable Ethernet disks. In *20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies, 2003.(MSST 2003). Proceedings.* (pp. 189-193). IEEE.
- Wang, Pu, Yan, Z., Wang, N., & Zeng, K. (2022). Resource Allocation Optimization for Secure Multi-device Wirelessly Powered Backscatter Communication with Artificial Noise. *IEEE Transactions on Wireless Communications, PP*, 1. <https://doi.org/10.1109/TWC.2022.3162137>
- Wang, R., Kang, W., Liu, G., Ma, R., & Li, B. (2020). Admission Control and Power Allocation for NOMA-Based Satellite Multi-Beam Network. *IEEE Access*, 8, 33631–33643. <https://doi.org/10.1109/ACCESS.2020.2973395>
- Wang, Yitu, Wang, W., Cui, Y., Shin, K. G., & Zhang, Z. (2018). Distributed Packet Forwarding and Caching Based on Stochastic Network Utility Maximization. *IEEE/ACM Transactions on Networking*, 26(3), 1264–1277. <https://doi.org/10.1109/TNET.2018.2825460>
- Wang, Y., Xu, F., Chen, Z., Sun, Y., & Zhang, H. (2018). An application-level QoS control method based on local bandwidth scheduling. *Journal of Electrical and Computer Engineering*, 2018.
- Wang, Z. (2018). DyCache : Dynamic Multi-Grain Cache Management for Irregular Memory Accesses on GPU. *IEEE Access*, 6, 38881–38891. <https://doi.org/10.1109/ACCESS.2018.2818193>
- Winterton, J. (2008). Business Research Methods ALAN BRYMAN and EMMA BELL. Oxford: Oxford University Press, 2007. xxxii+ 786 pp.£ 34.99 (pbk). ISBN 9780199284986. *Management Learning*, 39(5), 628-632.
- Workshops, P. (2014). Can we identify NAT behavior by analyzing Traffic Flows?. In *2014 IEEE Security and Privacy Workshops* (pp. 132-139). IEEE.
- Wu, B., Wu, B., Yin, H., Liu, A., Liu, C., & Xing, F. (2017). Investigation and System Implementation of Flexible Bandwidth Switching for a Software-

- Defined Space Information Network. *IEEE Photonics Journal*, 9(3), 1–14. <https://doi.org/10.1109/JPHOT.2017.2705134>
- Wu, J. C., & Brandt, S. A. (2006, May). The design and implementation of AQUA: an adaptive quality of service aware object-based storage device. In *Proceedings of the 23rd IEEE/14th NASA Goddard Conference on Mass Storage Systems and Technologies* (pp. 209-218).
- Wu, Y., Wang, F., Hua, Y., Member, S., & Feng, D. (2017). I / O Stack Optimization for Efficient and Scalable Access in FCoE-Based SAN Storage. *IEEE Transactions on Parallel and Distributed Systems*, 28(9), 2514–2526. <https://doi.org/10.1109/TPDS.2017.2685139>
- Wu, Z., Yu, C., & Madhyastha, H. V. (2015). CosTLO: Cost-effective redundancy for lower latency variance on cloud storage services. *Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2015*, 543–557.
- Xiong, B., Wu, R., Zhao, J., & Wang, J. (2019). Efficient differentiated storage architecture for large-scale flow tables in software-defined wide-area networks. *IEEE Access*, 7, 141193–141208. <https://doi.org/10.1109/ACCESS.2019.2942208>
- Xu, W. (2018). A Novel Data Reuse Method to Reduce Demand on Memory Bandwidth and Power Consumption For True Motion Estimation. *IEEE Access*, 6, 10151–10159. <https://doi.org/10.1109/ACCESS.2018.2807406>
- Yahya-Imam, M. K., Sellapan, P., & Devi, V. (2014). An enhanced bandwidth Management scheme for improved quality of service in network communication system. *Int. J. Electron. Electr. Eng*, 2(2), 147-152.
- Yakti, I., & Salameh, W. A. (2012). ISCSI (internet small computer system interface) for your startup. *International Journal of Information and Communication Technology Research*, 20(3).
- Yan, C., Zhang, Y., Zhong, W., Zhang, C., & Xin, B. (2022). A truncated SVD-based ARIMA model for multiple QoS prediction in mobile edge computing. *Tsinghua Science and Technology*, 27(2), 315–324. <https://doi.org/10.26599/TST.2021.9010040>
- Yang, C. T., Liu, J. C., Ranjan, R., Shih, W. C., & Lin, C. H. (2013). On construction of heuristic QoS bandwidth management in clouds. *Concurrency and Computation: Practice and Experience*, 25(18), 2540-2560. <https://doi.org/10.1002/cpe>
- Yang, H., Li, B., Liu, G., & Ma, R. (2018). Physical-layer network coding based multi-user cooperative relay transmission with multi-antennas in cognitive wireless networks. *IEEE Access*, 6, 40189-40197.
- Yu, Y. A. O., Guo, L. E. I., Liu, Y. E., Zheng, J., & Zong, Y. U. E. (2018). An Efficient SDN-Based DDoS Attack Detection and Rapid Response Platform in Vehicular Networks. *IEEE Access*, 6, 44570–44579.

<https://doi.org/10.1109/ACCESS.2018.2854567>

- Zeng, Y. I., & Gu, H. (2019). *Deep-Full-Range: A Deep Learning Based Network Encrypted Traffic Classification and Intrusion Detection Framework*. 7. *IEEE Access*, 7, 45182-45190. <https://doi.org/10.1109/ACCESS.2019.2908225>
- Zhang, F., Deng, R., & Liang, H. (2018). An Optimal Real-Time Distributed Algorithm for Utility Maximization of Mobile Ad Hoc Cloud. *IEEE Communications Letters*, 22(4), 824-827. <https://doi.org/10.1109/LCOMM.2018.2804928>
- Zhang, S., Lei, W., Zhang, W. E. I., Guan, Y., & Li, H. A. O. (2019). Congestion Control and Packet Scheduling for Multipath Real Time Video Streaming. *IEEE Access*, 7, 59758-59770. <https://doi.org/10.1109/ACCESS.2019.2913902>
- Zhang, W., Wang, C., Xiao, F., Xiong, N. N., & Chang, J. (2019). Reliable Storage System with Priority Filter and Load Balance Collection Model for Large Scale Sensor Networks. *IEEE Access*, 7, 184078-184089. <https://doi.org/10.1109/ACCESS.2019.2960075>
- Zhang, Xian, & Peng, M. (2019). Testbed Design and Performance Emulation in Fog Radio Access Networks. *IEEE Network*, 33(3), 49-57. <https://doi.org/10.1109/MNET.2019.1800378>
- Zhang, Xiaolu, Li, D., Li, W. W., & Zhao, W. (2019). An Optimal Dynamic Admission Control Policy and Upper Bound Analysis in Wireless Sensor Networks. *IEEE Access*, 7, 53314-53329. <https://doi.org/10.1109/ACCESS.2019.2912396>
- Zhao, F., Ma, W., Zhou, M., & Zhang, C. (2018). A Graph-Based QoS-Aware Resource Management Scheme for OFDMA Femtocell Networks. *IEEE Access*, 6, 1870-1881. <https://doi.org/10.1109/ACCESS.2017.2780520>
- Zhao, L., Inoue, Y., & Yamamoto, H. (2004, July). Delay Reduction for Liner-Search Based Packet Filters. In *ITC-CSCC: International Technical Conference on Circuits Systems, Computers and Communications* (pp. 160-163).
- Zhao, L., Shimae, A., & Nagamochi, H. (2007, July). Linear-tree rule structure for firewall optimization. In *Communications, Internet, and Information Technology* (pp. 67-72).
- Zhao, Y. (2018). An Energy Efficient and QoS Aware Routing Algorithm Based on Data Classification for Industrial Wireless Sensor Networks. *IEEE Access*, 6, 46495-46504. <https://doi.org/10.1109/ACCESS.2018.2866165>
- Zheng, X., Jia, J., Guo, S., Chen, J., Sun, L., Xiong, Y., & Xu, W. (2021). Full Parameter Time Complexity (FPTC): A Method to Evaluate the Running Time of Machine Learning Classifiers for Land Use/Land Cover Classification. *IEEE Journal of Selected Topics in Applied Earth*

*Observations and Remote Sensing*, 14, 2222–2235.  
<https://doi.org/10.1109/JSTARS.2021.3050166>

Zhou, Z., Yan, Y., Berger, M., & Ruepp, S. (2018). Analysis and Modeling of Asynchronous Traffic Shaping in Time Sensitive Networks. *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, 1–4. <https://doi.org/10.1109/WFCS.2018.8402376>


Zhu, S., Sun, Z., Lu, Y., Zhang, L., Min, G., & Member, S. (2019). Centralized QoS Routing Using Network Calculus for SDN-Based Streaming Media Networks. *IEEE Access*, 7, 146566–146576.  
<https://doi.org/10.1109/ACCESS.2019.2943518>

Zhu, T., Tumanov, A., Kozuch, M. A., Harchol-Balter, M., & Ganger, G. R. (2014, November). Prioritymeister: Tail latency qos for shared networked storage. In *Proceedings of the ACM Symposium on Cloud Computing* (pp. 1-14).

Zuberek, W. M., & Strzeciwiłk, D. (2018). Modeling traffic shaping and traffic policing in packet-switched networks. *Journal of Computer Sciences and Applications*, 6(2), 75-81.




## Appendix A: Research Permit



**REPUBLIC OF KENYA**  
National Commission for Science, Technology and Innovation


**Ref No: 505007**



**NATIONAL COMMISSION FOR  
SCIENCE, TECHNOLOGY & INNOVATION**

**Date of Issue: 25/November/2020**

**RESEARCH LICENSE**




**This is to Certify that Mr. JOSEPH KITHINI of Meru University of Science and Technology, has been licensed to conduct research in Meru on the topic: INTEGRATED QOS MANAGEMENT TECHNIQUE FOR INTERNET PROTOCOL STORAGE AREA NETWORKS for the period ending : 25/November/2021.**

**License No: NACOSTI/P/20/7464**

**Applicant Identification Number**  
**505007**

**Director General**  
**NATIONAL COMMISSION FOR  
SCIENCE, TECHNOLOGY & INNOVATION**

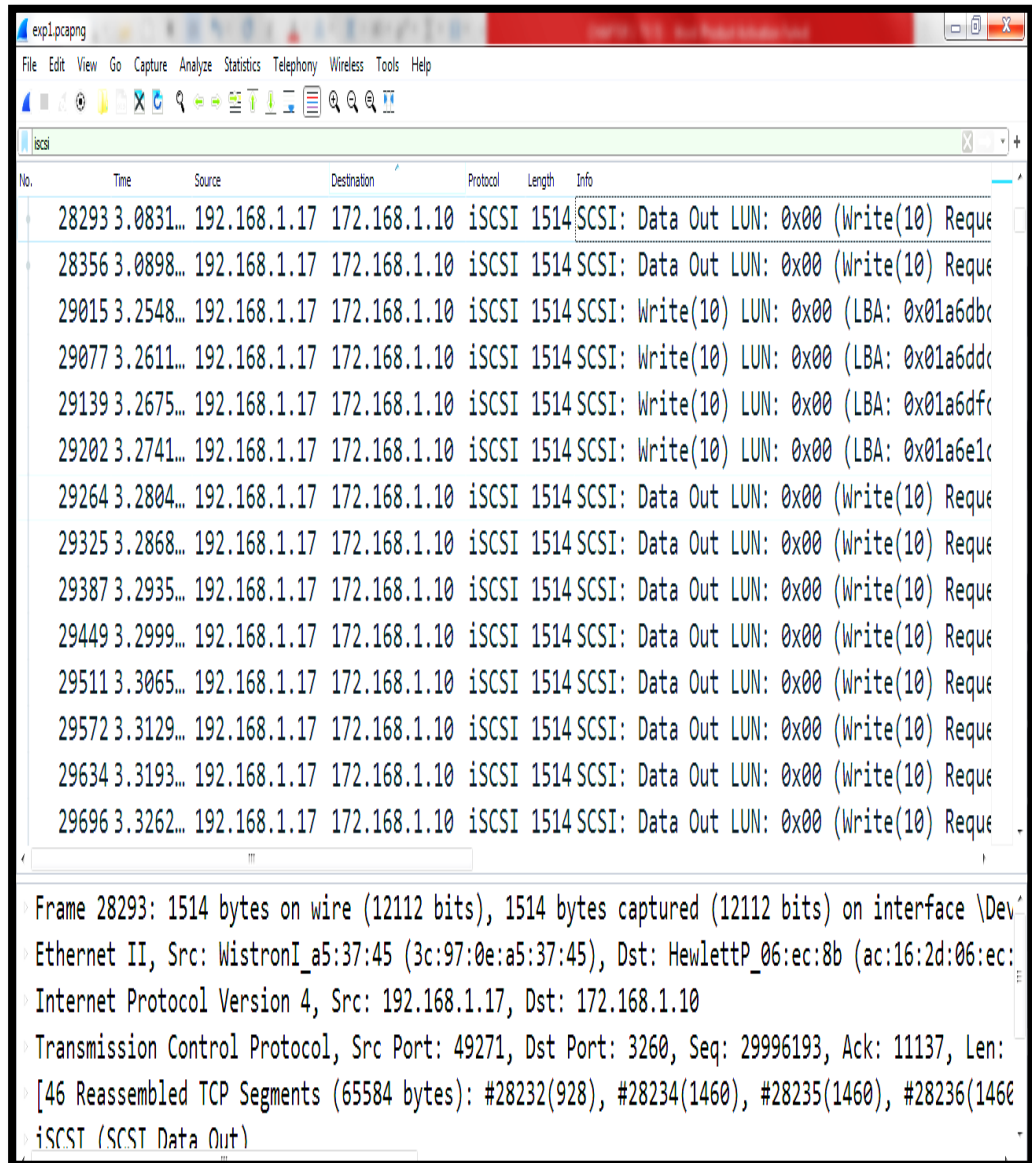
**Verification QR Code**



**NOTE: This is a computer generated License. To verify the authenticity of this document, Scan the QR Code using QR scanner application.**

263

## Appendix B: Wireshark Packets Capture



The screenshot shows a Wireshark capture of iSCSI traffic. The packet list pane displays several packets, with packet 28293 selected. The packet details pane shows the following information:

No.	Time	Source	Destination	Protocol	Length	Info
28293	3.0831...	192.168.1.17	172.168.1.10	iSCSI	1514	SCSI: Data Out LUN: 0x00 (Write(10) Reque
28356	3.0898...	192.168.1.17	172.168.1.10	iSCSI	1514	SCSI: Data Out LUN: 0x00 (Write(10) Reque
29015	3.2548...	192.168.1.17	172.168.1.10	iSCSI	1514	SCSI: Write(10) LUN: 0x00 (LBA: 0x01a6dbc
29077	3.2611...	192.168.1.17	172.168.1.10	iSCSI	1514	SCSI: Write(10) LUN: 0x00 (LBA: 0x01a6ddc
29139	3.2675...	192.168.1.17	172.168.1.10	iSCSI	1514	SCSI: Write(10) LUN: 0x00 (LBA: 0x01a6dfc
29202	3.2741...	192.168.1.17	172.168.1.10	iSCSI	1514	SCSI: Write(10) LUN: 0x00 (LBA: 0x01a6e1c
29264	3.2804...	192.168.1.17	172.168.1.10	iSCSI	1514	SCSI: Data Out LUN: 0x00 (Write(10) Reque
29325	3.2868...	192.168.1.17	172.168.1.10	iSCSI	1514	SCSI: Data Out LUN: 0x00 (Write(10) Reque
29387	3.2935...	192.168.1.17	172.168.1.10	iSCSI	1514	SCSI: Data Out LUN: 0x00 (Write(10) Reque
29449	3.2999...	192.168.1.17	172.168.1.10	iSCSI	1514	SCSI: Data Out LUN: 0x00 (Write(10) Reque
29511	3.3065...	192.168.1.17	172.168.1.10	iSCSI	1514	SCSI: Data Out LUN: 0x00 (Write(10) Reque
29572	3.3129...	192.168.1.17	172.168.1.10	iSCSI	1514	SCSI: Data Out LUN: 0x00 (Write(10) Reque
29634	3.3193...	192.168.1.17	172.168.1.10	iSCSI	1514	SCSI: Data Out LUN: 0x00 (Write(10) Reque
29696	3.3262...	192.168.1.17	172.168.1.10	iSCSI	1514	SCSI: Data Out LUN: 0x00 (Write(10) Reque

Frame 28293: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface \Dev Ethernet II, Src: WistronI\_a5:37:45 (3c:97:0e:a5:37:45), Dst: HewlettP\_06:ec:8b (ac:16:2d:06:ec: Internet Protocol Version 4, Src: 192.168.1.17, Dst: 172.168.1.10  
Transmission Control Protocol, Src Port: 49271, Dst Port: 3260, Seq: 29996193, Ack: 11137, Len: [46 Reassembled TCP Segments (65584 bytes): #28232(928), #28234(1460), #28235(1460), #28236(1460 iSCSI (SCSI Data Out)

## Appendix C: Parkdale Output Screen

Parkdale [3.02] - HGST HTS - 545050A7E380 - GG2ZBF40 97.6 GB Local Disk (C:); 102.0 MByte/sec read, 61.8 MByte/sec write

**97.6 GB Local Disk (C:)**  
HGST HTS - 545050A7E380 - GG2ZBF40

**358.8 GB Local Disk (D:)**  
HGST HTS - 545050A7E380 - GG2ZBF40

**DVD RW Drive (E:)**  
PLDS - DVD-RW DS8A8SH - KL31

**Removable Disk (H:)**  
Multiple - Card Reader - 1.00

File Size: 30 MByte

Block Size: 64 kByte [Windows Default]

Seq. Write Speed: **61.8 MByte/sec**

Random QD32: **2.1 MByte/sec, 549.7 IOPS, 1.82 ms**

Seq. Read Speed: **102.0 MByte/sec**

Random QD32: **615.1 kByte/sec, 153.8 IOPS, 6.50 ms**

[Copy test results](#) [Compare and Submit your test results](#)

[www.the-sz.com](http://www.the-sz.com)

## Appendix D: First Publication

*I.J. Information Technology and Computer Science*, 2021, 5, 51-63

Published Online October 2021 in MECS (<http://www.mecspress.org/>)

DOI: 10.5815/ijitcs.2021.05.05



# An Enhanced List Based Packet Classifier for Performance Isolation in Internet Protocol Storage Area Networks

**Joseph Kithinji**

Meru University of Science and Technology, Meru, Kenya

E-mail: [joskithinji2014@gmail.com](mailto:joskithinji2014@gmail.com)

**Makau S. Mutua and Gitonga D. Mwathi**

Department of Computer Science, Meru University of Science and Technology, Meru, Kenya

Department of Computer Science, Chuka University, Chuka Kenya

E-mail: {[smutua@must.ac.ke](mailto:smutua@must.ac.ke), [dgmwathi@chuka.ac.ke](mailto:dgmwathi@chuka.ac.ke)}

## **An Enhanced List Based Packet Classifier for Performance Isolation in Internet Protocol Storage Area Networks.**

**Abstract:** Consolidation of storage into IP SANs (Internet protocol storage area network) has led to a combination of multiple workloads of varying demands and importance. To ensure that users get their Service level objective (SLO) a technique for isolating workloads is required. Solutions that exist include cache partitioning and throttling of workloads. However, all these techniques require workloads to be classified in order to be isolated. Previous works on performance isolation overlooked the classification process as a source of overhead in implementing performance isolation. However, it's known that linear search based classifiers search linearly for rules that match packets in order to classify flows which results in delays among other problems especially when rules are many. This paper looks at the various limitation of list based classifiers. In addition, the paper proposes a technique that includes rule sorting, rule partitioning and building a tree rule firewall to reduce the cost of matching packets to rules during classification. Experiments were used to evaluate the proposed solution against the existing solutions and proved that the linear search based classification process could result in performance degradation if not optimized. The results of the experiments showed that the proposed solution when implemented would considerably reduce the time required for matching packets to their classes during classification as evident in the throughput and latency experienced.

**Index Terms:** Performance Isolation, Storage Area Network, Throttling, Optimization, Metrics

### **Methods:**

This study embarked on optimizing the process of packet matching during classification process for linear search based classifier. The methods of sorting the rule list, partitioning the rule list, jump search and building a linear tree rule structure to optimize the classification process.

To begin with, the system to be emulated was modelled for comparison purposes. To obtain the standard SLO requirements for each class of users, we used Table 2 and equation (3) and (4) to derive the SLO for classes of users based on the IOPs, block size and queue depth. The values for the SLO are throughput in Kb/s followed by IOPS and then response time. For a block size of 4kb the SLO for task, knowledge and power users is as follows; task users(20kb/s,5IOPS,6.4ms), Knowledge users(60kb/s,15IOPS,1.6-3.2ms) and power users(100kb/s,25IOPS,1.3ms). The same case applies for 64kb and 1 Mb block sizes.

### **Results**

An Enhanced List Based Packet Classifier for Performance Isolation in Internet Protocol Storage Area Networks was found to have an improvement by 20% in terms of operational cost compared to conventional list based packet classifier. In addition it was found to have a higher accuracy by 33% compared to conventional list based packet classifiers.

### **Conclusion and Future work**

The proposed solution has been tested and compared with traditional implementation of a linear search based classifier and established that the proposed solution gives better performance in terms of throughput and response time when used to classify traffic for performance isolation. The proposed solution was tested on an IPSAN and was found to be more suitable than the traditional implementation of linear search. In future we would also like to explore techniques of using performance isolation in providing availability and reliability guarantees. In addition, we would like to implement our proposed solution in non-storage systems where performance isolation is required

## Appendix D: Second Publication

# HPDDRR: Optimized Scheduler Shaper for Bandwidth Management and Traffic Shaping in Internet Protocol Storage Area Networks



IJCA Social Web  
Research (LEARN MORE)



International Journal of Computer Applications

Foundation of Computer Science (FCS), NY, USA

Volume 183 - Number 36

Year of Publication: 2021

Authors: *Kithinji Joseph, Makau S. Mutua, Gitonga D. Mwathi*

10.5120/ijca2021921746



### Citation

Kithinji Joseph, Makau S Mutua and Gitonga D Mwathi. HPDDRR: Optimized Scheduler Shaper for Bandwidth Management and Traffic Shaping in Internet Protocol Storage Area Networks. *International Journal of Computer Applications* 183(36):20-32, November 2021.



## **HPDDRR: Optimized Scheduler Shaper for Bandwidth Management and Traffic Shaping in Internet Protocol Storage Area Networks.**

### **ABSTRACT**

Providing QOS (quality of service) is a vital problem in storage area networks. In this paper a technique known as HPDDRR(hierarchical priority based dynamic deficit round robin) which is scheduler shaper that uses hit ration for flow prioritization and a dynamic quantum calculated based on the priority for scheduling is presented. Based on the applications used, packets may vary in sizes and belonging to different priority classes. To ensure that big low priority packets don't delay small high priority packets this study uses hierarchical priority queues instead of FIFO (first in first out) queues for scheduling. This allows for performance isolation as well as resource sharing. The evaluation results proof that HPDDRR is able to optimize bandwidth utilization as well as latency for competing traffic flows under Service level objectives constraints.

### **Keywords**

Dynamic Bandwidth management, Burst Handling, ISCSI, IP SAN, Quantum, Policing.

### **Methodology**

In achieving bandwidth management and traffic shaping the study adopted an experimental research design. Experiment is a research instrument that involves finding causal relationships between variables through the effect of manipulating one variable on another. It is suitable for phenomenon with known variables or initial hypothesis that aimed at testing or manipulating a theory .It is also used to test and answer „how“ and „why“ research questions and lies in the deductive approach and positivism philosophy domain. Experiments were set up to evaluate the proposed system on bandwidth allocation, bandwidth borrowing and burst handling. The proposed optimization of bandwidth management and traffic shaping was evaluated using the throughput and latency QOS metrics.

**Results:** HPDDRR was found to be able to allocate bandwidth based on the  $p_i$  values. The larger the  $p_i$  value the larger the allocation. HPDDRR was also found to adopt quickly to network changes with a convergence time of 10 seconds.

### **Conclusion and Future Work**

Evaluation done on HPDDRR shows that it is able to provide proportional allocation of bandwidth to classes of users based on priority and adopt the utilization experienced by traffic classes of users based on network conditions .HPDDRR has also been proven through experiments that it is able to absorb bursts from classes of user's flows. A hierarchical shaper can support more precise scheduling for the high rate traffic, this can significantly reduce cell shaping and jitter relative to existing approaches. With the hierarchical structure the sorting granularity for connection is reduced due to grouping. This reduces the implementation overhead and interference between competing connections. As future work the research would like to test the performance of HPDDRR on non-storage systems.