Implementation and Comparison of Packet Classification Modules: Hierarchical Trie, Set-Pruning Trie, Grid of Tries

A thesis

Submitted by

Tapan Kumar Swain

Roll no. 710EE1137

In the partial fulfilment of the requirements

for the award of the degree

Bachelor of Technology

and

Master of Technology

(DUAL DEGREE)

Under the Supervision of

Prof. P.K.Sahu

Department of Electrical Engineering

National Institute of Technology

Rourkela

Department of Electrical Engineering

National Institute of Technology

Rourkela

CERTIFICATE

This is to certify that the thesis entitled "**Implementation and Comparison of Packet Classification Modules: Hierarchical Trie, Set-Pruning Trie, Grid of Tries**" being submitted by **Tapan Kumar Swain (710EE1137)**, for the award of the degree of **Bachelor of Technology and Master of Technology (Dual Degree)** in Electrical Engineering, is a bonafide research work carried out by him in the Department of Electrical Engineering, National Institute of Technology, Rourkela under my supervision and guidance

The research reports and the results embodied in this thesis have not been submitted in parts or full to any other University or Institute for award of any other degree.

Prof. Prasanna Kumar Sahu
Department of Electrical Engineering
NIT Rourkela, Odisha

Date:
Place: Rourkela

ACKNOWLEDGEMENT

I express my deepest gratitude and sincere thanks to my supervisor **Prof. Prasanna Kumar Sahu** for his constant motivation and support during the course of my research work. Working with him has opened up a new horizon of state of art knowledge. His continuous monitoring, valuable guidance and input, have been always the source of inspiration and courage which are the driving forces to complete my work. My thanks and deep gratitude to **Mr.Guido Maier**, who have equally given me valuable guidance, advice, inspired me and patiently helped me in my work. I am overwhelmed with his immeasurable valuable input and help received during my research work. I wish to extend my sincere thanks to **Prof. A. K. Panda,** Head of our Department, for approving my project work with great interest.

I would like to earnestly extend my deepest gratitude to **Mr Koustabh Dolui**, for constantly motivating me and inspiring me with his valuable suggestions. My deepest love, appreciation and indebtedness go to my parents for their wholehearted support, encouragement and time sharing. Last but not the least I would like to thanks the GOD for their blessing to help me raised my academic level to this stage.

Tapan Kumar Swain

710EE1137

Abstract

In a network, the Routers provide best possible services by handling or processing each incoming packet independently (i.e. in the same manner). The process of categorizing the packets into 'flows' in an internet router is called "packet classification". All packets which belong to the same flow obey a predefined rule as mentioned in the forwarding table and are processed in the same manner by the router. For ex, all the packets which have same source and destination IP addresses are categorized to form a flow. Packet classification is required for non-best effort services that requires the routers to have the capability to identify and classify traffic in different flows for suitable processing.

In general, packet classification on multiple fields is a difficult problem, so we have implemented basic classification algorithms in Python programming language by taking five fields as per which the packets will be classified. The five fields are source IP address, destination IP address, input and output port number and the transmission protocol. Then these algorithm files were executed in NOX (an open source software providing sophisticated network functionality). Then the performance of the three algorithms i.e. the time taken by the algorithms to classify each packet is calculated and compared. This is repeated for different number of packets varying in size and compared and it was found that the grid of tries has the best performance among the three algorithms.

But, none of the three trie algorithms simultaneously avoids empty internal nodes and back tracking. This thesis proposes a new efficient packet classification algorithm using the set pruning trie approach. In the proposed algorithm, a hierarchical binary search tree is built for the pruned set of rules, which does not have empty internal nodes. Therefore, backtracking and empty internal nodes both are avoided by implementing the proposed algorithm. Simulation results prove that the proposed algorithm provides an improvement in search performance without increasing the memory requirement when compared with other existing trie based algorithms.

Contents

List of Figures

List of Abbreviations

ADSL	Asymmetric Digital Subscriber Line
VDSL	Very high speed Digital Subscriber Line
D2H	Direct To Home
DWDM	Dense Wavelength Division Multiplexing
OXC	Optical Cross Connect
MPLS	Multiple Protocol Layer Switching
SDN	Software Defined Network
TCP	Transmission Control Protocol
UDP	User Defined Protocol
IP	Internet Protocol
ISP	Internet Service Provider
DDR3	Double Data Rate type 3
SRAM	Static Random Access Memory
DRAM	Dynamic Random Access Memory
ASIC	Application Specific Integrated Circuit
TCAM	Ternary Content Addressable Memory
POP	Point of Presence
WCDMA	Wideband Code Division Multiple Access
HSDPA	High Speed Data Packet Access
LTE	Long Term Evolution
BGP	Border Gateway Protocol
OSPF	Open Shortest Path Forward
RIP	Routing Internet Protocol
SDH	Synchronous Digital Hierarchy
OC	Optical Channel
OTN	Optical Transport Network

List of Tables

Chapter-1

1.1 Introduction

The Internet is widely considered as the most reachable platform for the current and the next generation of information infrastructure. The virtually infinite bandwidth of optical fiber communication system has tremendously enhanced the data transmission speed during the past decade. The availability of unlimited bandwidth has prompted intensive multimedia services such as online gaming, music, distance learning and video download, meeting and conferences via videoconferencing. Today's technologies through which we access broadband internet such as very high bit rate digital subscriber lines (VDSLs) and Direct to home satellite television (D2H), are providing affordable broadband solutions for connecting to the Internet from home. Also, with Terabit Ethernet access over optical fiber to the enterprise on its way, internet access speeds are going to increase tremendously. It is clear that the implementation of these technologies to access broadband will definitely result in a high demand for higher Internet bandwidth. To meet the demands of the Internet traffic growth, researchers are continuously exploring faster transmission and switching technologies.

The invention of optical transmission technologies, such as dense wave division multiplexing (DWDM), optical cross connects, optical add drop multiplexers, etc, have contributed towards lowering the costs of digital transmission. For example, the transatlantic cables connecting North America and Western Europe carry 300 channels, each with a link capacity of 11.6 Gbps which are wavelength-division multiplexed on to a single fiber and is transmitted 7000 km across the Atlantic. Recently in Alcatel-Lucent a 1296 × 1296 optical cross-connect (OXC) switching system which use micro-electro-mechanical systems (MEMS) with a total switching speed of 2.07 petabits⁄s have been successfully tested.

To meet such stringent standards set by the growing demand for high speed Internet access, the routers have to efficiently handle the packets generated from various IP addresses, i.e route the packets to their destination within very limited time(generally in nano seconds). So we have implemented efficient algorithms in NOX (software providing sophisticated network functionality) to classify and route the packets to their destination. The three trie based packet

classification algorithms which are implanted in this project are hierarchical trie, set pruning trie and grid of tries.

1.2 Literature survey

Traditionally demultiplexing was used by the packet processing systems to handle the incoming packets. When a packet arrived, the header field was used by the Protocol Software to decide how payload contained in the packet will be processed. For example, the Next Header field is used by the IPv6 system for selecting the correct transport layer protocol module. Modern, high-speed digital networks take a completely different view of packet processing. Instead of demultiplexing, they employ a technique known as classification. In demultiplexing, a packet passes through a protocol stack one layer at a time, whereas classification allowed processing across layers. Owing to this property of the classification, it is not only used by companies such as Cisco and Juniper, but also has been used in Linux and in network processors by companies such as Intel.

Packet classification is relevant to the three key technologies in networking. First, Ethernet switches use classification instead of demultiplexing for forwarding packets. Second, a router which sends incoming packets over Multiple Protocol Layer Switching (MPLS) flows uses packet classification to select the appropriate flow. Third, classification paved the way for Software-Defined Networking (SDN) and the Open Flow protocol.

For understanding the need for classification, consider a network having protocol software arranged in a traditional layered stack. Packet processing depends on demultiplexing at every layer of the protocol stack. When a frame arrives, the protocol software reads the field Type to decode the contents of the payload frame. If the frame carries an IP datagram, the payload is dispatched to the IP protocol module for processing. IP protocol module uses the destination address of the packet to select the next-hop address. If the datagram is in transfer (i.e., passing through the router on its way to its destination), IP protocol modules sends the datagram back to one of its output ports. A datagram reaches TCP layer only if the datagram is destined for the router itself. TCP then uses the port numbers in the TCP segment for demultiplexing the incoming datagram between multiple application programs.

For understanding the inability of traditional layering to solve all problems, we consider MPLS processing. Consider a router at the boundary of a traditional internet and an MPLS core. Such a router should accept the packets which arrive from the traditional internet and select an MPLS path to send the packet. In many cases, network managers use transport layer protocol containing input and output port numbers when choosing a path. For example, suppose a packet manager wants to transport all packets down a specific MPLS path. All the packets which will use TCP port 80, meaning that the selection must examine the TCP port number. But in a traditional demultiplexing scheme, a datagram does not reach the transport layer except the datagram is intended for the local network system. Therefore, protocol software must be restructured to handle MPLS path selection as path selection often consists of transport layer information and a traditional stack will not send transit datagrams to the transport layer.

With the development of new applications, Internet service providers (ISPs) need routers to adapt to different quality of service (QoS) requirements for diverse applications. QoS necessitates the router be able to classify the incoming packets according to a set of predefined rules, which is the purpose of packet classification. The set of predefined rules is called a classifier and it is application-specific. Packet classification has been known to be a challenging problem. Without taking advantage of special properties of classifiers, any solution would be either exceedingly storage-consuming or time-consuming in the worst case. The classification (search/query) process is performed field by field. With proper implementation, each classification requires a few memory access on an average, thus, very high-speed classification can be achieved. Even though node sharing makes the updating process such as insertion and deletion less straightforward, the complexity of the updating the classifier remains low because each operation affects a small part of the data structure.

The major functional blocks of a packet forwarding engine (for ex, a router) which require processing at wire speed are the IP address lookup and the packet classification. IP address lookup decides an output port for an incoming packet, using the destination IP address of an incoming packet. Packet classification classifies the incoming packet into a particular flow based on predefined rules so that the packet can be accorded with the service defined for the class to which it belongs. For Internet routers to provide diverse service qualities for each packet flow, packet classification is a critical function. The time allotted to classify each packet in a flow is in nanoseconds since the packets arrive at speeds of gigabits or terabits per second. Therefore, the packet handling speed is of prime importance.

The use of application-specific integrated circuits (ASICs) with off-chip Ternary content addressable memories (TCAMs) has been the best option in providing packet forwarding at wire speeds. The demerits of using TCAMs is that it consume 150 times more power per bit than SRAMs. TCAMs cost around 30 times more per bit of storage than SRAMs (DDR). Another problem is the rule repetition related to the port number fields. The port number fields can be a fixed number or a range bounded by low and high boundaries. If a rule contains field specified by a range, the range has to be changed into multiple prefixes, and the rules with each changed prefixes are then stored into a TCAM entry. A port range can be converted into a maximum of 30 prefixes and if two port fields are given by ranges, a total of 900 TCAM entries are needed. Therefore, efficient packet classification algorithms using conventional memories such as SRAMs are necessary.

The speed at which the packet classification occurs is calculated by using the number of memory accesses in an algorithm since memory lookup is the slowest operation in a search process. Hierarchical trie based approaches are very efficient in enabling high-speed lookup performance as the search space is reduced by a substantial amount as compared to a one-dimensional search. The existing hierarchical trie based methodologies has two main drawbacks: back-tracking and empty nodes in the interior of the destination trie. In order to eradicate back-tracking, the hierarchical set-pruning trie employs a set-pruning of the rules, in which rules included in the ancestor nodes are copied into all the descendant nodes. In the grid-of-tries structure, it has to compute switch pointers in the destination trie in advance.

V. Srinivasan and G. Varghese [4] proposed the grid-of-tries data structure for 2D (2-dimensional) classification, which reduces the storage complexity compared to hierarchical trie.

T. V. Lakshman and D. Stiliadis [8] A formal description of the rule, classifier, and packet classification and how the rules are used to construct a tree and classify the packet.

P. Gupta and N. Mckeown [3] Proposed on how to include multiple fields in the classification module and effectively classify packets in corresponding flows. How the performance matrices such as search speed, memory requirement, scalability in classifier size, update time are used to classify the various packet classification algorithms.

S. Lyer, and A.Shelat [1] Given a classifier defining packet attributes or content, packet classification is the process of identifying the rule or rules within this set to which a packet conforms or matches.

1.3 Objective of work

With the growing demand of high bitrate and bandwidth intensive services such as video streaming, online gaming and cable television, the load on the digital infrastructure is huge. The Routers in the backbone network of any Tier of ISP's have their task cut out three fold, packets should be classified in as much less time as possible, minimum memory requirement of the routers and low power consumption.

Classification is a fundamental network performance enhancement that allows a packet-processing system to cross layers of stack of protocol to classify packets. A classifier handles each packet as an array of bits and examines the contents of header fields at specific locations in the array. Classification offers high-speed forwarding for network systems such as Ethernet switches and routers that send packets across MPLS tunnels.

The grid of tries structure simultaneously avoids back-tracking and eliminates empty internal nodes. But in grid of tries structure, we have to pre compute the pointers to the destination trie which enormously increases the complexity of the trie structure if the number of fields in the rules increases to more than 2 .This thesis illustrates the hierarchical packet classification algorithms such as hierarchical trie, set pruning trie and grid of tries and proposes a new hierarchical algorithm. The proposed algorithm constructs a hierarchical binary search tree, which does not include empty internal nodes, for the pruned set of rules. In other words, the proposed algorithm uses set-pruning to avoid back-tracking and uses a binary search tree to eliminate empty internal nodes. Hence the proposed algorithm improves the search performance as well as eliminates the memory overhead of empty nodes.

The objective of this project are:-

- To implement the trie based packet classification algorithms viz. Hierarchical trie, Set pruning trie and Grid of tries in Python programming language.

- To implement a new classification algorithm that uses the modified set pruning trie as the rule storing data structure and uses binary search in order to classify the packet.
- To implement the proposed modified set pruning trie in Python.
- To simulate the classification algorithms in NOX by varying the parameters such as packets size and number of packets.
- To compare the performance characteristics of the classification algorithms.

1.4 Thesis Layout

- Chapter 2 discusses the overview of Internet and its working. The architecture of the Internet w.r.t to the ISP's is briefly discussed. Then the router architecture and a brief description of its various components and their functions.
- Chapter 3 provides a detail insight into the working of the routers i.e. how it decodes the packet header and classifies the packet into flows. Then the basic description of the three packet classifying algorithms and their performance are highlighted. Owing to the demerits of the current trie based algorithms, a new trie based algorithm is proposed which fares better in terms of memory occupied and search speed .
- Chapter 4 gives a clear understanding of the Openflow and SDN technologies and how these trie algorithms coded in python are used to simulate the packet classification process as in real routers. The results obtained prove that the proposed algorithm is superior to the existing classification algorithms.
- Chapter 5 contains the concluding remarks of the project and the scope of future work.

CHAPTER 2
INTRODUCTION AND WORKING OF ROUTERS

2.1 Internet architecture

Internet is an interconnection of thousands of networks spanning all over the globe. It is a network of networks consisting of all commercial and service providers networks. It is not practically feasible for a single service provider to connect two distant lying on opposite sides of the earth. Depending on the area of coverage and size of the network, the Internet service providers (ISP's) are divided into three major categories. In the first category are the TIER 1 ISP's whose high speed networks form foundations of the internet. Large telecommunication and network providers such as Verizon, Sprint, AT&T, Vodafone etc form the Tier 1 of internet service providers. These telecomm giants sell or lease out their network capacities to the smaller tier 2 and tier 3 ISP's. Tier 2 ISP's are generally the smaller national service providers such as BSNL, China mobile which use the high speed network of the Tier 1 ISP's. Tier 3 ISP's are generally the regional service providers providing broadband internet connection over a region or a cluster of cities such as Ortel and ActTV.

The link between ISP's of same tier is called peering link and it is used by the service providers to share traffic heading towards each other's network. This sharing of the network via peering agreement link is for mutual benefit of the service providers, so it is free of any charges. But if a lower tier ISP has to use the link of a higher tier ISP to connect to the Internet, then it has to pay a transit fees. So Local and Tier 3 ISP's are the customers of the higher Tier ISP's. Tier 1 ISP's operate via IP and MPLS (multiple protocol layer switching) technology which uses optical fibre standards such as OC-192 (10 gbps) to OTN-3 (40 gbps), or even higher standards to meet the ever growing demand of bandwidth intensive services and applications. The high speed peering link agreements between the tier 1 ISP's form the backbone of the Internet. The primary aim of this high speed backbone network is to move large amounts of traffic as fast as possible between networks.

The service providers are connected to the rest of the Internet through a wide range of links, varying from a STS-1 to STS-48 to OC-3 link and these links generally use the Transport layer protocols i.e layer 2 in the protocol stack. The point of the network via which

the users and enterprises connect to the network is called the point of presence or POP. An Edge router is one which assimilates various connection links and all these edge routers are interconnected at a POP. The POP's are usually connected via a high speed connection to a backbone network and via a peering link to another POP's.

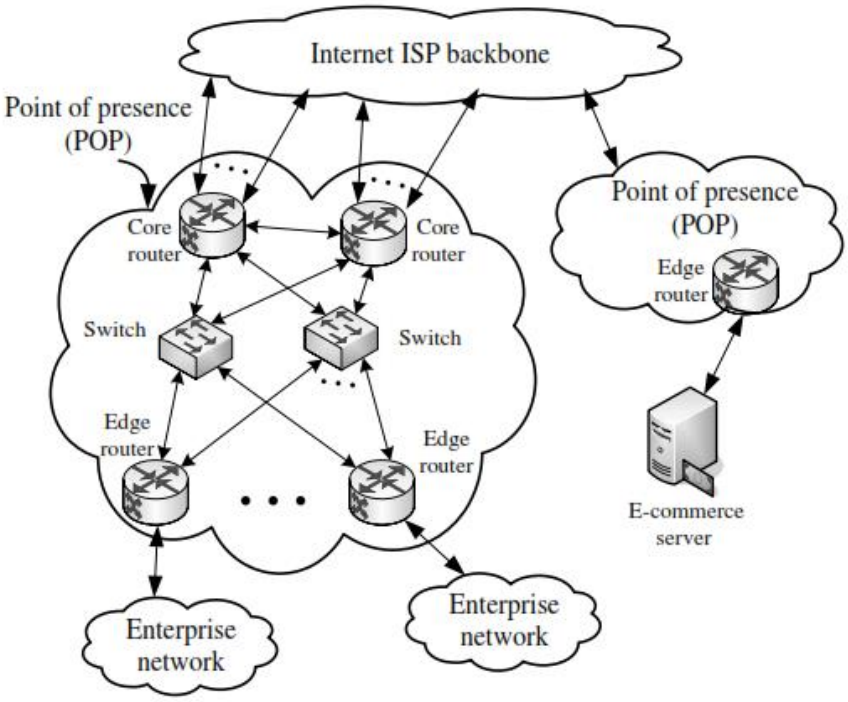


FIGURE 1. INTERNET ARCHITECTURE WITH POINT OF PRESENCE (POP)

Generally users can connect to the Internet via mobile phone, modem etc. Links to connect to the internet are telephone cables, via dial up modem we can connect to the Internet but the speed is limited to 56 kbps. Then comes ADSL and VDSL which uses the unused telephone slots to connect to the Internet at an improved speeds up to 10 mbps. Moving over to Mobile phones which connect to the Internet via radio waves use 3G and 4G techniques such as WCDMA, HSDPA can get internet access speeds up to 2-3 Mbps to tens of mbps. But the connection providing highest access speeds is the Ethernet connection via optic fibre which provides speeds upto 100 mbps.

The services provided via the internet such as e-mails, e-commerce, video streaming, online gaming are all client server type of models in which the client requests for a service and the server responds to the request by providing the appropriate service. All the client server models are distributed type of applications which are supported by the communication infrastructure. The packets generated at a source is transported to a router, where through a

switch fabric it is dispatched to the destination. The switching and routing of packets takes place in the layer 3 i.e the network layer. The network layer acts like a communication medium for applications and processes running simultaneously at two remote locations.

2.2 The Router Network comprising the Internet

The design of the routers is largely market driven and it has to satisfy the both the economic and operational requirements of the service providers. Service providers want high bandwidth and consistency but at the same time want to minimize their infrastructure and maintenance costs. The routers or MPLS switches are connected to OXC (optical cross-connect), which in turn are interconnected by DWDM transmission equipment and then to physical link, i.e the optical fiber cable. The core routers used in the network backbone have very specific requirement and that is to transfer large amounts of traffic as quickly as possible. The mid-size IP routers used at edge of the network have a wide range of evolving requirements because of the diversified services they offer at the network edge. Therefore, POP's have a pool of interconnected edge routers for this purpose of providing unique point to point solutions. But this not only increases the complexity of POPs but also raises the infrastructure and maintenance costs. For this reason, the current generation of edge routers are designed accordingly to handle the diverse services and evolving demands of the network providers.

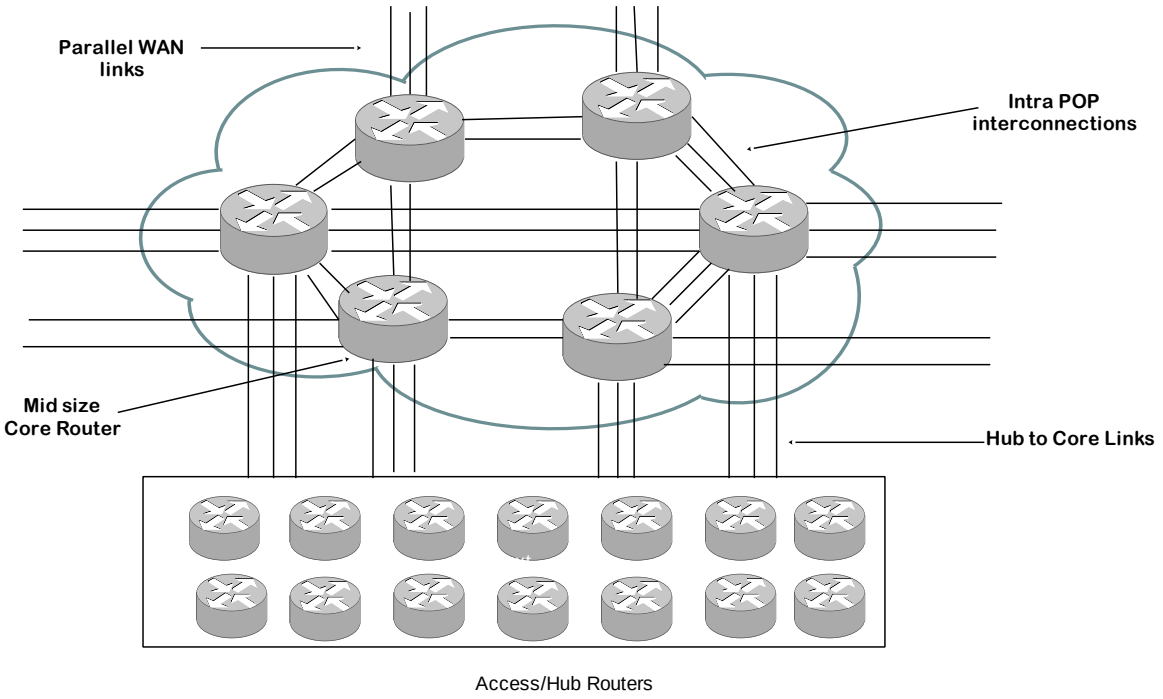


FIGURE 2. CURRENT TOPOLOGY OF THE ROUTERS IN THE NETWORK

In the future, the number of edge routers will increase manifold, so it will increase complexity of pops and result in unreliable network infrastructure. So the way forward for designing of future networks is the removal of edge routers altogether and replacing them with fewer number of core routers to increase reliability and performance, and keeping a check on the network costs. The future routers have to assimilate the strict performance characteristics of the core routers and diverse serviceability of the dynamic edge routers forged into a single body.

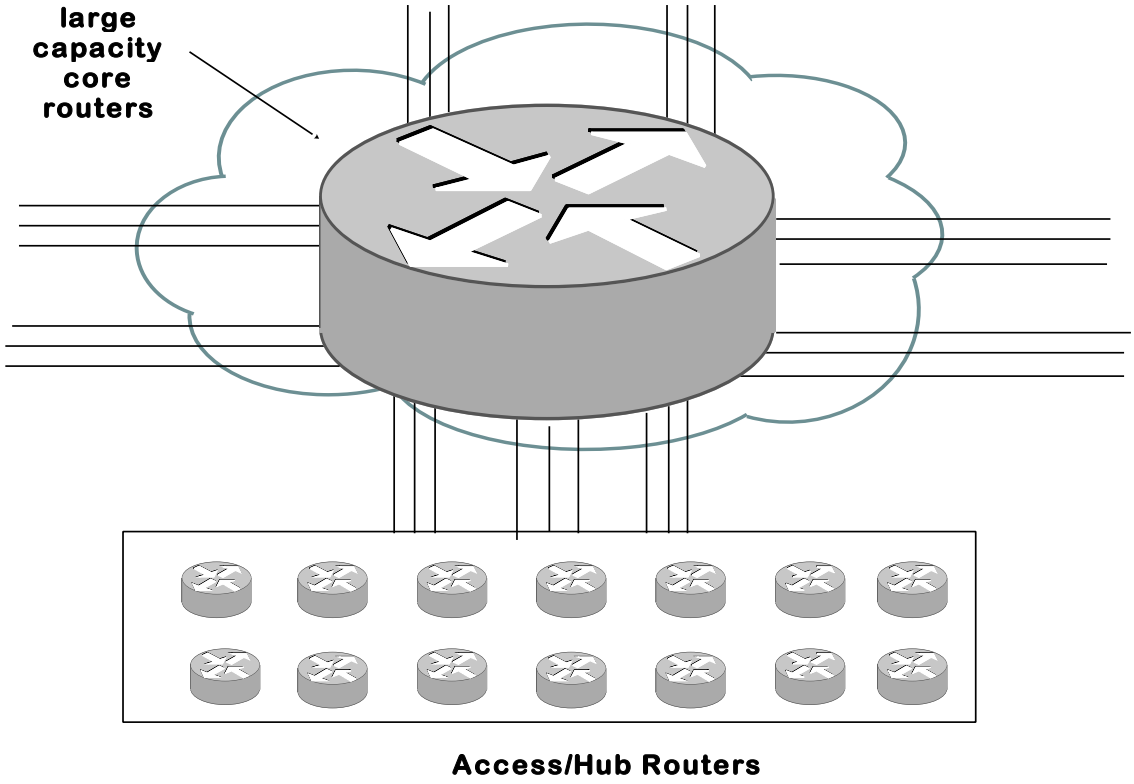


FIGURE 3. FUTURE TOPOLOGY OF THE ROUTERS NETWORK

2.3 Router architecture

The main job of a router is to receive packets through its input ports, process its header in the route controller and then switch the packet to the appropriate output port via the switch fabric. Router have two different types of architectures viz. centralized and distributed. In the centralized type of architecture the input and output ports are connected to the interfaces and these interfaces are connected to switch fabric. And all forwarding engines, a route controller and a management controller are interconnected via the switch fabric. The input ports at the interfaces sends the packets to the forwarding engine through the switch fabric. The

forwarding engine decodes the packet header and determines which output port it has to be sent to. This information is send to the input port and accordingly the packet is switched to the corresponding output port through the switching fabric. It is worth noting that in centralized type of router architecture, the packets have to cross the switching fabric twice, once for header processing in the forwarding engine and the second time for being switched to the appropriate output port. So in centralized type of architecture the switch fabric has to operate at least twice the line speed. In centralized architecture, the forwarding engines are shared by the interfaces. And administrative and management functions such as updating routing protocols to generate the forwarding tables, resource reserving are done by the route controller and the management controller.

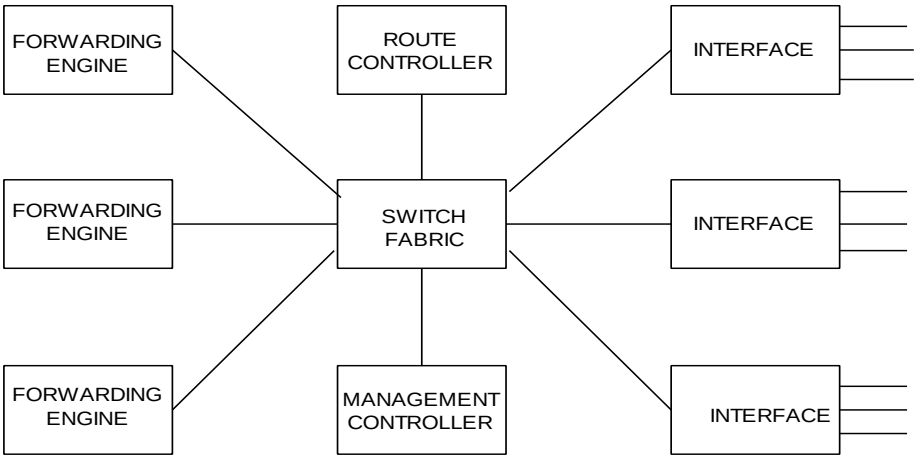


FIGURE 4. CENTRALIZED ARCHITECTURE OF A ROUTER

Moving on to the distributed type of architecture of the routers, the forwarding engine are integrated with the line cards. This type of architecture is followed by most of the current high performance routers. The packets coming through the input ports are processed there itself by the forwarding engines and then these packets are sent through the switching fabric only once. The forwarding tables in the forwarding engines are continuously updated by the Route controller based on the routing protocols used. Route updates are frequent but it is not compulsory to download a new forwarding table for each and every route. The forwarding tables are optimized for lookup speed whereas Route controllers are designed for dynamic and fast updates.

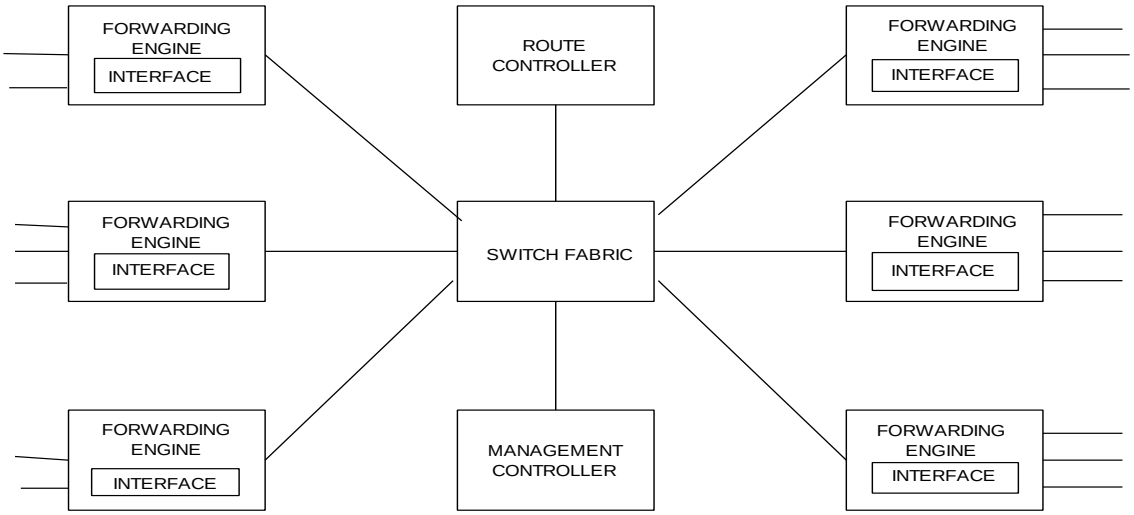


FIGURE 5. CENTRALIZED ARCHITECTURE OF A ROUTER

The functions of an IP router are of two types- data path functions and control plane functions. The data path functions of an IP router are packet forwarding decisions and output link scheduling for every datagram passing through the router. When a packet arrives at the forwarding engine, its destination address is masked by the subnet mask. It is a logical AND operation. It is done because only the network part of the destination IP address is required for forwarding the packet. The address obtained after the AND operation is used to lookup the forwarding table and this done by the forwarding engine. After that longest prefix matching is done to find the appropriate output port.

Functionalities such as configuring the system, exchange and management of routing table updates and information which are not as frequent as the data path functions fall under the control plane functions. Some examples of routing protocols are BGP (border gateway protocol), OSPF (open shortest path forwarding) and RIP (routing information protocol). The routers use these protocols to build a routing table and share these topologies with each other. The control functions are implemented on software as they not exercised on every inbound packet, they don't have a time bound constraint.

A typical router contains multiple line cards, a route controller (RC), a management controller (MC) and a switching fabric to interconnect them all. The RC, line cards, MC communicate with each other through a separate link known as Ethernet switch or via the switch fabric. The line card interfaces acts like a gateway from the physical link layer

i.e. layer 1 to the higher layers such as data layer (layer 2) layer 3 i.e. the network layer, transport layer (layer 4). The line card interfaces are the input and output ports for the data coming in and going out from a router. With the emergence of new applications, the protocols are evolving and the line cards are becoming increasingly complex. Current generation of line cards support full duplex optical fiber connection on the network side and either one incoming or one outgoing connection to the switch fabric backplane where the actual switching takes place. The router network connections aggregate the lower speed lines into a high speed link for high bandwidth applications such as video streaming. The line cards also have the provisions to regulate the flow of packets from thousands of input and output queues in order to adjust incoming and outbound traffic to and from the switch fabric.

TABLE 1. POPULAR PRACTICAL ROUTERS IN THE MARKET

MODEL	**CAPACITY(sum of ingress and egress capacities)**	**MEMORY**	**POWER**	**FEATURES**
Cisco 10000	51.2 Gbps	---	1200 W	QoS , MPLS
Dell N3000	240 Gbps	2 GB	3000 W	MPLS, VRRP-6
Juniper M-320	320 Gbps	2 GB	3150 W	MPLS, QoS, VPN
Cisco 7600	720 Gbps	1 GB	370 W	QoS, MPLS, Aggregation
Cisco 12000	1.28 Tbps	4 GB	4706 W	MPLS, Peering
HP 8812 Classic	1.44 Tbps	4 GB	3500 W	Management Interface control, high port density
lJuniper TX/T-640	2.5 Tbps	2 GB	4550 W	MPLS, QoS, Peering
Cisco CRS	9.2 Tbps	4 GB	16,560 W	MPLS, QoS, Peering

The components of a line card are transponders, a framer, a central processing unit, a traffic manager and a network processor.

Transponders: These are basically transrecievers responsible for full duplex communication between the line card and the optical fibre connection. They convert the optical signals to electrical signals (digital) for incoming network traffic and the vice versa for outgoing traffic.

Framer: A framer synchronises the header frame of the incoming packet, processes it and delineation of the packet. While transmitting the packets, a SONET/SDH framer generates the section, line and path overhead. The framer aligns the parity bits, inserts framing pattern for monitoring the performance of the router. And while receiving the packets it performs frame descrambling, bit parity monitoring and error count accrual for monitoring the performance.

Central processing unit: it performs the control plane functions such as setting up and closing the connection, updating the routing tables, supervising the register and buffers and handling the exceptions. The CPU does not operate at line speeds. Line speed is the speed at which the packets move in the interfaces and between the interfaces and the switching fabric.

Network processor: It is responsible for performing routing table lookup, packet classification and modifying the packet and it performs all these functions at line speed using some external memory such as static random access memory (SRAM) or dynamic random access memory (DRAM). It also uses external ternary content addressable memory (TCAM) for dedicated deep packet classification at higher levels (for multiple fields).

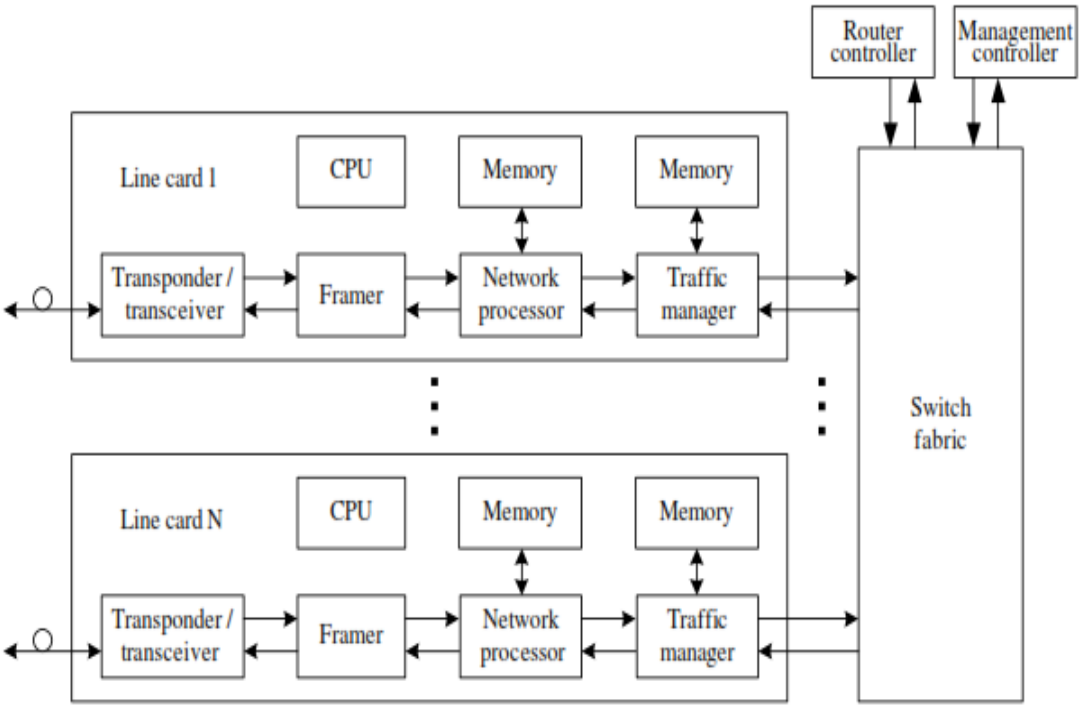


FIGURE 6. TYPICAL ROUTER ARCHITECTURE

Traffic Manager: Every connection and service class have different kinds of demands depending on the restrictions on classification speed and memory available. The traffic

manager handles the control plane functions for packet streams such as managing the buffer, traffic access control and scheduling the packets. The traffic access control consists of mechanisms which specify the required traffic flow characteristics and service requirements for example peak rate, average delivery time, buffering time for each packet stream, Shaping (delaying or adjusting their burstiness) data flows, Policing the data flows and taking corrective measures when the traffic deviates from the specified flow restrictions.

A practical router contains hundreds of line cards for classifying the data streams grouped into different stacks. Line cards are like miniaturised form of an actual router.

Chapter 3
PACKET CLASSIFICATION

3.1 INTRODUCTION

Conventionally routers try to deliver best services by handling each packet in the same manner. As new applications are being developed each day, the ISP's try to match up by providing different quality of services to the emerging new applications. To provide these quality of services, routers have to apply new classification algorithms, new packet handling methods like packet scheduling in a fair manner, resource reservation, and admission control. But prior to applying these techniques, the router should be checked if it is flow aware or not, i.e. it should be able to identify, differentiate and classify the incoming packets into different flows. A flow aware router is different from a traditional router in the sense that it is proficient enough to keep track of flows passing through it and it applies different classes of service to different types of flows. Each flow is specified by a particular set of rules. Each rule consists of different fields and an associated action with that rule is applied on that particular flow. A set of rules is called is called a classifier and it is based on the benchmarks to be applied to classify the packets of a certain network application. A Classifier should be able to identify the packet header attributes as packet classification done by the routers is simply matching a packet to a certain rule or rules and then take the corresponding action mentioned in that particular rule. To provide a clear picture of type of services delivered by a flow aware router, an example of a classifier is given below and this classifier is stored in Router X illustrated in the Figure 6.

TABLE 2. EXAMPLE OF A PRACTICAL CLASSIFIER

Rule	**Network Layer**		**Transport Layer**		**Application Layer Protocol**	**Action**
	Destination	**Source**	**Protocol**	**Destination**		
R1	208.68.16/24	192.168.25/24	RTP	*	RTP	Forward the traffic to port VI
R2	112.183/16	134.28.16/24	TCP	*	*	Drop traffic if rate > 10 Mbps
R3	134.28.16/24	*	UDP	*	*	Deny
R4	*	*	*	*	*	Permit

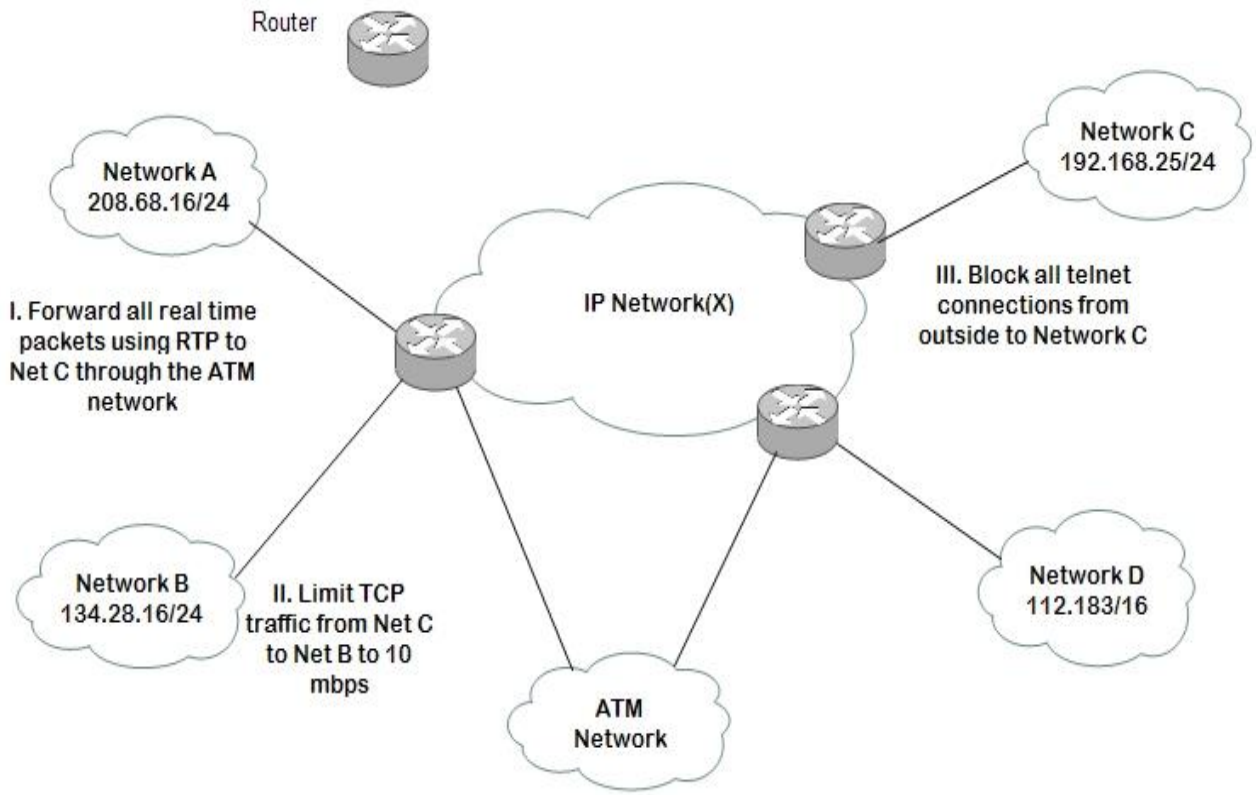


FIGURE 7. NETWORK EXAMPLE WITH A CLASSIFIER

The classifier example in the above figure has four rules in the classifier (but a practical router has hundreds of rules), the Router X offers the below mentioned functionalities:-

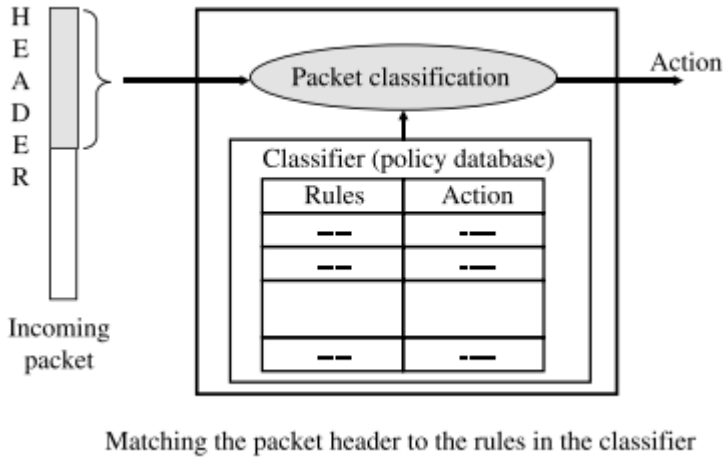
Packet Routing – Rule R1 forwards all packets using real time protocol (RTP) from Network A to Network C through the asynchronous transfer network. All the routing action takes place in the application layer.

Traffic Monitoring – Rule R2 restricts the flow of the traffic using transmission control protocol (TCP) from Net C to Net B up to a maximum of 10 mbps.

Packet Filtering – Network C is a private network accessible only by users inside the system, for example in some research laboratories or any confidential government network. Rule R3 inhibits the flow of any packet traffic employing user defined protocol (UDP) from an external network into Network C.

Here, the Rule R4 matches any incoming packet since it acts like a wildcard for all its fields

are without any specified value and the priorities of the rules is taken into account only when the packets matches R4 along with any other rule.



Matching the packet header to the rules in the classifier

FIGURE 8. PACKET CLASSIFICATION

A classifier is made up of N rules, Rj where $1 \leq j \leq N$ and Rj consists of three sub classification:

(i) A normal symbol Rj[x] where$1 \leq x \leq m$, on each of the m header fields of a packet.

(ii) A number P(Rj) points towards the priority of the rule among all the rules in the classifier.

(iii) An action which is denoted as Act (Rj).

A packet P arriving at the router with the header considered as a d-tuple (P1, P2, P3…. Pm) is considered to be matching Rj provided Pi matches Rj[x] where$1 \leq x \leq m$. Given that the incoming packet is P with its header considered to be a d-tuple, then the m dimensional packet classification method is to find a rule Rj having the highest priority in midst of all rules matching the d-tuple. The composition of packet header is a 32 bit source IP address, a 32 bit destination IP address, a 16 bit source port number, a 16 bit destination port number and 8 bits specifying the transmission protocol type. Highest priority among the rules is chosen by the longest prefix matching condition, the longer the number of bits of the header is matching the rule fields bit, higher is the priority. The rule having the highest priority among all the matched rules is chosen by the classifier and its corresponding action is performed on the packet.

The classifier example given in the table 3, each rule has five fields specifying the source and destination IP addresses in bits, source and destination port numbers in ranges. The prefix length specification is the same as in IP lookups, where the longest matching prefix is chosen for obtaining the next hop information of the packet. The source and destination port numbers is generally a number for example 36, or a range of numbers 243-1024, and greater than 1024.

TABLE 3. CLASSIFIER EXAMPLE WITH SEVEN RULES IN FOUR FIELDS

RULE	F1	F2	F3	F4	ACTION
R1	00*	110*	6	(10,12)	Act0
R2	00*	11*	(4,8)	15	Act1
R3	10*	1*	7	9	Act2
R4	0*	01*	10	(10,12)	Act1
R5	0*	10*	(4,8)	15	Act0
R6	0*	1*	10	(10,12)	Act3
R7	*	00*	7	15	Act1

A classifier consists of a Rule set, C = Rj, $1 \leq j \leq N$, here N specifies the number of rules in the rules set and each rule Rj has 'm' fields. The fields are labelled as Fx, where $1 \leq x \leq m$. So the rule Rj can be said to be consisting of 'm' sub rules, i.e. Rj = {Rj_1, $Rj_{2,\ldots\ldots,}$Rjm}. The fields F1 and F2 basically represent the source IP and destination IP addresses. These are specified in prefixes and so they can effectively be represented by a hierarchical trie or by a TCAM for faster memory searches. The fields F3 and F4 represent the particular or range of the input and output ports for the traffic flow. These two fields can be proficiently handled by segregating into different ranges and executing range lookup for matching the rule sub field.

The example of the classifier provided in the table above has seven rules in the rule set and each rule has four fields. The seven rules are arranged in decreasing order of their priorities i.e. rule R1 has the highest priority. This rule set serves as the basis of packet classification algorithms explained in details in the forthcoming sections.

The performance of the classification algorithms are measured and compared on several indices such as search speed, memory requirement, update time, scalability in operation and the flexibility in specification.

Search speed - A high speed optical fibre internet connection carrying traffic for example at 20 Giga bits per second require very high classification speed. Let say, the minimum IP packet size is 40 byte, then the link carries packets at speeds of 62.5 million packets per second. So the packets have to be classified within a maximum time of 16 nanoseconds.

Memory requirement – Lower the memory required, faster is speed at which the memory is accessed and lower is power consumed. So small storage space is essential for software algorithms based on cache memory and the hardware algorithms based on SRAM and DRAM.

Update time – When there is a modification in the classifier, such as addition or deletion of a field entry of any rule or any rule is added, then the structure of the algorithm has to be modified accordingly. Some particular applications such as flow recognition essentiate the fact that the update time should be small or else it affects the classification performance.

Scalability in operation – The link speed and the type of application determines the size of the classifier. An edge router handling traffic to and from a town handles anywhere between 100k to 1 million flows. So as the number of services provided by the router increase, the number of header fields increase, i.e. the router should be able to scale as per the service required.

Flexibility in operation – The classification algorithm should be able to handle varied specifications in the rules such as variable prefix length, port number ranges and wildcards.

So the algorithms described below emulate the above mentioned parameters to be efficient in their performance. The simplest of all packet classification algorithms is the binary search algorithm. In this algorithm, the rule set is arranged as an array or as a linked list in increasing order of their priorities. When a packet arrives, its packet header is decoded and is compared sequentially with the fields in the rule set in the classifier till a match is found. The classification time and storage complexity is O(N) for a N rule set. So, a binary search is somewhat useful in small rule sets, but practically not feasible for large rule sets.

3.2 Trie based Classification Algorithms.

3.2.1 Hierarchical trie

Hierarchical trie is a multi-dimensional form a binary trie where each dimension represents a field in the rules set. It is also known as backtracking tries and muti level tries.

Rule storing structure

In the figure 8, a two dimensional trie is shown having two tier structure representing the first and second fields of the classifier rule set provided in table 3. Only fields F1 and F2 are considered in the hierarchical trie as they are address prefixes and can be represented easily in the form of tries. The nodes representing F1 trie are elliptical in shape and the nodes representing F2 trie are circular in shape. The nodes in the F1 trie are connected to the nodes in the F2 tries by means of a next tree pointer represented in the form of a bold curved arrow. The grey nodes are labelled with a rule name such as Rj which means that if that node is reached during a search process, then that packet is said to have matched that particular rule. Since there are only four distinct prefixes in the field F1 of the rule set, so there exists four distinct F2 tries. A binary radix trie is formed using the F1 field of all the rules Rj_1 of the classifier. Binary radix trie is a particular form of a binary trie in which all the trie nodes are disjoint sets. For each distinct F1 node a (d-1) dimensional trie Ti is built recursively using the prefixes in the field F2 and each root node of F2 trie is connected to a F1 Trie elliptical node by means of a pointer stored in that F1 trie node.

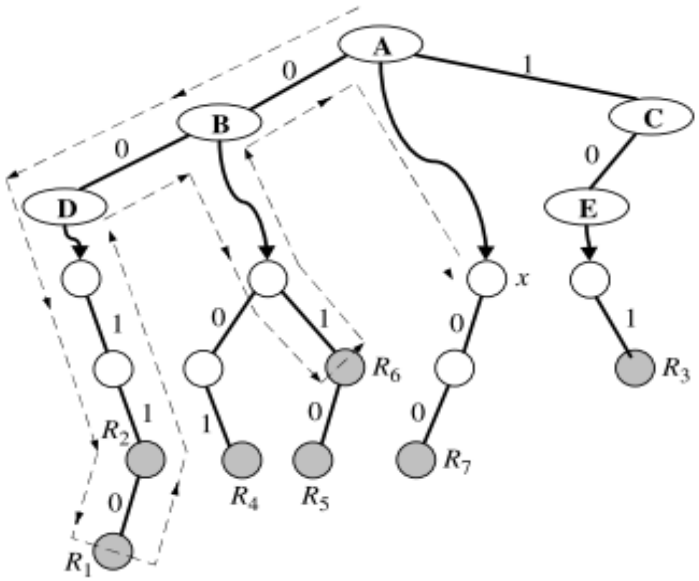


FIGURE 9. HIERARCHICAL TRIE DATA STRUCTURE FOR F1 AND F2 FIELDS

Classification scheme

Let the inbound packet has header in the form of (h1, h2, h3, ….hx) and it is compared to the trie based structure by the following method – the search algorithm first navigates the F1 trie by the help of header h1, after matching with a f1 node it travels to the next trie level using the next trie pointer. The algorithm recursively searches the d-1 dimensional F2 trie nodes for a suitable grey node to match a rule Rj.

For the trie shown in the figure 8 formed using the rule sets provide in table3. An incoming packet header is (001,110), so the search process begins by comparing the first part of the header prefix i.e. 001 with the F1 trie nodes. The first bit is 0, so the search moves left from node A to node B , and then again 0 is present in prefix so the search process lands at node D. Then the next bit is 1 but since there is not more nodes to traverse in the F1 trie so the search for F1 trie node stops at Node D and using the next trie pointer the search moves to the level 2, i.e. to the F2 trie. The F2 trie prefix to be searched is 110. So it moves using o to move left and 1 to move right in the trie, the search lands at nodes R1 and R2, but only R1 is recorded since it has higher priority. After recording node R1, the search backtracks to the 1^{st} ancestor of node D in F1 trie, i.e. node B and this process is continued till all the ancestors of node D have been traversed. Finally the search ends at mode x of the ancestor node A. The rule matched during the entire process is R1, R2 and R6, but only R1 is returned as the matching rule since it has the highest priority. Here the backtracking is essential because the first part of the prefix'001' may match many nodes in the F1 trie level and also we do not know in advance as to which F2 trie will match second filed of the prefix '110'. So backtracking is essential to find all the possible matches and return the highest priority rule.

Performance

From the storage requirement point of view, hierarchical trie is most efficient classification algorithm. The storage complexity is O(dW). Here N is the number of rules in the rule set of the classifier, d is the number sub fields in each rule and W is the maximum depth of each sub field. But on the negative side, it takes a long time in selecting the matched rule, backtracking is performed in the search to take into account all the rules matched by the packet and the highest priority one is chosen. Backtracking is also essential since the trie structure does not effectively show the priority levels of the rules. The search time complexity is $O(W^d)$ since for

searching each trie the time taken is O(W) and there are d subfields in the rule set , so intuitively the search time complexity will be $O(W^d)$. The update complexity is $O(d^2W)$.

3.2.2 Set Pruning Trie

The set pruning trie is a tweaked form of the hierarchical trie which solves the backtracking problem prevalent in hierarchical trie.

Rule storing structure

In this trie structure, each F2 trie node containing a valid prefix is duplicated into its descendant trie structure and then the F2 trie structure is built as per the new rule set. In the example given below , the two dimensional set pruning trie is constructed using the rule set of the table 3. In the original hierarchical trie, the rules sets in the F2 trie structure for the F1 trie nodes A,B and D are (R7), (R4, R5, R6) and (R1, R2) respectively. But here in the set pruning trie structure the corresponding F2 trie nodes of the F1 trie nodes A, B and D are (R7), (R4, R5, R6, R7) and (R1, R2, R4, R5, R6, R7) respectively since the rule sets have been copied into the descendants from the ancestor's trie nodes.

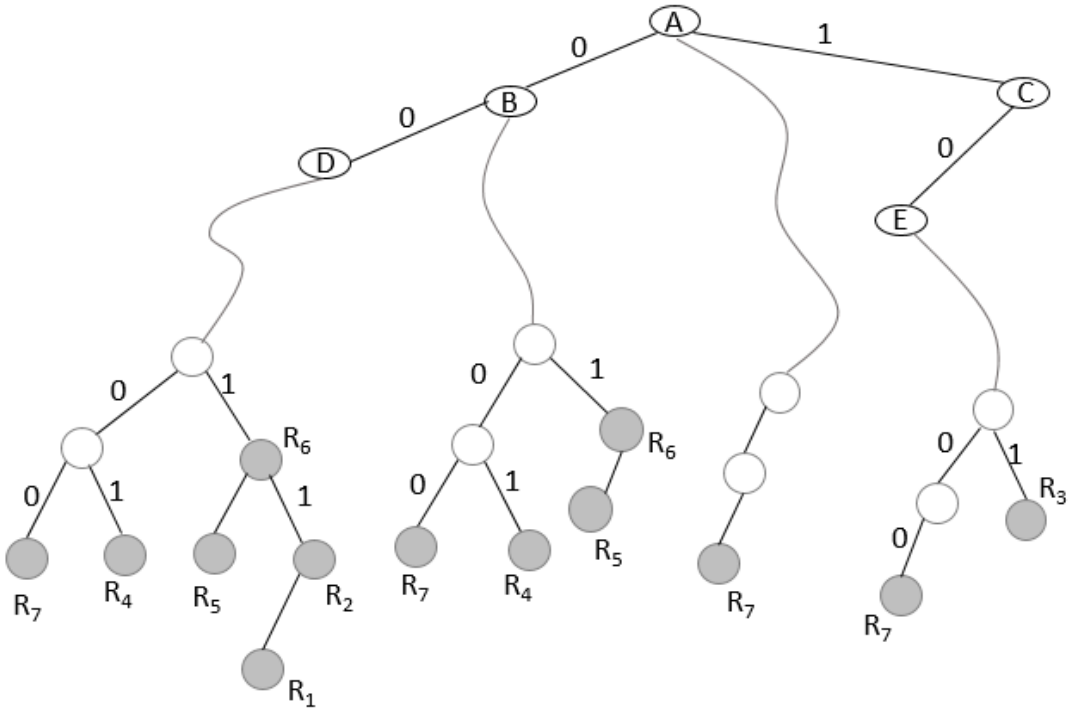


FIGURE 10. SET-PRUNING TRIE DATA STRUCTURE FOR THE RULE SET IN TABLE 3

Classification scheme

Packet with header prefix (001,110) is used to demonstrate the packet classification using set pruning trie. The process of rule matching is done by matching the d-tuple packet header with each of the d dimension of the set pruning trie. Here also the search process matches with multiple rules R1, R2 and R6 , and again R1 is chosen as the matching since it satisfies the longest prefix matching and has the highest priority. Here there is no backtracking because all the similar prefixes of the ancestor nodes have already been copied into all the descendant nodes.

Performance

The backtacking in the hierarchical trie were a necessity because the rule sets of the F1 trie are disjoint sets, also we don't know in advance as to which F2 trie contains the matching rule. Since the set pruning trie eliminates this need, so the search time complexity is reduced to O(dW) instead of $O(W^d)$ as in the case of hierarchical tries. But the elimination of backtracking process comes at the cost of increased memory requirement. The storage complexity is $O(N^d dW)$ as a rule is needed to replicated maximum N^d number of times. The complexity to update rules in the trie structure is of the order $O(N^d)$.

3.2.3 Grid of Tries

Rule storing structure

The grid of tries trie structure solves both the demerits of the hierarchical trie and the set pruning trie i.e. the backtracking process and excess memory requirement respectively. Here the switching pointers to the F2 trie are pre computed and stored in selected F2 tries. This process is equivalent to merging of F2 trie nodes of the ancestors into the F2 trie nodes of the descendant. The switching pointer which are labelled as Ps are 0/1 and inserted at node y whenever its corresponding node in the set-pruning trie has a 0/1-pointer to another node z while y does not. Node z may have many corresponding nodes in the set pruning trie but y points to the one present in the F2-trie that is most similar to the F2 trie having node y.

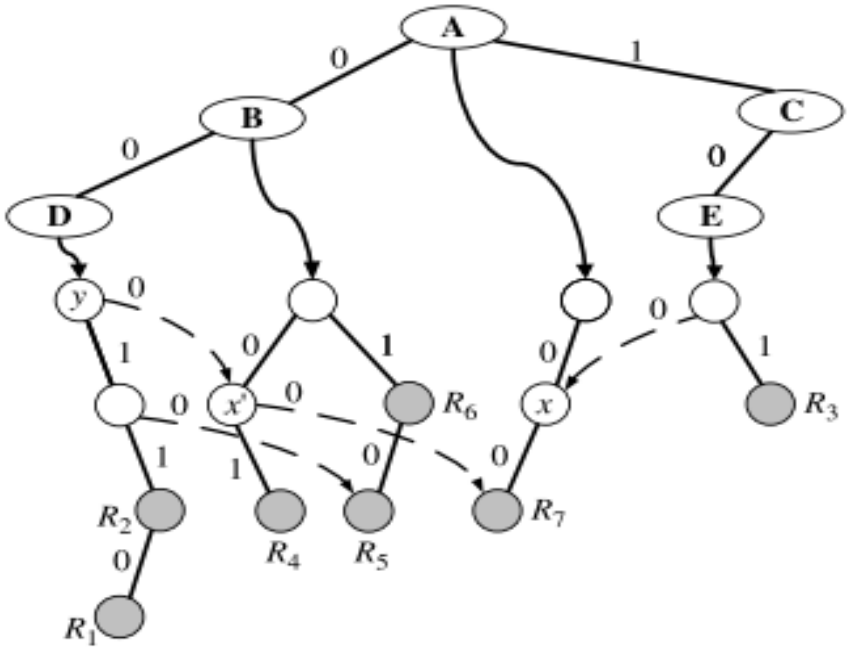


FIGURE 11. GRID OF TRIES DATA STRUCTURE FOR THE RULE SET IN TABLE 3

Classification scheme

If the switching pointers in the grid of tries are considered as 0/1-pointers, then the search procedure is the same as in the set-pruning trie. For the incoming packet header is (001,110), the F2 trie node R2 which is denoted as F2 B-trie, the basic change between the trie structures in the figure 8 and figure 9 is that node R2 is duplicated to its ancestor nodes in the set pruning structure where as in the grid of trie structure pre computed switching pointer '0' is used to point to node x' from node x i.e. R7. The pre computed switching pointers are represented by the dashed arrows.

Performance

The storage complexity is reduced to O(NdW) as compared to the complexity of $O(N^d dW)$ of the set pruning trie. The search time complexity of O(dW) is the same as set pruning trie but vastly improved from hierarchical trie due to the pre computed switch pointers. But the problem with grid of tries structure is that it is very complex to build when the number of fields increases. Also incremental updates to be applied to the trie structure are quite complex since many pointers are to be altered. If any node needs to be deleted, then the pointers pointing to that node needs to be reoriented to the new node created in its place.

3.3 Proposed Packet Classification Algorithm

The heiarachial trie algorithms described earlier suffer from either longer search process problem or large memory consumption or increasing complexity. The set pruning trie solves the problem of backtracking to ancestor nodes but it results in occupying a large amount of memory space. Also there are grid of tries but their complexity increase enormously when the number of fields in the rule set increases. So we have proposed a modified tree structure based on the fact that more specific prefixes i.e. longer prefixes generally have higher priorities. If a packet header prefix is matched first by a matching rule with more specific prefix, it is in the offing that particular rule is the best matching rule and so the tree search can be circumvented for most of the inputs by just comparing the packet header prefix with the matching rule with the highest priority of each level 2 trie containing the destination trie addresses. The ancestor or parent tree pointer is saved in each destination tree root node so that it will be linked to the that ancestor whenever the search process commences. So it is like a set pruning trie employing binary search tree with deletion of the empty nodes in the destination trie.

3.3.1 Construction of a set pruning trie employing binary search

The proposed modified set pruning trie algorithm is an amalgamation of binary search tree and set pruning trie. The trie construction is similar to the set pruning trie but with the removal of empty internal nodes in the destination trie by employing the binary search trie algorithm. The two field modified set pruning trie is constructed using the table 3 field numbered 1 and 2, 1 for the source IP prefixes and 2 for the destination IP prefixes.

For employing this scheme, the IP addresses in the destination trie should be stored in increasing order of length of the prefixes. While comparing two prefixes having different lengths, the shorter length prefix is equated to a similar length substring of the longer length prefix. If the matching turns out to be equal, then the next bit of the prefix with longer length is observed. If the next bit is one, then the loner length prefix is denoted as bigger or else the prefix with smaller length is denoted as bigger. So the binary search trie sanctions a restriction on constructing the binary destination trie, in way it forces a ancestor node to be located at a higher level than a child or descendant node in order that the ancestor prefix is

compared earlier. Therefore the shorter length ancestor prefix will be matched prior to a longer length prefix provided they are in a prefix nesting liaison.

3.3.2 Deleting the empty internal nodes

In binary search trie structure, the empty internal nodes are removed if they satisfy the following two conditions, firstly they should have no child or only one child and secondly, they should not be a grey node, i.e. it should not contain the next hop information or any matching rule. The level 2 of the trie structure is built using the destination table IP prefixes. The valid bit's entry is set to one when a rule is saved in the node or else the node is deleted when it has no stored rule. If the node is deleted the child pointers pointing to it are reset to the next available ancestor so it maintains a continuity in the search process for all other field comparisons. The linked list pointer are used for connecting the source and destination prefix pairs and the rules associated are arranged in decreasing order of their importance by the link list.

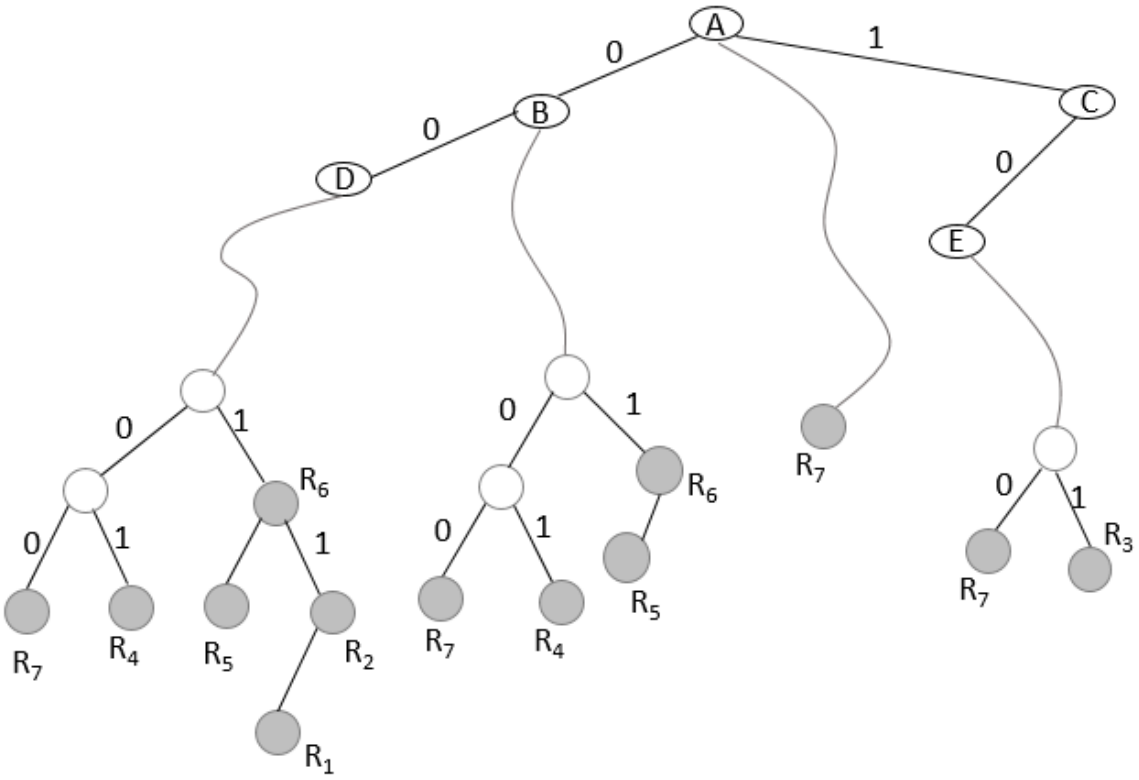


FIGURE 12. THE PROPOSED MODIFIED SET PRUNING TRIE USING BINARY SEARCH

3.3.3 Search procedure

The search process is similar to as in stet pruning trie, the incoming packet is searched for the longest matching rule with the same fields. The incoming packet's source address is compared with the level 1 trie for field F1 with 0 prefix denoting to the left pointer, it points out that the input address is smaller in length than the prefix stored in that node and 1 prefix pointing towards the right pointer. If the packet matches any node in the source IP prefix trie or there are no more nodes to be traversed then it follows the next tree pointer pointing to the next level of trie .i.e. to the destination IP tree. The rule which is best matching is given the lowest priority number. Now the search in the destination IP trie is done recursively as in the source IP trie. For the same packet used in previous tries (001,110), the nodes in the destination IP trie traversed and matches are R2, R1 and R6, but R1 is best matching rule due to its highest priority.

3.3.4 Algorithm Performance

The search complexity of the proposed algorithm is O(dlogN), replacing W which is depth od each field with logN, where N is the number of rules in the rule set. The memory requirement is less than or at maximum equal to that of the set pruning trie i.e. O (N^d) which is the same as the hierarchical set-pruning trie, in the worst case.

CHAPTER 4

Implementation of Algorithms in SDN

4.1 Openflow

Openflow is an open standard for software defined network (SDN) based communication protocol facilitating access in between the forwarding planes of switches and routers. It is basically an open network protocol for switches and routers providing sophisticated traffic management. It is a standardized platform for network researchers to perform experiments on internal working of network devices such as routers and switches. It also provides greater programmable control of network which results in the development of new applications in virtual environment.

Protocol is nothing but a specific set of rules that is used between two end point users in a communication system. Protocols specify the way in which the communication packets between the users. Some examples of the protocols used in today's network are- IP, UDP, RTP, TCP, BGP, OSPF, DHCP etc.

In traditional routers and switches, the data path functions such as fast packet classification and control path functions such as high level routing are both handled by the same device. An open flow switch splits up the two tasks. The high level routing decisions are communicated by a separate controller to the data path functions performing switch. The open flow protocol is the communication protocol used by the open flow controller and the switch for communicating with each other. The communication occurring between these two entities includes packets sending and receiving status, updating the forwarding tables and the classifier rule sets, and packet traffic statistics. The data path of the switch contains a flow table consisting of a certain set of rules and certain fields in each rule and an action associated with that rule such as sending it to a particular output port, altering certain fields of the rules or dropping the packet. The open flow switches receive a packet and sends it to the controller and the controller processes a path for the packet by decoding its header information.

So basically openflow creates an open platform for network researchers to program and test innovative network routing and switching protocols without having the need to buy sophisticated network equipment for applications such as secure networks, virtual machine mobility, internet of things, IP based mobile communication, cloud services etc.

4.2 Software Defined Network

SDN is the technology on which the open switches using open flow protocols operate. It is used to execute the control plane functionalities on commodity servers and general purpose hardware such as personal computers. It excludes the need for specific network hardware components. It helps in maintaining, control and program the data plane from a central controller. It is designed to not only control not just a networking equipment but an entire network consisting of several hosts and a controller. So SDN is as sort of revolution in the networking industry that facilitated the shift from static networks to programmable flexible networks.

NOX is a form of the software defined network. It is an open platform for budding network scientists to test and run their programmable networks. The first SDN technology to be implemented was Ethane that was developed by a Stanford based network researcher. NOX provides a network programmer fast and asynchronous user interface on linux platform using Ubuntu OS.

The network topology considered in this project is that of 4 hosts connected to a switch and the switch is managed by a controller.

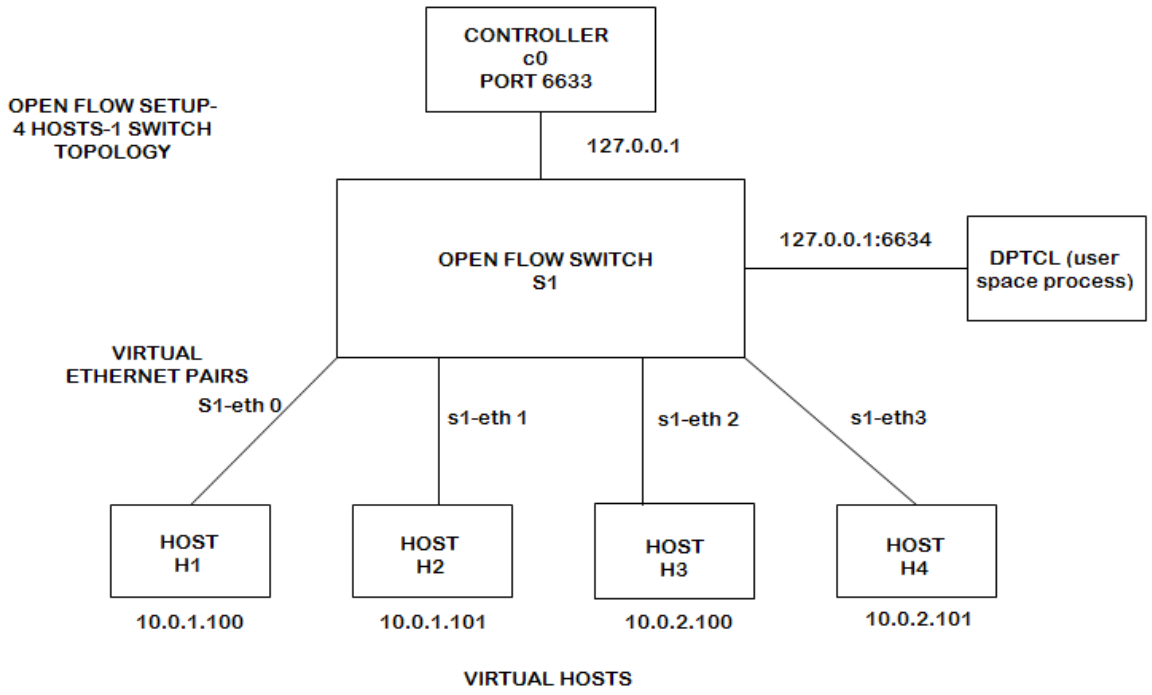


FIGURE 13. OPEN FLOW SETUP TOPOLOGY

4.3. Results and Discussions

The trie algorithms which are coded in python and then codes are simulated on openflow platform NOX, using 4 host and 1switch with a controller configuration shown in the previous algorithm. There are three separate terminal emulators run on Linux platforms simultaneously on openflow SDN, one of the windows is for configuring the network topology as mentioned earlier, second window is for loading the "python.py" file in which coding of the respective trie algorithms are loaded along with some proposed modification. The modifications include a separate module of codes for including the serial port of the NOX and setting up the timing and packet counters along the trie algorithms.

Given below are some snapshots of the programming window running in the virtual box. By using some particular functions such as "packETH" and "packETH.cli" we can create and modify the type and size of packets we want to send to the switch.

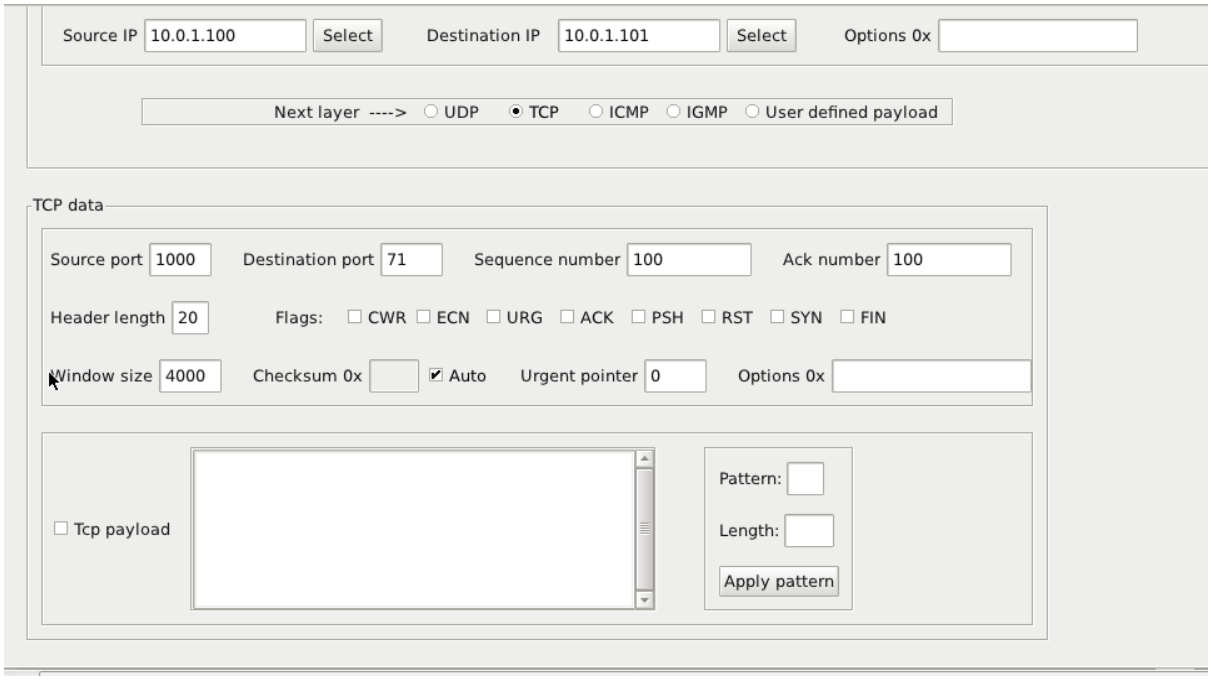


FIGURE 14. USE OF PACKETH FUNCTION TO GENERATE PACKETS

```
('The value of rule and path 2', '0000101000000000', '')
('I got called with', 16, 32)
('\t\t\t\t\t\t\t\t', 4)
('The value of rule and path 1', '', '00001010000000000000000101100100')
('The value of rule and path 2', '', '')
('I got called with', 0, 32)
The F2 type node is 51
('Returned with', '00001010000000000000000101100100')

The F1 Type Node is: 32
('Active Rule value: ', 'R2')
('The Active Rule Here is: ', '2')
('Value of R_True is: ', '542')
('Time Elapsed is', 17.62899999999945, 'ms')
('Counter values :', 42, 20, 20, 0, 0)
```

FIGURE 15. OUTPUT SNAPSHOT OF THE SIMULATION OF THE PROPOSED ALGORITHM

```
t@openflowtutorial:~/packETH-1.8/cli# sudo ./packETHcli                    Usage: packETHcli -i <interface> -m <mode> [-d <delay> -n <number of packets> [-s <startsi
stopsize stepsize] -p period] -f <file>

 <1,2,3>  - 1: send one packet (builder mode), no further options
          - 2: send sequence of one packet (Gen-b mode)
                  -d <us, 0> - delay between packets in micro seconds (-1 for maximum speed without counters, 0 for max speed with counters)
                  -n <number, 0> - number of packets to send or 0 for infinite
                  -s "<startsize stopsize stepsize> (please note that checksum in not recalculated for shorter lengths!!!)"
                  -p <period between steps>
          - 3: send sequence packets (Gen-s mode) - not yet done...

 <file name> - file name where packet is stored in pcap format

mples:

/packETHcli -i lo -m 1 -f packet1.pcap                                  - send packet1.pcap once on lo
/packETHcli -i eth0 -m 2 -d 1000 -n 300 -f packet2.pcap                 - send packet2.pcap 300 times with 1000 us (1ms) between them
/packETHcli -i eth0 -m 2 -d -1 -n 0 -f packet2.pcap                     - send packet2.pcap at max speed, infinite times, no counters
/packETHcli -i eth0 -m 2 -d 0 -n 0 -f packet2.pcap                      - send packet2.pcap at max speed, infinite times, with counters
/packETHcli -i eth1 -m 2 -d 0 -n 0 -s "1000 1500 100" -p 10 -f packet3.pcap  - send packet2.pcap at max speed, start with packet length of 1000 bytes
                                                                        - send 10 packets with this packet length, increase packet length by 100 bytes till 1500

/packETHcli -i eth0 -m 2 -d 100 -n 0 -s "8500 8500" -f packet2.pcap     - send packet2.pcap infinite times with 300us between them
                                                                        - with the size of 8500 bytes (even if packet2 is longer)


t@openflowtutorial:~/packETH-1.8/cli# sudo ./packETHcli -i h1-eth0  -m 2 -d 0 -n 21 -f rule2
bash)                                                                                                                    2:19 30/03/2015
```

FIGURE 16. USE OF PACKETH.CLI FUNCTION TO SEND THE PACKETS

There are various ways in which the proposed algorithm can be optimised and it depends on the characteristics of the rule sets used. Using class bench database, some particular rule sets have been used for comparing the classification algorithms and these rule sets accurately replicate the characteristics of a real time classifier. The rule sets employed for performing the comparison are firewall, access control list (ACL) and IP chain (IPC) and these rule sets have

sizes of 1000 rules and 5000 rules. The physiognomies of the various rule sets depends on the distribution of rules according to their source and destination prefix lengths. In the access control list rule set majority of the rule sets have long source and destination prefix lengths. But in the IP chain and firewall rule sets, there is an even distribution of long length and short length source and destination prefixes.

Zero length source and destination prefixes are known as wildcards. If a wildcard is present in a rule set, then for a packet to match a rule it has to match both, the wildcard as well as the longest matching rule. So if a rule set has many wildcard rule sets, then the wildcard acts as an ancestor of every node, so these rules have to be copied into all the descendant trie nodes which contain the next hop destination addresses. This results in excess number of duplicated rules which in turn increase the amount of memory occupied. In the access rule sets there are no wildcards in the top 80% priority. The importance of the wildcards are generally low. And in the IP chain and the Firewall rule sets, there are no wildcards in top 60% rules as per priority in the rule sets. The search process becomes more efficient if the non-wildcards are searched first as they have higher priority as it automatically avoids the searching for comparatively low priority wildcards. The number of incoming packets used in the simulations are three times the number of rules used in the rule sets.

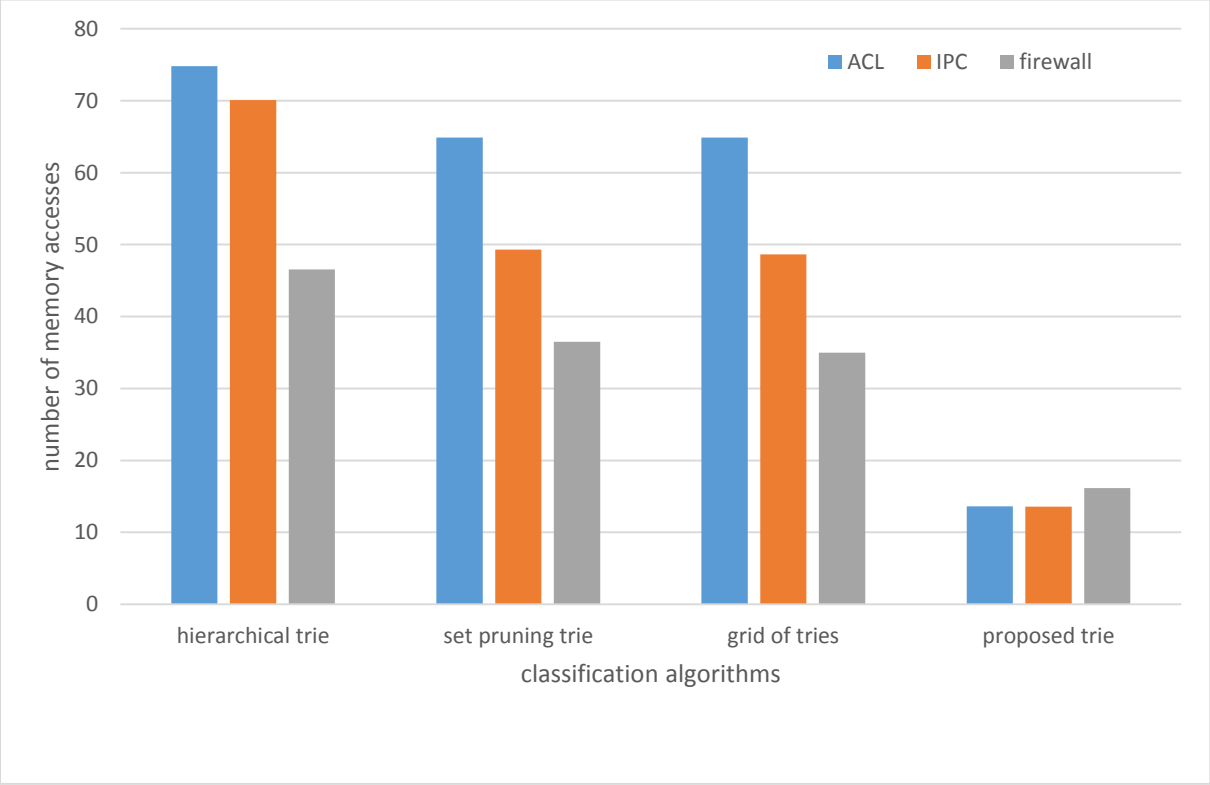


FIGURE 17 GRAPH DEPICTING NUMBER OF MEMORY ACCESSES FOR 1K RULE SETS

The classification speed is calculated by taking the number of memory accesses into account. The number of nodes accessed in the source and destination tries is equivalent to the number of memory accesses i.e. the number of visits in the memory entries in the linked list. The performance of the Hierarchical Trie, Set Pruning Trie, Grid of Tries and the proposed trie algorithm have been compared. The performance comparison are done by taking the Rule sets containing 1000 and 5000 rules each of Access control list, firewall and IP chain rule sets.

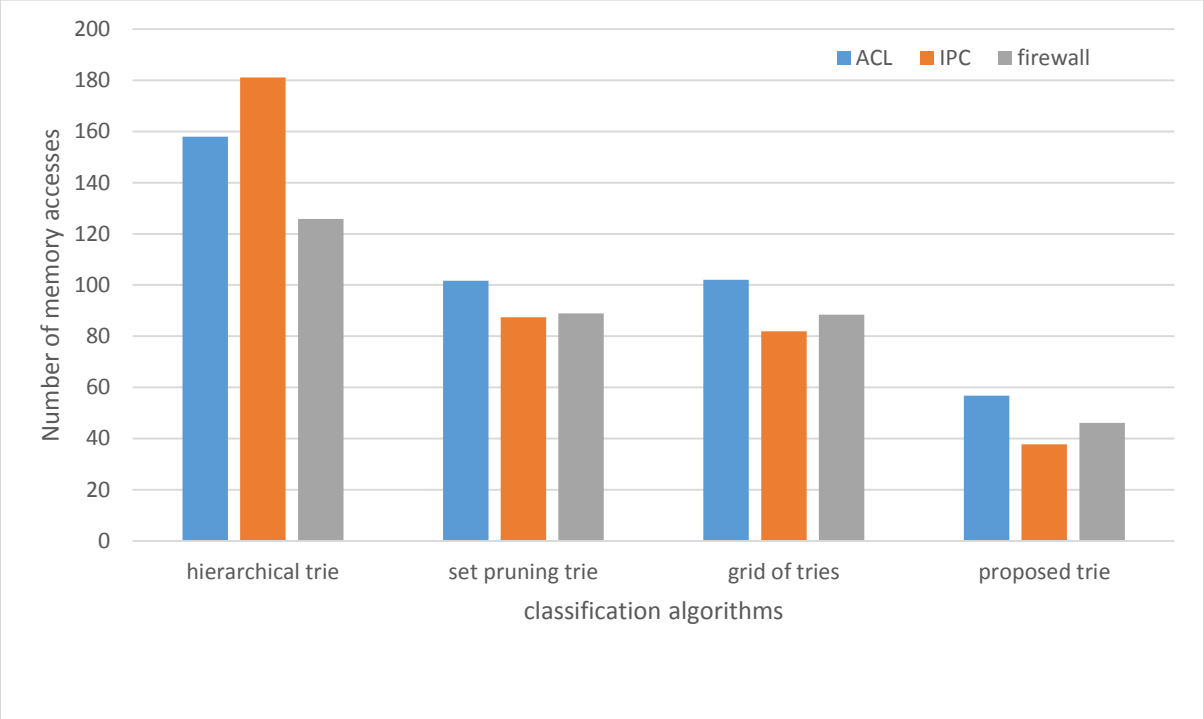


FIGURE 18 GRAPH DEPICTING NUMBER OF MEMORY ACCESSES FOR 5K RULE SETS

The proposed algorithm is designed to reduce the memory overburden caused by the set pruning tries. It has performed better than all the algorithms while using both the 1000 and 5000 rule sets.

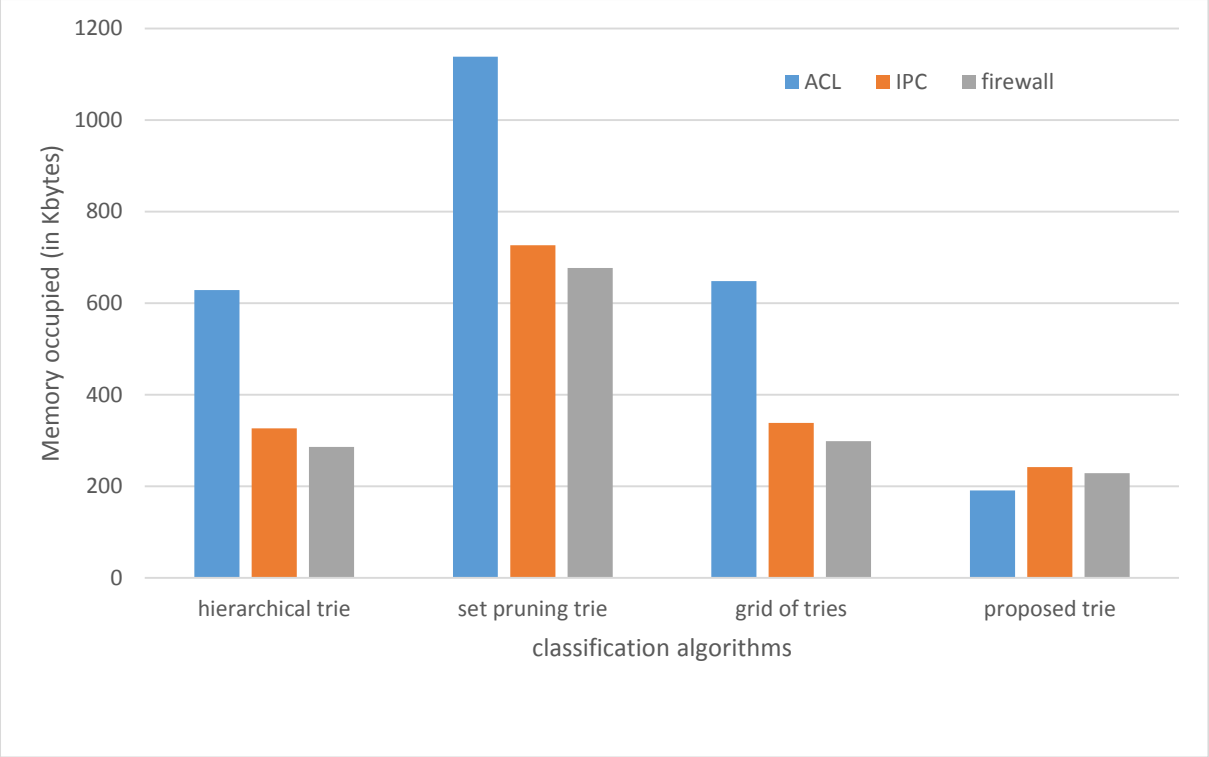


FIGURE 19. MEMORY REQUIREMENT OF THE 1K RULE SETS (IN KBYTES)

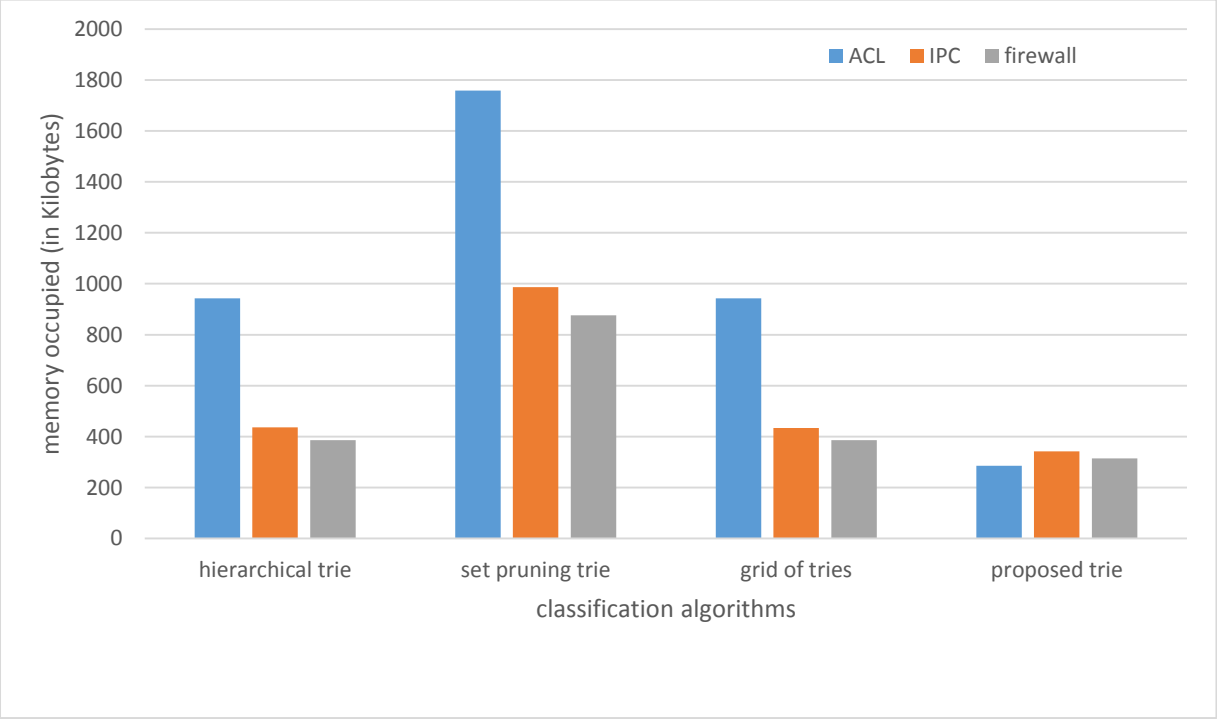


FIGURE 20. MEMORY REQUIREMENT OF THE 5K RULE SETS (IN KBYTES)

Also the number of memory accesses of the efficient algorithm proposed do not change much with type of rule sets. As expected the set pruning tries occupy memory in excess as compared to any of the trie based algorithm. The proposed algorithm consumes very less memory because of the removal of the empty internal nodes and also the search speed is improved a lot as the search algorithm does not have to traverse the empty internal nodes and also backtracking have been removed by the use of set pruning trie structure.This improvement in the performance is due to both, avoidance of backtracking and deletion of empty nodes. It has been seen that the memory required for the proposed algorithm are less than 500 Kbyte for each rule sets, so the proposed algorithm can be easily be embedded in a IC chip.

Chapter-5

CONCLUSION AND SCOPE FOR FUTURE WORK

5.1 Conclusion

The research work presented in the thesis mainly deals with analysis, comparison and development of trie based classification algorithms. The simulation of the packet classification algorithms have been carried out in Software defined Networks operating on NOX environment for varying rule sets. The different rule sets used in the simulations are IP chain, access control list and firewall. The simulation results show that the proposed algorithm fares better than the existing trie based algorithms in terms of memory space occupied and search speed. It consumes much smaller space to be executed and has faster search speed than the existing techniques as shown in the graphs plotted in chapter 4 obtained after simulating the algorithms in test bench environment. Since the algorithm consumes very less space in the memory, so it can be easily embedded in a chip to be used in a router to get improved packet classification performance.

5.2 Scope for future work

The search speed and memory requirement indices have been investigated for the trie based classification algorithms. Since the networks are evolving to cater to the diversified distributed applications so the routers have to cater to the needs of such a network. So the areas in which further research work can be done are:

1. Increasing the number of the fields in the rule sets to accommodate more and more distributed network services.
2. Implementing the same algorithm in Range based classification scheme and observe their performance characteristics.
3. Increasing the real time rule sets containing tens of thousands of rule sets and tracking the performance of the algorithms with the rule sets continuously updated by the routing protocols.

REFERENCES

[1] S. Lyer, R. R. Kompella, and A. Shelat, "ClassiPI: an architecture for fast and flexible packet classification," *IEEE Trans. On Networks*, vol. 15, no. 2, pp. 33–41 (Mar. 2001).

[2] P. Gupta, "Routing lookups and packet classifications: theory and practice," in *Proc. HOT Interconnects*, Stanford, California (Aug. 2000).

[3] P. Gupta and N. Mckeown, "Algorithms for packet classification," *IEEE Trans. On Networks*, vol. 15, no. 2, pp. 24–32 (Mar. 2001).

[4] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvagel, "Fast and scalable layer four switching," in *Proc. ACM SIGCOMM*, Vancouver, Canada, pp. 191–202 (Aug. 1998).

[5] F. Baboescu, S. Singh, and G. Varghese, "Packet classification for core routers: Is there an alternative to CAMs?" in *Proc. IEEE INFOCOM'03*, San Francisco, California, vol. 1, pp. 53–63 (2003).

[6] G. Zhang and H. J. Chao, "Fast packet classification using field-level trie," in *Proc. IEEE GLOBECOM'03*, San Francisco, California, vol. 6, pp. 3201–3205 (Dec. 2003).

[7] P. Gupta and N. Mckeown, "Packet classification on multiple fields," in *Proc. ACM SIGCOMM'99*, Harvard University, vol. 29, no. 4, pp. 147–160 (Aug. 1999).

[8] T. V. Lakshman and D. Stiliadis, "High-speed policy-based packet forwarding using efficient multi-dimensional range matching," in *Proc. ACMSIGCOMM*, Vancouver, Canada, pp. 203–214 (Sep. 1998).

[9] J. van Lunteren and T. Engbersen, "Fast and scalable packet classification," *IEEE Journal on Selected Areas in Communications*, vol. 21, pp. 560–571 (May 2003).

[10] M. M. Buddhikot, S. Suri, and M. Waldvogel, “Space decomposition techniques for fast layer-4 switching,” in *Conf. Protocols for High Speed Networks*, Holmdel, New Jersey, vol. 66, no. 6, pp. 277–283 (Aug. 1999).

[11] S. Shenker and J. Wroclawski, “Network element service specification template”, RFC 2216 (Informational), Sept. 1997. [Online]. Available at: http://www.ietf.org/rfc/rfc2216.txt

[12] H. J. Lee, M. S. Kim, W. K. Hong, and G. H. Lee, “QoS parameters to network performance metrics mapping for SLA monitoring,” *KNOM Review*, vol. 5, no. 2 (Dec.2002).

[13] Z. Wang, “Internet QoS: Architectures and Mechanisms for Quality of Service”, Morgan Kaufmann, San Francisco, California, 2001.

[14] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, “An architecture for differentiated service”, RFC 2475 (Informational), Dec. 1998, updated by RFC 3260.

[15] Douglas E. Comer and David L. Stevens, “Internetworking with TCP/IP” Volume 2: Design, Implementation, and Internals, Prentice-Hall, Upper Saddle River, NJ, Third edition, 1999.

[16] H. Lim, C. Yim, E.E. Swartzlander Jr, “Priority tries for IP address lookup”, *IEEE Trans. on Computers* 59 (6) (2010) 784–794.

[17] H. Park, H. Hong, S. Kang, “An efficient IP address lookup algorithm based on a small balanced tree using entry reduction”, *IEEE Trans. On Computer Networks* 56 (1) (2012) 231–243.

[18] H. Lim, H. Kim, C. Yim, “IP address lookup for Internet routers using balanced binary search with prefix vector”, *IEEE Transactions on Communications* 57 (3) (2009) 618–621.

[19] S. Dharmapurikar, P. Krishamurthy, D.E. Taylor,” Longest prefix matching using bloom filters”, *IEEE/ACM Transactions on Networking* 14 (2) (2006) 397–409.

[20] M. Waldvogel, G. Varghese, J. Turner, B. Plattner, “Scalable high speed IP routing lookups”, in: Proc. ACM SIGCOMM, 1997, pp. 25–35.

[21] N. Yazdani, P.S. Min, “Fast and scalable schemes for the IP address lookup problem”, in: Proc. IEEE HPSR, September 2000, pp.83–92.

[22] David E. Taylor, Survey and taxonomy of packet classification techniques, ACM Computing Surveys 37 (3) (2005) 238–275.

[23] C.-N. Lu, C.-Y. Huang, Y.-D. Lin, Y.-C. Lai, Session level flow classification by packet size distribution and session grouping, *IEEE Transactions on Computer Networks* 56 (1) (2012) 260–272.

[24] F. Baboescu, P. Warkhede, S. Suri, G. Varghese, Fast packet classification for two-dimensional conflict-free filters, *IEEE Transactions on Computer Networks* 50 (11) (2006) 1831–1842.

[25] K. Lakshminarayanan, A. Rangarajan, S. Venkatachary, Algorithms for advanced packet classification with ternary CAMs, in: *Proc. ACM SIGCOMM*, 2005, pp. 193–204.

[26] M. Faezipour, M. Nourani, Wire-speed TCAM-based architectures for multimatch packet classification, *IEEE Transactions on Computers* 58 (1) (2009) 5–17.

[27] D.E. Taylor, J.S. Turner, Classbench: a packet classification benchmark, *IEEE/ACM Transactions on Networking* 15 (3) (2007) 499–511.