IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 27, 2020, accepted February 10, 2020, date of publication February 14, 2020, date of current version March 2, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2973932

Effective Routing Technique: Augmenting Data Center Switch Fabric Performance

MUHAMMAD KASHIF KHATTAK[1,2], YAZHE TANG[1], HAMZA FAHIM[1], EID REHMAN[3], AND MUHAMMAD FARAN MAJEED[4]

[1]School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049, China
[2]Science and Technology on Communication Networks Laboratory, Shijiazhuang 050081, China
[3]Department of Software Engineering, Foundation University Islamabad (Rawalpindi Campus), Rawalpindi 44000, Pakistan
[4]Department of Computer Science, Shaheed Benazir Bhutto University, Sheringal 18000, Pakistan

Corresponding authors: Muhammad Kashif Khattak (kashif@stu.xjtu.edu.cn) and Yazhe Tang (yztang@mail.xjtu.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB0801703, in part by the NSFC under Grant 61672425, and in part by the Open Research Project of Science and Technology on Communication Networks Laboratory under Grant SXX18641X024.

ABSTRACT Today, data center networks (DCNs) are built using multi-tier architecture. These large-scale networks face many challenges, such as security, delay, low throughput, loops, link oversubscription, TCP Incast and Outcast, etc. In this paper, a TCAM (Ternary Content Addressable Memory) based routing technique is proposed, augmenting the routing capabilities of multi-tier architectures in large scale networks. The routing complexities in these architectures are rectified and improved by implementing an additional TCAM based routing table in Leaf/ToR switches for a specific number of compute nodes in particular Pods, and it is scalable to whole datacenter nodes. To test the model, we implemented two prototype models: one depicting our proposed TCAM based switch and the other is a typical Top-of-the-Rack (ToR) switch and compared the performance of the proposed model and if any overhead introduced in it. The preliminary results show that our TCAM based routing table technique is fast and it forwards the network packets at line-rate, does not introduce considerable latency, on-chip resources power consumptions is less than 3%, and helps to solve or mitigate the above critical problems that are present in the current large DC's three-tier architecture, especially in Top of the Rack and aggregation layers switches.

INDEX TERMS Data center networks, TCAM tables, top of the rack switches, link oversubscription, TCP incast and outcast, loops.

I. INTRODUCTION

Datacenter (DC) is a large, dedicated cluster of compute servers and switching devices, that is built to deliver online services to their customers. Today, not only large online service providers like Google, Amazon, Facebook and Alibaba, etc. are building huge data center networks, but large universities and private enterprises are also shifting towards deploying hundreds to thousands of compute nodes and routing devices to support their extended services like web servers, E-mail and online libraries etc. For this, these data centers need high throughput and low latency packet processing, by mitigating the problems which cause them to abandon these goals.

In recent years, the data center architecture has changed from Three-tier, Fat-tree architecture to Leaf-Spine architecture. The leaf-spine architecture has several advantages over previous three-tier clos architecture, as it gives scalability and a fixed number of hop count for the network packets to travel from one compute node to another compute node in the same data center. Today, the east-west traffic ratio has increased significantly in datacenter networks due to the extensive use of network virtualization technologies like VxLAN and Software-Defined Networking (SDN) Controllers. These components relay a huge amount of traffic inside the server racks in a DC [1]. We know, as the technologies evolved, there are certain anomalies and side effects have also arisen in the system. Today, these large datacenter networks face the challenges of Performance, Delay, Link Oversubscription, Loops, TCP Incast/Outcast, etc.

To overcome these problems, few new technologies are used, e.g. Spanning Tree Protocol (STP) is used to overcome the problem of infinite loops by shutting down some links on the switches. Large datacenters like Google, which is

The associate editor coordinating the review of this manuscript and approving it for publication was Matti Hämäläinen.

well-known for implementing Leaf-spine architecture, deals with the oversubscription problem by adding spine switches in the network, so that traffic pressure can be divided and reduced on particular oversubscribed links. Facebook's principal network building block of data centers is a *Pod*, in which there are a fixed number of spine and leaf switches. It eliminates the oversubscription problem by ensuring that each server in the pod is equally and redundantly connected to the network. Active queue management techniques can solve the Outcast problem but the round trip time (RTT) bias occurs and can harm the throughput fairness [2]. Qin *et al.* in [3] proposed a solution for the TCP Incast and Outcast problem by proposing new protocols named TCP Congestion Windows Replacement (TCP-CWR) and TCP Acknowledgment Change Rate TCP-ACR).

All these solutions solve or at least mitigate the above problems, but at the expense of some compromises. For example, to overcome oversubscription, adding another hardware switch in the network every time, costs hardware price, configurations and labor charges, etc. STP can solve the *Loop* problem at the expense of shutting down some useful links. Which can cause problems in case of link failures and the switching device may need to use these redundant links closed by STP. It still suffers from loop-free topology for VLANs [4]. Equal-Cost Multi-Path (ECMP) mitigates this issue by splitting up flows on the available equal-cost paths. But the ECMP works well for short flows. If the network is dominated by the long flows, then it can lead the network congestion on some links [5].

In this paper, we proposed a TCAM based solution to overcome or mitigate these issues in large data centers. TCAM memory is used by most of the switch vendors e.g. Cisco, for Layer 2 and Layer 3 routing for fast lookup and Quality-of-Service (QoS). TCAM memory is useful for fast lookup because it traverses the whole memory in one cycle. We propose that an additional TCAM table may be implemented in ToR/leaf switches, containing flow entries to connect the servers linked to the same leaf/ToR switch and the intra-Pod and inter-Pod servers. This solution also proposes to make groups of certain pods in the data center according to the requirements of the network administrator, to implement these TCAM flow tables more efficiently, and it can be scaled to the entire DC. The solution may help to avoid many large DCs problems, which in turn, improves the performance in terms of overall throughput and latency in the east-west traffic of a data center. The preliminary simulation results, implemented on hardware ONetSwitch45 [6], [7] FPGA board and tested on state of the art Spirent TestCenter (TC) machine, show that implementing additional TCAM based routing table does not degrade the performance of switch in terms of latency and gives the line rate throughput. The on-chip resources power consumptions of both the proposed and the simple ToR switch has been evaluated and neither of the overhead exceeds 3%.

The structure is organized as follows: Section-2 describes Background and Motivation, Section 3 draws the Design., Section-4 demonstrates the implementation and evaluation. Finally, Section 5 concludes the paper and points out further work.

II. BACKGROUND AND MOTIVATION

A. EVOLUTION OF DATACENTER NETWORKS

The data centers are big houses of computational and storage servers, that serve a large number of popular services on Internet such as search engines (e.g. Baidu, Bing), E-commerce (e.g. Taobao), web-based E-mail (e.g. QQ, Gmail), social networking (e.g. Weibo and twitter) and video sharing (e.g. Youku). They may deal with a variety of services that demand heavy infrastructure which requires the service providers to procure, build and maintain large DC Networks [8].

Over the last decade, many new design architectures have been proposed for data center networks to improve the computational and storage power of the data centers, such as [9]–[12]. The closely related work to our model is a patent [4], in which they proposed to implement one or more tables in leaf node to forward the traffic locally without passing it to the upper-level spine switch. The leaf-spine architecture given in Fig. 1, got popularity because of its flexibility to adapt the required bandwidth. It has many advantages over the two-tier architectures. And currently, large well-known internet giants have implemented this architecture. But, as the size of the data center increases, new technologies replace the old ones. Consequently, new anomalies and problems also arise, which are briefly described in the next subsection.

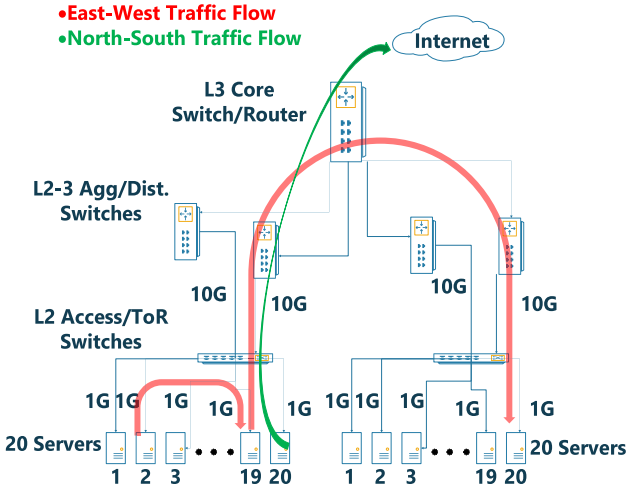


FIGURE 1. Leaf-Spine architecture: Depicting North-South and East-West traffic.

B. PERFORMANCE OVERHEAD IN EXISTING ARCHITECTURES

The challenge in building large DCs is that it must be developed as one logical and high-performance unit. As Fig 1 shows, the data center network servers send traffic inside and outside of DC. These data packets are switched through

the switching boxes deployed in a hierarchy, known as Core, Spine and Leaf switches. There are also redundant links between these components in order to keep the flow of traffic continue in case of a link failure. Sometimes we feel that these intra-DC packets should not visit unnecessary switches in between the sender and receiver. Therefore, eliminating such hops helps in augmenting the performance of the overall operations, as these intra-DC packets have dependencies. These redundant links also cause another problem if some packets revolve around a certain number of nodes for infinite time, this is called *loop*. To overcome the loop problem, STP is introduced, which in turn shuts down some links that cause a loop in the network.

But, STP has some limitations, and it cannot completely prevent the loops in a large number of VLANs. Also, these redundant shut-down links could be utilized to route traffic in case of another link failure. There could be a problem of link *Oversubscription*, in which a link is asked to handle more traffic than it can. For example, as shown in Fig. 1, if there are 20 1G links sending packet towards spine switch, but the link between the leaf and spine is only 10G, then this link will be oversubscribed at the rate of 2:1. To solve this problem, another spine switch is added to the network. Hence, the network size increases as well. But this is an expensive solution. Other problems that such large data centers face is TCP Incast and Outcast. In which, many incoming packets compete for one outgoing port. This might be due to different Round Trip Time or many senders send traffic to the same receiver.

As a result, there must be packets-drop or performance degradation in terms of throughput and delay. Most data centers use Commercial Off-the-Shelf (COTS) ToR switches to save the high development cost. Which are, in turn, usually unable to queue and process traffic in these kinds of abnormal conditions described above, and eventually leads to discarding many packets. For this reason, this study proposes a novel technique to improve the intra-DC traffic performance.

III. DESIGN

The switches and routers are the packet switching boxes, consist of software and hardware components. The packets are received via the ingress ports. Then it is processed by the packet processing module, implemented using some high-level packet processing language and tools, and then it is buffered and sent through egress ports. The packet processing module consists of three main steps: Parsers, Match and Actions, and Deparser. The parsing module parses the packet headers and extracts the desired header fields. Then these header fields are matched as a Key against flow entries in flow tables. If the match is found, then the action is performed accordingly. The match and action tables are high-performance lookup tables, usually implemented on Ternary Content Addressable Memory (TCAM).

In this section, we will discuss the relevant devices involved in our proposed model, the challenges associated with them, and the overall design of our proposed switch model.

A. TCAM CAPACITY

TCAM has become the de facto standard to implement high-speed packet classification schemes. Most of the packet classification devices delivered were TCAM based in 2003 [13]. TCAM memory is used for fast lookup because it traverses the whole memory in one operation. Therefore, the time it takes to find the match for a given packet is *O(1)*. But this is an expensive memory, that also consumes high power and ASIC resources. Therefore, the switches and routers are usually equipped with limited sized TCAM memory. Some studies showed that the TCAM memory available in most of the switches can store up to 2K to 4K flow rules [14] and they propose a Virtual TCAM that can store up to 40K rules, or even much larger flow tables in emerging switches [15]. In [16], the 5406zl switch can support 1500 OpenFlow rules, whereas the switch can support up to 64000 rules for Ethernet switching. The huge difference is because that the OpenFlow rules are stored in TCAM memory which is limited in size, while the Ethernet forwarding rules are stored on standard memory. Authors in [17] mentioned that, at that time, the maximum TCAM capacity in a network switch is 36Mb and it can accommodate up to 64K to 128K flow rules. High capacity switches like Cisco Catalyst 4506-E switches with Ample TCAM resources support up to 384,000 flow entries for fast routing and switching [18].

If we look at the current numbers of compute nodes connected in a large datacenter, we could see that by proper optimization and the TCAM memory available in today's high-end switches, it is quite feasible to implement an extra TCAM table for our proposed solution, and it is scalable too. Also, an OpenFlow [19] rule supported 10 header fields (248bits), whereas, current version of OpenFlow *OFv1.5* supports 44 fields, totaling more large number of bits (773bits). While an Ethernet forwarding-flow descriptor is only 60b (48b MAC + 12b VLAN ID or 20bits label), occupying clearly more little memory resources. Therefore, our proposed scheme occupies comparatively low memory. Current TCAM memory support more than 133M searches in one second for 144bit key [13].

B. 20GBPS TOP OF THE RACK SWITCHES

To understand how our model can be implemented in these ToR switches, first we need to understand that, how the ToR switches in a large DCs are placed and connected, and how it works. The switches that are connected directly to compute nodes are called "ToR/Edge/Leaf/Access" switches. The ToR switch makes the first connection between the compute servers in the network, and also to the more reliable backbone network within the building, out to the internet. Each server might have a single connection to a single ToR/Access switch, but everything above the ToR switch should be designed with redundant links. The ToR switches usually have the following key features: **high port density and low per-port costs**.

TABLE 1. Ethernet switch models with different numbers of ports and capacities.

	ToR 1Gig	ToR 10Gig	EoR
Giga bits Eth Ports	48	0	0
10Giga bits Eth Ports	4	24	128
Power 'W'	200	200	11,500
Size (rack unit)	1	1	33

High port density is critical because of the costs associated with managing them. Generally available Ethernet ToR switches models with the different numbers of ports and capacities are given in Table 1. As the number of switches increases, the complexities to manage them and the links between them also increases. Ultimately, the associated costs and potential network downtime become the critical factors that can lead to degrade the network performance significantly.

The development of 10G ToR switches has improved the performance significantly and decreased the cost per port. 10G Ethernet switches have laid the foundation for modern DCs applications while also opened the way for scalability. 10G ToR switches process Layer-2 (L2) and Layer-3 (L3) traffic, DC bridging and Fiber Channel Over Ethernet (FCoE) for an entire rack of compute nodes, and it consumes little power, gives more scalability and reduces cabling complexity. As, a 10G Ethernet leaf switch is just one hop away from another leaf switch, avoiding to jump up and down in the tree. Consequently, it improves latency and minimizes the bottlenecks. Network programmers can quickly add new leaf/ToR switches to fulfill data center growth, without re-architecting the whole network. In small networks of a few hundred users, ToR/Access switches can be connected directly to Routers/Core switches. However, in larger DCNs, another layer of switching, called the aggregation/distribution layer is built between the ToR/Access and the Core layer, which aggregates the lower-layer switches.

The network switching devices (such as switches and routers), and servers are all mounted on large racks. A typical rack is 2.0m high, 1.0m deep and 0.6m wide. The unloaded weight of these racks is usually 170 kg, and it supports a load of 900Kg[20]. The 2m high rack is divided into 42 vertical rack units (denoted as RU, or simply U). The height of each RU is 44.45mm (1.75in). The types of equipment mounted on these racks usually occupy one or more RU. In high capacity virtualized data centers, ToR switches may have all 10Gbps ports, handling 40Gbps speed today. This is an expensive but useful high-performance deployment. Therefore, we also implemented and tested the proposed model having 4×10GE ports for overall 40Gbps high-performance throughput.

The top-of-rack switches architectures are designed a little differently from Edge switches, often supporting 2-4×10Gbps uplink ports, 48×1Gbps server ports, more than one power supplies, and a more high-speed inter-switch stacking system than general user edge switches. The general architecture of the ToR switch is given in Fig. 2 [20]. The Power Supply Unit (PSU) and the FANs are isolated from the circuit board.

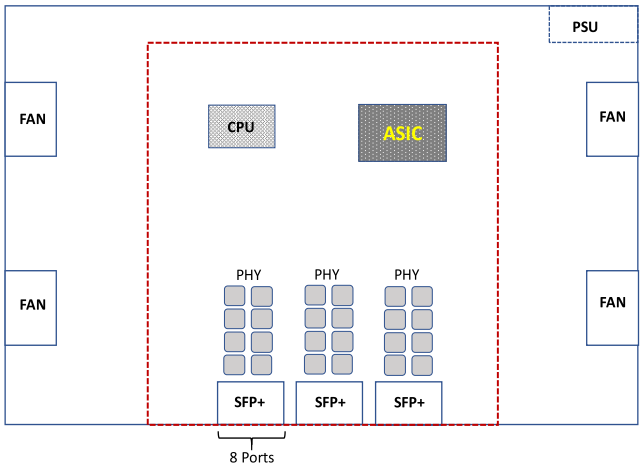


FIGURE 2. The internal architecture of a ToR switch.

C. WORKING OF THE PROPOSED TCAMSWITCH AND SIMPLE TOR (SIMPLESWITCH)

The protocols usually parsed/processed by these switches can be similar, but their processing could be different. For example, Multiprotocol Label Switching (MPLS) routing technique, that is used to route packets from one node to another node, based on short labels instead of long lookup flows, and it also helps to ease the software-defined based architectures, cloud networks, and data centers. MPLS creates tunnels that maps the traffic on specific queues to route traffic from various Edge/ToR switches on different levels. The proposed TCAM based model (TCAMSwitch) and the SimpleSwitch are expecting the packets with the same headers stack as shown in Fig. 3, so that the performance evaluation could be made fair. They are only different with the logic implemented on the TCAM based switch to handle the local east-west traffic more efficiently. Packet headers depicted in Fig. 3 are commonly used in most of the commercial routers with different combinations. The packets are assumed carrying MPLS headers and the IPv4 or IPv6 headers with TCP/UDP headers appended.

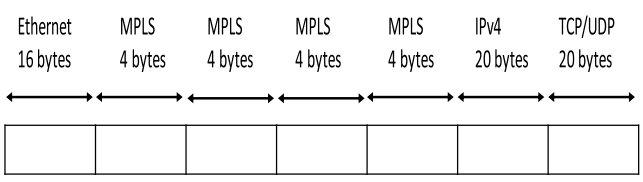


FIGURE 3. A realistic packet headers sequence in commercial edge routers.

In our TCAMSwitch, the data packets are forwarded based on the labels assigned to them or based on the

104bits Five_tuple key comprising source and destination IP addresses, source and destination ports and protocol. As the packets are received, the parsing module parses the headers in ParsingEngine, expecting Ethernet header at offset zero, and extracts the source and destination MAC addresses. The next header is recognized by the next header field 'type'. Similarly, all the packet headers are traversed and the required fields are extracted. There are two interfaces as shown in Fig 4, the first is to LookupEngineI, that gives the resultant value based on the MPLS label, and the second is to the LookupEngineII, that matches the Five_tuple key and returns the resultant output port. Packets are classified on the basis of destination by finding MPLS header in the packet. If the destination is a local server in the datacenter, by looking at the MPLS label inserted in the packet, then it is forwarded to the next router based on the assigned label, matching the Key against a value in LookupEngineI. If the destination is a remote host, then the packet is sent to LookupEngineII and is matched against its right output port value against the Five_tuple key, and the packet is forwarded to it.

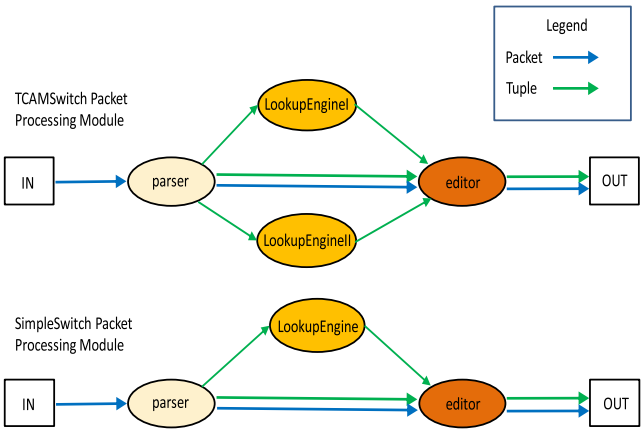


FIGURE 4. Simple DOT graphs of packet and tuple flow of our TCAMSwitch and the SimpleSwitch in packet processing module.

The SimpleSwitch is an Edge router, which is expecting the Ethernet, MPLS, IPv4/IPv6, and TCP/UDP headers. The ParsingEngine parses the packet headers and extracts the required fields as a *Five_tuple* key, and the key is sent to the *LookupEngine* for match and action, and the result is obtained based on the key. The *editor* engine could modify/update the label and source/destination fields of packet headers.

The difference between the working of the two switches is that the SimpleSwitch does not classify packets between the two TCAM tables. Instead, it implements one large table as shown in Fig. 4. Traversing such a large TCAM table in one cycle is a hard job both in terms of using processing resources and delay. This also encourages to move the north-south Ethernet forwarding flow entries (that is less than 25% of the total traffic [21]) to a standard memory instead of TCAM, or implement an additional TCAM table for it. Also, we believe that in an SDN environment, the SDN controller need to update the flow tables too frequently, but the TCAM memory is not suitable for updating records. Secondly, it will leave space in TCAM memory. Consequently, these saved fabric resources could be used for storing more east-west flow entries or it could be used to implement other user logic. Hence, in either case, instead of creating one large TCAM table, it is proposed to implement an extra TCAM flow table for the intra-DC traffic, depending upon the inter-dependent traffic in the DC. This will help to give high match and action performance, high speed for the inter-dependent intra-DC traffic, and security to the east-west traffic which is the larger proportion of the overall datacenter traffic generated by servers on racks.

Apparently it looks like a multi table pipeline that is supported by today state-of-the-art generic switching chips and somehow implemented in previous work, ServerSwitch [22]. But the use and application of these works are quite different. For example, in [22], [23], the authors focused on utilizing general low-price CPU with switching ASIC to perform data plane functions (besides control plane functionalities). For this, they offload some traffic to CPU for processing. However, in the predecessor, they did not mention that what kind of traffic will they process by the commodity CPU and how they will decide to process it there. These works are closer to our previous work [1]. While this model does not offload any traffic to CPU or any other switching chip like them. Instead we focus on implementing an additional TCAM table for processing east-west traffic to provide speed, reducing hop count, QoS and separating it from the north-south traffic to protect it from the compromised north-south packets. They also used three types of forwarding schemes, i.e. Destination based, Label based (tag or MPLS based) and Source routing. For our prototyping, we also implemented Five-tuple based lookup table and MPLS based forwarding. But, in our work, we mentioned that how we implement MPLS based forwarding in a medium-large DC and connect servers in different Pods and why we do so, in next subsection 3.4.

D. TCAM TABLE IMPLEMENTATION BASED ON A MEDIUM-LARGE DATACENTER

The large DC networks have hundreds of thousands of data servers. These servers are arranged in racks and connected via a hierarchy of switches and routers with other servers inside DC and to the outside world via the internet. Servers racks have Leaf or Top of Rack (ToR) switches or End of Row (EoR) switches, which are then connected to the higher level of more high capacity Aggregation and Core switches. Several such racks of servers and switches form Pods, as shown in Fig. 5.

For example, in a medium-large DC, where ten thousand servers are deployed for different online services. In a DC, each standard cabinet has 44U (rack unit) of vertical space. Where 1U is used for a patch panel and 1U for ToR/Leaf switch. The remaining 42U can accommodate 42 1U servers. Whereas, 1U server can typically hold four 10TB Large Form Factor (LFF) or eight 2TB Small Form Factor (SSF)

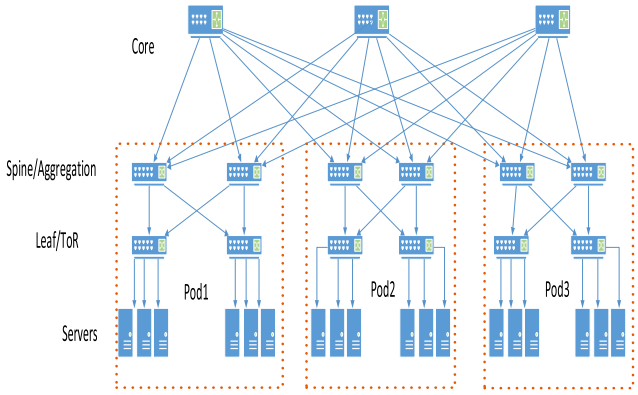


FIGURE 5. Three-tier data center architecture depicting Leaf/ToR-switches and servers' arrangement in Pods.

drives, which is standard storage capacity for modern DC servers. Therefore, we assume it a medium-to-large DC. Hence, 24 racks can hold ∼1000 servers. If one Pod consists of 24 racks, then ten thousand servers can make 10 Pods in a data center. Hence, we have 24 and 240 Leaf/ToR switches in a Pod and DC, respectively. Thus, for this, we can implement an extra TCAM table in every Leaf/ToR switch. This TCAM table will store flow entries for particular servers in specific Pods. For example, we create a TCAM table consisting of 2K (two thousand) flow entries. These flow entries will enable us to connect servers on Pod_k to the servers on $Pod_{(k,k+1)}$, where 'k' is the Pod number, and so on.

The studies focused on data center networks traffic characteristics reveal that generally, the DCNs switches handle 1000 to 10,000 active flows at a 1s time interval [21]. Recall that a rack of 42 servers and each server generates 25 flows, then the rack will generate ∼1000 flows per second. If the flow entry timeout of a flow table is 10s, then the ToR switch of the rack would have 10,000 flows at a time. This is a huge number. It can be reduced to 2000 flows or 1000 flows by optimizing the flow entry timeout to 2s or 1s respectively. The study like [24] shows that the 50% of flows in a data center remain active for less than 1s and 80% of flows remain active for less than 10s. Our model is based on the Ethernet forwarding scheme, which is per destination rather than per-flow (e.g. OpenFlow). Hence, each host connected will typically require one entry per host, also occupying low memory as compared to OpenFlow rules, as we mentioned in subsection 3.1. Hence, we can connect hosts as demonstrated by the following steps:

How the extra TCAM table is implemented connecting Pods in DC with 1s flow entry timeout:

a. If a Pod has 24 racks, 24 ToR switches, and 1000 servers,
   i. Each ToR switch adds a TCAM table of 2000 flow entries, half for Pod_k and half for Pod_{k+1}, where 'k' is the Pod number.
   ii. Each ToR switch of Pod_{k+1} store 2000 flow entries, half for Pod_{k+1}, and half for $Pod_{((k+1)+1)}$, i.e. Pod_{k+2}.
   iii. Repeat above steps until Pod_{n-1}, Pod_n, where 'n' is the last Pod in the DC.
   iv. Each ToR switch of Pod_n adds TCAM table of 2000 flow entries, half for Pod_n and half for Pod_k.
b. We have connected all the Pods and racks in a DC in a circle using an extra TCAM table.

Keeping in view that we are implementing an extra TCAM table of 2000 flow entries only, this way, we can connect two Pods in a DC. But as discussed in section 3.1, by optimizing TCAM memory, or implementing large TCAM memory, as found in high-end, high capacity switches, more and more flow entries can be added to the TCAM table. As a result, this technique could be scaled to connect the compute nodes in the whole DC. Consequently, it improves the performance of the whole network and has the ability to solve many problems faced by DC, as mentioned in the next subsection 3.5. We conclude that the TCAM based routing technique is fully scalable without the risk of degradation in performance.

E. ADVANTAGES OF TCAM BASED MODEL

1) PROS

The advantages of this TCAMSwitch model are more diverse and they are all separate research problems on their own. Therefore, here we hope that many future research works will use this model and will present more solid results on how to solve these problems. For example, this TCAMSwitch model helps to increase the speed of the packet processing by forwarding it using short lookup key and fast TCAM memory for east-west/intra-DC traffic, avoiding processing long lookup keys such as Five_tuple, for all the packets. This model also helps to reduce one hop for the inter-rack traffic between the servers in the same Pod, as the packets would not need to go the upper layer of high-level core switches. Instead, it could be routed via shortest programmable path. In turn, this also reduces the delay and increase the performance of the traffic that is dependent and stored on different servers in a DC. Besides, this model will also help in avoiding Loops in the network, as the traffic intended for those particular nodes will always follow the same route resulted by the TCAM table and would not need to compete for the alternative links. In case of link failure, an SDN controller would intervene. This model could also help in avoiding oversubscription on links, as the traffic could be routed using these specialized TCAM tables such that any link in a network is utilized according to its capacity. It is done by dividing flows on alternative links in the TCAM table in such a manner. Ultimately, TCP Incast and TCP Outcast problems are avoided too, as the traffic from many input ports could be customized to be forwarded via many output ports. Because, using these additional TCAM tables, network traffic can be routed, customized and controlled more perfectly on switch output ports.

This approach would also help in connecting more servers on the next Pod, i.e. $Pod_{(k+1)}$ and so on. Hence, it could be scaled to address all the problems mentioned above for the inter-pod traffic and the entire DC. This technique would also

help in security and QoS of data servers, as the traffic would flow through specialized paths. Hence, the north-south traffic (north-south traffic could be malicious) would not cause the risk for the east-west traffic. QoS can be improved by routing inter-dependent traffic in customized ways so that data could be retrieved in a more specialized way, that is stored on different servers, in other particular Pods. Hence, this solution can lay the foundation for many future researches in this field.

2) CONS

The challenge of this approach could be the TCAM table update. As the flow entries in the flow tables would need to be changed over time. The TCAM tables do not perform well in updating records, because the TCAM is a specialized memory built for fast lookup, as it traverses the whole memory in one cycle. But it is not suitable for updating records, because an update would also need to look for the priority or addresses of the flow entries in the TCAM memory, which may require several changes of positions and hence several cycles. This is also one of the reasons why we encourage to move the other Ethernet addresses to standard memories instead of TCAM, in an SDN environment. Because the SDN controllers consistently need to update the flow tables. These optimizations will also create space in lookup tables resources of switching fabric for our model. Another challenge of this approach is it consumes power and ASIC resources. But our simulation results show that there is no considerable degradation of implementing extra TCAM table in terms of performance and power consumption (Section 4.2). Also, this can be managed as discussed in the previous section, as now the switches have enough TCAM memory, so that we can implement our TCAM table inexpensively.

Hence, the overall advantages of this TCAM based routing technique is very convincing, and it can be used in more other specialized ways in future research works to increase the performance of these large modern data centers.

IV. IMPLEMENTATION AND EVALUATION

A. SIMULATION OF HARDWARE PLATFORM

ONetSwitch45 [6] has been used to simulate and test this model. The ONetSwitch45 FPGA board consists of Xilinx Zynq 7045 FPGA SoC, XC7Z045 as the main silicon platform. SD Card is the primary configuration interface. The processing system has Dual ARM Cortex-A9@800MHz processor. The Programmable Logic is based on Xilinx-7 series Kintex-7 FPGA. It has 350K Logic Cells, ∼5.2M ASIC gates. This board has 4xGE RJ45 Ethernet ports and 4×10GE SFP+ Ethernet ports. The FPGA fabric can give 40Gbps of line rate either as a single 1×40g or 4×10G output ports. The ONetSwitch45 board has been chosen for this evaluation because of its rich and less costly resources. It can be easily configured in software and hardware. These features make it highly suitable for research and development. Studies like [25], [26] have proposed the ONetSwitch family for network devices development and prototyping. It is more suitable then already worldwide adopted NetFPGA [27].

To evaluate the design, we implemented two switch cores on our FPGA board, one as our proposed TCAMSwitch and other for evaluation comparison as a simple ToR switch (SimpleSwitch). Both depict a typical ToR switch, expecting Ethernet header as the start of the packet, followed by the IPv4 header or MPLS header, which is maybe then followed by another MPLS header or IPv4 or IPv6 header, and so on, which is then followed by TCP/UDP header as shown in Fig. 6.

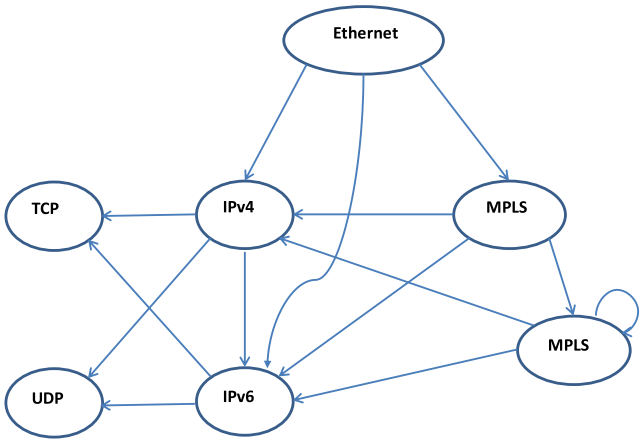


FIGURE 6. Parse Graph: depicting a typical ToR switch.

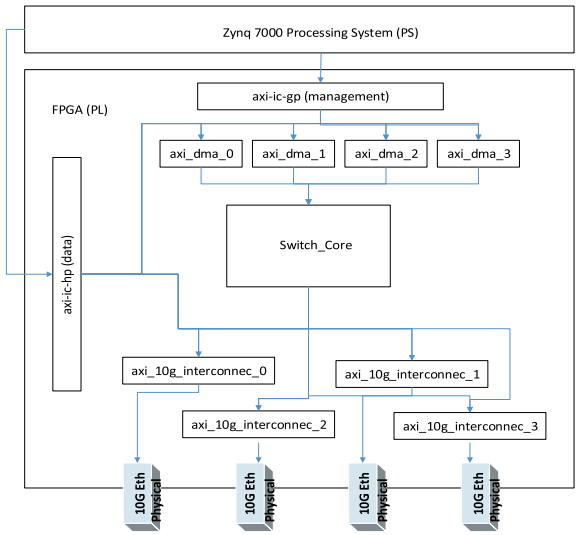


FIGURE 7. Hardware switch architecture connecting IP (Intellectual Property) cores.

Hardware Description Language Verilog is used to design digital logic circuits to implement hardware switching logics and added as IP cores in Block Design using the Xilinx Vivado 2017.4. The architecture is shown in Fig. 7. The architecture works in two parts, the Processing System (PS) part is loaded with Linux OS. The PS controls the Programmable Logic (PL) part. Different IP cores are implemented on the PL as shown in Fig. 7, to receive, process,

store and transmit packets between ingress and egress ports. Four internal logical ports 1-4 are created that are linked to physical Ethernet ports. The high-level domain-specific language Xilinx SDNet specification (PX) is used to program packet processing logics. To generate the Verilog code and Switch IP block, we used Xilinx SDNet compiler. Eventually, Vivado 2017.4 is used to implement it on target FPGA.

Next, the Xilinx Software Development Kit XSDK 2017.4 is used to generate First Stage Boot Loader (FSBL) and boot.bin files. We customized the FSBL and uboot files according to our chip architecture. To describe the physical hardware in the system, a customized devicetree.dtb is also needed. It is used to pass low-level hardware information from the first stage boot loader to the Kernel.

Finally, bin, dtb, uboot and init.sh files are copied to SD Card and it is inserted into its proper slot on the FPGA board to configure and initialize the switch. Also, we copy our TCAM API file 'app' and flow entries text files lookup.tbl on SD Card, to download the flow entries into the TCAM flow tables.

B. EVALUATIONS

To evaluate the performance of our TCAMSwitch, we use state of the art Spirent TestCenter (TC) machine. Spirent TC is used by ISPs, NEMs, Enterprise Networks and Academia to test, measure and validate their devices. Our Spirent TC supports two 10G SFP+ ports per slot. We reserved two 10G ports on slot 2 (2/1, 2/2) and connected it to 10G Ethernet ports on our FPGA board using fiber optic cable, for transmitting packets to/from the device under test (DUT). Stream blocks are configured in conformance with the data plane configurations of the switch cores. UDP headers are appended to the Ethernet frames in the Spirent TC.

We use four different frame sizes for this evaluation i.e. 64, 500, 1000 and 1500 Bytes, that lie between the undersize frame threshold 64B and jumbo frame threshold 1518B according to IEEE Ethernet standard 802.3 frames sizes. The gap between the packets is 20B. The inter-packet gap also causes a decrease in the throughput given by the device. So, naturally, the customers cannot get the hundred percent throughput. The FPGA processing fabric can process the packets at 40Gbps, and we can get the throughput as a whole or it can be divided on 4×10G output ports. So, we set the topology to send and receive the packets from one 10G port to another 10G port.

Fig. 8 shows that our proposed TCAMSwitch forwards the packets at line-rate and there is no considerable performance degradation in terms of throughput. The slight degradation is because of the inter-packet gaps. The results show that the performance of the packets forwarded through the TCAMSwitch is quite close to that of the SimpleSwitch. This shows that implementing an additional large TCAM lookup table for this purpose has no performance overhead and forwards the packets at line-rate. Naturally, we cannot exceed the performance more than that, and achieving performance close to 10G verifies that the implemented switch can catch up the

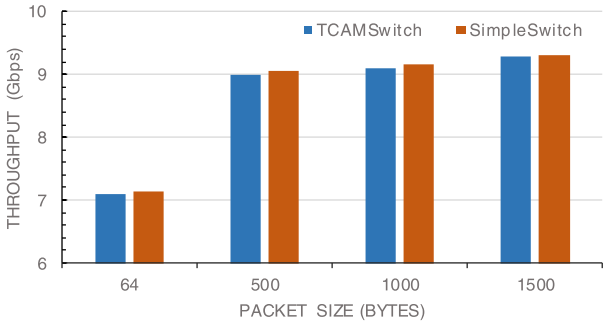


FIGURE 8. Performance comparison of our TCAMSwitch model and a simple ToR (SimpleSwitch).

speed of the physical port, that is the line-rate. Most of the state-of-the-art architectures implemented on NetFPGA such as [28]–[30], ONetSwitch [1] and DPPSN [31] show the same performance outcomes.

Fig. 9 also shows the same trend, in which the latency outcomes of both the switches do not differ considerably. The lookup performance is the same for all the tables implemented on TCAM. That is why it is highly suitable to implement routing tables in TCAM without the risk of performance degradation and to ensure QoS for the high proportion of intra-DC traffic. Our proposed model classifies the packets at the very early stage of packet parsing engine and take similar time to process and forward all the packets. The latency outcomes of our proposed model are quite close to any state-of-the-art FPGA implementation such as [32].

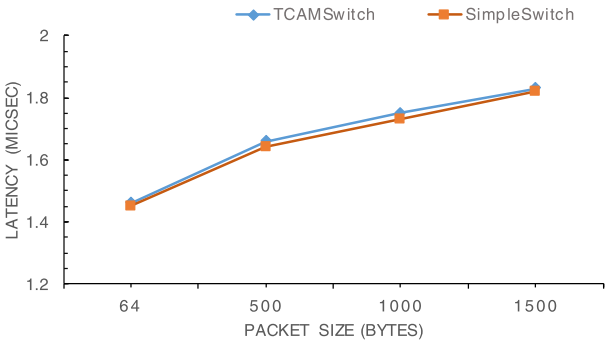


FIGURE 9. Latency comparison of our TCAMSwitch model and SimpleSwitch.

As the SimpleSwitch is implemented with only 256 flow rules, that's why the performance is close to the TCAMSwitch. But, in large data center, there are many more flow rules and switch contains large flow tables, thus the performance of our TCAM model would be lot better than SimpleSwitch because of its split and small TCAM lookup tables.

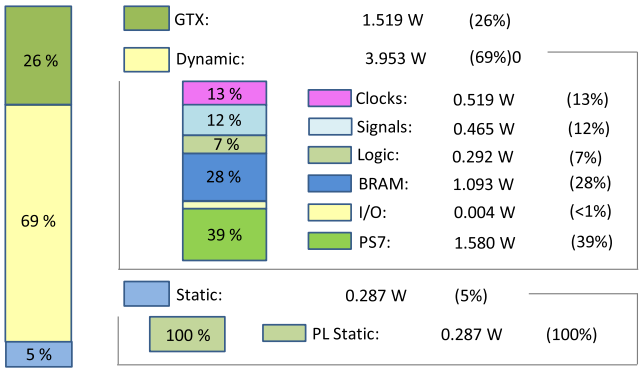
The post-implementation resources consumption overhead of SimpleSwitch (simple ToR) and our TCAMSwitch are given in Table 2. The disparity in the LUTs (lookup tables) resources consumption is because we implemented an additional Lookup table consisting of 1500 flow entries on TCAM

TABLE 2. Logic cells resources overhead comparison. (*SS = SimpleSwitch; TS = TCAMSwitch*).

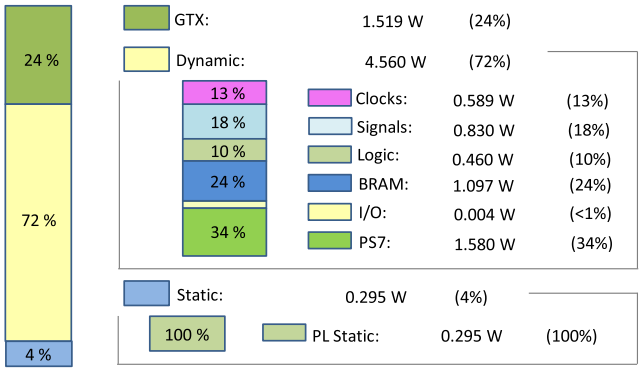
Resources	Available	Utilization		Utilization %	
		SS	*TS*	*SS*	*TS*
LUT	218600	80251	195324	36.71	89.35
LUTRAM	70400	7540	33583	10.71	47.70
FF	437200	96853	160324	22.15	36.67
BRAM	545	234.5	237.5	43.03	43.58

TABLE 3. Summary of the post-implementation on-chip power consumptions of simpleswitch and TCAMSwitch.

	SimpleSwitch	TCAMSwitch
Total On-Chip Power	5.759 W	6.374 W
Junction Temperature	35.8 °C	37.0 °C
Thermal Margin	49.2 °C (25.3 W)	48.0 °C (24.7 W)
Effective ΘJA	1.9 °C/W	1.9 °C/W



(a). SimpleSwitch On-Chip power consumption of resources.



(b). TCAMSwitch On-Chip power consumption of resources

FIGURE 10. (a). SimpleSwitch On-Chip power consumption of resources. (b). TCAMSwitch On-Chip power consumption of resources.

memory for our effective routing model, whilst there are only 256 flow entries in the SimpleSwitch Ethernet forwarding lookup table. Unfortunately, comparatively, FPGAs come up with low memory and poorly simulate the TCAM. Most of the resources are consumed by just match stages of the packet processing pipeline, which otherwise could be used for other user logics.

Our FPGA switch fabric has limited resources, therefore we added 1500 instead of 2000 flow entries. We conclude that, as the modern switching chips in the latest switches and routers are more powerful and have high logic resources, therefore implementing such an additional TCAM table for effective routing does not affect the chip resources limitation and performance of the switches considerably. The resources are compared in terms of LUTs, LUTRAM, FFs (Flip Flops) and DRAM. As compared to other state-of-the-art hardware switch platforms NetFPGA 1G and 10G, the resources consumptions as measured in [29], our model consumes less resources.

The power consumption overhead can be calculated from Fig. 10 (a, b). Generally, 1Mb of TCAM chip consumes 15 to 30 Watts of power[13], and number of ternary rules stored on TCAM is directly proportional to power consumption. Still, it can be seen that the overall difference between the power consumption of the on-chip resources such as Core Dynamic, I/O and Device Static of *SimpleSwitch* and the *TCAMSwitch* is less than 3%. Table 3 gives a summary of the post-implementation on-chip power consumption of both the prototype switches.

The *Total On-Chip Power* = *Core Dynamic* + *I/O* + *Transceiver* + *Device Static*. It includes the power dissipated on-chip from all the supply sources. It does not include the power supplied to the off-chip devices. This is the operating power, not the startup power. The Thermal Margin is negative when the estimated junction temperature exceeds the maximum specified value. This information can be used to address the excess power consumed on-chip. Hence, we conclude that the power consumption of our TCAMSwitch is also negligible.

V. CONCLUSION

In this paper, we proposed an effective TCAM based routing technique for data center ToR switches. The study proposes that an additional TCAM table may be implemented in ToR/leaf switches containing flow entries to connect the servers linked to the same leaf/ToR switch and the servers in other particular Pods. The model is useful in providing security and QoS, as well as may help to overcome many problems faced by the ToR/Leaf switches in large data centers, e.g. High-Throughput, Link Oversubscription, Packet Loops, and TCP Incast and Outcast. The model is tested for its validation, performance, power and resource consumptions on state-of-the-art testing and implementation tools such as ONetSwitch45 hardware FPGA board and Spirent TC. We implemented two switch cores: one for our proposed architecture and the other is a simple ToR/Edge switch (SimpleSwitch), to evaluate any overhead introduced in our model. The preliminary results show that there is no considerable performance overhead introduced in our model,

and it gives the line-rate throughput, almost similar to that of (SimpleSwitch). The on-chip power consumption overhead of the proposed model is also negligible. However, to fully test the model and to support the theoretical assumptions made in this article, it is needed to be tested in real large multi-tier DC.

We are very hopeful that this model could be used in many future research works in implementing QoS and security in modern large DCs.

ACKNOWLEDGMENT

The authors would like to thank C. Hu from Xilinx and J. Meng, X. Sun, Q. S. Yi, X. Tian, and H. Li from Xi'an Jiaotong University for their valuable suggestions and help to work. They would also like to thank to the anonymous reviewers for their valuable suggestions to improve this article.

REFERENCES

[1] M. K. Khattak, Y. Tang, and U. S. Khan, "TOSwitch: Programmable and high-throughput switch using hybrid switching chips," *IEEE Commun. Lett.*, vol. 23, no. 12, pp. 2266–2270, Dec. 2019.

[2] J. Huang, S. Li, R. Han, and J. Wang, "Receiver-driven fair congestion control for TCP outcast in data center networks," *J. Netw. Comput. Appl.*, vol. 131, pp. 75–88, Apr. 2019.

[3] Y. Qin, W. Yang, Y. Ye, and Y. Shi, "Analysis for TCP in data center networks: Outcast and Incast," *J. Netw. Comput. Appl.*, vol. 68, pp. 140–150, Jun. 2016.

[4] A. Ghanwani, K. Subramanian, P. Janardhanan, S. Sundaram, and M. Anumala, "Routing in spine-leaf networking systems," U.S. Patent 9 584 397 B2, Feb. 28, 2017.

[5] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: Fine grained traffic engineering for data centers," in *Proc. 7th Conf. Emerg. Netw. Exp. Technol.*, 2011, p. 8.

[6] C. Hu, J. Yang, H. Zhao, and J. Lu, "Design of all programable innovation platform for software defined networking," Presented as the Part Open Netw. Summit (ONS), 2014.

[7] *ONetSwitch45: All Programmable Open Networking Innovation Platform*. Accessed: Feb. 14, 2020. [Online]. Available: https://github.com/MeshSr/ONetSwitch/wiki/ONetSwitch45

[8] S. Zafar, A. Bashir, and S. A. Chaudhry, "On implementation of DCTCP on three-tier and fat-tree data center network topologies," *SpringerPlus*, vol. 5, no. 1, p. 766, Dec. 2016.

[9] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, Oct. 2008.

[10] K. Bilal, S. U. Khan, J. Kolodziej, L. Zhang, K. Hayat, and S. A. Madani, "A comparative study of data center network architectures," in *Proc. ECMS*, 2012, pp. 526–532.

[11] A. Greenberg, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Towards a next generation data center architecture: Scalability and commoditization," in *Proc. Workshop Program. Routers Extensible Services Tomorrow (PRESTO)*, 2008, pp. 57–62.

[12] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: A scalable and flexible data center network," in *Proc. ACM SIGCOMM Conf. Data Commun.*, 2009, pp. 51–62.

[13] A. X. Liu, C. R. Meiners, and Y. Zhou, "All-match based complete redundancy removal for packet classifiers in TCAMs," in *Proc. 27th Conf. Comput. Commun.*, Apr. 2008, pp. 111–115.

[14] Q. Maqbool, S. Ayub, J. Zulfiqar, and A. Shafi, "Virtual TCAM for data center switches," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2015, pp. 61–66.

[15] P. Bosshart, G. Gibb, H.-S. Kim, G. Varghese, N. McKeown, M. Izzard, F. Mujica, and M. Horowitz, "Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 99–110, Aug. 2013.

[16] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: Scaling flow management for high-performance networks," in *Proc. ACM SIGCOMM Conf.*, 2011, pp. 254–265.

[17] Y. Chiba, Y. Shinohara, and H. Shimonishi, "Source flow: Handling millions of flows on flow-based nodes," in *Proc. Conf. SIGCOMM*, 2010, pp. 465–466.

[18] Cisco. *Cisco Catalyst 4500 Series Switch Data Sheet*. Accessed: Feb. 14, 2020. [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/switches/catalyst-4500-series-switches/product_data_sheet0900aecd801792b1.html

[19] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[20] N. Farrington, E. Rubow, and A. Vahdat, "Data center switch architecture in the age of merchant silicon," in *Proc. 17th IEEE Symp. High Perform. Interconnects*, Aug. 2009, pp. 93–102.

[21] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. 10th Annu. Conf. Internet Meas. (IMC)*, 2010, pp. 267–280.

[22] G. Lu, C. Guo, Y. Li, Z. Zhou, T. Yuan, and H. Wu, "Serverswitch: A programmable and high performance platform for data center networks," *Nsdi*, vol. 2011, p. 2, Mar. 2011.

[23] G. Lu, R. Miao, Y. Xiong, and C. Guo, "Using CPU as a traffic co-processing unit in commodity switches," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw.*, 2012, pp. 31–36.

[24] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: Measurements & analysis," in *Proc. 9th ACM SIGCOMM Conf. Internet Meas.*, 2009, pp. 202–208.

[25] T. Zahid, F. Y. Dar, X. Hei, and W. Cheng, "An empirical study of the design space of smart home routers," in *Proc. Int. Conf. Smart Homes Health Telematics*, 2016, pp. 109–120.

[26] J. Kang, X. Hei, and J. Song, "A comparative study of zynq-based openflow switches in a software/hardware co-design," in *Proc. Int. Conf. Secur., Privacy Anonymity Comput., Commun. Storage*, 2017, pp. 369–378.

[27] J. Naous, G. Gibb, S. Bolouki, and N. McKeown, "NetFPGA: Reusable router architecture for experimental research," in *Proc. Workshop Program. Routers Extensible Services Tomorrow (PRESTO)*, 2008, pp. 1–7.

[28] T. Su, L. You, Q. Wang, and C. Hou, "The high speed switching experiment based on NetFPGA SUME," in *Proc. 11th Int. Conf. Comput. Sci. Edu. (ICCSE)*, Aug. 2016, pp. 652–657.

[29] T.-W. Chu, C.-A. Shen, and C.-W. Wu, "The hardware and software co-design of a configurable QoS for video streaming based on openflow protocol and NetFPGA platform," *Multimedia Tools Appl.*, vol. 77, no. 7, pp. 9071–9091, Apr. 2018.

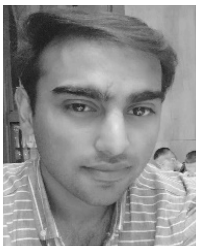
[30] T. Lukaseder, L. Bradatsch, B. Erb, R. W. Van Der Heijden, and F. Kargl, "A comparison of TCP congestion control algorithms in 10G networks," in *Proc. IEEE 41st Conf. Local Comput. Netw. (LCN)*, Nov. 2016, pp. 706–714.

[31] E. Kaljic, A. Maric, and P. Njemcevic, "An implementation of a deeply programmable SDN switch based on a hybrid FPGA/CPU architecture," in *Proc. 18th Int. Symp. INFOTEH-JAHORINA (INFOTEH)*, Mar. 2019, pp. 1–6.

[32] G. Zhong, H. Javaid, H. Saadat, L. Xu, C. Hu, and G. Brebner, "FastProxy: Hardware and software acceleration of stratum mining proxy," in *Proc. Crypto Valley Conf. Blockchain Technol.(CVCBT)*, Jun. 2019, pp. 73–76.

MUHAMMAD KASHIF KHATTAK received the master's degree in computer science and technology from the Kohat University of Science and Technology, Kohat, Pakistan, in 2008. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Xi'an Jiaotong University (XJTU), China. His research interests include data center networks, designing and programming network switches and routers, data plane programmability, FPGAs, and ASICs.

YAZHE TANG received the B.Sc., M.Sc., and Ph.D. degrees from Xi'an Jiaotong University, Xi'an, China, in 1993, 1996, and 2002, respectively. He is currently an Associate Professor with the Department of Computer Science and Technology, Xi'an Jiaotong University. His main research interests include computer networking systems, network security, and network measurement and monitoring.



HAMZA FAHIM received the B.S. and M.S. degrees from the Computer Science Department, COMSATS University Islamabad (CUI), Pakistan, in 2012 and 2015, respectively. He is currently pursuing the Ph.D. degree in computer science and networks with the School of Computer Science and Technology, Xi'an Jiaotong University, China. His current research interests include intra-body nanonetworks, software defined networks, and wireless sensor networks.

EID REHMAN received the M.S. (Hons.) and Ph.D. degrees from International Islamic University Islamabad, Pakistan, in 2011 and 2018, respectively. He is currently working as an Assistant Professor with Foundation University Islamabad. His research interests include networking, WSN, and next-generation networks.

MUHAMMAD FARAN MAJEED received the M.S. degree (Hons.) in computer science from the CECOS University of IT and Emerging Science, Peshawar, Pakistan, in 2011, and the Ph.D. degree in computer science from the Department of Information and Communication Technologies, Asian Institute of Technology, Thailand. He is currently a Faculty Member with the Department of Computer Science, Shaheed Benazir Bhutto University, Sheringal, Pakistan. His research interests include future Internet architectures, the Internet of Things, network robotics, and sensor networks.