A Literature Review on Software-Defined Networking (SDN) Research Topics, Challenges and Solutions

Kannan Govindarajan[1], Kong Chee Meng[1], Hong Ong[1]

[1]Advance Computing Lab,
MIMOS BERHAD,
Malaysia
kannan.darajan@mimos.my, cm.kong@mimos.my, hh.ong@mimos.my

Abstract— **Cloud computing data centers are becoming increasingly popular for the provisioning of computing resources. In the past, most of the research works focused on the effective use of the computational and storage resources by employing the Virtualization technology. Network automation and virtualization of data center LAN and WAN were not the primary focus. Recently, a key emerging trend in Cloud computing is that the core systems infrastructure, including compute resources, storage and networking, is increasingly becoming *Software-Defined.* In particular, instead of being limited by the physical infrastructure, applications and platforms will be able to specify their fine-grained needs, thus precisely defining the virtual environment in which they wish to run. Software-Defined Networking (SDN) plays an important role in paving the way for effectively virtualizing and managing the network resources in an on demand manner. Still, many research challenges remain: how to achieve network Quality of Service (QoS), optimal load balancing, scalability, and security. Hence, it is the main objective of this article to survey the current research work and describes the ongoing efforts to address these challenging issues.**

Keywords — Cloud Computing; Virtualization; Software-Defined Networking; OpenFlow; Quality of Service; Load Balancing; Security; Scalability.

I. Introduction

Cloud Computing [1] is generally categorized into Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). Virtualization technology acts as the backbone for IaaS service delivery model to virtualize and provide the Cloud resources in an effective manner. However, most of the existing research efforts in the recent past years mainly focused on the effective use of the compute and storage resources using the virtualization technology such as Xen [2], Kernel Virtual Machine (KVM) [3], VMWare [4] and etc. Network automation and virtualization of data center LAN and WAN were not the primary focus of most researchers and users. However, virtualization and cloud computing are pushing data center operators to think beyond their traditional network set up.

Ethernet networks have evolved significantly since their inception in the late 1980s, with many evolutionary changes leading to the various switch categories that are available today. Data center LAN and WAN switching has emerged as a unique category, with highly dense 10Gbps, 40Gbps, and now 100Gbps port-to-port wire-rate switching as one of the leading Ethernet networking product areas. Beyond these considerable speed progressions, the other significant advancements are 1) data center switching offers sub-microsecond switch latency (measured in nanoseconds) and zero-drop packet failover when failing over to redundant links for addressing QoS, 2) sophisticated traffic load balancing algorithms are developed for addressing increased asset optimization, 3) scaling in support of large carrier-class virtualized infrastructures, and 4) built-in network security mechanisms to enforce policy and reduce cyber-threat incidents. While these state-of-the-art switching features leverage 30 years of progressive hardware and software technology evolution, successful implementation of network virtualization and automation requires a fundamental shift from closed, vendor-specific proprietary network operating systems to open, extensible, externally programmable operating systems. This open extensibility requirement is driven by the guiding principles of cloud data centers in which resources are managed dynamically as one integrated system made up of compute, network, and storage.

Software-Defined Networking (SDN) [5] is recently an emerging technique that paves the way for virtualizing the network resources in an on demand manner. It provides an abstraction of the underlying network to the applications residing in upper layers. Conventionally, the network devices such as switches and routers have control plane, management plane and data plane whereas in SDN, the logic of control and data plane is decoupled separately. The control plane logic is implemented as a software component that is residing in a server and data plane is located in network devices. The decoupling of control and data plane logic has transformed the network resources into programmable, automation and network control, highly scalable and flexible networks based on the business needs. Moreover, SDN [6] replaces the functionality of networking devices as just forwarding devices. The intelligence of where and how to make forwarding is residing in control plane. The control plane logic is implemented in the software called controller. OpenFlow [7] is the protocol for communicating the controller with network devices. Some of the popular SDN controllers in the market and research are Floodlight [8], Beacon [9], NOX [10], and OpenDayLight [11]. The SDN architecture used in current networking world is shown in Figure 1. The SDN application 1 to N represents the features such as Quality of Service (QoS), Load Balancing (LB), Firewall (FW) and etc. that is deployed on top of SDN controllers. The Controller receives the packet and forward to OpenFlow based switches for example OpenVSwitch [12].

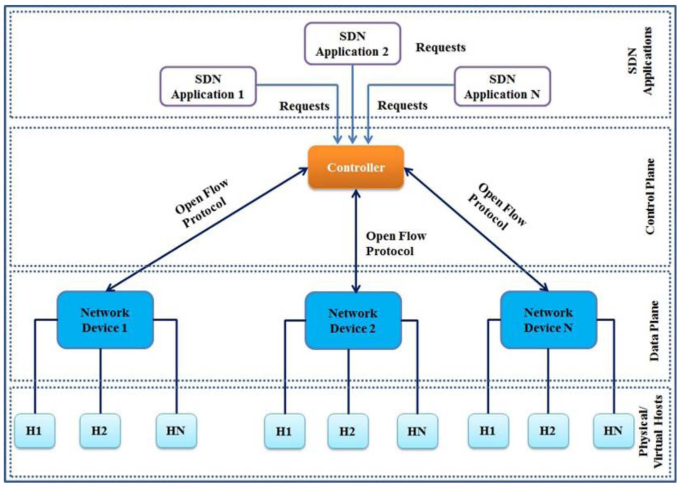

Figure 1: SDN Architecture

The OpenFlow based switches maintain the flow table as shown in Table 1. It matches the header fields in the flow table entries, based on the entries in the flow table it makes the decision to forward the packet to appropriate port or discard the packet. If the packet is not matching with the entries which are available in flow table, it encapsulates and sends back the packet to the controller. Finally, the controller takes the decision of how to handle the packet such as notifying the switch to drop the packet or making an entry in the flow table for supporting the new flow.

Although SDN has lot of advantages over conventional networking, it has its own challenging issues. As per our literature survey, we have identified four major research and challenge issues in SDN such as Quality of Service (QoS), Load Balancing (LB), Security and Scalability as shown in Figure 2. From the identified research areas, we have represented some of the ongoing efforts to solve those challenging issues.

In summary, the main contributions of this research paper are: i) Identification and classification of research and challenging issues in SDN, ii) Survey of ongoing research efforts to solve the identified research and challenging issues, iii) Comparison of the presented techniques focusing on their features related to QoS, LB, Scalability and Security.

The rest of this paper is organized as follows. Section 2 discusses the literature review on Quality of Service (QoS). Section 4 represents the literature survey on load balancing in SDN; Section 5 and section 6 discusses the literature reviews on scalability and security respectively; Section 6 discusses the summary of this research paper with future wok.

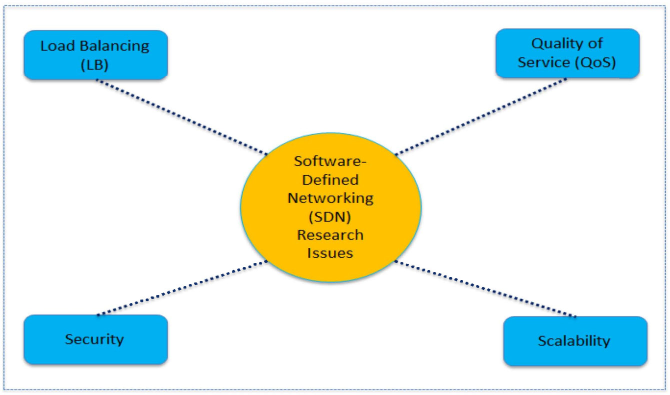

Figure 2: Major Research Issues

II. Quality of Service (QoS)

Quality of Service (QoS) is defined as an ability to provide a service. It is an essential property in networks and it is difficult to achieve the desired QoS parameters for long years. The main QoS parameters to achieve in the network are guaranteeing the bandwidth, minimize the delay, reduce the packet loss and congestion control that is shown in Figure 3. In this article, we present some of the research works which are mainly focused on solving the QoS problems in the SDN based networking world.

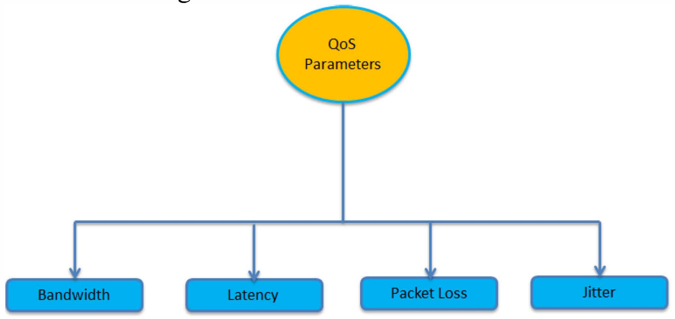

Figure 3: Quality of Service (QoS) Parameters

A. OpenQoS

Egilmez et al. [13] proposed an OpenFlow protocol based controller namely OpenQoS to achieve end-to-end Quality of Service (QoS) for multimedia based applications. In this work, the traffic is classified into data flows and multimedia flows. The multimedia flows are diverted into dynamic QoS guaranteed routing algorithm whereas the other data flows are following the shortest routing algorithm. The dynamic QoS routing is defined as Constraint Shortest Path (CSP) problem which is NP-Complete in nature [30]. The dynamic QoS routing calculates the shortest path based on congestion and delay factors. The route management module is responsible for collecting those two factors and the routing calculation function is responsible for achieving dynamic QoS routing.

Table 1: Flow Table

The routing calculation uses the Lagrangian Relaxation Based Aggregated Cost (LRBAC) polynomial algorithm [31].

Switch port	MAC src	MAC dst	Eth type	VLAN ID	IP src	IP dst	IP prot	TCP sport	TCP dport

OpenQoS introduce three interfaces: 1) controller-controller interface that is responsible for interacting with other OpenFlow controllers and enhance the scalability, 2) controller-service interface to interact with multimedia and other applications and 3) controller-forwarder interface to interact with switches for performing various actions such as controlling the traffic flows, enforcing the firewall rules and etc. The OpenQoS is implemented over the Floodlight Controller which is the most stable one. The proposed approach is evaluated by streaming of videos over User Datagram Protocol (UDP) and Transmission Control Protocol (TCP). The performance metric of Packet Signal to Noise Ratio (PSNR) is calculated that reflects the quality loss from the original packets.

Finally, OpenQoS has indicated that Multiple Description Decoding, Load Balancing in Content Distribution Networks and enabling cross layer design in the Internet and OpenFlow wireless networks as their future works. However, OpenQoS proposed approach does not consider resource reservation and priority based queuing mechanisms that paves the way for minimizing packet loss and latency.

B. OpenQFlow

Airton Ishimori et al. [14] proposed a QoS management framework called QoSFlow that enables the QoS management functions in Openflow based network environment. The proposed work controls and manages the QoS parameters such as bandwidth, queue size and delay in an on demand manner. The QoSFlow architecture has two major modules namely QoSFlow Controller and QoSFlow Datapath.

The QoSFlow Controller is based on NOX that is responsible for managing and monitoring actions and controlling signal messages. In addition to the controller, it is built with the following four components namely QoSFlow Agent, QoSFlow Manager, QoSFlow Monitor and DB-QoSFlow Client. The QoSFlow Agent establishes the communication between management tool and QoSFlow Monitor and QoSFlow Manager components. The QoSFlow Monitor monitors the QoS flows and QoSFlow Manager manages the QoS flows.

The QoSFlow Datapath component creates the low-level actions on the networking devices. The proposed work limits the total bandwidth to a known rate, limit the bandwidth of a particular user, service or client, reserve bandwidth for a particular application or user, manage oversubscribed bandwidth and it allows equitable distribution of unreserved bandwidth. The policies which are implemented in QoSFlow framework is responsible for handling configuration of switches, scaling and management of hundreds of switches and controlling the behavior of end-to-end QoS requirements.

C. Secondnet

In Cloud it is essential to provide the guaranteed bandwidth to the virtual machines (VMs) which are residing in different data centers. Secondnet [15] is a data center virtualization framework that provides an abstraction for Virtual Data Center (VDC) that guarantees the bandwidth for user requests allocated between every pair of virtual machines. It is worthwhile to note that the bandwidth allocation is a NP-hard problem. As such, the Secondnet proposed a VDC Allocation Algorithm, which is a low time-complexity heuristic algorithm.

The VDC Allocation Algorithm is scalable in nature as it distributes the traffic from virtual to physical mapping with guaranteed bandwidth. The VDC Allocation Algorithm works like the following: 1) servers are clustered into various sizes based on the proximity of closeness i.e. hop count value; 2) based on the proximity, the algorithm minimizes the allocation time by searching the servers only in the suitable clusters other than the whole physical network. Essentially, the first step in VDC allocation algorithm is selecting a cluster C_k which are closer to satisfy the user application requests. The second step is constructing the bipartite graph by putting required virtual machines in left side and the available physical servers at right side. Then, it selects the server as feasible candidate for hosting virtual machine, only if it satisfies the processor speed, memory, disk space, and ingress and egress bandwidth.

In addition to bandwidth allocation, SecondNet also proposed a min-cost flow algorithm for path allocation that selects the best connectivity available in the physical network. The proposed work is implemented with Port-Switching Source Routing (PSSR) to allocate the routing path as a sequence of output ports of switches. It is possible to implement the PSSR mechanism with Multi-Protocol Label Switching (MPLS) to the existing commodity switches. This framework is well-suited for enterprise workloads to provide guaranteed application performance.

D. CloudNaaS

Benson et al. [16] presented the design, implementation and evaluation of Cloud Networking as a Service (CloudNaaS) framework. It extends the self-provisioning model of providing network devices in an on demand manner in addition to compute and storage devices. The proposed system uses the policy language to specify the users' application requirements, and then it translates the high-level user application requirements into communication matrix that indicates the virtual network between source and destination virtual machine can able to transfer packets.

The CloudNaaS architecture interfaces the two components namely cloud controller and network controller. The cloud controller is responsible for managing the virtual machines and physical hosts. The network controller is responsible for managing the configuration of network devices and placing the virtual machines within the cloud. The functionality of cloud controller is extended to accept the network policy specifications for generating the communicating matrix. The network controller is integrated with placement algorithm that provides the input to cloud controller for placing the virtual machines in the physical hosts. Moreover, the proposed work provides the virtual network functions such as network isolation, custom addressing, service differentiation and deployment of middleware boxes for intrusion detection, caching or application acceleration to deploy the customer's application in Cloud resources.

E. Automated and Scalable QoS Control for Network Convergence

Network convergence is one of the recently emerging concepts which received greater attention, because it is highly desirable to serve the traffic from multiple applications on to a single network. It has two dimensions namely convergence of traffic from different applications and convergence of traffic from different tenants.

Wonho Kim et al. [17] proposed a QoS control framework for the automation and management of converged network traffic. The proposed QoS controller creates the network slices and assigns different traffic applications into created network slices dynamically to satisfy the QoS requirements. The main motivations of the proposed system are automatically finds and apply the best configuration for the flows; controller is a dynamically adaptable based on the workloads, the controller framework can able to deploy and manage the existing and large scale networks and controller is provided with network optimization technique to optimally use the network resources. The proposed architecture has three main components namely QoS controller, Adaptive Aggregator and Network-wide Optimization. The controller invokes the adaptive aggregator to achieve better scalability and makes the decision for QoS configurations by measuring the states of network and applying network wide optimization technique. QoS APIs are provided with controller to automate the configuration and management. The proposed work is implemented with Flow Spec and Slice Spec which is used to represent a set of flows in network and performance requirement in network such as maximum bandwidth, minimum delay, etc. respectively.

III. LOAD BALANCING

Load-balancing [18] is a smart congestion aware routing in Software-Defined Networking (SDN). It is an essential entity in SDN based network environment to improve the availability and scalability of applications in Figure 4 which are deployed in Cloud infrastructure that leads to achieve the minimal response time of the applications. The load balancing mechanisms such as Equal Cost Multi Path (ECMP) and Valiant Load Balancing (VLB) are used in data centers. The ECMP based routing strategy calculates the cost for multipath and spreads the traffic over multiple paths based on the calculated cost. VLB forwards an incoming flow to the corresponding destination by selecting a random switch.

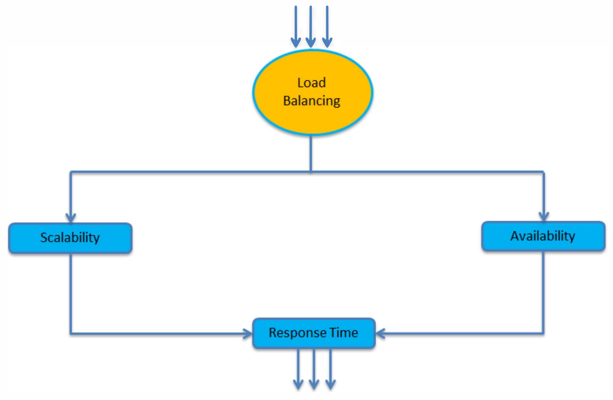

Figure 4: Load Balancing in SDN

A. AsterX

Nikhil Handigol et al. [18] proposed the load balancer named Aster *x. The main objective of the proposed load balancer is to minimize the response time of the applications by jointly consider the server and network load. The load balancer considers a single request or bunch of requests as flow. Finally, it makes the decision to route as individual request or in any combination of requests using Equal Cost Multi Path (ECMP) [19] like oblivious load balancing. It has the following characteristics such as distributed throughout the network that enables scalability, logically centralized and flexible in nature. The load balancing decision in Aster * x is proactive versus reactive during the arrival of every request, individual versus aggregated requests and static and dynamic.

The AsterX architecture consists of three main modules namely Flow Manager, Net Manager and Host Manager. Flow Manager is responsible for managing and routing the flows based on the selected load balancing algorithms such as disjoint, server based selection and joint selection. Net Manager monitors the network topology and utilization level of network. Host Manager monitors the status and load of the available servers. The proposed work is tested in GENI infrastructure that is spanned across three University campuses and web servers are hosted in PlanetLab nodes connected through OpenFlow based networking devices.

B. OpenFlow-Based Server Load Balancing Gone Wild

Richard Wang et al. [20] proposed an in-network load balancer that installs the wildcard rules in the switches for redirecting the requests in a proactive manner. The load balancing architecture is implemented with partitioning algorithm to generate the wildcard rules and transitioning algorithm to change one set of rules to another. The partitioning algorithm is a centralized controller program to determine the global optimal wildcard rules. It divides the client traffic based on the weights calculated for the client traffic and then a binary tree is used to construct the IP prefixes, where each node is corresponding to an IP prefix and nodes which are closer to leaves represent longer prefixes. Hence, the partitioning algorithm minimizes the wildcard rules by implementing the concept of aggregating sibling nodes in association with same server replica. The transitioning algorithm is used to change those rules to adjust the newly calculated load balancing weights. The proposed has been implemented and being evaluated using OpenVswitch, NOX and MiniNet [21].

IV. SCALABILITY

S. No.	Scalability Level	Description
1	**Level 1**	It indicates the number of switches that an SDN based controller can able to support
2	**Level 2**	It describes the flow table entries that occur for each flow
3	**Level 3**	It represents how SDN controller is capable of handling the switches which are spanned across multiple sites.

Scalability [22] is one of the essential key factors in SDN based networking environment. Scalability in SDN is classified into three levels as shown in Table 2.

Table 2: Scalability Issues in SDN

A. DIFANE

Minlan Yu et al. [23] proposed the DIstributed Flow Architecture for Networked Enterprises (DIFANE). The main objective of proposed work is to avoid the bottlenecks in controller and achieving better performance and scalability to preserve the traffic in the data plane. The DIFANE architecture has two main motivations such as (i) distribute the rules across switches called authority switches and scaling to large number of topologies by running the partitioning algorithm (ii) handle all packets in the data plane by diverting packets through authority switches.

To implement the DIFANE architecture, it is required to change only in the control plane not in the data plane. The proposed work is experimented on top of OpenFlow based switches to achieve lower delay, higher throughput, and better scalability in a distributed manner than directing packets through a separate controller. It makes four high-level design decisions for reducing the overhead of handling cache misses and allows the system to scale to a large number of hosts, rules, and switches. It handles wildcard rules efficiently, reacts quickly to network dynamics such as policy and topology changes and host mobility. The controller in DIFANE generates the rules and the generated rules are allocated to the authority switches. It is the subset of available switches which has larger memory and processing capability. The controller first partitions the rules and distributes the partition and authority rules to the switches. Using link-state routing, it computes the path and caching the rules in the authority switches.

B. Maestro

Zheng Cai et al. [24] proposed a system called Maestro to achieve the scalability by enabling parallelism and throughput based optimization technique. It sends and receives the OpenFlow messages through TCP connections. Maestro is implemented with four applications namely Discovery, IntradomainRouting, Authentication and RouteFlow.

The Discovery application in Maestro sends out the probe message to the neighbors whenever the new switch is joined in the network. The flow request is first checked with security policies implemented in authentication application. Once the security policy is validated true, RouteFlow application finds the path and generates a message for flow configuration in every switch. The RouteFlow and Authentication is called as flow process stage. Once the flow configuration messages are sent to their destination, the flow request packet is sent back to the origin, it is called as flow request execution path. The task manager in the Maestro system provides a unified interface; it generates ‘n’ number of worker threads based on the number of cores in the controller machine to complete the submitted tasks. The main design goals of multi-threading concepts implemented in Maestro system are to distribute the work evenly among available core/threads, minimize the overhead introduced by cross-core and cache synchronization and minimize the memory consumption of system. The proposed system is evaluated by comparing the performance with NOX in an emulated environment using the performance metrics of throughput in requests per second and delay experience by the flow requests. Maestro shows better performance in all aspects compared to NOX.

C. DevoFlow

Mogul et al. [25] proposed a model called Devolved OpenFlow (DevoFlow). The main motivation of the proposed work is developing a simple, cost-effective hardware and redistribution of decisions by the switches itself. It reduces the number of switch-controller interactions, TCAM entries and detecting the QoS flows in an efficient manner. Additionally, it provides a mechanism for making routing decisions in switches itself.

DevoFlow resolves the control issues by invoking the controller on every flow setup and using the OpenFlow based flow match wildcards in an aggressive manner to reduce the control-plane load. Similarly, it resolves the statistics issues by aggregating counters from microflows using pull-based Read State mechanism and aggregating counters over multiple microflows using wild-card mechanism. It is implemented with two mechanisms namely a) rule cloning and b) local routing actions for transferring the control to switches. The rule cloning mechanism is integrated with a Boolean CLONE flag value, based on the flag value, the switch makes the decision to follow standard wildcard behavior or locally clone the wild card rule to create a new rule. This rule exactly matches the lookup table that reduces the cost of TCAM by decreasing the usage of TCAM. The local routing actions mechanism is helpful for taking decisions by the switch itself without increasing the overhead in the controller. The OpenFlow statistics collection efficiency is improved by integrating sFlow [26] based sampling technique and threshold-based triggering and reports.

V. SECURITY

Security is another major threat in SDN based network. The first biggest security challenge is to protect the controller which has more intelligence for controlling the data planes. The other securities challenges reside in the SDN based networking environment are protecting Distributed Denial of Service (DDoS) attacks, Intrusion Prevention and etc.

A. NetFuse

Ye Wang et al. [27] proposed a scalable mechanism named NetFuse that resides in between OpenFlow controllers and switches to protect the Cloud based data centers from traffic overload. Nowadays, the data centers are largely affected by DDoS attacks and workload changes, misconfigurations and etc. It makes use of passively-collected OpenFlow control messages for detecting active network flows, multi-dimensional flow aggregation to identify the network flows overloading behavior, toxin-antitoxin mechanism to shape the rate of traffic flow.

The monitoring component is employed with active query and passive listening mechanism to aggregate the network information. It intercepts the control messages to acquire the global view of the network information. If the packet received by the switch does not match with flow table entries it sends a *PacketIn* message to the controller, it replies to the switch for installing forwarding rule using *FlowMod* message. The switches send the *FlowRemoved* message to the controller, once the flow time is expired. Moreover, it uses the OpenFlow *ReadState* message to know the network resource utilization. NetFuse is implemented with flow aggregation mechanism. It is formulated as a NP-hard combinatorial optimization algorithm. The flow aggregation is modeled as threshold-based aggregation that will identify the flow which overloads the behavior. NetFuse is implemented with adaptive control mechanism to modify or reissue the new flow rules to the switches. Finally, NetFuse has improved the scalability of the system by implementing the flow redirection, delay injection, and packet blocking.

B. Fresco

Seungwon Shin et al. [28] developed an OpenFlow security application development framework named FRESCO to provide OpenFlow enabled detection and mitigation modules. This framework consists of an application layer which is implemented using python modules available in NOX and a security enforcement kernel.

Each module is organized with five interfaces such as (1) input (2) output (3) parameter (4) action and (5) event. The modules are implemented as an event-driven processing function. The FRESCO Development Environment (DE) provides the platform with bunch of useful information about security for researchers. It has four main purposes such as (1) script to module translation (2) database management (3) event management (4) instance execution. The script to module translation function is responsible for translating the FRESCO scripts to modules and creation of instances from modules. The database management function is responsible for aggregating the network and switch state information and providing an interface for using that information. The event manager function is responsible to notify the previously defined events. The instance execution is responsible for loading the created instances into memory. The FRESCO security enforcement kernel in this framework monitors and keeps track the status of OpenFlow switches in a regular interval. The security policies such as DROP, REDIRECT and QUARANTINE are enforced by the security applications written in the proposed framework based on the threats to the network.

VI. CONCLUSION AND FUTURE WORK

Cloud resources such as compute, storage and network become the worthwhile infrastructure for computation, data storage and hosting network based applications. Software-Defined Networking (SDN) solves the issues in the conventional networking and virtualizes the network resources in an on demand manner to maximize the utilization by effectively using the network resources and satisfying the user application constraints. In this paper, we surveyed the state of the art in Software-Defined Networking (SDN) research in four areas: Network Quality of Service (QoS), Load Balancing, Scalability and Security. From the literature survey, we have identified that, there is no common architecture or solution to address all the four issues that should be addressed in the context of Software-Defined Networking (SDN). Hence, our future work is mainly focused to develop our own Software-Defined Networking (SDN)

Platform to address the above discussed four challenging issues.

ACKNOWLEDGEMENT

The authors sincerely thank the MIMOS BERHAD, for financially supporting the Advanced Computing Laboratory, in MIMOS BERHAD, Kuala Lumpur, Malaysia to do extensive research in the latest technology.

References

[1] P. Mell and T. Grance. "NIST definition of cloud computing". National Institute of Standards and Technology. October 7, 2009.

[2] Xen (2013). [Online]. Available: http://www.xen.org/.

[3] Vmware (2013). [Online]. Available: http://www.vmware.com/.

[4] Kernel-based Virtual Machine (KVM) (2013). [Online]. Available: http://www.linux-kvm.org/.

[5] Open Networking Foundation (ONF) (2013). [Online]. Available: "Software-Defined networking: the new norm for networks," https://www.opennetworking.org/images/stories/downloads/openflow/wpsdn-newnorm.pdf.

[6] OpenFlow Consortium (2013). [Online]. Available: http://openflowswitch.org

[7] 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, 2008.

[8] Floodlight (2013). [Online]. Available: http://www.projectfloodlight.org/floodlight/.

[9] Beacon (2013). [Online]. Available: https://openflow.stanford.edu/display/Beacon/Home.

[10] NOX (2013). [Online]. Available: http://noxrepo.org/wp/.

[11] OpenDayLight (2013). [Online]. Available: http://www.opendaylight.org/.

[12] OpenVSwitch (2013). [Online]. Available: http://openvswitch.org/.

[13] Hilmi E. Egilmez, Seyhan Civanlar, and A. Murat Tekalp, "An Optimization Framework for QoS-Enabled Adaptive Video Streaming Over OpenFlow Networks", IEEE TRANSACTIONS ON MULTIMEDIA, VOL. 15, NO. 3, APRIL 2013.

[14] Airton Ishimori, Fernando Farias, Igor Furtado, Eduardo Cerqueira, Antônio Abelém, Automatic QoS Management on OpenFlow Software-Defined Networks

[15] Guo, Chuanxiong, Guohan Lu, Helen J. Wang, Shuang Yang, Chao Kong, Peng Sun, Wenfei Wu, and Yongguang Zhang. "Secondnet: a data center network virtualization architecture with bandwidth guarantees." In Proceedings of the 6th International Conference, p. 15. ACM, 2010.

[16] Benson, T., Akella, A., Shaikh, A., & Sahu, S. (2011). CloudNaaS. Proceedings of the 2nd ACM Symposium on Cloud Computing - SOCC '11 (pp. 1-13). New York, New York, USA: ACM Press. doi:10.1145/2038916. 2038924.

[17] Wonho Kim, Puneet Sharma, Jeongkeun Lee, Sujata Banerjee, Jean Tourrilhes, Sung-Ju Lee, and Praveen Yalagandula "Automated and Scalable QoS Control for Network Convergence" Proceedings of USENIX INM/WREN 2010, San Jose, CA, April 2010.

[18] Nikhil Handigol, Srini Seetharaman, Mario Flajslik, Aaron Gember,Nick McKeown, Guru Parulkar, Aditya Akella, Nick Feamster, Russ Clark, Arvind Krishnamurthy, Vjekoslav Brajkovic×, Tom Anderson, "Aster*x: Load-Balancing Web Traffic over Wide-Area Networks".

[19] Equal Cost Multi Path (ECMP) (2012). [Online]. Available: http://lib.tkk.fi/Dipl/2011/urn100416.pdf.

[20] Richard Wang, Dana Butnariu, and Jennifer Rexford, "OpenFlow-Based Server Load Balancing Gone Wild".

[21] MiniNet. (2013). [Online]. Available: http://mininet.org/

[22] Soheil Hassas Yaganeh, Amin Tootoonchian, Yashar Ganjali, "On the Scalability of Software-Defined Networking", IEEE Communications Magazine Feb 2013.

[23] Minlan Yu, Jennifer Rexford, Michael J. Freedman, Jia Wang, "Scalable Flow-based Networking with DIFANE", Sigcomm 2010.

[24] Zheng Cai et al Zheng Cai, Alan L. Cox, T.S. Eugene Ng, "Maestro: A System for Scalable OpenFlow Control", 2011.

[25] Andrew R. Curtis, Jeffrey C. Mogul, Jean Tourrilhes, Praveen Yalagandula, Puneet Sharma, Sujata Banerjee, "DevoFlow: Scaling Flow Management for High-performance Networks", Sigcomm 2011.

[26] SFlow (2013). [Online]. Available: http://sflow.org/about/index.php.

[27] Ye Wang, Yueping Zhang, Vishal Singh, Cristian Lumezanu, Geoff Jiang, "NetFuse: Short-circuiting Traffic Surges in the Cloud", ICC 2013.

[28] Seungwon Shin, Phil Porras, Vinod Yagneswaran, Martin Fong, Guofei Gu, Mabry Tyson, "FRESCO: Modular Composable Security Services for Software-Defined Networks", NDSS 2013.

[29] *Rodrigo Braga, Edjard Mota, Alexandre Passit*o, "Lightweight DDoS Flooding Attack Detection using NOX/OpenFlow", LCN 2010.

[30] Jeffrey D. Ullman, NP-Complete Scheduling Problems, J. Comput. Syst. Sci., Vol. 10, No. 3, 1975, Pp.384-393.

[31] A. Juttner, B. Szviatovski, I. Mecs, and Z. Rajko, "Lagrange relaxation based method for the QoS routing problem," in Proc. IEEE INFOCOM, vol. 2, Apr.2001, pp. 859-868.