Network Security Virtualization With Minimal Management Cost

Vrushali N. Huchhe[1], S.S.Ponde[2]

[1] Department of Computer Science and Engineering Marathwada Shikshan Prasarak Mandal's Deogiri Institute of Engineering & Management Studies, Aurangabad Maharashtra State, India 2017-2018

[2] Associate Professor, Department of Computer Science and Engineering Marathwada Shikshan Prasarak Mandal's Deogiri Institute of Engineering & Management Studies, Aurangabad Maharashtra state, India 2017-2018

Abstract: . ***Network management became so complex due increased requirement of using security devices It is very difficult to provide required security at correct places at needed time with less amount of time NSV presents a concept of network security virtualization which virtualizes security resources to network administrators users and thus maximally use existing security devices It provides security to the networks with minimum cost We developed a prototype that do the maximum use of pre installed static security devices and SDN to virtualizes security functions. It contains-***
(1) a simple language to record security services and policies
(2) a routing algorithm to decide shortest routing paths for different requirement and
(3) a set of security response functions to handle security incidents.

Keywords: SDN, Openflow.

1. Introduction

Network management became very complicated because of big requirement of security devices in the network.
One example of this type of network is cloud network. A cloud network commonly consists of a large amount of hosts and network devices to serve to a large number of dynamic users, each having a logically separated network. Network becomes complex due to many security devices. Many security devices are used to improve the performance, sturdiness, and security of networks. The security devices can serve many applications to networks, but it makes the network more complex to handle. So, there is need to solve this problem.
Extra security devices make network security compex to handle. The security devices have different security functions to solve different problems. For example, firewall & Network intrusion detection system (NIDS) controls the network access & observes the attacks. So, the network administrator should select proper security functions/devices and employ them into proper places. It is very critical for the administrator, to know network attacks of various network users and the administrator cannot understand the demands of different tenants before. The pre-installed security devices can not be in the proper positions that can serve the different security demands of different network users.
To solve this problem, it is required to leverage fixed security devices, and abstract these security devices to serve an interface for network users.
So, this is a new concept of Network Security Virtualization (NSV) that makes great use of fixed location, security devices and serves active, flexible, and on-demand security services to the users. So it is not needed to have the knowledge of pre-installed security devices.

NSV has two methods.
(i) Properly manage the flows to required network Security services, and
(ii) Provides network security response Functions on a network device.
It leverages the use of pre-installed security devices, NSV clearly redirect network flows to desired security devices when needed. For example, if a security policy wants that a network flow should be monitored by a security service, NSV technology reroutes the flow to the mentioned security middle devices.
It provides a security reaction on each network device. Latest techniques provide a method to manage network flows actively at a network device, e.g. SDN; can understand some basic security reply functions at a network device. It can conduct required security response functions on a network device when needed. In SDN, a network administrator can route the traffic from a centralized control console without having to touch individual switches. It can dynamically control network flows and monitor whole network status easily. It is a modern approach to networking that removes the complex and static nature of network architectures through the use of a standard-based software abstraction between the network control planes and underlying data forwarding plane, including both physical & virtual devices.

2. Literature Survey

In [1] paper, R. Ballard proposed OpenSAFE, a system which enables the random way of traffic for security observing applications at line rates. It presents a flow specification language ALARMS which easily handles management of network monitoring appliances. It shows a validation of currently undertaking to observe traffic across the network. OpenSAFE has three components: a set of design abstractions about the flow of network traffic; ALARMS (A flow specification language, and an OpenFlow component which implements the policy. For the ease of handling monitoring architecture to the network administrators, it uses ALARMS, a language for random route management for security traffic. ALARMS utilize the abstractions to create simple policy language syntax to describe paths. Paths are defined between named components, and each component cause to a distribution rule in the situation of multiple, parallel components. ALARMS are a high-level programming language which depends on a low-level programmatic interface to a network switch.

In [2] Sekar explores NIDS or NIPS deployment through discerning monitoring packets at diverse nodes. In this paper, V. Sekar describes a design that leverage spatial, network wide chances for sharing NIDS and NIPS functions. In case of NIDS, it assures that no node is overloaded by using a linear programming arrangement while giving detection responsibilities to nodes. It shows a prototype NIDS implementation to examine traffic per these assignments, and shows that the approach can be achieved. In case of NIPS, it presents how to do the maximum use of specialized hardware (e.g., TCAMs) to decrease the outline of redundant traffic on the network. These hardware conditions make the optimization problem NP-hard, and also give practical nearly exact algorithms based on randomized rounding. In this paper, a systematic formulation is given for effectively handling NIDS and NIPS deployments. Network-wide organized method, is used where various NIDS/NIPS abilities can be optimally shared throughout the various network locations depending on the operating conditions – traffic profiles, routing policies, and the resources ready at each location.

In [3], an infrastructure is proposed for Network-wide NIDS deployment that maximally uses three scaling opportunities: on-path sharing to divide responsibilities, repeating traffic to NIDS clusters, and collecting together intermediate results to divide expensive NIDS processing. It is challenging to equalize both the compute load across the network and the total communication cost incurred via replication and aggregation. It implements a backwards-compatible method to enable existing NIDS architecture to maximally use these benefits. It shows that the proposed method can significantly decrease the maximum computation time, also provides best elasticity under traffic variability, and gives enhanced detection coverage. A general NIDS design is proposed to maximum use of three opportunities: offloading processing to other nodes on a packet's routing path, traffic replication to off-path nodes (e.g., to NIDS clusters), and aggregation to split expensive NIDS tasks. It allows networks to understand these benefits with fewer changes to existing NIDS software. Many real-world arrangement of networks show that this system decrease the maximum compute load substantially, provides best elasticity under traffic variability, and offers improved detection coverage.
In [4] paper, describes FRESCO, an OpenFlow security application development framework proposed to simplify the process of rapid design, and modular arrangement of OF-enabled detection and mitigation modules. FRESCO, is an OpenFlow application, which gives a Click-inspired programming framework that allows security researchers to implement, share, and compose together, various security detection and mitigation modules. It shows the application of FRESCO with the implementation of many well-known security defenses as OpenFlow security services, and use them to analyze various performance and efficiency of proposed framework. FRESCO is used to solve the issues that can accelerate the constitution of new OF-enabled security services. FRESCO exports a scripting API that allows security practitioners to code security monitoring and threat detection logic as modular libraries. These modular libraries present the basic processing units in FRESCO, and may be distributed and connected together to provide complicated network security applications. It presents the FRESCO security enforcement kernel. It shows that FRESCO produces less overhead and allows active creation of well-known security functions with substantially fewer lines of code.

In [5] consider two aspects of OpenFlow that accept security challenges, and propose two solutions that could solve the problem. The first challenge is the inherent communication traffic constriction that comes between the data plane and the control plane, which an opponent could take advantage by supporting device a control plane saturation attack that interrupts network operations. Even well mined relating to conflict models, such as scanning or denial-of-service (DoS) activity, can produce more strong impact on OpenFlow networks than usual networks. To solve this problem, introduced an extension to the OpenFlow data plane called connection relocation, which considerably decreases the amount of data to- control-plane communications that comes during such attacks. The second problem is that of allowing the control plane to expedite detection of, and reaction to, the changing flow dynamics within the data plane. For this, introduced actuating triggers over the data plane's existing statistics collection services. These triggers are fixed by control layer applications to both record for asynchronous call backs, and fix conditional flow rules that are only activated when a trigger condition is validated within the data plane's statistics module. It describes AVANT-GUARD, an implementation of two data plane extensions, explains the performance impact, and analyze its use for forming more scalable and flexible SDN security services. The aim of AVANTGUARD is to create SDN security applications more scalable and reactive to active network threats. The challenge, which to be solved here, is the inherent traffic constriction introduced by the interface between the control plane and the data plane that known opponent can take the advantage. Connection migration allows the data plane to shield the control plane from such saturation attacks. The second problem is the issue of reactivates. A SDN security application requires expeditious access to network statistics from the data plane as a method for quickly responding to network threats. To solve this, it introduces actuating triggers that automatically fix flow rules when the network is under illegal coercion.

In [6], presented an architecture called Jingling, which adds operations to networks through outsourcing. Here the commercial company network forwards data and extra process is done by external Feature Providers (FPs). It gives the advantages such as decreased cost and complicated management. Feature API(FAPI) allows communication between enterprise control and configure features. Here SDN concept is used. SDN solves the problem of middlebox placement.

3 SYSTEM DESIGN

NETSEC contains five main modules: (i) Device and policy manager, (ii) Routing rule generator, (iii) Flow rule enforcer, (iv) Response manager, and (v) Data manager.
Device and policy manager performs two main functions. First, it takes the information of security devices from a cloud administrator, and registers that information into a device table for usage. Second, this module also takes security requests from each network users, and it converts them into security policies and registers the policies into a policy table. So, this module has two type of information: (i) locations/types of security devices from a cloud administrator and (ii) security policies from each user. It makes system to manage network security devices easily.

Response manager takes detection results from security devices, and it enables security response strategies that are mentioned in security policies, when it is required. For e.g. if a user mentions a security policy to drop all corresponding packets when a threat is detected by a NIDS, the response manager will enable drop function to remove network packets belonging to the detected network flows on a network device. Enabled functions will be identified as a set of network flow rules, which are forwarded to routers or switches, and so the system can maximally use each network device as a kind of security device (e.g., firewall).

Routing rule generator forms routing paths to control each network flow. When forming routing paths, this module checks security polices of each user to fulfill their needs. For e.g., if a user defines a security policy that mentions all network flows to port 80 should be checked by a NIDS attached to a router A, then this module created (a) routing path(s) which allows all network packets routing to port 80 pass through the router A. It helps the system assign security needs to each security device depending on value and usefulness

Flow rule enforcer allows flow rules to each OpenFlow router and switch. If the response manager allows response strategies or the routing rule generator creates routing paths, this module converts them into flow rules that could be recognized by OpenFlow routers/switches. After conversion, it sends converted rules to concerning routers or switches.

Data manager acquires network packets from routers or switches to hold up to some security devices send their detection results to NETSEC. It holds packets are for allowing some in-line style security functions as how generic Intrusion Prevention Systems provide. This module does not hold packets all the time, but only captures and stores when needed.

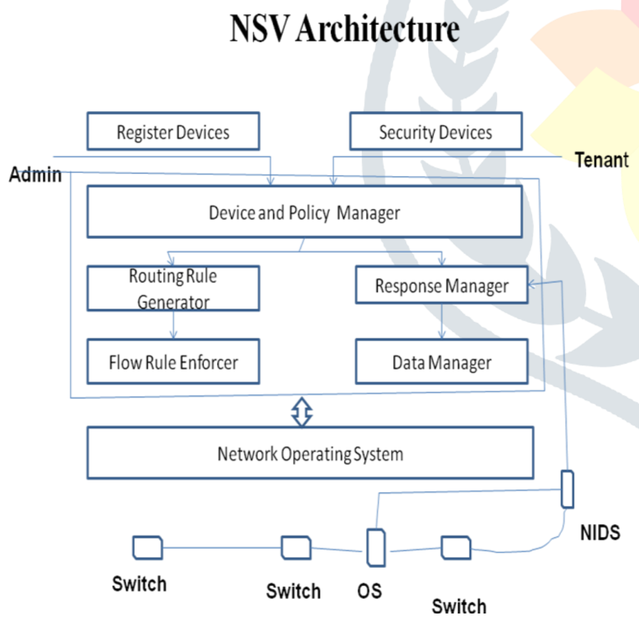

Figure 1: NSV Architecture

Working of NETSEC

A network administrator records network security devices both physical devices and virtual appliances to NETSEC. After registration cloud users required to create their security requests and send them into NETSEC. Then NETSEC analyzes the submitted security requests to understand the aim of users and writes the corresponding security policies to policy table. Next if NETSEC receives a new flow setup request from a network device it checks whether this flow is matched with any submitted policies. If it is NETSEC will create a new routing path and corresponding flow rules for the path, at this time NETSEC assures that the routing path includes required security devices that are defined in a matched policy i e the first NSV function. After this operation it allows flow rules to each corresponding network device to forward a network flow. If any of security devices detects malicious connection/content from monitored traffic they will report this information to NETSEC. Based on the report and submitted policies NETSEC enables a security response function to respond to malicious flows accordingly.

3.1 Registration of Security Devices

To use pre-installed fixed security devices, a cloud administrator requires to record them to NETSEC using a simple script language. The script language asks for the following information in registration: (i) device ID, (ii) device type (e.g. firewall and IDS), (iii) device location (e.g., attached to a router A), (iv) device mode (passive or in-line), (v) supported functions (e.g., detect HTTP attacks).

3.2 Creation of Security Policies

After a network administrator record security devices for a cloud network to NETSEC, the information of the recorded security devices is shown to users using the cloud network by NETSEC. Then, the users can define their security requests taking into account recorded security devices and security functions allowed by NETSEC. The script for a request consists of 3 fields: (i) flow condition, which describes the flow to be observed, (ii) function set, which defines the needed security devices for observing or investigating, and (iii) response strategy, which defines how to manage the flow if a threat is detected. The policy syntax is:
{{flow condition}, {function-list}, {action-list}}. Currently, NETSEC supports 5 different response strategies and they are drop, isolate for passive mode and drop, isolate, redirect for in-line mode. Here, it provides an example script for the following security request: one user (IP = 10.0.0.1) wants all HTTP traffic regarding to his IP to be observed by a firewall and IDS, and it wants to drop all packets detected as attacks by the firewall and the IDS. This request can be sent to NETSEC with the following script:
{{((DstIP = 10.0.0.1 OR SrcIP = 10.0.0.1) AND (DstPort = 80 OR SrcPort = 80))}, {firewall, IDS}, {drop}}.
Finally, NETSEC receives security requests from each user, and it converts them into security policies that can be suitable to a SDN enabled cloud network. At this time, NETSEC requires to convert user described high-level constraints into more specific network level conditions, and it also maps function set into security devices registered before.

3.3 Decision of Routing Paths

If NETSEC finds network packets meeting a flow condition specified by a security policy, then it will direct these packets to fulfill security requirements. When NETSEC forward network packets, it should take into account the following two things: (i) network packets should pass through specific security devices to meet the security needs, and (ii) the produced routing paths for network packets should be optimized. There are various existing routing algorithms for intradomain to find shortest paths. However, they cannot be used directly for our case. Since network packets only consist of the source and destination information, existing routing methods cannot discover needed ways to locations where security devices are fixed. .
NETSEC supports two modes of security devices which are passive mode and in-line mode. For a passive mode device, it can route the traffic to pass through the device, or just mirror a

duplicate to the device and forward the original traffic in another way. For an in-line mode device, all traffic should pass through and be observed by this device. The generated routing path should meet the needs from different modes of security devices. Also, a network may contain only passive mode devices or in-line mode devices, or both the two kinds.

Latest software-defined networking technologies (e.g., OpenFlow) serves several interesting functions and one of them is to control network flows as per our desire. With the help of this function, we propose a routing algorithms, which can fulfill requirements. It defines the following 4 terms to explain our algorithms more clearly: (i) start node, a node sends network packets, (ii) end node, a node receives the packets, (iii) security node, a node mirror packets to a passive security devices, and (iv) security link, a link on which in-line security devices are located. . To describe the proposed algorithm are clearly, we will provide concrete example to illustrate the key concept of each algorithm. For the illustration, we use a simple network structure as shown in Figure 2.

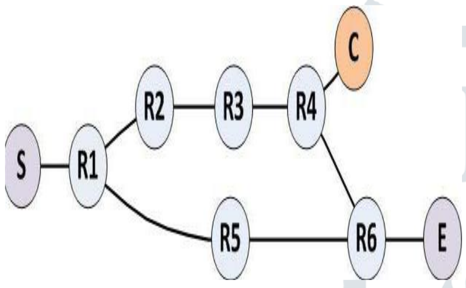

Figure 2: Layout

It contains six routers (R1 - R6), a start node (S), an end node (E), and a security device (C) attached to node R4 (thus R4 is a security node). We assume that node S sends packets to node E, and our example security policy is specified that all packets from node S to node E should be inspected by security device C. Furthermore, Figure 3 shows the traditional packet delivery based on the shortest path routing without considering the need of security monitoring.

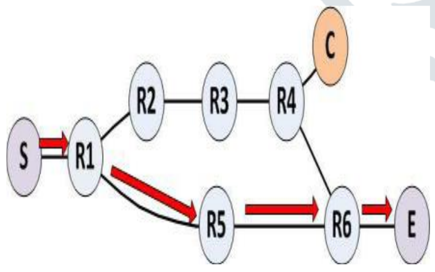

Figure 3: Shortest Path

Thus, packets from node S are simply sent through the path of (S → R1 → R5 → R6 →E), and obviously in this case they cannot be checked by the security device C. Next we will describe how our new algorithm work and illustrate them on the same network structure.

3.3.1 Advanced -Shortest

OpenFlow supports the function of sending out network packets to multiple outports of a router simultaneously, and it can create multiple redundant network flows. Thus, we try to propose an enhanced version of Algorithm. This approach finds the shortest path between a start node and each a node, which is nearest to a security node and in the shortest path between a start node and an end node. If it finds the node, it asks this node to send packets to multiple output ports: (i) a port, which is connected to a next node in the shortest path, and (ii) (a) port(s), which is (are) connected to (a) node(s) heading to (a) security node(s). Therefore, network packets are sent through the shortest path, and they are sent to each security node as well.

Here shortest path is found by considering the number of hops & energy being consumed by the system. This approach is presented in Algorithm.

Advanced Shortest Algorithm

```
Input: S (start node)
Input: E (end node)
Input: Ci = security node i , i = 1, 2, 3, .., n
Output: FPj , multiple shortest paths)
P0 = find_shortest_path(S, E);
FP ← P0;
foreach Ci do
foreach n j in P0 do
T Pi, j ← find_shortest_path(Ci , n j );
for each Ci do
        for each nj in P0 do
               ej = Cal energy(nj);
             TPi,j ← find_ shortest_path(Ci, nj, ej);
              FP ← T Pi, j ;
       End
End
Find_shortest_path(Ci, nj, ej)
      t1=√(ci – nj)2;
      t2=ej*0.5;
      Mj=(t1*0.5)+t2;
Return min(m);
```

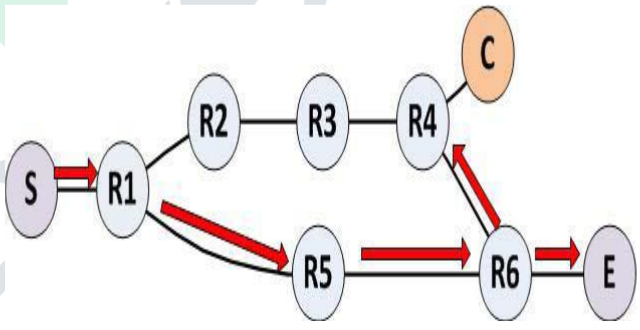

Figure 4: Multipath- Shortest

Figure 4 presents an example scenario for this algorithm. It first finds the shortest path between S and E, and it discovers the shortest path between R4 and nodes on the found shortest path, which is R6 → R4.

Enabling Security Response Functions

NETSEC gives a way of 5 security response strategies, and they do not require adding physical security devices or changing network configurations for managing packets.

In this passive mode, NETSEC supports two response strategies. First, NETSEC can drop packets that relate to detected network flows. This strategy is beneficial to stop some later malicious packets in the flow, but it does not give surety that no malicious packets are delivered to the target host`. Second, NETSEC can separate a specific host or a VM, if it is detected as malicious. In this strategy, NETSEC is able to avoid sending network packets to a detected host or a VM, or from a detected host or a VM. A user can specify which kind of packets should be blocked.

3 Evaluation

Virtual Network Environment: We select Network Simulator -3, which is popularly used for emulating OpenFlow network environments, to emulate 6-router network topology. We create a 6- router network topology with 6 OpenFlow-enabled switches (2 LinkSys switches and 4 TP-Link switches), and 2 hosts for a client a server.

Generation Time and Network Cost Measurement

We calculate four metrics to find the performance overhead of NETSEC. First, we calculate the flow rule generation time of Multipath and Advanced shortest algorithm. Second, we estimate the response time between a client host and a server host, when NETSEC sets up a routing path between them.

Third, we measure the network cost which represents total cost when packets are delivering a packet between a start node and an end node, and this cost can be formulized as the following formula: ∑i,j∈M ci,j, is the unit cost for flow along the arc between two nodes I and j, and I, j are pairs of nodes belonging to path.

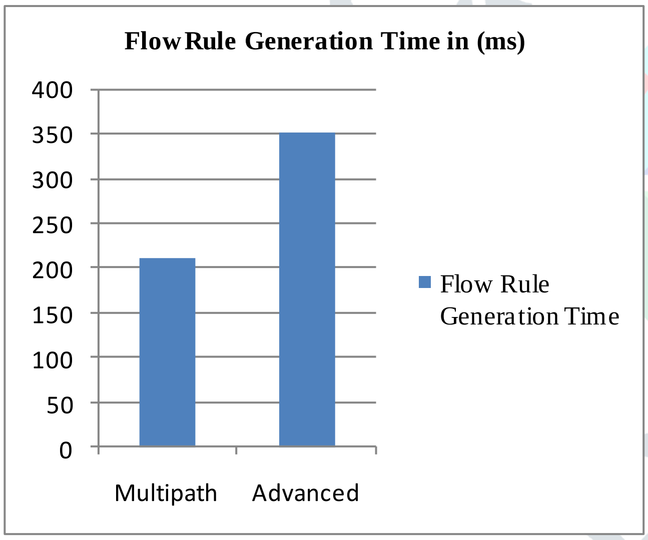

Figure 5: Flow Rule Generation Time

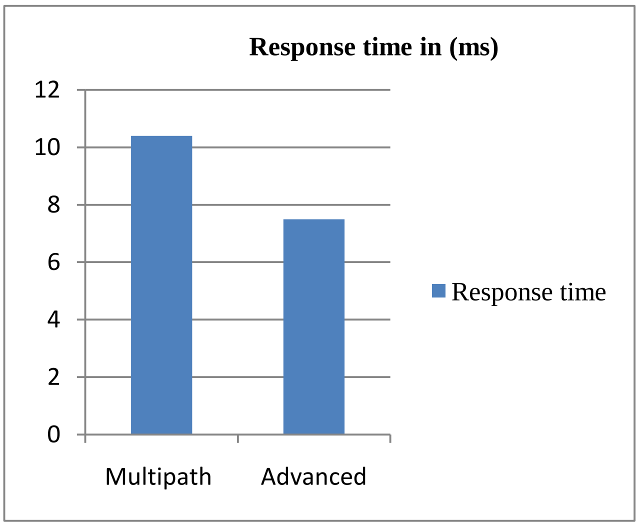

Figure 6: Response Time

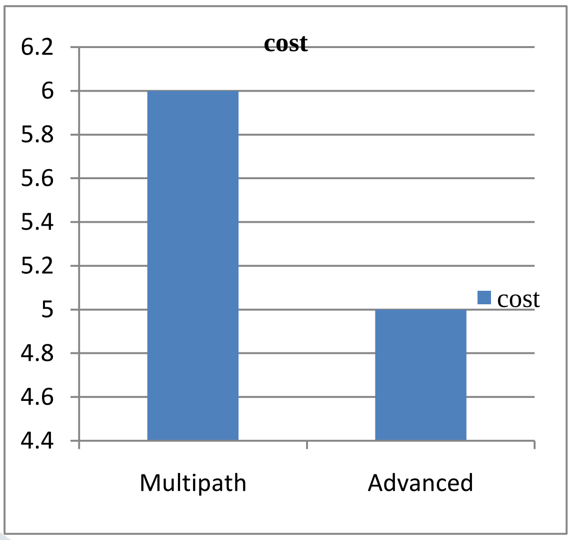

Figure 6: Network Cost Measurement (in no. of Hops)

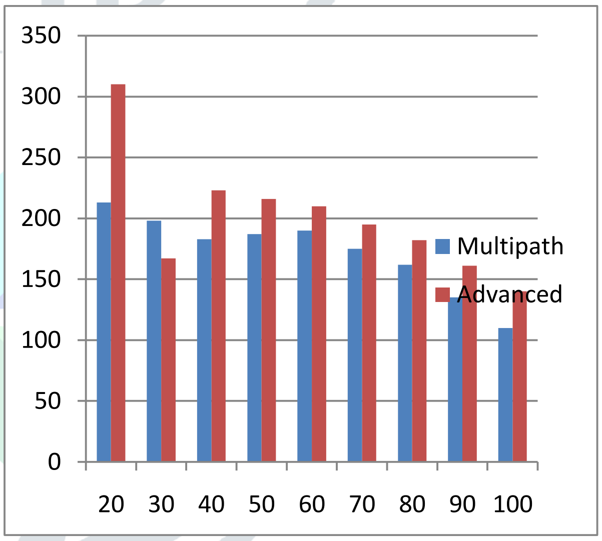

Figure 7: Throughput (in Mb)

Fourth, we measure Throughput, i.e. No. of packets delivered over a channel. Fifth, we measure energy consumption of devices. Sixth, we calculate PDR i.e. packet delivery ratio.
When we measure each metric, we compare our routing algorithm with multi-path shortest. Based on comparison, we can estimate the overhead of the proposed routing algorithm
.

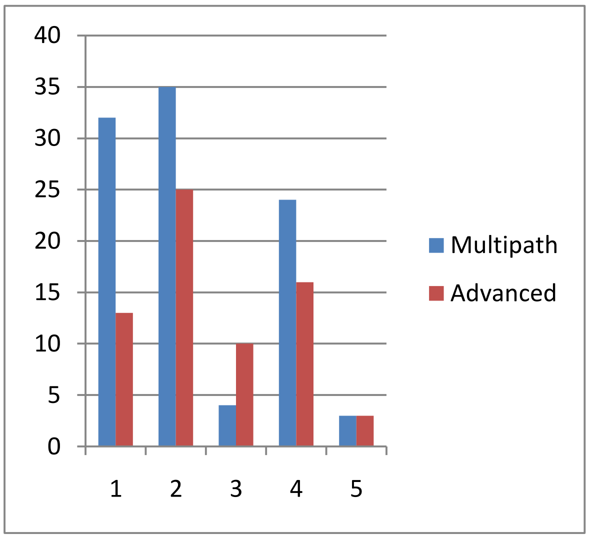

Figure 8: Energy Consumption Multipath shortest and Advanced shortest algorithm in microwatts

4 Comparison

Table 1: Comparison of Multipath Shortest algorithm with Advanced Shortest algorithm

Parameters	Multipath-Shortest	Advanced Shortest
Response Time	10.4 ms	7.5 ms
Network cost	6 hops	5 hops
Rule generation Time	210 ms	350 ms
Energy Consumption	153 mw	67 mw

The results for routing path generation time are shown in table. We can observe that the proposed routing algorithm add relatively high overhead compared with the baseline module. However, the time required for generating the floe rules greater because it generate the flow by considering the no. of hops and also the energy consumed by the devices.

5 Conclusion

This paper introduces a Concept of Network security virtualization (NSV) that can virtualize security resources/ functions and provide security response functions to network devices at required time. It implements a new prototype system NETSEC, to show the application of NSV. NSV prototype system can be used in complex networks like cloud. NSV enables the Administrator a great control over a network infrastructure. NSV can virtualizes specific network functions and allow then to run as individual nodes connecting with other communication and network services.

References

[1]J. R. Ballard; I Rae, and A. Akella, “Extensible and Scalable network monitoring using openSAFE” in Proc. USENIX Internet Netw. Manage Conf. Res. Enterprise Netw 2010,p.8.

[2]V. Heorhiadi, V. Sekar, and M. K. Reiter, “New opportunities for load balancing in network-wide intrusion detection systems,” in Proc. ACM CoNEXT, 2012, pp. 361–372.

[3]R. Wang, D. Butnariu, and J. Rexford, “OpenFlow-based server load balancing gone wild,” in Proc. 11th Hot-ICE, 2011, p. 12.

[4]S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, and M. Tyson, “FRESCO: Modular composable security services for software-defined networks,” in Proc. 20th Annu. Netw. Distrib. Syst. Secur. Symp. (NDSS), Feb. 2013, pp. 1–16.

[5]S. Shin, V. Yegneswaran, P. Porras, and G. Gu, “AVANT-GUARD: Scalable and vigilant switch flow management in software-defined networks,” in Proc. 20th ACM Conf. Comput. Commun. Secur. (CCS), 2013, pp. 413–424.

[6]G. Gibb, H. Zeng, and N. McKeown, “Outsourcing network functionality”, in Proc. ACM SIGCOM workshop Hot Topics Softw.Defined Netw. {HotSDN), Aug.2012,pp. 73-78.