Deployable SDN architecture for network applications: an investigative survey

Shravanya Gangadhara
Department of Computer Science and Engineering
PES University
Bengaluru, India
shravanya.g17@gmail.com

Swati Nagaraj Hasyagar
Department of Computer Science and Engineering
PES University
Bengaluru, India
swatinagaraj2@gmail.com

Uma Damotharan
Department of Computer Science and Engineering
PES University
Bengaluru, India
umaprabha@pes.edu

***Abstract*— Software Defined Networking (SDN) is acclaimed as the future of networking as it provides disaggregation between software and hardware and enables a central point of control over the entire network. Many network applications such as load balancers, IPv4-IPv6 converters, VMs, IDS and IPS systems can be realised through SDN deployments. Implementing an SDN solution for the aforementioned network applications would guarantee better performance. As part of this survey, research papers have been reviewed to understand various network applications, their shortcomings in the current deployments and a corresponding proposed SDN solution to tackle these shortcomings. Accordingly, proposals have been made with respect to the deployment strategies of these SDN network applications. The three approaches to deploy SDN solutions are: the rip and replace model, the overlay model and the hybrid model. The main focus of this paper is to propose the best model out of the three models for an SDN deployment for each of the discussed network applications.**

***Index Terms*— DDoS Attacks, Honeypots, Hybrid SDN, IPv4/IPv6 coexistence, Load Balancing, Software Defined Networks, Virtual Machine Migration**

I. INTRODUCTION

Software Defined Networking is an upcoming networking paradigm that has the potential to replace the Internet and all enterprise networks. Numerous research undertakings are probing into various SDN deployment techniques in place of the existing legacy networks. There are two primary reasons for the triumph of SDN architecture over the legacy networks [1]. Firstly, SDN is arranged into a three layered architecture consisting of the data plane layer, the control plane layer and the application layer as depicted in the figure alongside. Secondly, SDN provides the control plane layer with a central view of the entire network thus providing a single point of contact and control.

The three layered architecture of SDN introduces abstraction between hardware and software components of the network. This ensures hardware and software disaggregation. The data plane layer is responsible for the forwarding action. The control plane layer or the SDN controller is responsible for executing the forwarding logic programs and protocols. Thus the controller controls the data plane layer and it's forwarding action. The SDN application runs on the application layer. The layered architecture enables software to change periodically, without disrupting the hardware or the forwarding action it is associated with. This level of abstraction remains unachievable in the existing legacy network as the proprietary network devices have inter-dependant components which enforces a lock-down of the hardware and the software.

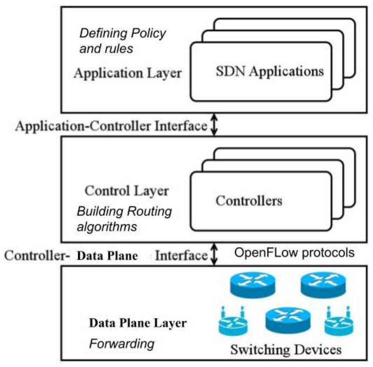


Figure 1: Three-layered architecture of SDN

From the above architecture, it can be observed that the data plane layer switches have to interact with the controller when a forwarding decision has to be made. The data plane layer is only responsible for the hardware components of the network and remains unaware of any intelligent routing algorithms. Since the control plane has access to these applications, it governs the logic behind packet forwarding. Consequently, the data plane has to contact the control plane whenever a new route has to be created. When a data plane layer switch contacts the controller, the controller ensures that flow tables(routes) are installed in all the switches that forward the corresponding packets. Since the controller possesses a central view of the entire network, it can optimally create the flow table. In the case of legacy network architectures, each router runs an algorithm to determine where the packet should be forwarded to. Legacy network routers do not possess a central view of the entire network.As a result, they contribute to excess amounts of inter router communication traffic generated from distance vector or link state routing algorithms. The SDN architecture can forward packets in a similar manner as compared to legacy network, but in a more optimal manner with less horizontal traffic. This flexibility is highly appreciated in emerging technologies such as mobile computing, cloud-computing and big data environment as it provides high accessibility, bandwidth and efficiency in handling network traffic. The main focus of this survey paper is to help deploy SDN based network application solutions by considering a suitable deployment model. In spite of a fair amount of research on deployment mechanisms, there still persist several key challenges in this field. One of the major challenges with building SDN enabled solutions is that there are no established standards for it in the network industry. Different vendors offer different implementation approaches to SDN solutions. Adoption of this technology among enterprises that have smaller networks and fewer resources has also been relatively slow due to various implementation issues. The following section explains various models that are available to be deployed or to be implemented by SDN architectures.

There are three models to implement any SDN network viz. Rip and Replace, Overlay and Hybrid [24]. The Rip and Replace model can be achieved only with pure SDN-enabled devices that only use Open Flow protocols. The Overlay model works with SDN controllers that build an SDN overlay virtual network on top of an existing legacy network. The Hybrid model works with networking equipment that supports OpenFlow as an additional protocol within their software stack. Therefore, the data plane flow tables are dictated by the SDN controller and also by the protocols the switch runs on its own like the BGP. This model offers the advantage of low disruption in the existing network to implement SDN architectures.

There are various network applications in the legacy networks for which an SDN solution would act as their better counterpart. These network applications could lie in the access network such as load balancers, IPv4 to IPv6 converters and VMs (Virtual Machines). Alternatively, they could form a part of the core network like IDS (Intrusion Detection Systems) or IPS (Intrusion Prevention Systems) inclusive of Honeypots. There are many shortcomings of the existing network applications that can be resolved by a corresponding SDN deployment of the same. The remaining part of this section addresses the shortcomings of network applications and the corresponding SDN solutions.

With an explosion in the number of IPv4 network devices, the IPv4 pool is exhausting. The slow deployment of IPv6 has created an urgent need for exploring feasible solutions to aid the transition. By leveraging SDN, an efficient solution can be built to translate IPv4 addresses into IPv6 address and vice versa [2]. With growing internet and web applications, the traffic load on the network and servers are very elastic. In order to prevent individual servers from being overwhelmed by the sudden fluctuations, we deploy load balancers for distributing traffic effectively among the servers. Consequently, this leads to better power conservation, reduces server downtime and enhancing resource utilization [3]. In the perspective of making IT more flexible and cost-effective, employing virtualization technology helps users to pool their hardware resources and efficiently use the resources. As suggested in [4] nowadays, data centers host significant web content and web services. Due to improvement in server virtualization, live VM migration presents ways to optimally manage hosted services. Live VM Migration helps administrators to dynamically reallocate VMs to provide load balancing, energy saving means, etc without significantly disrupting any services. Security is an important aspect of VM migration.

Securing the network is critical because attacks can cause disruption of services. DoS/DDoS attacks are fatal to servers and may cause failure in a particular section of the internet if not tended to immediately. Since legacy network security appliances are moving towards software enabled approaches, SDN could act as an enabler to build a network security application that concentrates on fatal attacks like DoS/DDoS [5]. Honeypots are advanced security applications in the network that possess the capability to stop or completely deactivate an attacker system through intelligent algorithms [6]. This forms a vital part of the Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) in the networking domain. Legacy networks cannot satisfy the increasing infrastructural demands of these network applications. Hence the literature also observes and analyses the deployable SDN models.

The remainder of the paper is organized as follows: Section 2 reviews related work done on the above-mentioned network applications and the limitations they face. Section 3 discusses the approach taken by each paper to design an architecture for the SDN network applications. Section 4 analyses how these architectural designs have been implemented to realize SDN network applications. Section 5 examines the results obtained by the SDN implementations of the network applications. Section 6 critiques the SDN model used in the previous sections and suggests an alternative if any. This section also elaborates on any future work that could be done.

II. RELATED WORK

The research work done in the domain of chosen network applications and its limitations are investigated in the following section. The traditional approaches for *IPv4-IPv6 coexistence* discussed in [2] are described in this subsection. The first approach is Dual Stack which involves revamping IPv4 only devices to make them compatible with IPv6 only devices. Its key issue is that it is significantly costly. The second approach is tunneling which encapsulates IPv6 packets in IPv4 packets and routes them in IPv4 traffic. Its key issue is lack of optimality, where users of new architecture cannot use services of the underlying architecture. The third approach is Address Translation using NAT64 which performs IP header and address translation and facilitates communication between IPv6 only and IPv4 only hosts. Its key issue is that NAT64 has limited capacity.

Since *Load balancing* is crucial to the growing network traffic for optimal energy and resource utilization, an SDN implementation of the same is highly necessary. The existing ways to achieve load balancing are: through protocols, specialized software, specialized gateways and splitters. The protocol approach is achieved through OpenFlow protocols [7] which provides an open protocol [8] to program the flow tables in different switches. Although this is an SDN approach for implementing load balancers, it will bring additional flow expenses for collecting information from servers. The specialized gateways make use of algorithms like RR-DNS scheduling algorithm [9]. This approach however, faces the problem of poor reliability and maintenance. In case of fail-over where one or more of the servers happens to fail, the client gets delayed by a considerable amount of time until the new server IP address is suggested by the load balancers. Reference [3] proposes a better approach to build SDN based load balancers using Finite State Machines (FSMs).

Reference [4] discusses the challenges of *Live Virtual Machine(VM) Migration and concerned network issues* of traditional approaches that are described in this subsection. One of the challenges in traditional approaches limits VM migration to LAN environment since the requirement of live VM migration is to maintain its initial Media Access Control(MAC) and IP addresses post migration. The other challenge is that network state updates are hard to control and predict. We could update each and every router and switch after VM migration, but the distributed nature of routing functionality causes unpredictability that may result in routing loops.

Traditionally for cross data center live VM migration using SDN, an approach which associated two suites of IP and MAC addresses with each VM was proposed. Assigning two suites of address makes it necessary to replace headers and map very often. This approach is highly dependent on underlying network topology, compelling it to be similar across data centers. Key challenges to inter-data center and intra-data center live VM migration are described as follows:

1) Inconsistent view of VMs information between network controller and Cloud OS: During migration process, the live application is associated with the initial VM by the network controller. This will result in incorrect routing paths of packets that are concerned with the running instance . There exists one network controller for each data center. Hence, making the topology even more complicated for multiple data centers.

2) Insufficient flexible routing under subnet location expansion: The VMs belonging having same IP and gateway, thus belonging to the same subnet, but when located in different places are described to be in subnet location expanded topology. Across subnets traditional layer 3 will fail. But in SDN, each data center has a network controller. So, subnets of the same data center are controlled by the network controller, which handles routing effectively. Subnets across data centers, cannot be controlled by network controllers in data centers since the Internet is involved in routing Inter-data center routing.

3) Inadequate support for Live Migration: It is necessary for the service downtime caused by migration to be within a tolerable range of service level agreement.One of the significant network Constraints in Virtualisation discussed in [10] is described in the below subsection.

In the below subsection, there is a paradigm shift from network applications in access network to the core network. The core network applications discussed are Intrusion Detection Systems and Honeypots. In spite of the paradigm shift, the SDN solutions and their deployment strategies remain constant. Several attacks can disrupt live migration. Single domain or inter-domain security is mandatory. Since OpenStack does not authorize all possible operations when interdomain operations are involved, it is a limitation in the real deployment. Reference [23] aims to exploit benefits of SDN and NFV to virtualise and migrate multimedia network functions across hosts.

Since *DoS/DDoS attacks* are fatal to servers, reaction time of the security tools must be considerably good. DoS/DDoS attacks occur at an exponential rate. This is the reason that the existing network security tools that are compatible with legacy networks are insufficient to address the issue. Hence all network security tools being built are highly time sensitive. New challenges faced in this domain is that few new DoS/DDoS attacks have the capability of slowly attacking the host making it all the more difficult for time sensitive systems to detect it [11]. Hence many SDN based solutions which cover all these cases have been looked into as part of the literature. While few of them talk about the identification and mitigation of the attacks and how they cause congestion in the network, few other talk about how secure they make the network for other network applications to run in a secure way [12], [13]. Few other SDN solutions cited also talk about how security can protect the end user from malicious attacks [14] - [18]. Thus these focus on the security issues targeted at the access network. The approaches taken include OpenFLow, sFlow and similar protocols which have few constraints. The constraints include the unpleasant effect of congesting the control plane during the monitoring, detection and mitigation phases where the major role is played only by the control plane. Another approach that has been looked into is redirecting all the traffic to a Intrusion Detection System (IDS). This would suffer from decreased throughput and increased latency issues. Reference [5] addresses such issues related to time delay and congestion control by proposing an in-switching mechanism in SDN implementation of DoS/DDoS network security application. *Honeypots*, as stated earlier, is an advanced network security tool. Many hybrid networks are using the hybrid architecture from the early 2000s since it provides a more secure network and the benefits associated with hybrid honeypot architectures have not yet been met by any other architecture thus far. The hybrid honeypot architecture refers to redirecting the traffic from a Low Interaction Honeypot(LIH) into a High Interaction Honeypot(HIH). However, the lack of a traffic filtering mechanism could easily result in invalid data flooding to the HIH. Other hybrid honeypots which use a different architectural styles exist. The architecture on of them use is by encapsulating the packets in GRE tunnel thus making the communication between LIH and HIH difficult to detect. Reference [19] use a mechanism that deals with handing over the connection from frontend to backend thus redirecting the traffic. In addition, Reference [20], [21] proposes another hybrid honeypot architecture specifically addressing the identical-fingerprint problem that most hybrid honeypots suffer from. The idea has few downsides although it looks very promising. The downsides to the approach is that the frequency of changing switch states up and down is enormously high hence limiting the possibility of large scale deployment. Reference [6] proposes a novel SDN solution for the TCP connection handover mechanism which addresses the above issues.

III. Methodology

The SDN architecture employed by various network applications is explained in detail in this section.
The necessary IPv4/IPv6 address conversion is given in [2]. The exchanging of packets between IPv4 and IPv6 is handled by programmable OpenFlow. OpenFlow Switches forward IPv6-IPv4 traffic to NAT64. NAT64 repacks IPv6 packets into IPv4 packets. SDN controller which is centralized manages the NAT64 pool. It monitors each NAT64s utilization and checks flow speed from OpenFlow Switches.

Load balancing is done using NAT64 service pool. Depending on IPv6 traffic, the corresponding number of NAT64 nodes are created in a virtualized way. An SDN approach to implement *load balancing* is better suited the needs of the solution as it requires better reliability. Also the SDN approach gives a logically centralized view[19] making it easier to address issues related to distributing loads and single point failures. Reference [3] propose an SDN approach with an advancement of FSMs to build load balancers. Programmable SDNs are used as the solution. Specifically Event-driven programmable SDNs [20]. In event-driven network controlling, each and every policy/functionality undertaken by the switch is anonymous to the change in states. These changes in states are triggered by various events like forwarding, intrusion detection and so on. Since the switch functionality can be reduced to a set of states as mentioned, it can be represented by a Finite State Machine(FSM). The FSM is present in each of the data plane component but the functionality of the FSM is dictated by the control plane thus ensuring a logically centralized control. Defining the FSM in one centralized place enables the capability to define the principle on which the entire network has to work on.

NFV, SDN, OpenStack are state of the art networking technologies in virtualization[10]. An SDN approach for *live VM migration* is more effective because it overcomes the traditional approach challenges by providing flexible programmability due to data plane and control plane separation. Hence it overcomes LAN environment restriction since it has single logical view of the entire network, thus enabling the controller to create new flows that can be deployed on the control plane. SDN technology provides centralized control which optimizes network state update.

An SDN approach to implement network security applications that detect and mitigate threats like *DoS/DDoS* is best suited due to its crucial role. The SDN approach helps in faster detection but most importantly an optimum way of mitigation. Since the control plane is responsible for detection and mitigation, it can do so in the most optimum way as it is aware of the entire network. The SDN is a powerful approach since the control plane can mitigate the attack to switches which may not be directly connected to it. This was not possible in the previous legacy network approach. Reference [21] proposes a unique solution using stateful SDNs to secure access networks from DoS or DDoS attacks. The normal SDN compatible switches are stateless as they are not required to retain session information or status about each communication (here forwarding packets or performing any other action on the packets). The controller on the other hand takes all these responsibilities of maintaining session information. But when building an application to detect DoS or DDoS attacks, one major factor we have to consider is reaction time. Most of the SDN solutions depend entirely on the control plane layer to perform functions like monitoring, detection and mitigation. With this approach, the control plane's functionality is not only limited to stateful forwarding decisions but also processing of the application, thus being overloaded. This might lead to improper functioning and bugs in the controller. To eliminate these limitations and constraints, [21] introduce in-switch processing functionality to identify and mitigate DoS/DDoS attacks. In stateful SDNs, the forwarding table is a combination of multiple Finite State Machines (FSMs). In the FSM approach, each switch is reduced to a state and each forwarding or dropping function is treated as an event that triggers the change of state. Statesec utilizes the effective monitoring capabilities provided by stateful SDN, which involves delegating local processing inside switches. This reduce the burden on the controller which can now achieve better detection percentages. The three common steps required to handle DDoS protection are : 1) traffic monitoring 2) attack detection 3) attack mitigation. In the in-switch processing approach, only the mitigation process is left for the controller and the processing inside switches include monitoring and attack detection. *Honeypots* are the most advanced network security applications. An SDN solution for Honeypots is crucial since the existing legacy network does not support Honeypots with greater processing capabilities and wider network reach. The related work section gave a brief of all SDN approaches taken thus far. The major challenge that persists is the communication between the LIH(frontend) and HIH(backend) in the hybrid architecture of honeypots. Reference [22] addresses these issues of hybrid honeypot architecture through SDN solutions. The problem this paper tries to address is the redirection of TCP traffic from the frontend to backend for further packet analysis. During this redirection, particularly the approach taken by TCP connection handover mechanisms are vulnerable to attacks such as port scans which could be easily visible to the black hat hackers. Thus an SDN based control plane layer is proposed in the paper under study. The paper aims at implementing a hybrid honeypot systems to control the packet transmissions. The controller ensures that the communication remains transparent by redirecting the communication to a specific honeypot for further investigations. To implement this approach, SDN switches which are OpenFlow compliant are used in the infrastructure layer while the control layer consists of an Ryu SDN controller accompanied with a Snort alert mechanism.

IV. IMPLEMENTATION

The analysis of implementation of SDN architectures of the network applications under survey is done in this section. To implement *IPv6/IPv4 coexistence* [2] using SDNs we need an SDN controller, like Ryu, a DNS64 service which provides IPv6 formatted IPv4 address, an IPv4 router, an IPv6 router, a NAT64 service pool which translates an IPv6 address to IPv4 address and vice versa, OpenFlow switches that forward network traffic. As shown in the below diagram, OF-SW forwards traffic to NAT64 node when IPv6 host wants to communicate with IPv4 host. NAT64 repacks IPv6 packets to IPv4 packets. SDN Controller monitor's NAT64 utilization and creates more NAT64 nodes if existing NAT64 nodes are overloaded, hence does load balancing. OpenStack updates flow table with addition or deletion of NAT64 nodes. An architecture like this can be easily adopted into a hybrid SDN implementation.

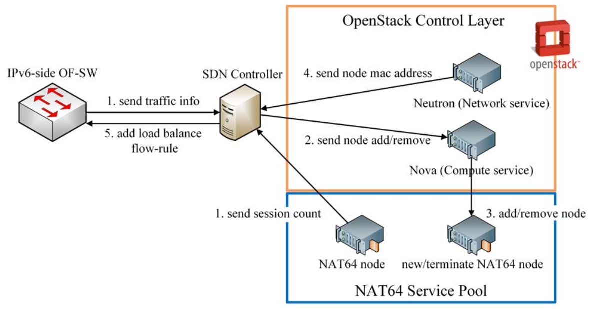
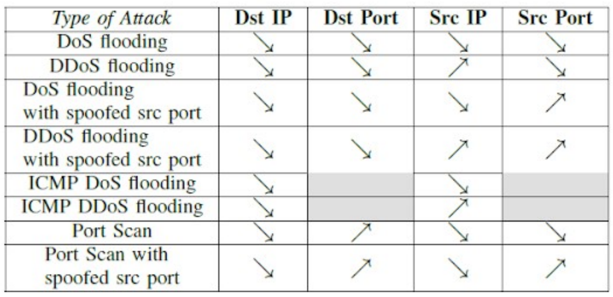


Figure 2: Message flow of IPv6 migration system over SDN

Reference [3] have implemented *Load Balancing* through an SDN approach using PyResonance which is Resonance implemented on Pyretic. Pyretic is used as the implementation platform for event-driven network controlling. The implementation of the above project was done in simulation only on the Mininet [19] platform. The FSM is implemented as follows with 4 modules to achieve load balancing:
Module 1 :Load balancing Module is responsible for monitoring the traffic to detect a trigger of an event.
Module 2 :Event Handler Module is used to receive the state of each switch from the previous Module and trigger the change through transitional signals.
Module 3 :Finite State Machine Module implements the Finite State Machine which is responsible for the actual transitions between states.
Module 4 : This module takes care of default action like forwarding that should occur.

This subsection proposes *Live VM migration procedure* [4] in the following phases.
1) Before migration: Cloud OS verifies migration condition. The commencement of this event is notified to the network controller, following which routing path is established in coordination with orchestrator.
2) During Migration: In this phase, memory and storage are transmitted to the destination server. Network controller executes partial network state migration in prior, hence reducing migration time.
3) After Migration: The initial VM is shut down by the Cloud OS. The new VM is started and these events are notified to the orchestrator and the network controller, which respectively update the necessary relevant information.

The SDN implementation of network security applications that detect and mitigate *DoS and DDoS Attacks :* can be best explained by dividing it into 3 functionalities - monitor, detect and mitigate.
Monitoring - The StateSec implementation involves two independent processes inside the switch - 1) forwarding packets as per controller instructions 2) monitoring of traffic features. The monitoring happens according to the FSM deployed on the switch by the controller, thus the states and transitions are known in advance. The controller initializes - a flow table, look-up scope and update scope for keeping track of necessary features in the packets it forwards.
Detection - An entropy based algorithm [5] as shown in Figure 3 is used to detect if incoming traffic is flowing only from one source IP address or same port addresses. The gradual increase or decrease in the entropy of the related counters gives some information regarding if the packets are DoS or DDoS attacks. The entropy based algorithm is used extensively in such applications and an instance of it is being presented in this paper. Having said that, it is not always ensured that the algorithms accuracy is the best. The algorithm also gives rise to few false positives and the same could be avoided by increasing the sensitivity of the algorithm through thresholding techniques.



Type of Attack	**Dst IP**	**Dst Port**	**Src IP**	**Src Port**
DoS flooding	↘	↘	↘	↘
DDoS flooding	↘	↘	↗	↘
DoS flooding with spoofed src port	↘	↘	↘	↗
DDoS flooding with spoofed src port	↘	↘	↗	↗
ICMP DoS flooding	↘		↘	
ICMP DDoS flooding	↘		↗	
Port Scan	↘	↗	↘	↘
Port Scan with spoofed src port	↘	↗	↘	↗

Figure 3: Entropy based algorithm based on the above parameters

Mitigation - Once an attack is detected by the switch, it lets the controller know of the same. The controller comes into action only at this point. The controller would now be responsible to deploy new flow rules in all switches that have potentially been exposed to packets that are part of the attack. The controller would also have an option to perform complex functionalities on the packets like redirecting the packets to a Deep Packet Inspection (DPI).

Reference [6] implement this approach with SDN switches which are OpenFlow compliant being used in the infrastructure layer and the control layer consisting of an Ryu SDN controller accompanied with a Snort alert mechanism. The

implementation can be divided into 3 phases namely:
Phase 1 - Establishment of communication between attacker and honeypot system.
Phase 2 - Transferring of the TCP packets from the LIH to the HIH by replaying the received TCP traffic.
Phase 3 - TCP session has reached the HIH and synchronization of packets is in process.

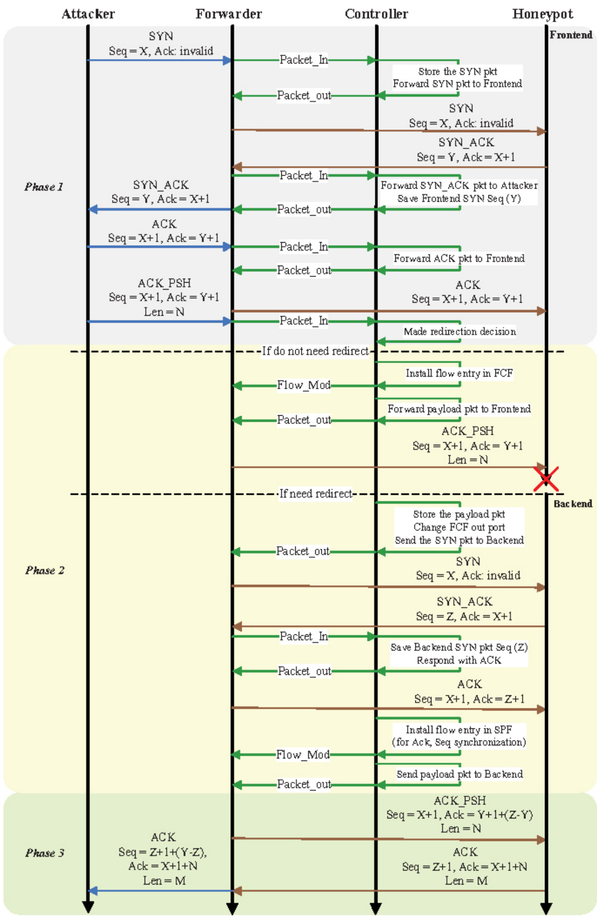

Attacker
Forwarder
Controller
Honeypot
Frontend
Backend
Phase 1
Phase 2
Phase 3
SYN Seq = X, Ack: invalid
Packet_In
Packet_out
Store the SYN pkt Forward SYN pkt to Frontend
SYN_ACK Seq = Y, Ack = X+1
Forward SYN_ACK pkt to Attacker Save Frontend SYN Seq (Y)
ACK Seq = X+1, Ack = Y+1
Forward ACK pkt to Frontend
ACK_PSH Seq = X+1, Ack = Y+1 Len = N
Made redirection decision
If do not need redirect
Install flow entry in FCF
Flow_Mod
Forward payload pkt to Frontend
If need redirect
Store the payload pkt Change FCF out port Send the SYN pkt to Backend
Save Backend SYN pkt Seq (Z) Respond with ACK
SYN_ACK Seq = Z, Ack = X+1
ACK Seq = X+1, Ack = Z+1
Install flow entry in SPF (for Ack, Seq synchronization)
Send payload pkt to Backend
ACK_PSH Seq = X+1, Ack = Y+1+(Z-Y) Len = N
ACK Seq = Z+1+(Y-Z), Ack = X+1+N Len = M
ACK Seq = Z+1, Ack = X+1+N Len = M

Figure 4: Transparent TCP connection handover mechanism

V. Results

The SDN implementation of *Load balancing* using FSM [3] reveal the following. Only during the start of a new policy being learnt by the switches from the controller, the time consumed is going to peak, the rest of the time an amortized curve is observed as given in the below figure.

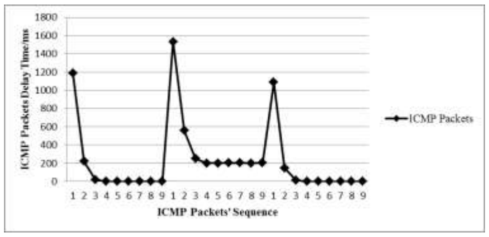


Figure 5: ICMP Packets Delay Time

The solution stated in [5] has resulted in the following. The proposed StateSec protocol can be observed to be much more time sensitive than sFlow protocols that were used previously. The In-switching concept introduced in the paper also proves to be an excellent design pattern that can implemented in many other network services and applications. StateSec has achieved 80% accuracy in identifying the types of attacks and 85% accuracy for true positives. Since slow attacks remain a challenge they result in false negatives that go undetected but the percentage of false negatives in StateSec is much lower than the one offered by sFlow.

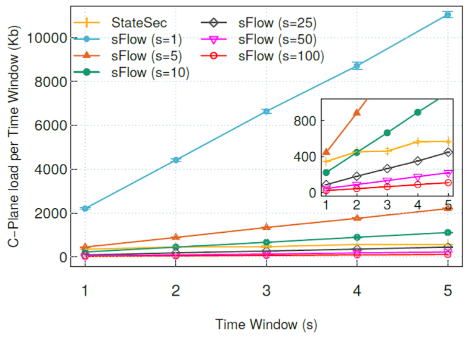


Figure 6: Comparison between sFlow and StateSec with respect to control plane load

The solution stated in [6] has resulted in the following. The graph obtained from the SDN approach is more Gaussian than it was in the non-SDN approaches which do not perform redirection. The equally-distributed packet I/O of Honeybrid is much sharper than the other two architectures explained in the related work section. Therefore, if the adversary uses the difference of the packet I/O performance to detect the redirection, the mechanism proposed by this paper is much stealthier than the Honeybrid gateways approach thus giving us better and desirable results.

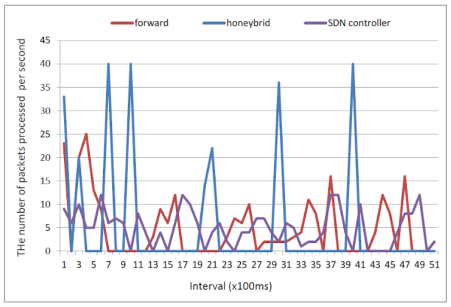


Figure 7: In-packet and out-packet graphs under different honeypot mechanisms

VI. Conclusions

In this survey, it is observed that SDN applications have successfully been implemented for various network functionalities. They include services offered by the access networks like NAT, IPv4 to IPv6 converters, VM Migration applications as well as services offered by the core networks like Intrusion Detection Systems and Honeypots. Although these papers have been successful in designing optimized solutions for these functionalities, deployment of these SDN approaches has not yet been achieved. This is due to the rip and replace paradigm that the above papers have considered, which might not be feasible anytime soon.

Rip and replace paradigm is an efficient architectural decision for data centre networks owing to the proprietary ease of replacing hardware. The same architecture would not work on an enterprise network since various stakeholders come into play. A deployable SDN alternative would be the Hybrid paradigm of SDN architecture. Since Hybrid paradigm consists of workarounds to allow both older protocols and newer SDN protocols to co-exist, it is a more deployable SDN approach. We have established the fact that SDN can address the growing needs of Information Communication Technology today. Thus it becomes crucial to deploy these SDN solutions at the earliest. We propose that Hybrid SDN Implementation model is the way forward to thoroughly exploit the benefits offered by SDNs.

REFERENCES

[1] W. Xia, Y. Wen, C. H. Foh, D. Niyato and H. Xie, "A Survey on Software-Defined Networking," in IEEE Communications Surveys & Tutorials, vol. 17, no. 1, pp. 27-51, Firstquarter 2015

[2] Lin, Jia-Jhun, Kai-Ching Wang, Shin-Ming Cheng and Yen-Chun Liu. On exploiting SDN to facilitate IPv4/IPv6 coexistence and transition. 2017 IEEE Conference on Dependable and Secure Computing (2017): 473-474.

[3] Zhou, Y., Ruan, L., Xiao, L., Liu, R. (2014). A method for load balancing based on software defined network. Advanced Science and Technology Letters, 45, 4348.

[4] Liu, Jiaqiang, and Depeng Jin. "Software defined live virtual machine migration." In Network Protocols (ICNP), 2013 21st IEEE International Conference on, pp. 1-3. IEEE, 2013.

[5] J. Boite, P. A. Nardin, F. Rebecchi, M. Bouet, V. Conan, "Statesec: Stateful monitoring for ddos protection in software defined networks", 2017 IEEE Conf. on Network Softwarization (NetSoft), Jul. 2017

[6] W. Fan and D. Fernandez, "A novel SDN based stealthy TCP connection handover mechanism for hybrid honeypot systems," 2017 IEEE Conference on Network Softwarization (NetSoft), Bologna, 2017, pp. 1-9. doi: 10.1109/NETSOFT.2017.8004194

[7] Zuo QY, Chen M, Zhao GS, Xing CY, Zhang GM, Jiang PC. OpenFlow-based SDN technologies. Ruanjian Xuebao/Journal of Software, 2013 (in Chinese).

[8] Open networking fundation. 2012. http://www.opennetworking.org

[9] Li Chuan Chen, Hyeon Ah Choi. Approximation algorithms for data distribution with load balancing of Web Servers[A]. In Proceedings of IEEE International Conference on Cluster Computing[C], 2001:274-281

[10] Ibn-Khedher, Hatem, Emad Abd-Elrahman, Hossam Afifi, and Jacky Forestier. "Network issues in virtual machine migration." In Networks, Computers and Communications (ISNCC), 2015 International Symposium on, pp. 1-6. IEEE, 2015.

[11] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments, Computer Networks, vol. 62, pp. 122136, 2014.

[12] M. Dhawan, R. Poddar, K. Mahajan, and V. Mann, Sphinx: Detecting security attacks in software-defined networks, in NDSS, 2015.

[13] N. G. Dharma, M. F. Muthohar, J. D. A. Prayuda, K. Priagung, and D. Choi, Time-based DDoS Detection and Mitigation for SDN Controller, in APNOMS, 2015, pp. 550553.

[14] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments, Computer Networks, vol. 62, pp. 122136, 2014.

[15] Y. E. Oktian, S. Lee, and H. Lee, Mitigating Denial of Service (DoS) attacks in OpenFlow networks, in ICTC, 2014, pp. 325330.

[16] C.-J. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang, NICE: Network Intrusion Detection and Countermeasure Selection in Virtual Network Systems, IEEE Trans Dependable Secure Comput, vol. 10, no. 4, pp. 198211, 2013.

[17] T. Chin, X. Mountrouidou, X. Li, and K. Xiong, An SDN-Supported Collaborative Approach for DDoS Flooding Detection and Containment, in MILCOM, 2015, pp. 659664.

[18] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, AVANT-GUARD: Scalable and Vigilant Switch Flow Management in Software-Defined Networks, in ACM CCS, 2013, pp. 413424.

[19] Open networking fundation. 2012. http://www.opennetworking.org

[20] H. Kim, A. Voellmy, S. Burnett, N. Feamster, R. Clark. Tech Report. Lithium: Event-Driven Network Control

[21] J. Boite, P. A. Nardin, F. Rebecchi, M. Bouet, V. Conan, "Statesec: Stateful monitoring for ddos protection in software defined networks", 2017 IEEE Conf. on Network Softwarization (NetSoft), Jul. 2017

[22] W. Fan and D. Fernandez, "A novel SDN based stealthy TCP connection handover mechanism for hybrid honeypot systems," 2017 IEEE Conference on Network Softwarization (NetSoft), Bologna, 2017, pp. 1-9.

[23] Ibn-Khedher, Hatem, Emad Abd-Elrahman, Hossam Afifi, and Jacky Forestier. Network issues in virtual machine migration. In Networks, Computers and Communications (ISNCC), 2015 International Symposium on, pp. 1-6. IEEE, 2015.

[24] Modern Networking and SDN, Edx, tinyurl.com/y83qgtzn.