VOL. 12, NO. 16, AUGUST 2017 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

DYNAMIC LOAD BALACING USING QUEUING INTERFACE SYSTEM

Sairam R, Neshan Anand and V Deeban Chakravarthy
SRM University of Science and Technology, India
E-Mail: sairam_r@icloud.com

ABSTRACT

Software-Defined Networking (SDN) has grown to become a very promising network architectural design in which network devices are controlled by a SDN Controller. Employing SDN offers an attractive solution for network security especially for Brute force attacks in SDN environments. This project analyzes the characteristics of the traffic flow up-streaming to an ISP server during both states of normal and brute force traffic attacks. Based on the traffic analysis, an SDN-based Attack Prevention design is constructed which proposes to capture and analyze incoming flows on-the-fly. A channel search prevention mechanism was then designed using both hard decision thresholds and Queuing Inference System to detect the Brute force attack. The system is designed in a way which releases attack flows based on the demands from the control plane in order to detect and determine the presence of attacks.

Keywords: load balancing, SDN, queuing interface system, channel search algorithm.

INTRODUCTION

Software-defined networking (SDN) [1] is a computerized architectural design serving to be elastic to changes, cost-effective, easily controllable, suitable for high bandwidth and dynamic in nature in comparison to currently existing applications. SDN design decouples network control and forwarding functions thus enabling the network control to become directly programmable and the allowing the underlying infrastructure to be obtained from applications as well as network services. SDN is meant to address the fact that the static architecture of traditional networks doesn't support the needs like storage and scalable computing in current modern computing environments like data centers. This is done by disassociating the system that makes decisions on where the traffic is sent from the underlying systems that transfers traffic to the selected destination.

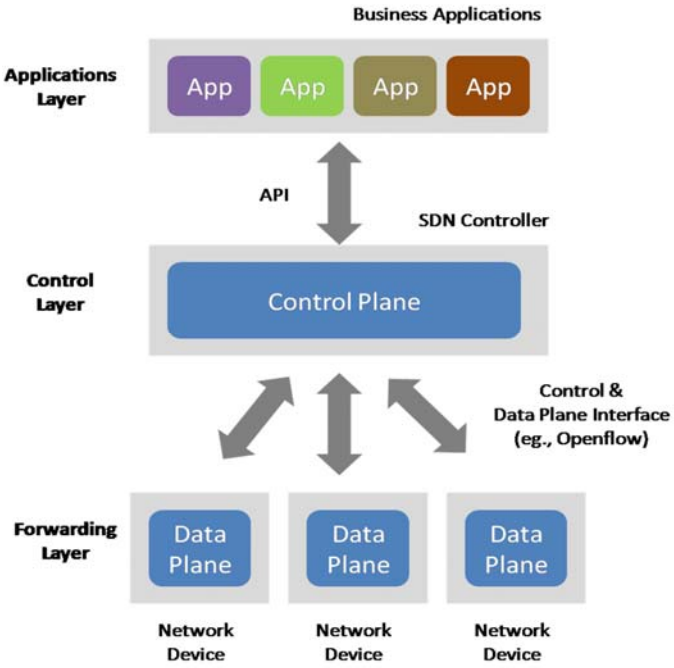


Figure-1. SDN architecture.

Load balancing [2] as the name suggests, refers to balancing the work that a computer has to perform by dividing the work a single computer performs between two/more computers in order to achieve the same amount of work done quicker. Load balancing can be implemented with hardware, software, or by using a combination of both. The primary reason for clustering of server is load balancing. Load balancing seeks to optimize resources and maximize throughput. It also minimizes response time, and avoids overloading of any single resource. Using

several components with load balancing instead of one component may improve reliability and also availability through redundancy.

In this paper we have introduced a new concept called the Queuing Interface System (QIS). The QIS is used to effectively manage congestion and balance the load effectively. The QIS makes use of a standard Queuing algorithm for collections and delivering packets. It helps in balancing the load as well as avoids congestion in the networks. We have also detected and prevented the attacks on the networks by using a channel search algorithm. In Channel Search rule, sinkhole attack is done by modifying sequence variety in RREQ, higher the sequence variety, then route is going to be more modern the packet contains. Sinkhole node selects the starting and ending pairs. With this we effectively manage the load, avoid congestion, detect and prevent any such attacks on the network.

RELATED WORKS

SDN is a growing concept and is expected to make the future as many people are researching and developing on this concept. Load balancing and security attacks have become common in SDN and are being researched widely. Using ONOS to effectively balance and control the congestion in a network and implement it real time have become a common sight. Some of the various algorithms stated by a few researchers are listed below:

Smriti bhandarkar *et al* [3] researched using round robin load balancer method which involved incoming packets or requests that were distributed across the available switches in the network. One can choose this load balancer if all the switches present in the cluster have the same capability and can handle equal amount of load. Under this consideration, the round robin load balancing method is known as the simplest and most effective method for load distribution. If the switches with equal capabilities are used for round robin load balancing it means that the less capable switch gets flooded with the number of packets or requests even though it is not able to process that many number of requests. Statistical configuration and uneven load balancing are the important limitations to round robin load balancers. Dynamic load balancer is a module integrated with the floodlight controller to set various custom costs on every link dynamically cached by the topology module. By using this module in the floodlight controller, new custom costs are setup on the link on both the directions. From this we say the dynamic load balancer is an algorithm used for load shifting from the best calculated path to reduce the collision and information loss when the load on the link will be greater than the bandwidth of the link. From the experiment we arrived at the results that prove that it can handle more packets and can have greater efficiency than round robin load balancer in both the modes.

Widhi Yahya's *et al* [4] works include using the extended Dijkstra's shortest path algorithm in SDN to balance the load effectively. The algorithm used not solely considers the edge weights, but additionally considers the node weights to find the closest server that the client requested. The algorithm additionally considers the link load so as to avoid congestion. Pyretic was established for implementing the algorithm and compare it with similar ones beneath the Abilene network with the Mininet emulation tool. As shown by the comparisons, the algorithm outperforms the others. The topology they used was Abilene network topology. According to this, they established an OpenFlow controller and eleven Open Flow switches as nodes, every of that was joined to the controller logically and POX was used as OpenFlow controller. The load balancing algorithm was implemented using pyretic. Thus from this work they concluded that the proposed algorithm was better than the other in terms of throughput and latency.

Seyed Mohammad Mousav's *et al* [5] work was of mitigation of DDoS attacks. To mitigate this threat, the paper used the central management of SDN for attack detection and conjointly introduces an answer that was effective and easy in terms of the resources that it used. To be precise the paper demonstrated how Distributed denial of service attacks will expend controller resources and provided an answer that detected attacks constructed on the entropy variation of the destination IP address. The paper presented an effective solution for detecting DDoS [6] attack in SDN. By making use of the main role the controller plays in SDN, they were able to use the destination IP address for detecting attacks within the first 250 packets of the traffic that has malicious packets. Also, the threshold that was chosen is set to the lowest possible rate of attack traffic. Nonetheless, the detection rate for this threshold was 96%. It provides the detection for both the controller and the attacked host. One of important advantage of this solution is its flexibility. Any parameter in the solution can be altered to fit the requirements of the controller. Destination IP address, window size and threshold can all be set to correspond to the desired values even in real-time. This is possible through the controller's interface. Thus this method was effective in detecting and preventing DDoS attacks.

PROPOSED WORK

The Open Network Operating System also known as ONOS [7] designed for Software Defined Networking (SDN) is a platform for service providers possessing high performance and scalability. As a cluster based Operating System, it must stand up to the demands of various service providers. ONOS provides numerous purposes such as granting resources, adding API's, permissions, in addition user-facing software system like a command line interface and a graphical user interface for system applications. The controller manages the entire network rather than a single device. ONOS [8] will run as a distributed system across various servers, to use the hardware and memory resources of various servers that offer fault tolerance during server failure and upgrades of hardware and software system without disturbing the network traffic. In this work, ONOS is used as the controller to balance the loads and carry out the various events with the help of the ONOS application. The application is useful to build the network for the real

time project. In order to create a virtual network with switches to balance the loads we have used a Network Animator (NAM) [9]. It is a Tcl/TK based animation tool accustomed to trace network simulations and packet movement. It aids numerous aspects like topology design, packet level animation, and multiple knowledge scrutiny tools. To plot the graph we have used Xgraph. The Xgraph [10] is a simple x-y information plotter with interactive buttons for moving, magnifying, printing and choosing various options. It plots information from various files on an equivalent graph and handles unlimited data-set sizes and numerous information files. The topology used here is fat tree topology. Fat-tree topology [11] is used in this work since the branches nearer the top of the hierarchy are thicker than branches further down the hierarchy. Since the branches are data links the varied thickness of the data links allows for more efficient and technology-specific use.

In many of defense systems or mechanisms based on SDN architecture mostly concentrated in a few key design, including attack detection, attack defense, attack mitigation, load balancing, traffic filtering and source tracking of attack. The common approaching method of Brute force detection and mitigation solutions is collecting traffic characteristics before applying different Channel Search algorithms in order to confirm whether or not an attack is taking place. Each time the system detects a Brute force attack, throughout the policies setting at firewalls or prevention devices, the attack traffics are identified and dropped. Also we have dynamically introduced a Queuing Inference System to detect Brute force attacks. In this paper we have introduced a new concept called the Queuing Interface System (QIS). The QIS makes use of a standard Queuing algorithm for collections and delivering packets. The QIS uses the concept of Red and Tail Drop algorithm to prevent congestion at the Queue. As packets arrive at the QIS, they are collected one after another in a First come First serve basis. When the queue becomes full, all the incoming packets after this are dropped and collected in order to add to the queue later. This is the implementation of Tail Drop algorithm. Whereas the packets collected in the queue are sent to their respective nodes.

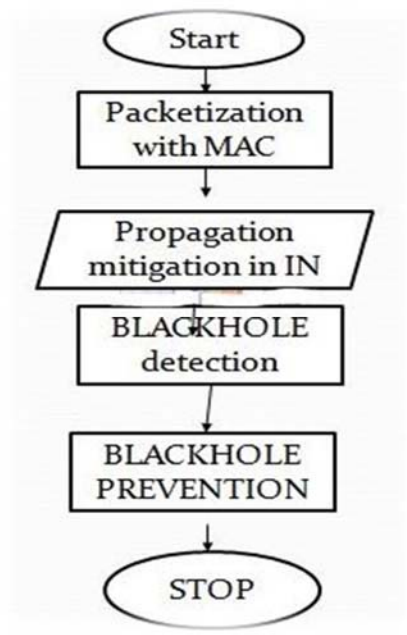
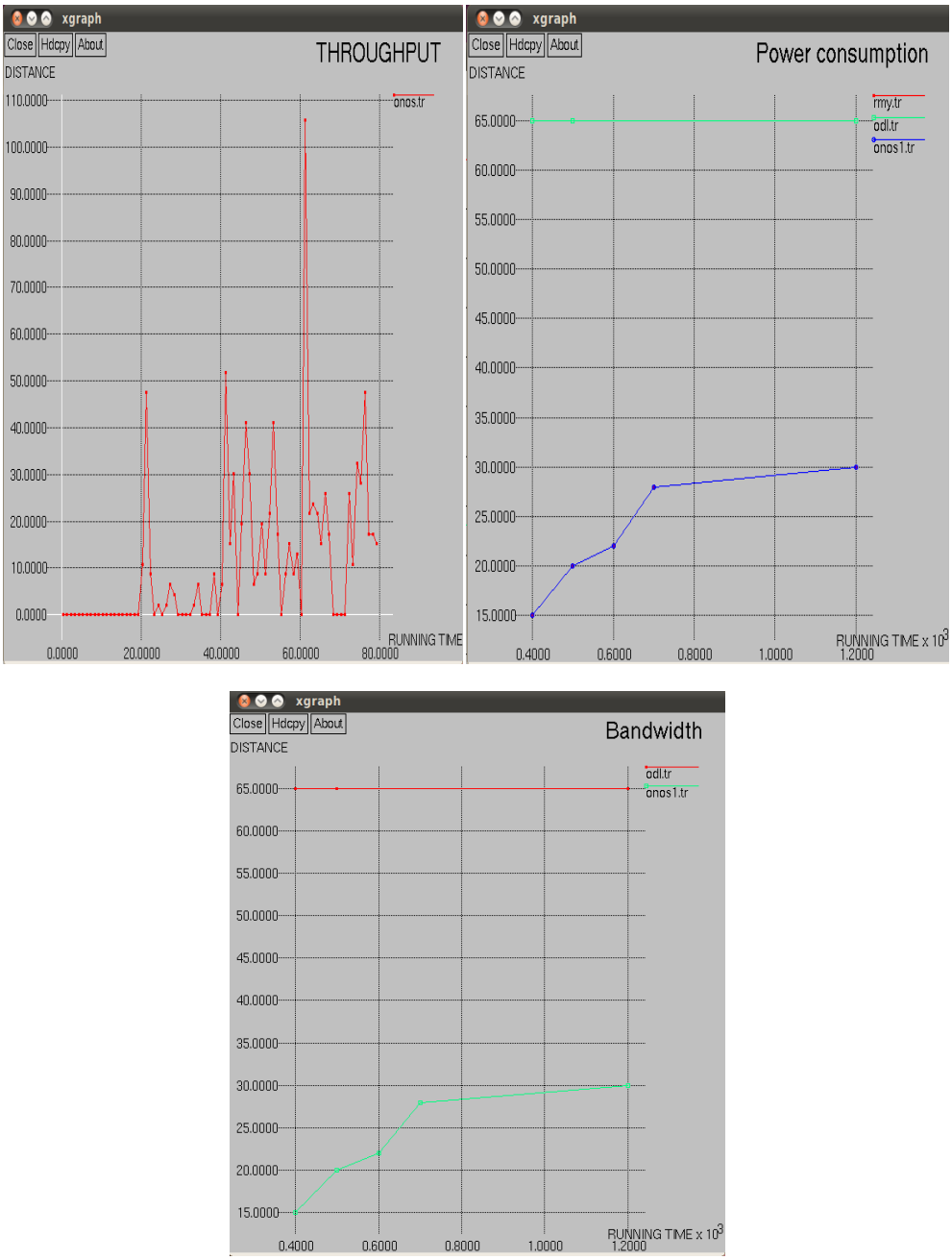


Figure-2. Flowchart for detection and prevention.

In Channel Search rule, sinkhole attack is done by modifying sequence variety in RREQ, higher the sequence variety, then route is going to be more modern the packet contains. Sinkhole node selects the starting and ending pairs. It observes the starting node's sequence variety rigorously from the RREQs of starting node and generates a phony RREQ with a larger sequence variety than the sequence variety of the starting node. It then broadcasts the phony RREQ. Nodes that take this phony RREQ acknowledge that this route may well be a desirable and recent route to the starting than the alternative route.

RESULTS

The results of the experiment conducted are shown below. We have used an Xgraph to plot the results. The results were compared with ONOS, OpenDayLight controller and Rosemary controller. The results were proved to be efficient in ONOS in terms of throughput, power consumption and bandwidth. The throughput was found be efficient in ONOS. The ONOS comparatively consumed less power than others. The bandwidth was less and efficient in ONOS. Thus with the results obtained we conclude that ONOS is a better bet than other controllers. Load balancing was executed well and good and also the detection and prevention of any attacks was done. Congestion was avoided in the network.



CONCLUSION AND FUTURE WORK

The proposed algorithm effectively balances the load, avoids congestion and effectively detects and prevents any attacks in the network. The throughput, power consumption and bandwidth were found be efficient. In future, we hope to add more elements to our approach such as the solution for the situation that when encountered a large number of attacks against sudden temporary ones. The mention of the defence mechanism, we also look forward to be able to put the capabilities that to trace the source of the attacker by making use of the characteristics and ability of SDN in our research projects. Furthermore, by making use of various other algorithms that are more effective than the one proposed and to effectively balance the load and also to detect and prevent attacks on a large scale. How to reinforced and enhance the effectiveness of the proposed method is one of the main research directions in the future, and we will be committed to expand our approach.

REFERENCES

[1] Cui Chen-Xiao and Xu Ya-bin. 2016. Research on Load Balance Method in SDN. International Journal of Grid and Distributed Computing. 9(1): 25-36 http://dx.doi.org/10.14257/ijgdc.2016.9.1.03.

[2] Yuanhao Zhou, Li Ruan, Limin Xiao, Rui Liu. 2014. A Method for Load Balancing based on Software-Defined Network. Advanced Science and Technology Letters. Vol. 45 (CCA 2014), pp. 43-48 http://dx.doi.org/10.14257/astl.2014.45.09.

[3] Smriti Bhandarkar, Kotla Amjath Khan. 2015. Load Balancing in Software-defined Network (SDN) Based on Traffic Volume. (Advances in Computer Science and Information Technology (ACSIT) Print ISSN: 2393-9907; Online ISSN: 2393-9915; 2(7): 72-76.

[4] Widhi Yahya Achmad Basuki, Jehn-Ruey Jiang. 2015. The Extended Dijkstra'sbased Load Balancing for OpenFlow Network. International Journal of Electrical and Computer Engineering (IJECE). 5(2): 289~296 ISSN: 2088-8708 ρ289.

[5] Seyed Mohammad Mousavi and Marc St-Hilaire. 2015. Early Detection of DDoS Attacks against SDN Controllers. International Conference on Computing, Networking and Communications, Communications and Information Security Symposium.

[6] Nayana Y, Mr. JustinGopinath, Girish. L. 2015. DDoS Mitigation using Software Defined Network. International Journal of Engineering Trends and Technology (IJETT). 24(5).

[7] Overview of ONOS. ONOS wiki. Retrieved from https://wiki.onosproject.org/display/ONOS/Wiki+Home.

[8] Introducing ONOS -a SDN network operating system for Service Providers. Retrieved from http://onosproject.org/wp-content/uploads/2014/11/Whitepaper-ONOS-final.pdf.

[9] NAM: Network Animator. Retrieved from http://www.isi.edu/nsnam/nam/.

[10] Monika Roopak, Dr. Bvr Reddy. 2011. Performance Analysis of Aodv Protocol under Black Hole Attack. International Journal of Scientific & Engineering Research. 2(8), ISSN 2229-5518.

[11] Li Yu and Deng Pan. 2013. OpenFlow based load balancing for Fat-Tree networks with multipath support. Proc. 12th IEEE International Conference on Communications (ICC'13), Budapest, Hungary.