Jamal et al LGURJCSIT 2019

ISSN: 2521-0122 (Online)
ISSN: 2519-7991 (Print)

LGU Research Jounral for
Computer Sciences & IT

Vol. 3 Issue 4, October - December 2019

Recovery Method for Disasters of Network Servers by Using POX controller in Software defined Networks.

[1] Asis Jamal , [2]Sarah Javed , [3]Arslan Akram, and [4]Shahzaib Jamal

[1]Department of Computer Science, Lahore Garrison University, Lahore, Pakistan
Email: asis@lgu.edu.pk, sarahjaved@lgu.edu.pk, aaadoula11@gmail.com, shahzaibjamal@live.com

Abstract:

The devices we used to automation made up of electric bunch called IoT devices so the more complex task is to manage them effectively. If the devices cannot connect or share anything correctly then these devices will be considered as useless. So the diversity of these devices will be increase the chance of survival. When we talk about the disaster the main difference between networks software and hardware needed to be overcome . for this we have to control the data traffic smartly so the software defined networks make this thing possible to give more programmability because in SDN the data plane is separated from control plane. When IoT devices got disconnected because of internet lack at that time these devices have to respond quickly. So for this purpose, software defined networks used to search another path for information transfer just to get that connection back we can say SDN provides reroute based on the routing information and routing flows they have already and they also have better understanding of pathways for communication. So in this paper our main focus is on this problem that will occur because of disaster and we intend to recover server and also multiple servers from this disaster of link failure , traffic engineering , power outage and rerouting of packets. For this we proposed a systematic approach for recovering these servers from disaster by using software defined networks. The separation of control plane from data plane provides programmability and also make the system flexible for getting back the connection soon. So for this recovery from disaster we are going to use OpenFlow protocol used by SDN and we using Mininet to implementation. The controller will be POX and also we are using Lipsflow mapping for disaster management and recovery.

Keywords: LISP, Locator/ID separation Protocol, SDN, software defined network, Open Flow, load balancing and latency)

1. INTRODUCTION

The recovery of disaster is more complex because the network system are becoming more complex to maintain and to control the network system manually. The maintenance of network system is difficult because of increasing growth of network infrastructure so the disaster management is more complex now a days. We can solve the problem by going on that node of network like vender specification solutions and also by debugging that error. For this network complexities we are going to use the technology named as software defined networking. So SDN is a new and innovative approach to provides the control and flexibility to the network and also used for the managing , building and designing the network infrastructure. In this paper we are going to recover our servers that effected by disaster and the disasters like : traffic engineering, rerouting of packets, device failures and also the link failure. So specifically in this paper we proposed an approach for disaster recovery using software defined networks. SDN provides more programmability by separating the control plane from data plane

and also it increases the flexibility and through this SDN can provides disaster recovery very fast[1]. This technology called SDN requires methods for the communication between data plane and control plane and that mechanism or method called OpenFlow protocol which is used by software defined networking for packet matching phenomena. We used POX controller which is written in python and with this controller we used LISP flow mapping for packet mapping and the LISP that will provides the flexible map and framework for the network application. For this purpose the emulator we used is mininet for creating the virtual network[2].

2. RELATED WORK

In network design and management Software defined network gives a different approach[3]. There is static nature of conventional networking even there is small changes in the condition of network that would at high cost of reconfiguring the switches that will be at large number and also routers and other resources of network. Shiaeles et al. (2018) described FHSD solution the FHSD stands for "Fuzzy hybrid spoofing detection". It is a multi-layered spoofing identification system. It uses MAC address, counts hops and web client[4]. It uses the empirical rules for detection of malfunctioned traffic and its mitigation. This strategy accompanies its own disadvantages as its solutions features values are stored in files, this comparisons is cumbersome with it comes with the database. Another method is HCF, the HCF stands for Hop Count Filtering. It utilizes the TTL estimation of the source header to recognize the disaster. Dou et al. (2016) filtering technique uses statistical correlation between different attributes is described. This is used both attack situation and the situation when there is no attack. When there is no attack, normal pattern is analyzed using attribute pairs from the transport layers of the network packets. The recurrence of event of these sets would be extricated and used to find the confidence value of the stream. The attributes that exist between these two layers were utilized to decide the authenticity of a packet[5]. During the attack, the same confidence value is used to find whether an incoming packet is valid or not. The procedure utilizes the Confidence score to learn the authenticity of an incoming packet with comparing it to a threshold. If confidence limit is satisfied then access is granted to this packet. Aroua and Zouari (2015) have introduced architectural approaches to introduce a coordinated detection with response strategy. The authors consider the existing network architectures that are used by most ISP providers where traffic behavior is analyzed, collected and stored in a central server[6]. However, the authors fears system failure when it comes to the central server or gets compromised by an attacker. In their work, they have improved the single point of failure by equally distributing the shared information using the Byzantine hypothesis of byzantine general problem[7]. When an attack is detected, the information is shared and defense system applied.

A. SDN architecture

The unique feature of SDN is the control plane is separated from data plane . Control plane is formatted by a set of controllers which acts as intelligent brain of SDN, while the data plane formed by multiple packet forwarding switches. This separation of control and data plane enables the network to be directly programmable and achieve benefits like, simple network management, improve the utilization of network, and network innovations and all that. For SDN the communication interface is open flow which is considered as single controller to gain the simplicity. When the network scale continually expands a protocol may suffer from scalability and performance issues. As needs be, various multi-controller approaches are then proposed, and luckily they accomplish a typical fundamental design with joint endeavors. SDN design comprises of three layers: data plane, control plane, and application plane. The data plane is made out of bundle sending switches that are overseen by controllers through southbound application programming interfaces (APIs)[8]. The controllers are associated with the application plane by means of northbound APIs to encourage organize control and system administrations. The main idea of Software Defined Networking (SDN) is the partition of control and information planes. With SDN, the generally circulated control plane of system components, for example, switches and routers, is coherently unified in a SDN controller SDN controller has a worldwide perspective of the system and can settle on better steering choices in view of the present condition of the system, for example interface use one of the hubs. Along

these lines, SDN empowers more proficient system control and administration. In spite of the fact that the control plane is legitimately incorporated, in excess of one controller may be required for adaptability and dependability . High accessibility and low control plane inactivity are important to ensure the information plane execution, which is particularly essential for mission basic applications. The SDN engineering can be separated into six sections, and each part is clarified in detail as takes after: (1) Management plane: It incorporates organizing applications like directing, observing, stack adjusting, and firewalls[9]. Administration plane is in charge of characterizing standards and arrangements. A few creators utilize the term application plane rather than administration plane. (2). Northbound interface: A northbound interface offers help for correspondence between administration plane and control plane. It gives low‐level directions to the southbound interface. It is otherwise called administration to control plane interface. Up till now, no standard conventions have been characterized for the northbound interface. (3). Control plane: It is in charge of programming the sending gadgets. It in this way goes about as the mind of the system. Concentrated controllers dwells on this plane[10]. The controller has the total worldwide perspective of the systems . It is otherwise called controller plane. The arrangement of controllers oversees control plane or controller plane. (4) East‐west interface convention: East‐West convention is utilized to deal with the correspondence among various SDN controllers[11].

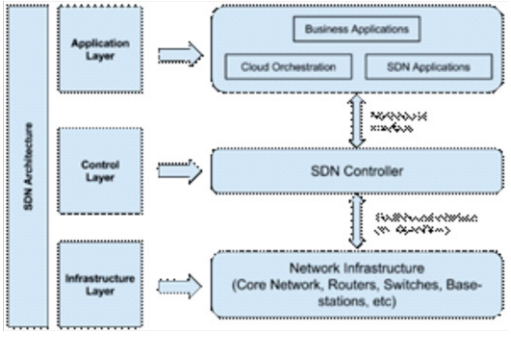


Fig 1. Architecture of software defined network.

B. OpenFlow

OpenFlow Is the protocol of software defined networks that provides the communication between controller and network device in the architecture of SDN. It was proposed to empower analysts to test new thoughts in a generation domain. OpenFlow gives a particular to relocate the control rationale from a switch into the controller. It likewise delivers a convention for the correspondence between the controller and the switches[12].

C. Pox controller:

Pox is the open source platform for development which is python based software defined networking application like OpenFlow and the other controllers. POX is the tool which enables the development and prototyping rapidly and this is commonly used platform than NOX which is java based platform for results getting[13].

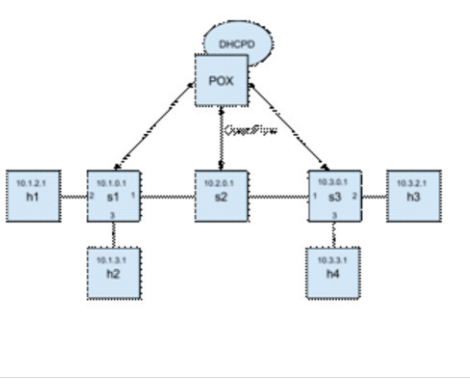


Figure 2.The architecture of POX controller of software defined networks.

D. LISP overview

Another protocol which we are going to use is LISP stands for location id separation protocol. It is the protocol which provides the separation between host locator and host identity which will done by creating two namespace[14]s. The two namespaces like end point identifier and the host locator so the LISP provides the mapping between virtual IP address and physical IP address . these two namespaces named as EID and RLOC [15].

E. Service of LISP flow mapping:

LISP flow mapping provides the system for mapping. This will have two entities named as map resolver and map server. This will provides the mapping of virtual IP address and physical address[16] . Mapping data can also include a variety of routing policies includes disaster recovery, traffic engineering and load balancing[17].

3. PROPOSED METHDOLOGY

Software defined network(SDN) use the controller named as POX and LISP flow mapping for the recovery of disaster in servers. For effective recovery from disaster using this technology SDN and also the lisp nodes that are mobile nodes and can installed in client and server nodes for the connectivity between server and client and for the stream less connectivity. Software defined networking (SDN) separates the data and control planes, removes the control plane from network hardware and implemented from software instead, which enables programmatic interface and, it increases flexibility of managing network. OpenFlow Is the protocol of software defined networks that provides the communication between controller and network device in the architecture of SDN. It was proposed to empower analysts to test new thoughts in a generation domain. OpenFlow gives a particular to relocate the control rationale from a switch into the controller. It likewise delivers a convention for the correspondence between the controller and the switches. This protocol is the combination of multiple components called plugins , pluggable controllers and also the applications.

Locator/identifier separation protocol (lisp) is another protocol which we are going to use is LISP stands for location id separation protocol. It is the protocol which provides the separation between host locator and host identity which will done by creating two namespaces. The two namespaces like end point identifier and the host locator so the LISP provides the mapping between virtual IP address and physical IP address . these two namespaces named as EID and RLOC.

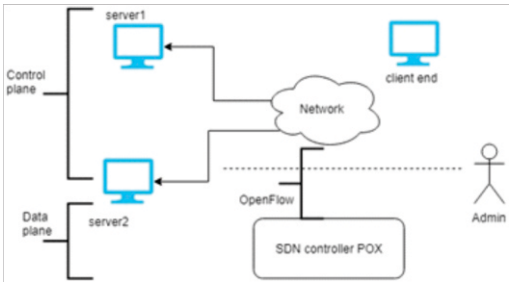


Fig 3. Architecture of proposed system for effective recovery of disaster by using SDN.

The LISP protocol have map server which is an open source environment for development , building and also enable the internet applications in it. The map server authenticate the EID-ROLC mappings by adding them to database. Here is also the command of map resolver which accepts the packets from ingress tunnel router(IRT) then solves the mapping of EID to ROLC by adding them to database. So we are going to resolve this problem or you can say to do effective recovery from disaster on server by using SDN. Because SDN have separated the control plane from data plane which will gives the more flexibility and programmability .control plane is the layer in networks that is responsible for the flow control which have the functions of management and configuration. Data plane is also called as forwarding plane which carries the data packets and you can say it carries the requests. The data plane is responsible for data transfer between clients, handling multiple conversations using various protocols, and manages communication with remote hosts. Data plane packets travels via routers, rather than to or from the. LISP mob is an application which can change their network attachment point without losing connection between host. Lisp mobile node: Lisp mobile node typically sends and receives LISP encapsulated packets. It uses the two name spaces endpoint identifier (EID) to name hosts in networks and routing locators (RLOCs) to locate a node.

4. IMPLEMENTATION

In this paper we have divided the implementation into four parts named as: SDN controller configuration(POX controller), Map registration, server authorization and the priority based checking of server.The flow chart for the recovery of disaster is discussed below in figure

4. and the flow of the implementation is:

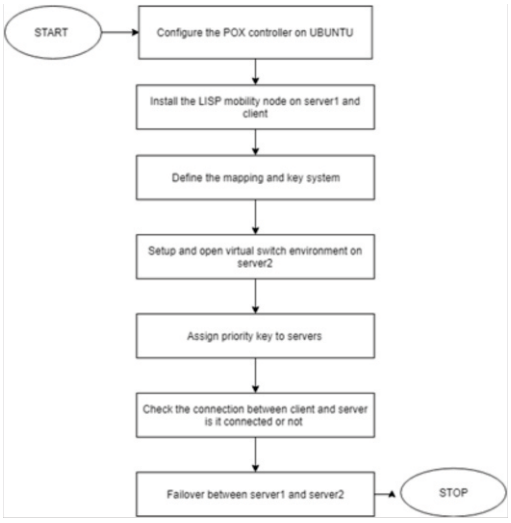


Fig4: Flow chart for proposed methodology of disaster recovery of server system

First we have to run the controller or you can say configure the SDN controller and in this paper we consider the POX controller for supporting the lisp flow mapping. Then we install the mobility nodes on the client and server end for the configuration of network. After that we define the mapping system and their key. Key means that the information and registration are stored in flow table and these entries will be identified by a key value generated. And the mapping we discussed is used for recovery as because it will map the physical address and the virtual address that will done by using LISP service. In next step of methodology we setup an environment of virtual switches just to note the fail over between the server 1 and the second server. By using northbound API and southbound API we create the bridge between switches and controller to assign the priority to server and to check the connectivity between server and client to see the failover between server1 and server2 by using mininet.

After that the data access object used to separate the database from the user, it will create the bridge between map resolver and the map server this will be done by API's. the main objective of the DAO is to access the database without knowing the implementation logic. So then the map server is used to registering and adding the server and making key and mapping system. A map resolver is used to processing a query and to receive a query to process that will received by client and server.

Map registering:

For the map registering the first step is to fetch the end point identifier EID and then store it in locator list or array. After that request for the message of map register. The notify message will received by server to map register after that add up the EID record to the database and then checking for the authentication of data which present. For this the condition is if the key_id matches with server key_id then send the notify message registering to server otherwise send the acknowledgement to server to notify that this no authentication of data have done.

Fig5. Flowchart for the map registering in disaster recovery.

Server authentication:

After the map registering there is need to authenticate the server so we have done server authentication for this the first step is to map server should be authenticated then get the key of authentication from the map resolver and then map server will sends iterate mask to map resolver and if the map resolve sends notify message then add key to authentication key if its not getting any notify message then remove authentication key from map server.

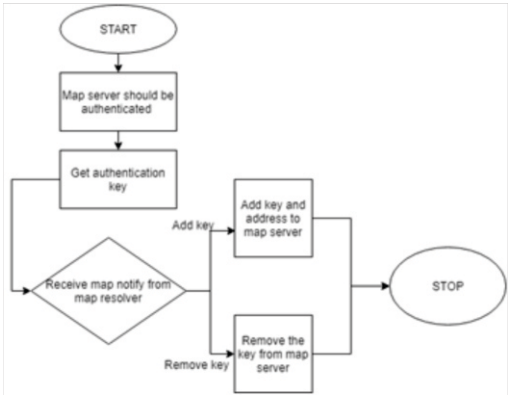


Fig 6. Figure or flow of steps for server authentication.

Servers priority checking:

The pseudocode for the priority checking this discussed bellow first, client send request to server by SDN controller named as POX which follows the OpenFlow protocol. Then if the client key_id matches with the server key_id it will send message that controller connects with server. If its not connects then will show message that the controller disconnects the client by sending message that key is not matched with map server key. After that or in next step disaster occurs at server1 and it crashes. Then controller will check the priority by map resolver. The condition is set if the server1 priority is less than priority of server2 then it will migrate towards the server2 while if its not then it will check for the next server which one having higher priority. Flow chart for priority checking:

In this flow chart figure 8 we intended to explain checking priorities server by SDN controller. The client sends request to server via controller, the controller checking key_id if key_id matches the controller connects to server for service otherwise controller send notify message called key not matches with map server key_id . the client get service on same time server gets disaster the controller searching for another server with higher priority value in this server2 having higher priority so it's connects to server2 if priority lower so it searching for another server.

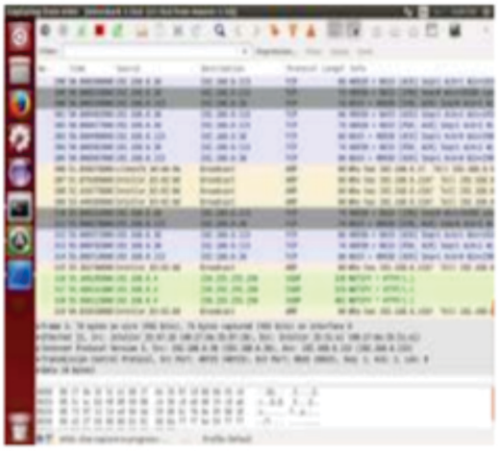
5. EXPERIMENTAL RESULTS

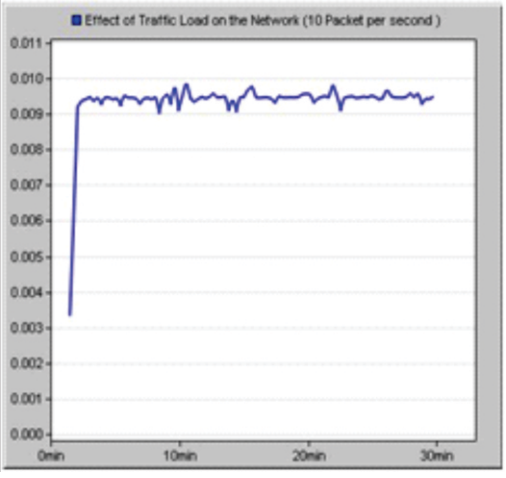
In this research there are two parameters Load balancing and latency. So, the Latency is

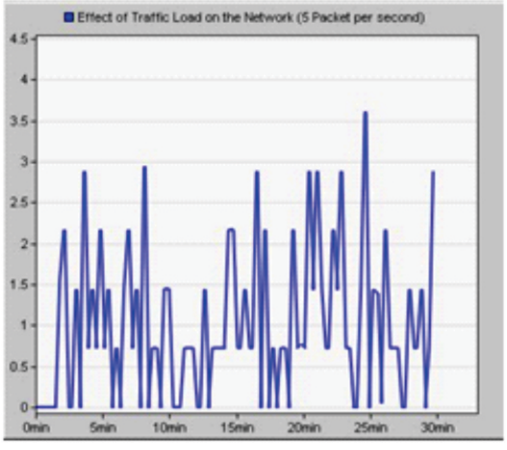
Fig7.Architecture for priority checking

the delay from input into a system to desired outcome; the term is understood slightly differently in various contexts and latency issues also vary from one system to another. Latency greatly affects how usable and enjoyable electronic and mechanical devices as well as communications are People connecting from distances to these live events can be seen to have to wait for responses. This latency is the wait time introduced by the signal travelling the geographical distance as well as over the various pieces of communications equipment. Even fiber optics are limited by more than just the speed of light, as the refractive index of the cable and all repeaters or amplifiers along their length introduce delays. And the second one is load balancing refers to efficiently distributing incoming network traffic across a group of

backend servers, also known as a server farm or server pool. To cost-effectively scale to meet these high volumes, modern computing best practice generally requires adding more servers.







Now the second parameter is latency or delay at same requirements when each node forward five packets per second and ten packets per second so the result is like and the simulation environment is:

6. CONDLUSION

In this paper we have implements the disaster recovery of server by using the technology software defined networks and the LISP the flow mapping service over UBUNTU operating system successfully. We have implemented the map registering by EID , the mapping service of the requests and by priority checking which have done by SDN controller named as POX controller which is open source the python based development environment. In future we can extend this work to deploy lisp flow mapping In android and also in windows. So we can extend the platform. And we can also see the disaster in case of load balancing by end to end communication.

References

[1] K. Nguyen, Q. T. Minh, and S. Yamada, "A software-defined networking approach for disaster-resilient WANs," in 2013 22nd International Conference on Computer Communication and Networks (ICCCN), 2013, pp. 1-5.

[2] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in 2014 IEEE 3rd International Conference on Cloud Networking (CloudNet), 2014, pp. 7-13.

[3] W. H. Muragaa, K. Seman, and M. F. Marhusin, "A pox controller module to collect web traffic statistics in SDN environment," World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering, vol. 10, pp. 2002-2007, 2016.

[4] F. Bannour, S. Souihi, and A. Mellouk, "Distributed SDN control: Survey, taxonomy, and challenges," IEEE Communications Surveys & Tutorials, vol. 20, pp. 333-354, 2018.

[5] M. Ziaullah, P. Shetty, and S. Kamal, "Image feature based authentication and digital signature for wireless data transmission," in

2016 International Conference on Computer Communication and Informatics (ICCCI), 2016, pp. 1-4.

[6] R. T. Baum, "IP based security applications using location, port and/or device identifier information," ed: Google Patents, 2011.

[7] M. Kamruzzaman, N. I. Sarkar, J. Gutierrez, and S. K. Ray, "A study of IoT-based post-disaster management," in 2017 International Conference on Information Networking (ICOIN), 2017, pp. 406-410.

[8] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," IEEE Communications Surveys & Tutorials, vol. 18, pp. 236-262, 2016.

[9] C. P. Dingman, P. Mahadevan, and J. J. Ordille, "Method and apparatus for supporting communications between a computing device within a network and an external computing device," ed: Google Patents, 2016.

[10] L. R. Dennison, "Software control plane for switches and routers," ed: Google Patents, 2013.

[11] M. Karakus and A. Durresi, "A survey: Control plane scalability issues and approaches in software-defined networking (SDN)," Computer Networks, vol. 112, pp. 279-293, 2017.

[12] J. Wang and M. Luo, "Packet prioritization in a software-defined network implementing OpenFlow," ed: Google Patents, 2018.

[13] V. Gramoli, G. Jourjon, and O. Mehani, "Disaster-tolerant storage with SDN," in International Conference on Networked Systems, 2015, pp. 293-307.

[14] J. R. Putman, M.-H. Nguyen, T. C. Hanson, and S. Srinivasan, "Communication application server for converged communication services," ed: Google Patents, 2015.

[15] P. Kakade, S. B. Raman, and R. Sharma, "Systems and methods for business impact analysis and disaster recovery," ed: Google Patents, 2019.

[16] S. Azodolmolky, Software defined networking with OpenFlow: Packt Publishing Ltd, 2013.

[17] G. Liu, M. Trotter, Y. Ren, and T. Wood, "Netalytics: Cloud-scale application performance monitoring with sdn and nfv," in Proceedings of the 17th International Middleware Conference, 2016, p. 8.