ABSTRACT

Software denned networking empowers network operators with more edibility to program their Networks. With SDN, network management moves from codifying functionality in terms of low level device conjurations to building software that facilitates network management and debugging. By separating the complexity of state distribution from network specification, SDN provides new Ways to solve long-standing problems in networking routing, for instance while simultaneously allowing the use of security and dependability techniques, such as access control or multi-path. However, the security and dependability of the SDN itself is still an open issue. In this position Paper we argue for the need to build secure and dependable SDNs by design. As a rest step in this Direction we describe several threat vectors that may enable the exploit of SDN vulnerabilities. We then sketch the design of a secure and dependable SDN control platform as a materialization of the concept here advocated. We hope that this paper will trigger discussions in the SDN community around these issues and serve as a catalyzer to join ports from the networking and security & Dependability communities in the ultimate goal of building resilient control planes.

INTRODUCTION

Software-defined networking (SDN) is an architecture that aims to make networks agile and flexible. The goal of SDN is to improve network control by enabling enterprises and service providers to respond quickly to changing business requirements.

In a software-defined network, a network engineer or administrator can shape traffic from a centralized control console without having to touch individual switches in the network. The centralized SDN controllers directs the switches to deliver network services wherever they're needed, regardless of the specific connections between a server and devices.

This process is a move away from traditional network architecture, in which individual network devices make traffic decisions based on their configured routing tables.

ARCHIETECTURE

A typical representation of SDN architecture comprises three layers: the application layer, the control layer and the infrastructure layer.

The control layer represents the centralized SDN controller software that acts as the brain of the software-defined network. This controller resides on a server and manages policies and the flow of traffic throughout the network.

The infrastructure layer is made up of the physical switches in the network. The application layer, not surprisingly, contains the typical network applications or functions organizations use, which can include intrusion detection systems, load balancing or firewalls. Where a traditional network would use a specialized appliance, such as a firewall or load balancer, a software-defined network replaces the appliance with an application that uses the controller to manage data plane behavior.

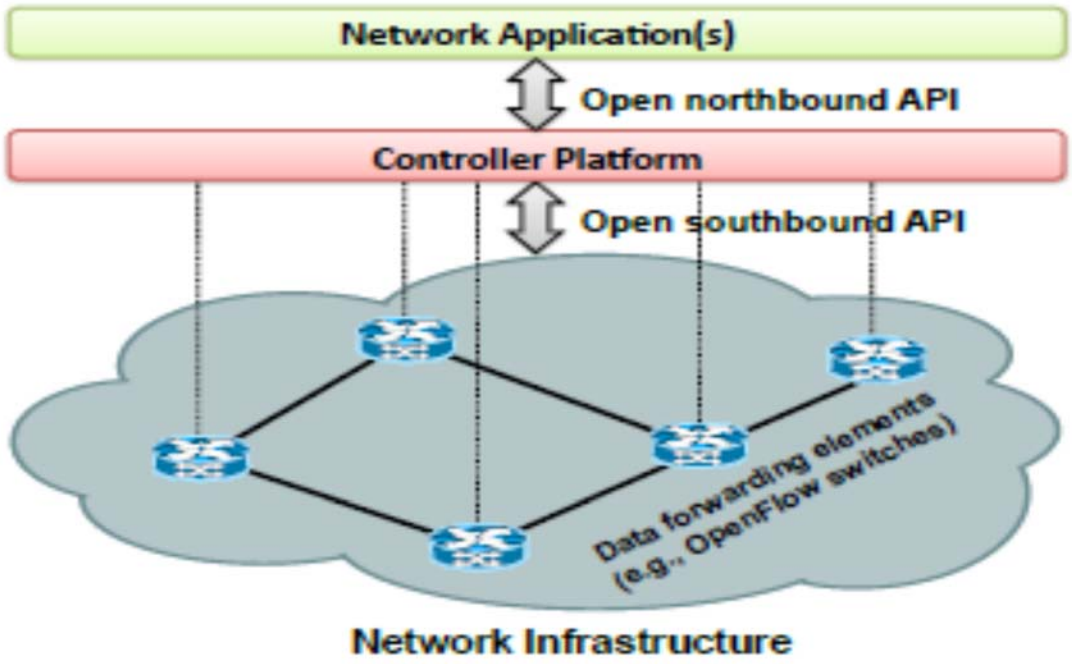


FIG 3.1: SDN architecture

SDN architecture separates the network into three distinguishable layers, connected through northbound and southbound APIs.

These three layers communicate using respective northbound and southbound application programming interfaces (APIs). For example, applications talk to the controller through its northbound interface, while the controller and switches communicate using southbound interfaces, such as OpenFlow -- although other protocols exist. There is currently no formal standard for the controller's northbound API to match OpenFlow as a general southbound interface. It is likely the OpenDaylight controller's northbound API may emerge as a de facto standard over time, given its broad vendor support.

WORKINGS

How SDN works?

SDN encompasses several types of technologies, including functional separation, network virtualization and automation through programmability. Originally, SDN technology focused solely on separation of the network control plane from the data plane. While the control plane makes decisions about how packets should flow through the network, the data plane actually moves packets from place to place.

In a classic SDN scenario, a packet arrives at a network switch, and rules built into the switch's proprietary firmware tell the switch where to forward the packet. These packet-handling rules are sent to the switch from the centralized controller.

The switch also known as a data plane device queries the controller for guidance as needed, and it provides the controller with information about traffic it handles. The switch sends every packet going to the same destination along the same path and treats all the packets the exact same way.

Software-defined networking uses an operation mode that is sometimes called adaptive or dynamic, in which a switch issues a route request to a controller for a packet that does not have a specific route. This process is separate from adaptive routing, which issues route requests through routers and algorithms based on the network topology, not through a controller.

The virtualization aspect of SDN comes into play through a virtual overlay, which is a logically separate network on top of the physical network. Users can implement end- to-end overlays to abstract the underlying network and segment network traffic. This micro segmentation is especially useful for service providers and operators with multitenant cloud environments and cloud services, as they can provision a separate virtual network with specific policies for each tenant.

BENEFITS OF SDN

With SDN, an administrator can change any network switch's rules when necessary prioritizing, deprioritizing or even blocking specific types of packets with a granular level of control and security. This is especially helpful in a cloud computing multi- tenant architecture, because it enables the administrator to manage traffic loads in a flexible and more efficient manner. Essentially, this enables the administrator to use less expensive commodity switches and have more control over network traffic flow than ever before.

Other benefits of SDN are network management and end-to-end visibility. A network administrator need only deal with one centralized controller to distribute policies to the connected switches, instead of configuring multiple individual devices. This capability is also a security advantage because the controller can monitor traffic and deploy security policies. If the controller deems traffic suspicious, for example, it can reroute or drop the packets.

SDN also virtualizes hardware and services that were previously carried out by dedicated hardware, resulting in the touted benefits of a reduced hardware footprint and lower operational costs.

Additionally, software-defined networking contributed to the emergence of software- defined wide area network (SD-WAN) technology. SD-WAN employs the virtual overlay aspect of SDN technology, abstracting an organization's connectivity links throughout its WAN and creating a virtual network that can use whichever connection the controller deems fit to send traffic.

CHALLENGES

Security is both a benefit and a concern with SDN technology. The centralized SDN controller Presents a single point of failure and, if targeted by an attacker, can prove detrimental to the network.

Ironically, another challenge with SDN is there's really no established definition of software-defined networking in the networking industry. Different vendors offer various approaches to SDN, ranging from hardware-centric models and virtualization platforms to hyper-converged networking designs and controller less methods.

Some networking initiatives are often mistaken for SDN, including white box networking, network disaggregation, network automation and programmable networking. While SDN can benefit and work with these technologies and processes, it remains a separate technology.

SDN technology emerged with a lot of hype around 2011, when it was introduced alongside the Open Flow protocol. Since then, adoption has been relatively slow, especially among enterprises that have smaller networks and fewer resources. Also, many enterprises cite the cost of SDN deployment to be a deterring factor.

Main adopters of SDN include service providers, network operators, telecoms and carriers, along with large companies, like Facebook and Google, all of which have the resources to tackle and contribute to an emerging technology.

USE CASE DIAGRAM

USE CASE 1:

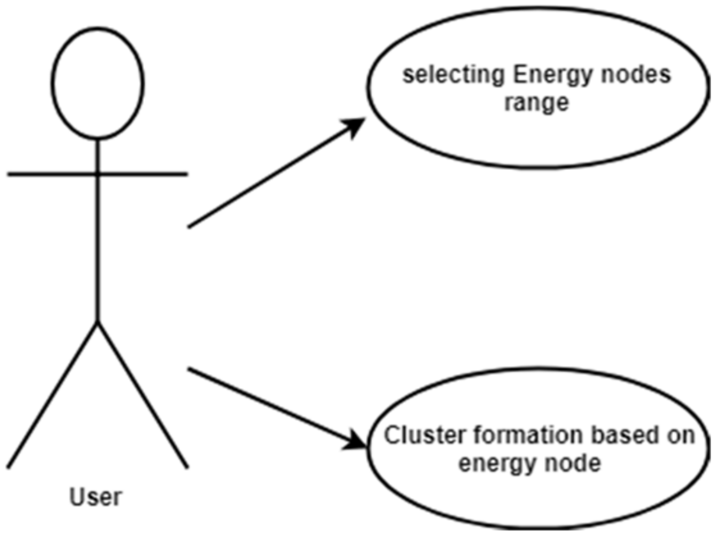

selecting Energy nodes range
Cluster formation based on energy node
User

Fig: 8.1 use case 1

USE CASE 2:

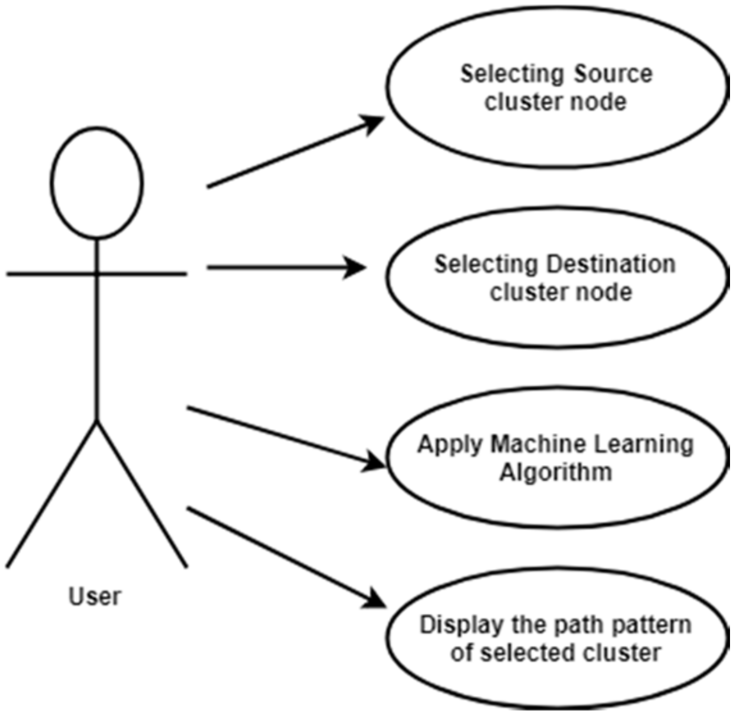

Selecting Source cluster node
Selecting Destination cluster node
Apply Machine Learning Algorithm
Display the path pattern of selected cluster
User

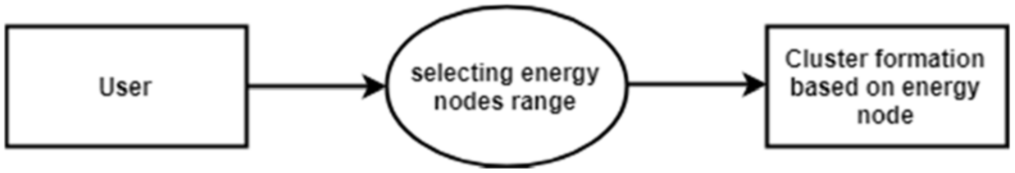
Fig: 8.2 use case 2

Data Flow Diagram

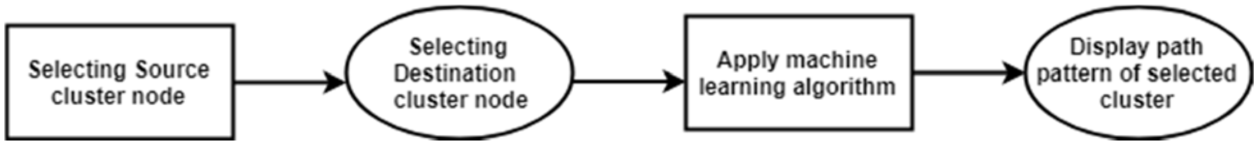
A data flow diagram is a graphical representation of the "flow" of data through an information system, modeling its *process* aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data, and the output data is generated by the system.

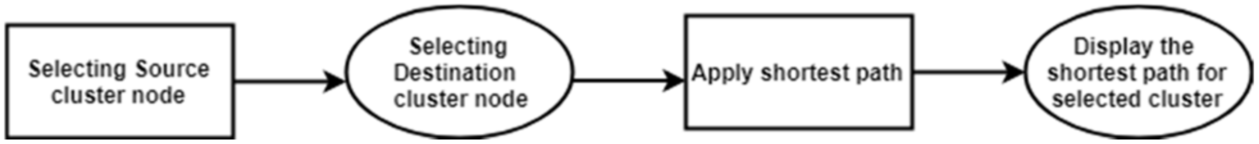
DFD 0


User
selecting energy nodes range
Cluster formation based on energy node

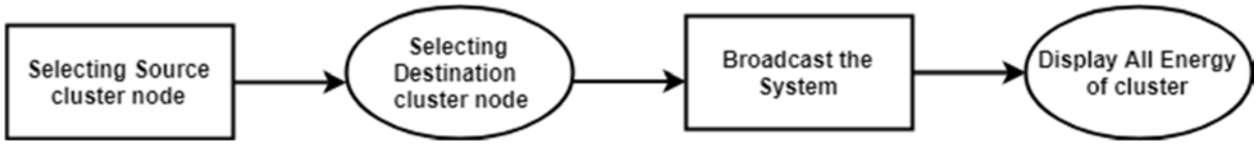
DFD 1


Selecting Source cluster node
Selecting Destination cluster node
Apply machine learning algorithm
Display path pattern of selected cluster

DFD 2

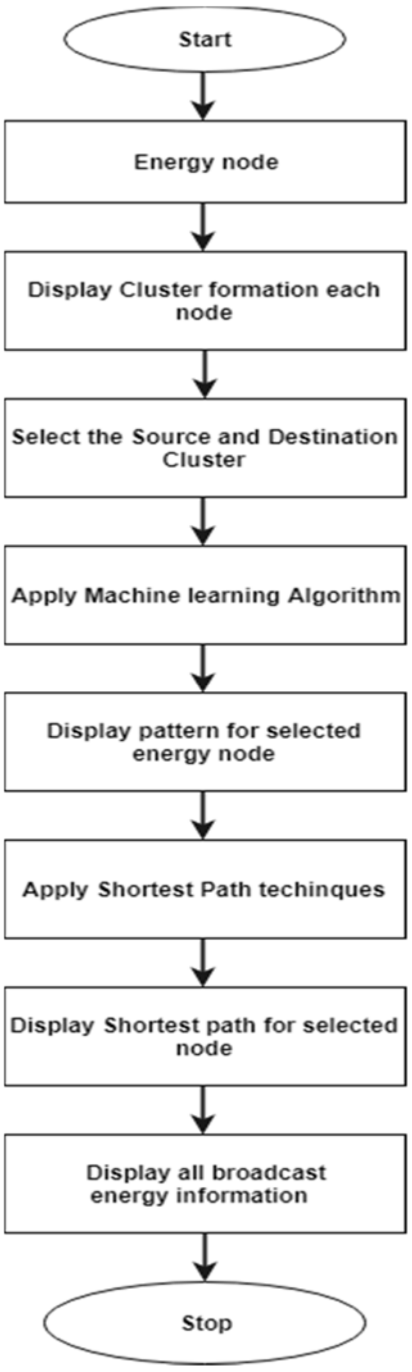

Selecting Source cluster node
Selecting Destination cluster node
Apply shortest path
Display the shortest path for selected cluster

DFD 3


Selecting Source cluster node
Selecting Destination cluster node
Broadcast the System
Display All Energy of cluster

FLOWCHARTS

A flow chart is a graphical or symbolic representation of a process. Each step in the process is represented by a different symbol and contains a short description of the process step. The flow chart symbols are linked together with arrows showing the process flow direction.


Start
Energy node
Display Cluster formation each node
Select the Source and Destination Cluster
Apply Machine learning Algorithm
Display pattern for selected energy node
Apply Shortest Path techinques
Display Shortest path for selected node
Display all broadcast energy information
Stop

SEQUENCE DIAGRAM:

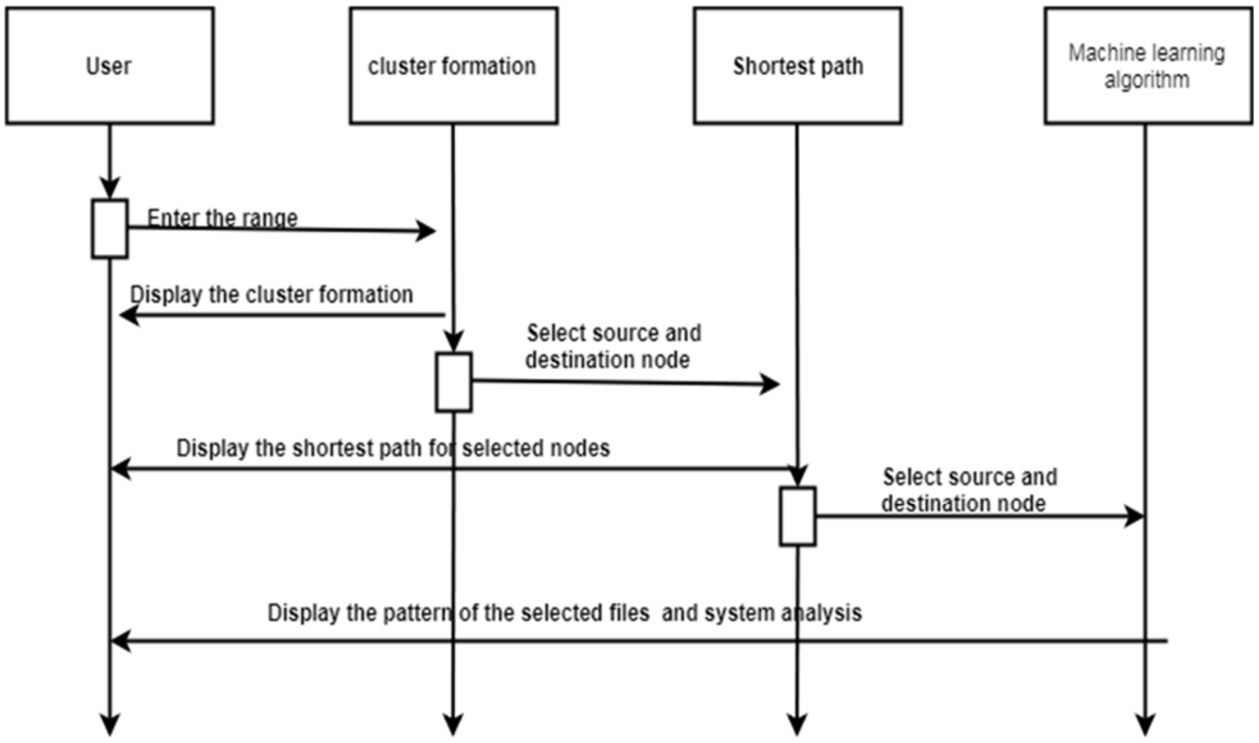

User
cluster formation
Shortest path
Machine learning algorithm
Enter the range
Display the cluster formation
Select source and destination node
Display the shortest path for selected nodes
Select source and destination node
Display the pattern of the selected files and system analysis

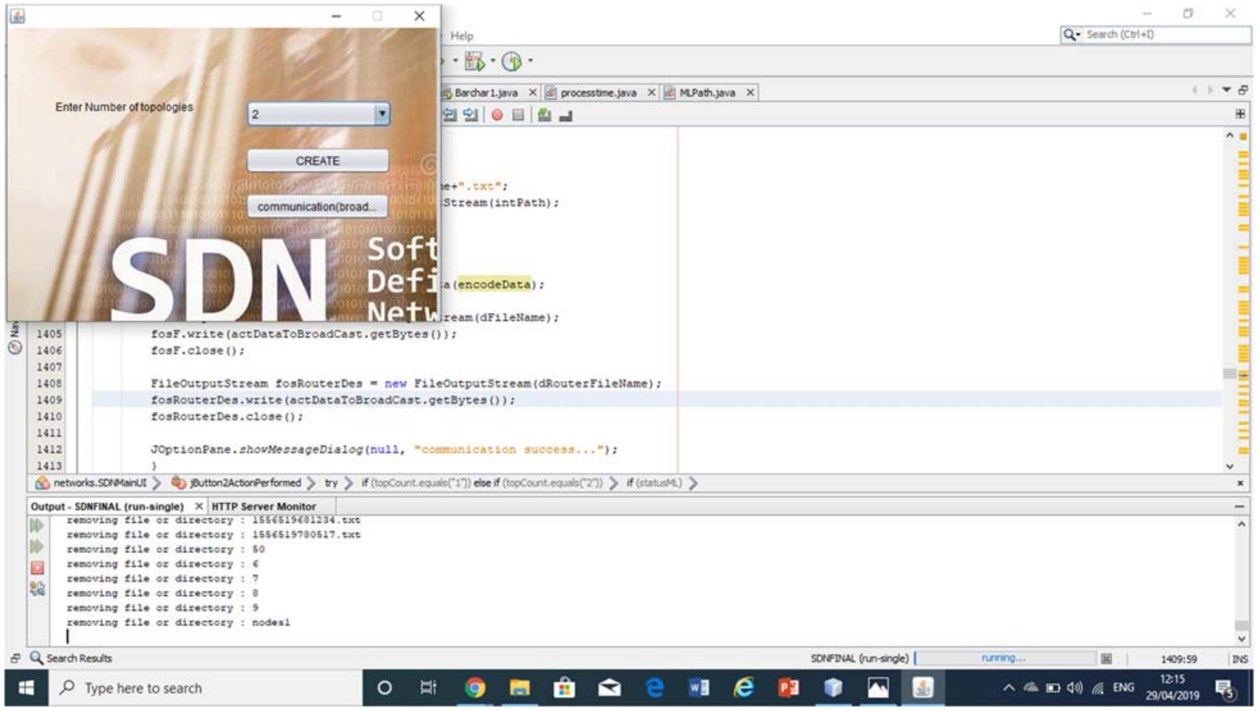
Fig: sequence

PSEUDO CODE

Random Uniform Matrix Generation

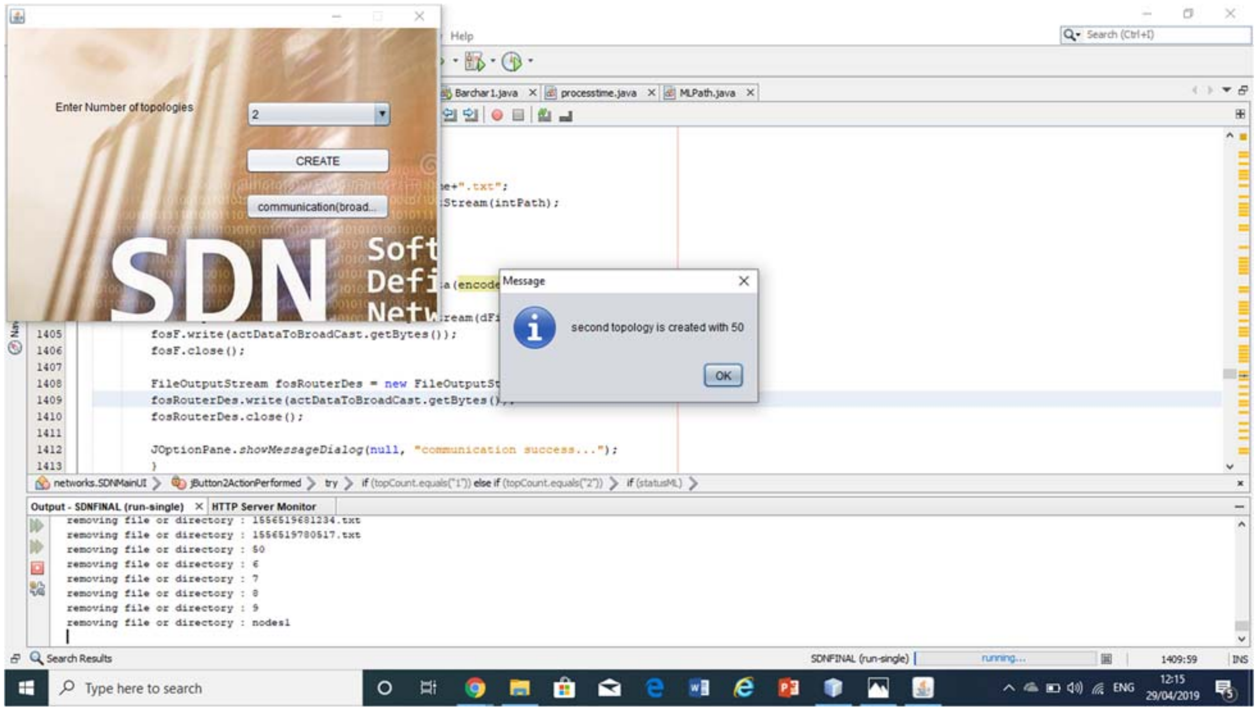
```
t1=MAKE(C1)
t2=MAKE(C2)
t3=MAKE(C3)
t4=MAKE(C4)
n1🡐RANDOM(1 - 150)
n2🡐RANDOM(1 - 150)
n3🡐RANDOM(1 - 150)
n4🡐RANDOM(1 - 150)
C1=ASSIGN ENERGIES(t1,n1)
C2=ASSIGN ENERGIES(t2,n2)
C3=ASSIGN ENERGIES(t3,n3)
C4=ASSIGN ENERGIES(t4,n4)
Ch1🡐MAXENERGY(C1)
Ch2🡐MAXENERGY(C2)
Ch3🡐MAXENERGY(C3)
Ch4🡐MAXENERGY(C4)
DISPLAYACK(C1,C2,C3,C4)
```

SCREENSHOTS



```
e+".txt";
Stream(intPath);

a(encodeData);

ream(dFileName);
        fosF.write(actDataToBroadCast.getBytes());
        fosF.close();

        FileOutputStream fosRouterDes = new FileOutputStream(dRouterFileName);
        fosRouterDes.write(actDataToBroadCast.getBytes());
        fosRouterDes.close();

        JOptionPane.showMessageDialog(null, "communication success...");
        }
```

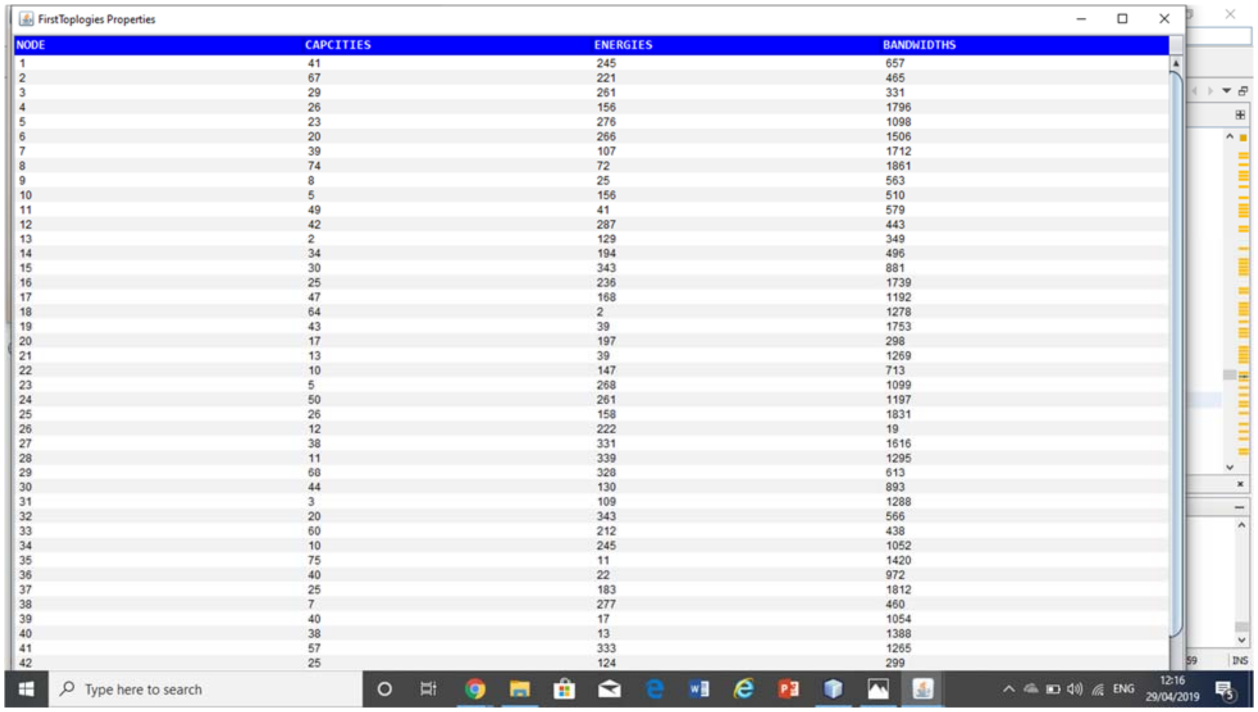
```
removing file or directory : 1556519681234.txt
removing file or directory : 1556519780517.txt
removing file or directory : 50
removing file or directory : 6
removing file or directory : 7
removing file or directory : 8
removing file or directory : 9
removing file or directory : nodes1
```



second topology is created with 50

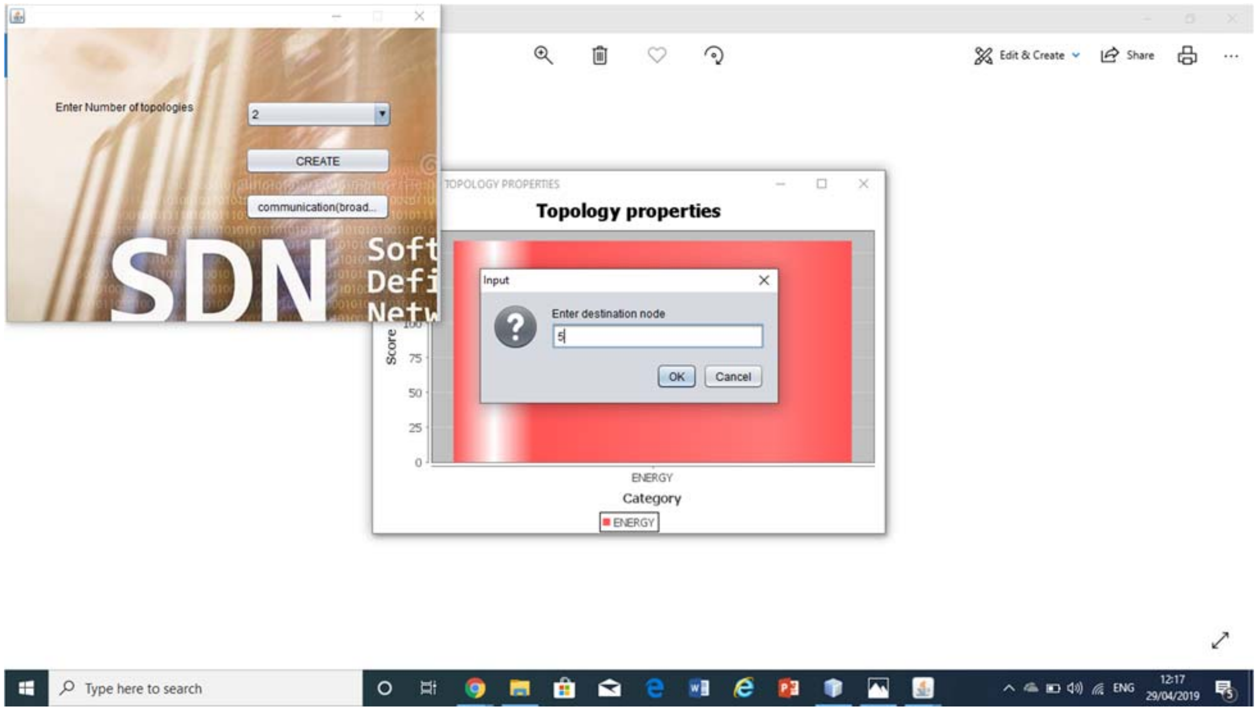
```
e+".txt";
Stream(intPath);

a(encode

ream(dFi
        fosF.write(actDataToBroadCast.getBytes());
        fosF.close();

        FileOutputStream fosRouterDes = new FileOutputSt
        fosRouterDes.write(actDataToBroadCast.getBytes(
        fosRouterDes.close();

        JOptionPane.showMessageDialog(null, "communication success...");
        }
```

```
removing file or directory : 1556519681234.txt
removing file or directory : 1556519780517.txt
removing file or directory : 50
removing file or directory : 6
removing file or directory : 7
removing file or directory : 8
removing file or directory : 9
removing file or directory : nodes1
```



FirstToplogies Properties

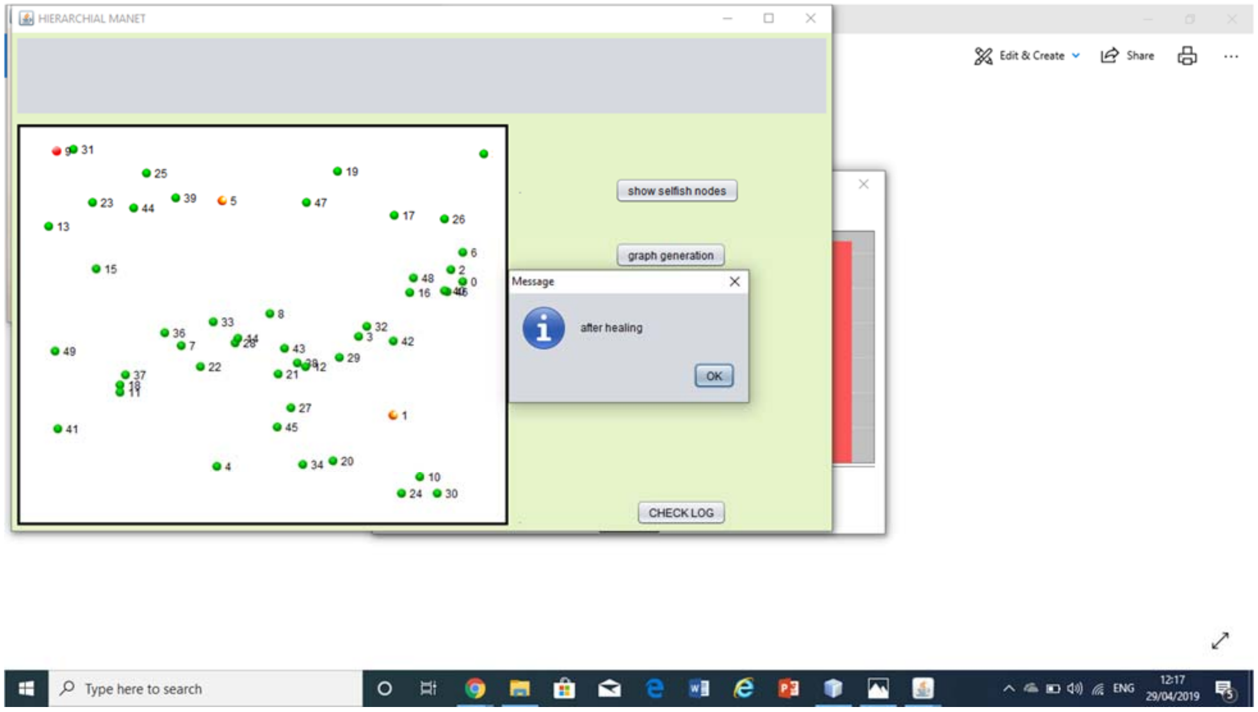
NODE	CAPCITIES	ENERGIES	BANDWIDTHS
1	41	245	657
2	67	221	465
3	29	261	331
4	26	156	1796
5	23	276	1098
6	20	266	1506
7	39	107	1712
8	74	72	1861
9	8	25	563
10	5	156	510
11	49	41	579
12	42	287	443
13	2	129	349
14	34	194	496
15	30	343	881
16	25	236	1739
17	47	168	1192
18	64	2	1278
19	43	39	1753
20	17	197	298
21	13	39	1269
22	10	147	713
23	5	268	1099
24	50	261	1197
25	26	158	1831
26	12	222	19
27	38	331	1616
28	11	339	1295
29	68	328	613
30	44	130	893
31	3	109	1288
32	20	343	566
33	60	212	438
34	10	245	1052
35	75	11	1420
36	40	22	972
37	25	183	1812
38	7	277	460
39	40	17	1054
40	38	13	1388
41	57	333	1265
42	25	124	299



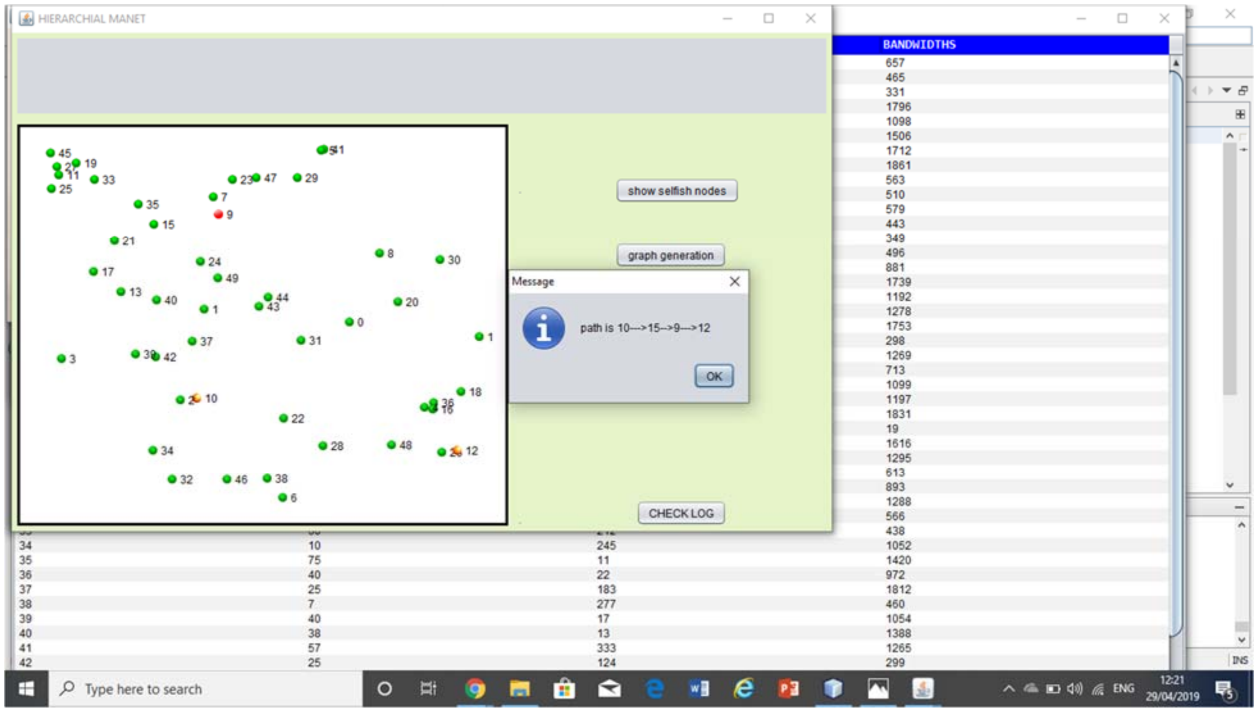
Enter Number of topologies: 2

TOPOLOGY PROPERTIES — Topology properties

Input — Enter destination node: 5

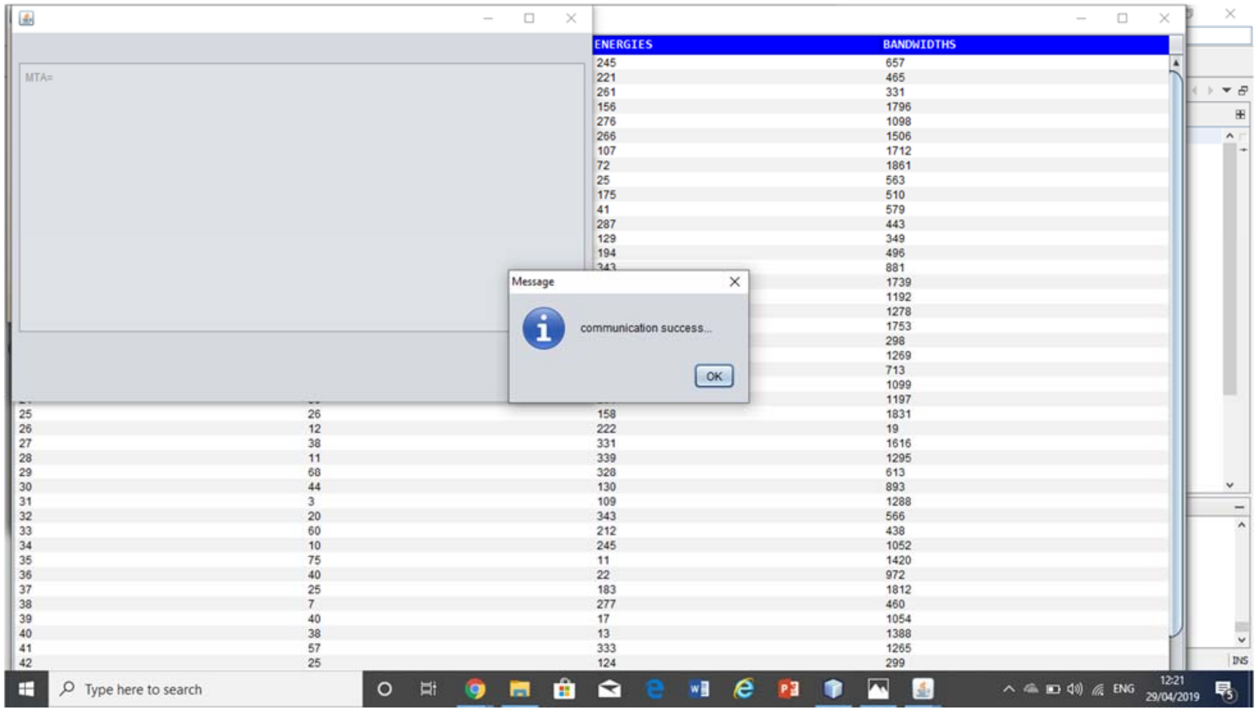


after healing



path is 10--->15-->9--->12

			BANDWIDTHS
			657
			465
			331
			1796
			1098
			1506
			1712
			1861
			563
			510
			579
			443
			349
			496
			881
			1739
			1192
			1278
			1753
			298
			1269
			713
			1099
			1197
			1831
			19
			1616
			1295
			613
			893
			1288
			566
			438
34	10	245	1052
35	75	11	1420
36	40	22	972
37	25	183	1812
38	7	277	460
39	40	17	1054
40	38	13	1388
41	57	333	1265
42	25	124	299



MTA=

Message

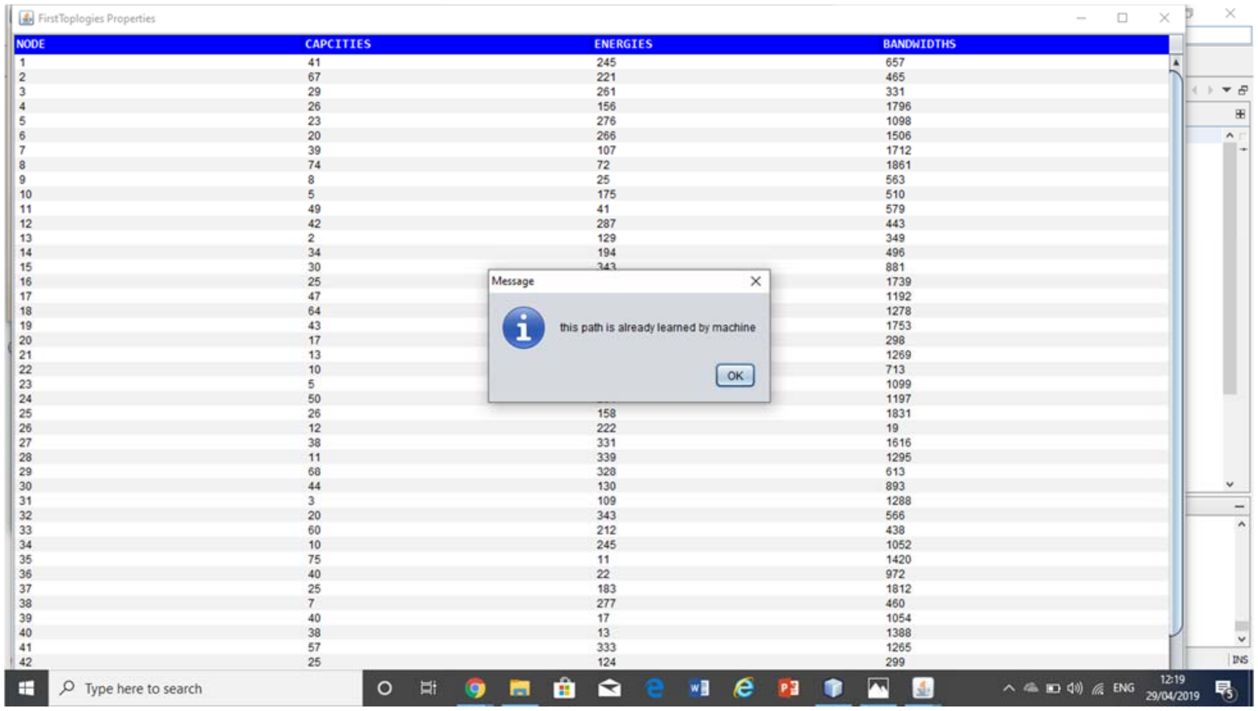
communication success...

OK

		ENERGIES	BANDWIDTHS
		245	657
		221	465
		261	331
		156	1796
		276	1098
		266	1506
		107	1712
		72	1861
		25	563
		175	510
		41	579
		287	443
		129	349
		194	496
		343	881
			1739
			1192
			1278
			1753
			298
			1269
			713
			1099
			1197
25	26	158	1831
26	12	222	19
27	38	331	1616
28	11	339	1295
29	68	328	613
30	44	130	893
31	3	109	1288
32	20	343	566
33	60	212	438
34	10	245	1052
35	75	11	1420
36	40	22	972
37	25	183	1812
38	7	277	460
39	40	17	1054
40	38	13	1388
41	57	333	1265
42	25	124	299

Type here to search

ENG 12:21 29/04/2019



FirstToplogies Properties

Message

this path is already learned by machine

OK

NODE	CAPCITIES	ENERGIES	BANDWIDTHS
1	41	245	657
2	67	221	465
3	29	261	331
4	26	156	1796
5	23	276	1098
6	20	266	1506
7	39	107	1712
8	74	72	1861
9	8	25	563
10	5	175	510
11	49	41	579
12	42	287	443
13	2	129	349
14	34	194	496
15	30	343	881
16	25		1739
17	47		1192
18	64		1278
19	43		1753
20	17		298
21	13		1269
22	10		713
23	5		1099
24	50		1197
25	26	158	1831
26	12	222	19
27	38	331	1616
28	11	339	1295
29	68	328	613
30	44	130	893
31	3	109	1288
32	20	343	566
33	60	212	438
34	10	245	1052
35	75	11	1420
36	40	22	972
37	25	183	1812
38	7	277	460
39	40	17	1054
40	38	13	1388
41	57	333	1265
42	25	124	299

Type here to search

ENG 12:19 29/04/2019

CONCLUSION

Based on the current existing scenario with the current work the conclusion is to make the edge node as cluster head for sure for broad casting. The proper broadcasting is getting done only with shortest path with which nodes are having more than aggregated threshold value. So to reduce duplicate broadcast this work is updated with machine learning model for updating of past paths.

REFERENCES

1. T. Koponen et al. Onix: a distributed control platform for large-scale production networks". In: OSDI. 2010.

2. N. Gude et al. NOX: towards an operating system for networks". In: Comp. Comm. Rev. (2008). M. Caesar et al. Design and implementation of a routing control platform". In: NSDI. 2005.

3. M. Casado et al. Rethinking Enterprise Network Control". In: IEEE/ACM Trans. on Networking 17.4 (2009).

4. P. Porras et al. A security enforcement kernel for OpenFlow networks". In: HotSDN. ACM, 2012. S. Shin et al. FRESCO: Modular Composable Security Services for Software-Dened Networks". In: Internet Society NDSS. 2013

5. N. McKeown et al. OpenFlow: enabling innovation in campus networks". In: Comput. Commun. Rev. (2008).

6. S. Sorensen. Security implications of software-dened networks. 2012. S. M. Kerner. Is SDN Secure? 2013.