Experimenting with Resilience and Scalability of Wifi Mininet on Small to Large SDN Networks

Ankit Kumar, Bhargavi Goswami, Peter Augustine

Abstract:Today everything is getting digitized where people want to be wireless by all aspects. There is a high demand of WiFi in every sector. Highest influence on network planning of newly developed network infrastructure is of SDN to meet the futuristic needs of upcoming technology. As a result, newly developed networks have become more adaptive to dynamic circumstances along with enhanced flexibility. Being globally connected, it is inevitable to obtain adequate services from data centers through Wi-Fi support on SDN Networks, which is still a dream. Thus, the target of the experiment performed and presented by the authors of this paper is to implement WiFi support on SDN. Further, authors have also demonstrated the scalability and resilience of SDN based WiFi Network on Mininet by testing performance parameters in various dynamic scenarios. This paper will have a high impact on the end users as SDN technology can be implemented as last mile technology using WiFi SDN.

I. INTRODUCTION

Entire world is going wireless in current scenario. Increasing demand of SDN based WiFi in today's scenario forecast the future of networks. As of now, wireless technologies like WiFi WiMAX 802.16, WLAN 802.11, Bluetooth 802.15, etc has minor support over SDN as the technology is implemented only on large data centers which has been experimented earlier by the authors[1]. While going through the literature available on research platforms and while referring the white paper along with RFCs, we could not come across any implementation of said technologies in SDN[2]. With this research experimentation, we bring out solutions that will implement WiFi on SDN with minimum complexities. Limitation of Existing Application includes following. 1) Very few functionalities of WiFi are available on SDN Architecture. 2) Existing WiFi networks does not support SDN implementation to end users[3].Software-Defined Networks (SDN) enables the users to create, configure, modify and troubleshoot, i.e. enables network management programmatically in an efficient manner, thus it is a profound concept which will build the future of networks[4]. The concept of SDN overcomes the limitations of static network architectures i.e. the traditional architectures which are more complex compared to the new ones that will be created programmatically, making it more flexible and easier to modify and troubleshoot[5].

Last mile technology is Wi-Fi and the world is growing to be connected through Wi-Fi. If SDN takes over the current infrastructure considering its prominent features of programmability and flexibility, it will be surely adapted by last mile users of Wi-Fi technology. It is inevitable to obtain adequate services from data centers through Wi-Fi support on technology such as Wi-Fi on SDN, which should become the reality. This creates a clear demand of Wi-Fi implementation on SDN. Further the newly developed SDN also needs to be tested under controlled conditions to check its performance stability in diversified criterions and circumstances.

Complete control using policy implementation over network devices from a central console happens through controller[6]. SDN permits a Network Administrators to change the way the network instruments such as Routers, Switches and Access Points handle the packets[7]. But, the Wi-Fi implementation has not been done on large scale SDN, that can actually help last mile users of Wi-Fi technology to adapt with SDN. While going through the literature available on the multiple research platforms, we found Ramon R. Fontes from University of Campinas (UNICAMP), Campinas, Sao Paulo, Brazil who came up with Mininet – WiFi emulator by developing a driver for existing WiFi devices making them communicate with Linux operating system to communicate with Mininet[8].

A methodology is being proposed in this article to perform the evaluation of Wi-Fi on SDN using TCP and UDP protocols. With this research project the team members brought out solutions to implement WiFi over SDN and test its stability along with resilience by testing performance parameters in various dynamic scenarios. Forthcoming sections include creation of topologies using Wifi-Mininet having network of switches, connected with each other. The network is designed to supports OpenFlow and forwarding using SDN Controller[9]. A reference to python script is taken into consideration for the creation of dynamic topology which was coded while experimenting with controllers like Beacon, Onos, Ryu, FloodLight and OpenDayLight[10]. While experimenting, use of graphing tools for displaying the results and drawing comparison is done[11]. Performance evaluation is done using multiple user platform diversified networking scenarios before coming to any conclusion. Comparison of the results with existing available solutions is done to determine futuristic approach towards research in the area of networking.

Revised Manuscript Received on December 22, 2018

Ankit Kumar ,CHRIST,Bangalore
Bhargavi Goswami,CHRIST,Bangalore
Peter Augustine,CHRIST,Bangalore

The paper is written with intention to demonstrate the experimentation done on Wifi Mininet to implement SDN Networks to test its resilience and scalability over emulation environment. Article is distributed in following sections. Next section explains about the wifi-mininet architecture, followed by Section III that explains tools and techniques used in the experiment conducted, followed by system design and implementation sections followed by results and discussion, finally conclusion, future scope and references.

II. WIFI MININET ARCHITECTURE

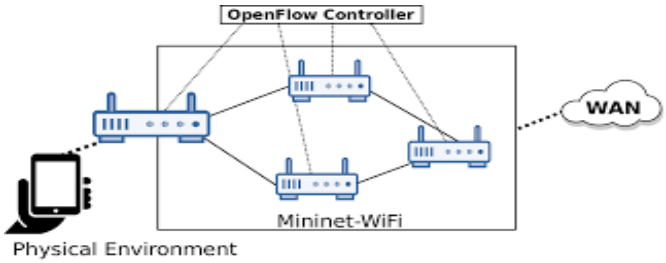


Fig.1 WiFi Mininet Architecture

In this section, the background and architecture of Wifi-Mininet is discussed. The system architecture as given in Fig 1 that shows the working of WiFi Mininet[12]. The physical environment communicates with the network of WLAN using Access Point. Further, network of wireless nodes is enabled to be controlled by policy of company using OpenFlow Controllers. Using the Mininet simulator and creating a network topology of maximum up to 500 nodes, the entire network is emulated in this experiment. Next, a connection between the client and server must be established and further communication continues. Using the concentrated result by filtering required parameters, Gnuplot is used to convert log files into graphs/plots and thus analysis is done for different topologies.

A python script is used to create topology and its execution. Using xterm for any two stations, client server connection must be established between them and iPerf input for traffic should be provided and the result must be obtained. As performance evaluation of resilience and scalability of Wifi Mininet is the objective, testing is done for both the TCP and UDP protocols and then conclusion is drawn after thorough comparison with each other.

The target of this experiment is to create a wireless support in SDN. Hence to attain this, a customized topology using python scripting is created as per our requirement on Mininet which would comprise of various components like switches, hosts and controller. Following mininet, Wi-Fi implementation is also done on MiniEdit which is out of bound of this paper. Finally the results are obtained for analysis with multiple executions as per the goal of the experiment to have Wi-Fi supported SDN system.

The development of the system is a challenge as the entire work is being developed on a virtual machine, which has its own limitations and complications due to space, RAM and various other factors which affect the smooth functioning of the system. Very limited content and literature survey is available for the system which is being implemented; hence it's a challenge to come up with such work with limited resources. Many a times, the system crashes or doesn't support the required necessary commands to be executed and thus it takes time to configure and bring back the system to the previous working state with the help of snapshots as there is no proper support from the network community for the errors and loopholes where the researchers get stuck. It also enables to transfer and reuse these components to further develop and modify, building a trial and error scenario for the developers to try-out different topologies, scripts and constraints and research to get the best out of it.

III. TOOLS AND TECHNIQUES

Mininet-WiFi: Mininet-WiFi is an extended version of Mininet software with added functionalities like WiFi stations and access points. Thus Mininet-WiFi is used for the implementation of this experiment by creating stations, access points, and controllers to emulate the WiFi scenario with mobility [13][14].

VMware: The major requirement of experimentation with WiFi Mininet was that virtual machine must be created that will run the Mininet-WiFi network emulator. VMware is a virtualization and cloud computing software which can be one of the options. Another option was VirtualBox because, it is open-source and runs on Windows, Mac OS, and Linux. VMware is opted as it can run on multiple OS can run on one physical server[15].

Gnuplot: Gnuplot is a portable command-line driven graphing utility. Gnuplot is used to generate graphs to display results and draw the comparison between the system. The graphs are generated based on various parameters. Widely acceptable by scientific community, Gnuplot is best fit for experiments executed on Linux platform[16].

Implementing Wi-Fi Support on SDN:
With the help of Mininet the topologies were created[17]. These topologies were implemented using Python script to generate tested efficiently[18].

Testing Wi-Fi Modules:
The performance evaluation will be tested using multiple user's networking scenarios. Use of graphing tools is made to display results based on parameters between the communicating system. A comparison is made between the existing solution and the proposed solution. The throughput, latency and packet delivery ratio is taken into consideration before coming up to a final conclusion.

Mininet WIFI installation [19]:
Step 1. Update ubuntu server using the following commands:

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

Step 2. Get Mininet-WiFi from source code:
$git clone git://github.com/intrig-unicamo /mininet-wifi.git.
Step 3. Open Mininet-WiFi folder:
$ cd mininet-wifi,
Step 4. Run the install script:
$ sudo util/install.sh –W ln fv.
Some of the install.sh options: -W: wireless dependencies, -n: mininet-wifi dependencies, -f: Open Flow, -v: Open v Switch, -l: w medium d, optional: -6: w pan tools.

SYSTEM DESIGN

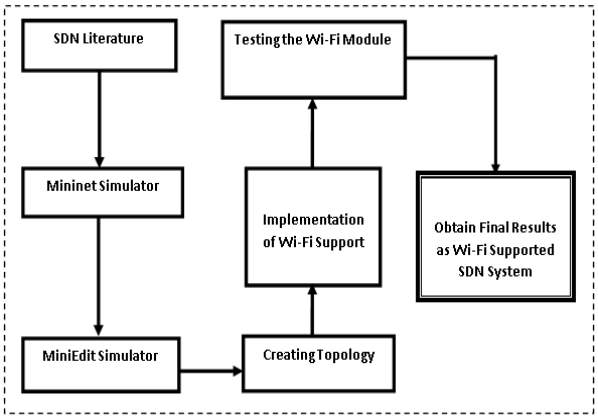


Fig 2. Block Diagram of the System

Software-Defined Networks (SDN) enables the users to build, configure, monitor and troubleshoot various networking devices and hence it has proved to be time-saving and prevents the tedious manual configuration of the setup, thus it is proves to be the future of networks. But SDN lacks the support of wireless technologies. As shown in the Fig. 2, block diagram of experiment has following steps. The reference of custom topology generation on minuet using python script as provided in [20] is taken to implement the stated scenario of the experiment using Wifi-Mininet simulator. The Wi-Fi based scenario generated is implemented which are tested using the TCP and UDP protocols for its stability in various dynamic scenarios. Finally, the results are obtained and use of graphing tools is done to visualize the comparison. There is a comparison drawn between the current and the existing scenarios to identify any futuristic approach through this experiment.

IV.IMPLIMENTATION

This section explains the generation of scripts to develop small to large WiFi Mininet Networks on SDN and further experimenting with diversified networking scenario. Two scripts were experimented during research. One with 10 WiFi Nodes Network and second with 30 WiFi Nodes. The scripts have repeated statements which are indicated by N as a generalized number for such repetition. Demonstration of 10 Nodes Script is explained as follows in Fig 3.

Steps to establish a connection between Client and Server station:
1. Run the mobility_10.py file which is a script file for creating a network of 10 stations.
2. Open terminal emulator with two stations (as client-server) using the command:
xterm sta1 sta9 ; where sta1 is source and sta9 is destination.
3. Establish Client-server connection with TCP and UDP protocols using following commands.
TCP Flow: Iperf -c (IP_of_Server) 10.0.0.150 -p 6653 -t 30 (Client side).
Iperf -s -p 6653 -i 1 > tcp_result (Server side).

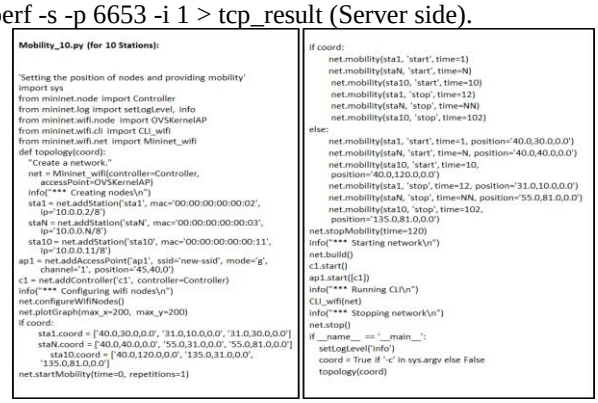


Fig. 3. Script to generate custom topology for experimentation.

UDP Flow: Iperf -c (IP_of_Server) 10.0.0.150 -u -p 6653 -t 30 (Client side).
Iperf -s -u -p 6653 -i 1> udp_result (Server side).

In a similar fashion, scripts with 30 nodes are created and experimented to test the performance of near to far neighbors in the complex uneven topology. The commands given above remains the same but, the number varies from 1 to 30. The same commands can be followed to generate scripts as per the requirement of the experiment. Once the experiment is in the execution state, logs are maintained for every single event occurring during the execution of the experiment. These obtained results are plotted in the form of graphs directly using the following set of commands.

Code to create graphs on various parameters: To study the behaviour of WiFi Mininet based SDN Network experiment is conducted for two different types of flows, TCP and UDP. To observe TCP flows behaviour in a wireless scenario, parameters taken into consideration are Bandwidth and Congestion Window and Retransmission. To observe the behavior of UDP flows over WiFi networking scenario, parameters considered are Bandwidth and Jitter. The reason behind different parameter set based on the type of flow is its basic behaviour. Connection-Oriented TCP follows the same path for each flow from source and destination which creates a chance of getting one path congested and bottlenecked, which can be proved if bandwidth and congestion windows are observed deeply.

Whereas, for Connectionless UDP follows multiple paths to make packets reach from source to destination and chances of a congested path is minimum as packets are distributed in multiple path. Here, what matters is the time taken by the flow to completely reach the destination where jitter plays a major role. Thus, for UDP flows, Bandwidth and Jitter are most suitable parameters to observe.

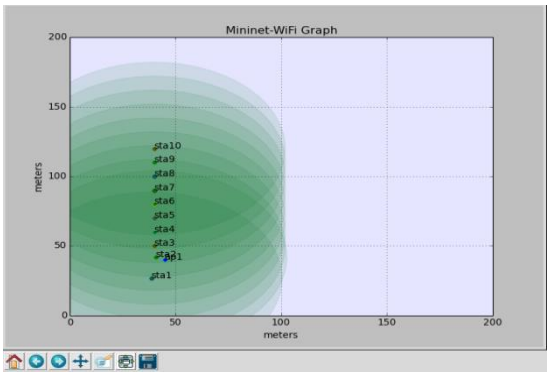
Creating TCP graphs:
For creating Bandwidth graph with 10 stations with TCP protocol, save the data of Time and Bandwidth from the above created tcp_result file:
$cat tcp_result | grep sec | head -100 | tr – " " | awk '{print $3,$7}' > tcp_band_10
The above command will create “tcp_band_10” file with the data of Time and Bandwidth. Plot the graph using gnuplot:
gnuplot> plot “tcp_band_10” title “TCP Bandwidth Flow_10” with linespoints.
gnuplot>set xlabel “Time(sec)”
gnuplot>set ylabel “Bandwidth(Mb/s)”
gnuplot>set xrange [0.0:30.0]
gnuplot>set yrange [0.0:30.0]
gnuplot>set xtics 0,1,30
gnuplot>set ytics 0.0,1,30.0
gnuplot>replot.

Similarly, Retransmission graph and Congestion window graph with 10 stations with TCP protocol can be created by copying their respective data into new files from the obtained tcp_result file.
For Retransmission graph:
$cat tcp_result | grep sec | head -100 | tr – " " | awk '{print $3,$9}' > tcp_retr_10.
For Congestion Window graph:
$cat tcp_result | grep sec | head -100 | tr – " " | awk '{print $3,$10}' > tcp_cwnd_10.
and plot the graph using gnuplot with appropriate range and labels.

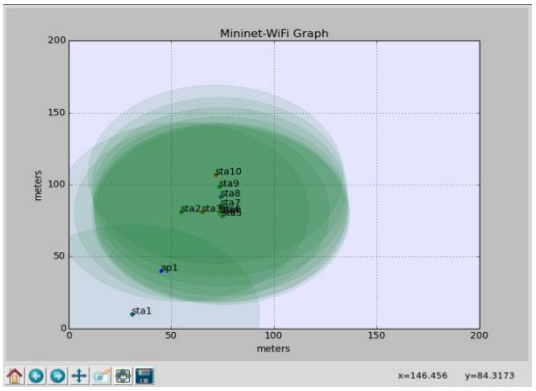
Creating UDP graphs:
Similarly, Bandwidth graph and Jitter graph with 10 stations with UCP protocol can be created by copying the respective data from the obtained udp_result file.
For Bandwidth graph:
$cat udp_result | grep sec | head -100 | tr – " " | awk '{print $3,$7}' > udp_band_10
For Jitter graph:
$cat udp_result | grep sec | head -100 | tr – " " | awk '{print $3,$9}' > udp_jitter_10
and plot the graph using gnuplot with appropriate range and labels.

The above steps are given for generating the graphs for 10 stations from the log files generated during experiment execution. Similar steps can be followed for creating graphs for 30 stations. The only difference is run mobility_30.py script instead of mobility_10.py and create the graphs for parameters with TCP and UDP flow.

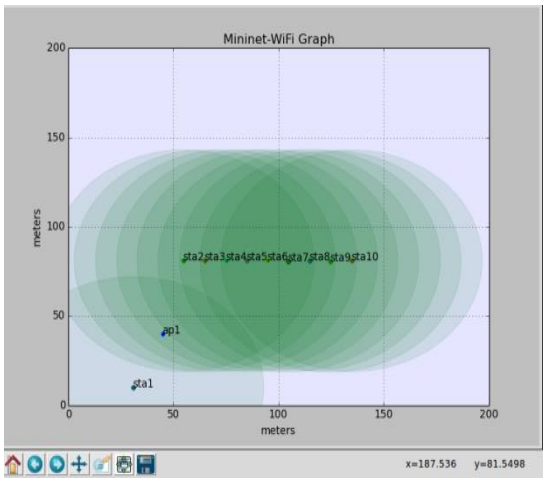
Mininet-WiFi (10 stations): Fig.4(a) shows the initial stage of the stations when we execute Mobility_10.py. Initially nodes are given position in vertical manner which changes while wireless nodes leaves its position and move towards horizontal axis. Fig.4(b) shows the intermediate stage of the stations. The stations are mobile hence they change their position. Fig.4(c) shows the final stage of the stations. Final position taken by the mobile stations at the end of the execution.



(a)Initial stage



(b) Intermediate stage

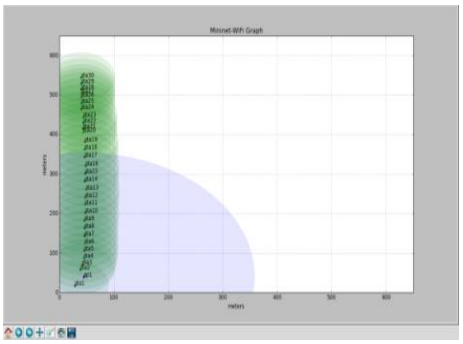


(c) Final stage

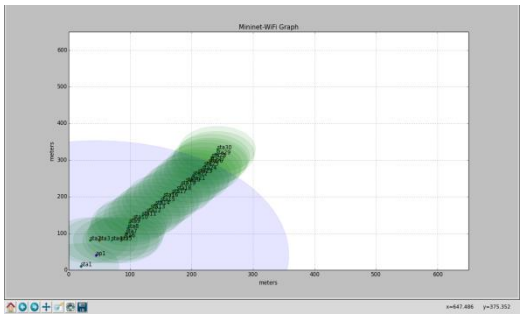
Fig 4. Mobility of 10 stations

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

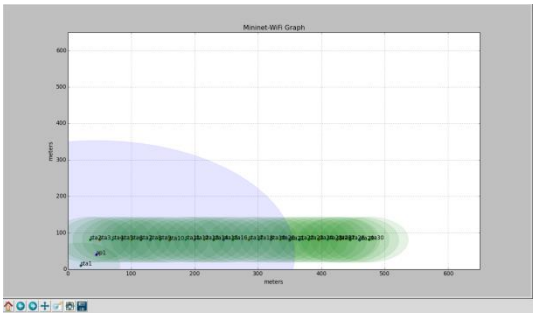
Mininet-WiFi (30 stations): Similarly, Fig.5(a),(b),(c) shows the different stages of the station when we execute Mobility_30.py



(a)Initial stage



(b) Intermediate stage

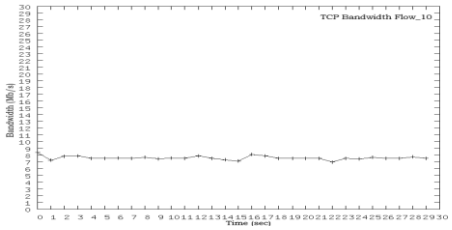


(c) Final stage

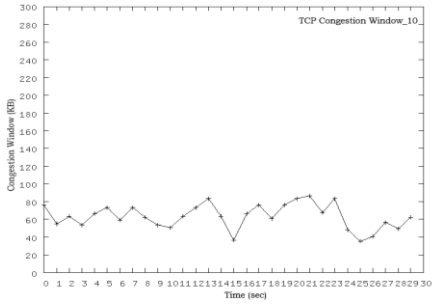
Fig 5. Mobility of 30 stations

V.RESULTS AND DISCUSSON

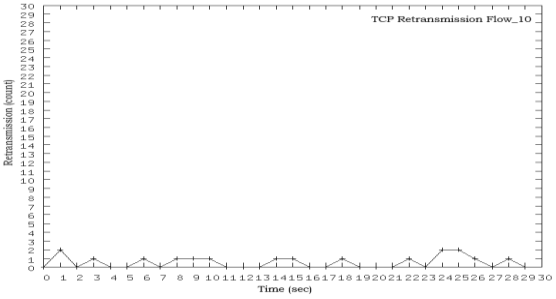
TCP-10 stations graphs are obtained based on the logged files are shown in Fig.6 where Fig.6(a) is Bandwidth, Fig.6(b) is Congestion Window and Fig.6(c) is Retransmission



(a) Bandwidth



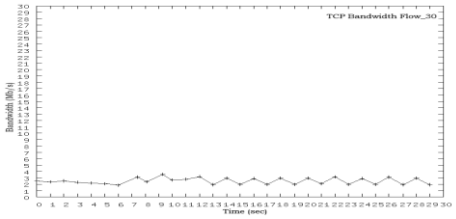
(b) Congestion Window



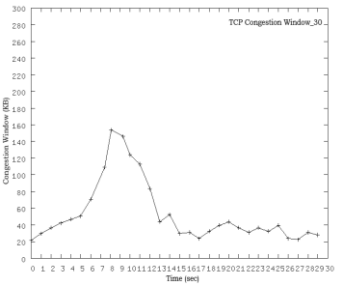
(c) Retransmission

Fig 6. TCP parameters for 10 stations

TCP-30 stations graphs are obtained based on the logged files are shown in Fig.7 where Fig.7(a) is Bandwidth, Fig.7(b) is Congestion Window and Fig.7(c) is Retransmission:

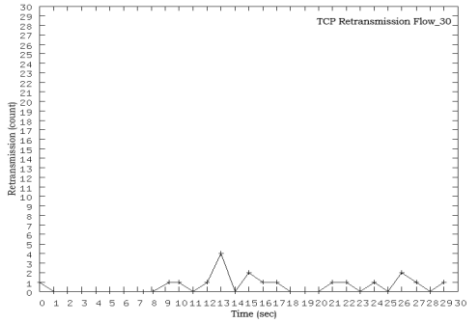


(a) Bandwidth



(b) Congestion Window

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

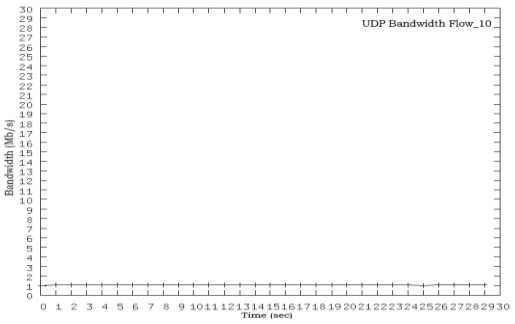


(c) Retransmission

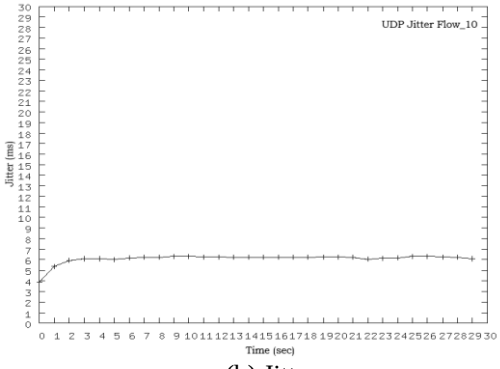
Fig 7. TCP parameters for 30 stations

Result comparison of TCP-10 and TCP-30: Bandwidth: With 10 stations, the bandwidth ranges from 7.50-8.40 Mbits/sec, while for 30 stations, the range is 1.85-3.54 Mbits/sec, thus we can observe that as the number of stations increases, the bandwidth decreases. Congestion Window: With 10 stations, the congestion window ranges from 35.4-86.3 KBytes, while for 30 stations, the range is 21.2-154 KBytes, thus we can observe that as the number of stations increases, the congestion window increases. Retransmission: With 10 stations, the retransmission count is 17, while for 30 stations, the range is 19, thus we can observe that as the number of stations increases, the retransmission also increases.

UDP-10 stations graphs are obtained based on the logged files are shown in Fig.8 where Fig.8(a) is Bandwidth and Fig.8(b) is Jitter:



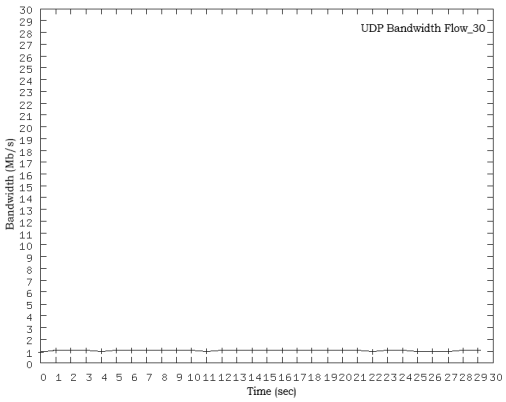
(a) Bandwidth



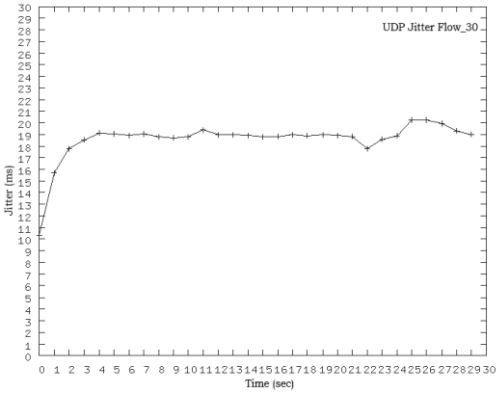
(b) Jitter

Fig 8. UDP parameters for 10 stations

UDP-30 stations graphs are obtained based on the logged files are shown in Fig.9 where Fig.9(a) is Bandwidth, and Fig.9(b) is Jitter



(a) Bandwidth



(b) Jitter

Fig 9. Resultant graphs of UDP parameters for 30 stations

Result comparison of UDP-10 and UDP-30: Bandwidth: Here with 10 stations and with 30 stations, we see that there is not much difference observed in the bandwidth,as it ranges between 0.9-1.1 Mbits/sec. Hence, we can say that with an increase in the number of stations, no significant change is observed with UDP protocol but in the TCP protocol, the bandwidth drastically reduces. Jitter: With 10 stations, the jitter ranges from is 3.875-6.306 ms, while for 30 stations, the range is from 10.314-20.242 ms, thus we can observe that as the number of stations increases, the jitter also increases in UDP protocol.

VI.CONCLUSION

Software-Defined Networking (SDN) enables us to configure, monitor and troubleshoot various network devices which prevents the tedious manual configuration setup work and in turn saves time and money of the users.

The required goal to evaluate the performance of the Wi-Fi on SDN was achieved using the TCP and UDP protocols. Majorly we checked the bandwidth, the congestion window and the retransmission for the TCP protocol, while we used the bandwidth and jitter for the UDP protocols. It was observed that the performance gradually reduced when the number of stations increased, we see that bandwidth reduced, and the retransmission also increased in case of the TCP protocol, while the UDP protocol shows that the Jitter increases as the number of stations increase but the bandwidth remains almost same. Hence, with implementation of Wi-Fi in SDN, we can figure out how the devices perform and what can be the result with increase in number of hosts.

VII .FUTURE SCOPE

While the authors have limited parameters to only Bandwidth, Congestion Window and Jitter, different parameters can be considered to evaluate the performance. The resource available with Desktop PC for experimentation has limitations of RAM, CPU and Cache Memory. More hosts, stations, access points can be taken into consideration to check the performance as enhancement of this paper. Other protocols can be used to draw comparison between them where as in this experiment observations are limited to OpenFlow Protocol. The number of stations on which the test can be performed is limited again due to resource limitations which can be enhanced with better research execution systems with different scenario that may vary the results.

REFERENCE

1. Goswami, Bhargavi, and Seyed Saleh Asadollahi. "Enhancement of LAN Infrastructure performance for data center in presence of Network Security." *Next-Generation Networks*. Springer, Singapore, 2018. 419-432.
2. S Das, B Goswami, S Asadollahi, “Investigating Software-Defined Network and Networks-Function Virtualization for Emergent Network-oriented Services”, International Journal of Innovative Research in Computer and Communication Engineering, vol. 5, no. 2, pp. 201 – 205, 2017, DOI:10.15680
3. Asadollahi, Saleh, and B. Goswami. "Revolution in Existing Network under the Influence of Software Defined Network." *Proceedings of the INDIACom 11th, Delhi, March* (2017): 1-3.
4. M Gosai, B Goswami, U Kar, (2014) Experimental Based Performance Testing of Different TCP Protocol Variants in comparison of RCP+ over Hybrid Network Scenario, International Journal of Innovations & Advancement in Computer Science (IJIACS), Vol 3, Issue 2, Pg. No. 31 to 37, India.
5. Asadollahi, S., Goswami, B., & Gonsai, A. M. “Implementation of SDN using OpenDayLight Controller”.International Journal of Innovative Research in Computer and Communication Engineering, vol. 5, no. 2, pp. 218–227, 2017.
6. Asadollahi, Saleh, et al. "Scalability of software defined network on floodlight controller using OFNet." *Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT), 2017 International Conference on*. IEEE, 2017.
7. B Goswami, S Asadollahi, “Performance Evaluation of Widely Implemented Congestion Control Algorithms over Diversified Networking Situations”, ICCSNIT – 2016, Pattaya, Thailand. Open Access.
8. Ramon dos Reis Fontes , Christian Esteve Rothenberg, “Mininet-WiFi: A Platform for Hybrid Physical-Virtual Software-Defined Wireless Networking Research”, Proceedings of the 2016 ACM SIGCOMM Conference, August 22-26, 2016, Florianopolis, Brazil [doi>10.1145/2934872.2959070
9. S Asadollahi, B Goswami (2017) “Experimenting with Scalability of Floodlight Controller in Software Defined Networks”, International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT), IEEE, Mysore, India.
10. Asadollahi, S., Goswami, B. (2017). Software Defined Network, Controller Comparison. International Journal of Innovative Research in Computer and Communication Engineering, vol. 5, no. 2, pp. 211–217, 2017. Open Access.
11. P.K. Janert, Gnuplot in Action, Manning Publications Co., Shelter Island, NY (2009)
12. R. R. Fontes, S. Afzal, S. H. B. Brito, M. A. S. Santos and C. E. Rothenberg, "Mininet-WiFi: Emulating software-defined wireless networks," 2015 11th International Conference on Network and Service Management (CNSM), Barcelona, 2015, pp. 384-389, DOI: 10.1109/CNSM.2015.7367387
online] Available: https://github.com/intrig-unicamp/mininet-wifi
14. Mininet-WiFi:SDN emulator supports WiFi networks, [online] Available: http://www.brianlinkletter.com/mininet-wifi-software-defined-network-emulator-supports-wifi-networks/
15. Sugerman, Jeremy, Ganesh Venkitachalam, and Beng-Hong Lim. "Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor." USENIX Annual Technical Conference, General Track. 2001.
16. Racine, Jeff. "gnuplot 4.0: a portable interactive plotting utility." Journal of Applied Econometrics 21.1 (2006): 133-141.
17. De Oliveira, Rogério Leão Santos, et al. "Using mininet for emulation and prototyping software-defined networks." 2014 IEEE Colombian Conference on Communications and Computing (COLCOM). IEEE, 2014.
18. [online]Available: https://stackoverflow.com/questions/31590526/how-to-use-python-scripts-or-py-files-in-mininet
19. dos Reis Fontes, Ramon, and Christian Esteve Rothenberg. "The User Manual." (2005).
20. Pal, Chandan, et al. "Implementation of simplified custom topology framework in Mininet." Computer Aided System Engineering (APCASE), 2014 Asia-Pacific Conference on. IEEE, 2014.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication