International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Index Copernicus Value (2016): 79.57 | Impact Factor (2017): 7.296

Survey for Traffic Engineering in Software Define Networks

ZAHRAA ALAA BAQER[1], Mohammed Najm Abdulla[2]

[1, 2]Department of Computer Engineering, University of Technology, Baghdad, Iraq

Abstract: ***As a result of the development in applications of communication networks with emergence of software define networks (SDN) decouples the control plane and data plane of the networks. therefore, the traffic engineering (TE)mechanisms is one of the most important technologies to be within the application layer which optimizing traffic, improve network robustness, and avoid network congestion. This papers reviews the TE methods for SDNs that exploited data networks depend on several criterial with respect to continuous update of network configuration in term of achievable network throughput.***

Keywords: software define networks (SDN), TE mechanisms, open flow (OF), SDN controller, Mininet, northbound interface(NBI), southbound interface (SBI)

1. Introduction

Software define networks is an emerging technology and is a shortcut to the next generation of infrastructure in network engineering. SDN is a design that means to make systems coordinated and adaptable. The objective of SDN is to enhance arrange control by empowering undertakings and specialist organizations to react rapidly to changing business necessities. that distinguished by separation control plane from forwarding data plane in network system[1], and centralized visibility and network control also network programmable by external applications as single, logical with entity and the control plane comprises the controller , the controller gives a high deliberation dimension of sending components the board which missing in the present systems. In this way, the controller is basic segment of the SDNs design that added achievement or disappointment of SDN , the SDN controller are given in [2].the traditional network is quite different from the SDN network as Figure [1] .SDN architecture include three layers as shown in Figure [2], that summarize the SDN architectures in network systems. The control plan layer contains the controller to manage data plane layer. The application layer connected to control plane layer via northbound interface(NBI) or known controller application interaction to interface between application layer and SDN controller .The NBI empower regular system administrations , for example security , TE , routing and quality of services (QoS) while the forwarding data plan layer connected to control plane layer via southbound interface (SBI)(e.g., open-flow protocol)[3] . The open flow is a protocol between SDN controller and switching devices that defines an API (application program interface) between forwarding layer and controller in SDN. The OF protocol[4] is supported network devices that correspondence between the control and data planes which access the forwarding data layer to provide an external application of network devices . A standout amongst the most generally utilized SDN empowering agent is the Open-Stream convention. This can be enables the SDN controller to deal with OF switches. The OF switches contain flow tables, a group table, and OF channel as Figure [3]. The flow tables and the group table are utilized for parcel query and after that to forward the bundles. The of channel is a reflection layer. It sets up a safe connection between every one of the switches and the controller by means of the convention. Table 1 records a portion of the as of now SDN accessible switches. In this paper, the idea of TE issues and mechanisms based on SDN open flow techniques. The rest of the paper surveys some of the TE techniques, and it is organized as follows: Section 2 describes an overview of TE in SDN and the assimilation from them. Section 3 describes implementation and testing of SDN. In Section 4, conclude the paper.

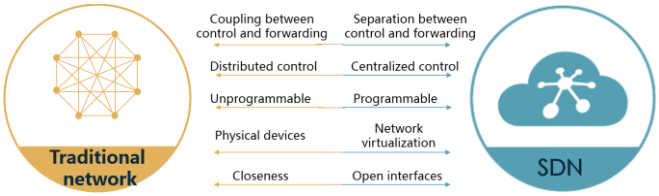

Figure 1: Distinction among the traditional and SDN

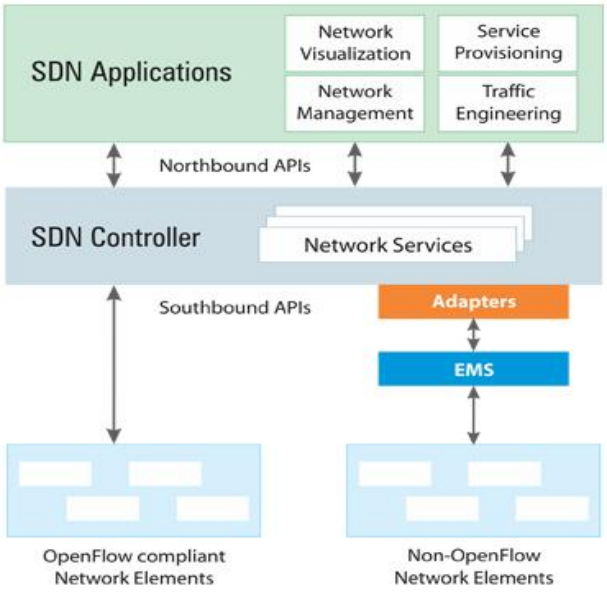

Figure 2: Architecture of SDN

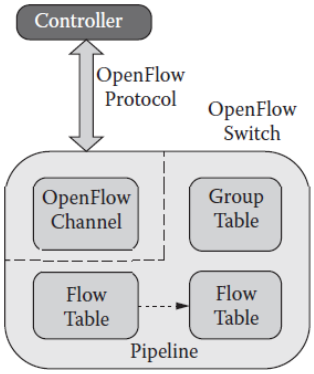

Figure 3: The OF protocol of SDN

Table 1: Switches in SDN

Switch	Type	Series/ Versions	OpenFlow version
Arista [38]	HW	7050 ,7150, 7500	1.0
Brocade [39]	HW	CES 2000, CER 2000, MLX	1.0, 1.3
HP [40]	HW	3500, 3500yl, 5400zl, 6200yl, 6600	1.0, 1.3
IBM [41, 42]	HW	IBM 8264, RackSwitch G8264, G8264T	1.0
LINC[43]	SW	-	1.2,1.3,1.4
NEC [44]	HW	PF5240, PF5248	1.0 , 1.3.1
Open vSwitch (OVS) [45]	SW	Latest version OVS 2.1.2	OpenFlow 1.0 for OVS 1.9 and earlier, OpenFlow 1.2 and 1.3 for OVS 1.10 and later with some missing features, OpenFlow 1.1 for OVS 2.0 and later with some missing features, experimental support of OpenFlow 1.4 for OVS 2.2
Pica8[46]	HW	P-3290, P-3295, P3930 , P-3297 , P-3922	1.0, 1.1, 1.2, 1.3, 1.4

2. Traffic engineering in SDN

The main goal of most applications is to engineer traffic with the aim of reducing power consumption, maximizing aggregate network utilization, providing optimized load balancing, and other generic traffic optimization techniques. In SDN-based systems the controller can powerfully change the system state, for instance, in conventional systems the connection cost for steering conventions, for example, IS-IS are kept static for a significant lot. On the off chance that clog occurs in the organize it might prompt poor conveyance of information till the connection costs are changed or the issue is settled. In any case, in SDN these qualities can be changed all the more powerfully to adjust to the changes. Progressively creative directing system can be actualized, or the current directing conventions can be altered, with the goal that they can change powerfully according to organize state to improve asset use, stay away from disappointment what's more, clog, and enhancement QoS. To summarize the traffic engineering techniques in the SDN technology by the research community as show in table 2. The conduct of the transmitted data must be management to optimize the performance of communication networks. The TE used in past and current as ATM,IP,MPLS networks [5-7]that show in figure[4] ,and management on important aspects such as load balance , characteristics of traffic analysis and traffic management for identify and remove errors from network systems , checking network covering various aspects of important network in SDN with data transmitting. When SDN are effective for TE into existing network and using algorithm to solving incremental introduce in SDN for TE, that depend on used Fast Fully Polynomial Time Approximation Schemes and deployment partial and improve FPTAS[8].

TE is a vita system application , which reference framework in SDN [9] that presented on two parts : management and measurement traffic in SDN .In measurement traffic based on networks parameters, traffic analysis, generic measurement and prediction that supported into management traffic to improve network availability and performance , due to the framework of management traffic in SDN include load balance, QoS , energy saving scheduling and management to hybrid IP-SDN [8].In energy saving for networks by using some algorithms to improve the TE as use routing algorithm which are based on SDN environment[10][11] .

QoS parameters to improve the performance of throughput in SDN and using bandwidth reservation methods[12] for user based and flow based in Figure[5] to evaluated the performance in SDN to suitability according to fixed parameters as in table 3 . Also concentrate on used new methods to cluster [13] related in the multi-dimensional traffic ,which can achieve better performance in TE clustering ,providing good quality routing when some overlap between clusters and flexibility on routing configuration based on TE to get optimal routing .

Figure 4: SDN in past and future

Table 2: TE technique in SDN

Technique	*Description*	*Routing*	*Comments*
B4[14]	• It uses a centralized TE, layered on top of the routing protocols, • To achieve fairness it allocates	• It uses hashed based ECMP to balance the load among multiple links.	• if TE service can be stopped so that the packets are forwarded using • short path forwarding mechanism.

	resources • Using Min-Max fairness technique.		
Hedera [15]	• Detects the elephant-flows at the edge switches, • If threshold is met, i.e. 10% of • Nic bandwidth, the flow is marked as elephant flow • Uses periodic pulling, every 5 s.	• Uses the global view of network • And calculate the better paths, • Which are non-conflicting, for the • Elephant flows.	• It achieves 15.4 b/s throughput, • Achieves better optimal bisection of bandwidth of network, in comparison • To ECMP, • Periodic pulling can cause high • Resource utilization in switches.
Devo Flow[16]	• Detects the elephant-flows at the edge switches, • If threshold is met, i.e. 1–10 mb, • It marks the flow as elephant flow.	• It uses aggressive use of wild Carded OF rules, and a Static multi-path routing algorithm to forward the traffic.	• It can improve throughput up to • 32% in CLOS network.
Mahout [17]	• Detects the elephant-flows at end host using a shim layer, the default • Threshold is 100 k, and then the • Flow is marked as elephant-flow, • It uses in-band signaling to inform the controller about the elephant flows.	• It computes the best path for Elephant-flow; otherwise it forwards • Other flows using ecmp, • It calculates the path that is least Congested by pulling the Elephant-flow statistics and link utilization from switches.	• It can detect elephant flow, if • Threshold is 100 k, in 1.53 ms, • It has 16% better bisection than ECMP.
Micro TE[18]	• Detect the elephant flows at end host, • It calculates the mean of traffic matrix between tor-tor, if the mean and traffic is between d of each other, default is 20%, then it is predictable.	• Uses short term predictability to route the traffic on multiple paths, • The remaining streams are routed by the EMCP scheme with heuristic Threshold.	• If traffic is predictable it performs • Close to optimal performance otherwise • It performs like ECMP.
Mice Trap[19]	• It addresses the mice-flows, • Uses end-host elephant-flow detection • To distinguish between mice flows, and elephant flows.	• It aggregates the mice-flows to improve scalability, • It route the mice-flows using a Weighted multi.	• n/a.
Rethinking Flow Classification in SDN[20]	• It is a tag-based classification, • Source-edge switch tags the packets based on the application classes.	• The tag is also an identifier for matching & forwarding the packets.	• It is 3 ms faster than the of field matching, • it requires introduction of new api's to the data plane.
Atlas[21]	• It classifies each application uniquely. • It uses C5.0 machine learning tool to classify the applications, • It requires user to install agents on their mobile devices to collect information to train ML trainer.	• It routes the flow based on applications, and network requirements.	• It has about 94% accuracy, • Requires extension to of.
MSDN-TE[22]	• It gathers network state information and considers the actual path's load to forward the flows on multiple paths.	• It dynamically selects the best • Shortest path among the available paths.	• It has better performance over other • Forwarding mechanisms such as • Shortest path first, • It reduced download time by 48%.

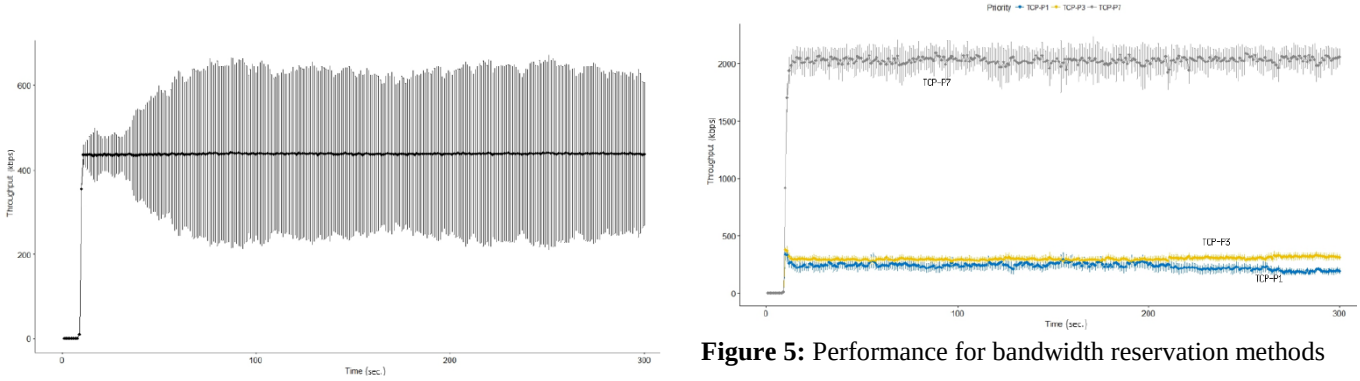

Figure 5: Performance for bandwidth reservation methods

Table 3: Parameters for user based bandwidth reservation

Parameter	Value
TCP packet size	66 bytes - 1514 bytes
UDP packet size	132 bytes - 1066 bytes
TCP packet generating interval	0.109 ms - 989 ms
UDP packet generating interval	2.16 ms - 90 ms
Wireless net. Total bandwidth	30 Mbps
Wired net. total bandwidth	100 Mbps
AP-OF switch-router link delay	0.5 ms
Access Point transmit power	-16 dbm
Max data rate supported by Access Point	150 Mbps
Frequency bands used by Access Point	2.4 GHz and 5 GHz
Client connection link delay	5 ms
Flow priority range	1-7 (lowest-highest)

2.1 Implementation of TE

TE App isn't an inside RYU App; it is a remote Python App that use the REST API of the RYU controller. RYU controller is high quality controller for production environments. As information, it needs an arrangement document in Json design that portrays the traffic relations, while the topology and the limit of the connections are recovered utilizing the REST API of the Topology module and of the OFCTL module, which give separately the topology and the speed of the ports. It is created by three sections: recovery of the information, heuristic calculation, and establishment of the guidelines. The last advance is accomplished utilizing the API rest "flowentry/include" of the OFCTL module, which permits to push governs in the OF switches as show in Figure [6].

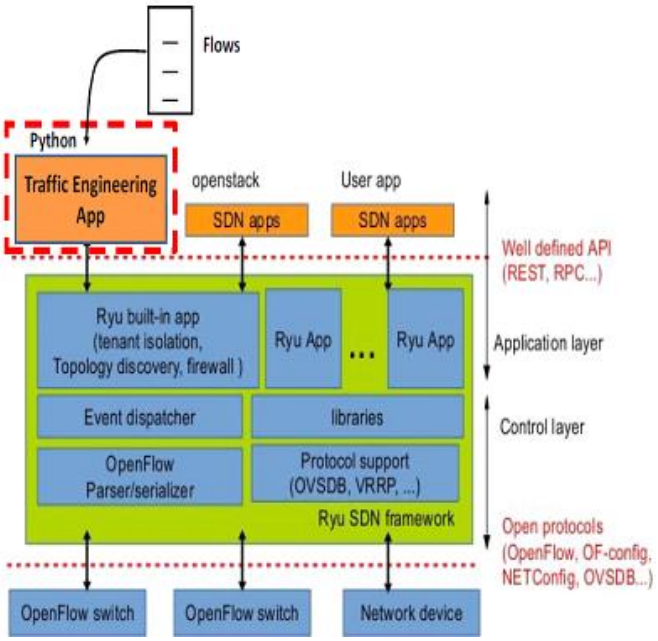

Figure 6: Implementation of TE

2.2 TE algorithm

In the OSHI(Open Source Hybrid IP/SDN) systems, the TE has been tended to from a specific perspective: we have understood a heuristic ready to settle with an estimate the stream task issue[23][24]. the heuristic arrangement is fundamental on the grounds that the correct arrangement is computationally unpredictable. Through the stream task, we can characterize an ideal parcel of the traffic on autonomous numerous ways. This strategy, adjusting the heap on the connections, can lessen the postponement experienced on the joins, subsequently staying away from situations where a few connections are more over-burden than the other. The stream task issue moves the streams in the system (the courses utilized by the streams are the variable of the issue) with the point of decrease the worldwide intersection time of the system T, which can be communicated as the entirety of the deferral experienced on the connections of the system on account of the consequences of Jackson. Specific conditions must be expected such Poisson landings, freedom among entries and takeoffs, and so forth (for further subtleties see[25]). In these conditions, each connection can be demonstrated as M/M/1 line, and we can figure effectively the reaction time of the line. Accepting a M/M/1 line, if the heap on the connection approaches the limit of the connection, the normal reaction time goes to limitlessness, so the stream task moves the traffic in the organize designating the heap on the other accessible connections and keeping away from the over-burden in the basic connections.

The heuristic is characterized by three stage:

1) Introduction stage;
2) Compelled Shortest Path First (CSPF) stage, where a first assignment of the streams is figured it out;
3) Heuristic re-task stage, where we understand the re-assignments of the streams all together to limit the intersection time of the system; Amid the instatement stage, the calculation recovers the information essential for its legitimate execution, at that point the second stage begins.

3. Implementation and testing of SDN

As referenced in past segment, numerous switches and controllers are accessible for actualizing a product characterized organize. However, there is likewise routes for testing thoughts in SDN, for example, simulators, emulators and test-beds.

3.1 SDN Controllers

SDN controllers or network operating system (NOS) can be classified on centralized or distributed architecture as show in table 4[26][27][28][29][30][31][32][33] and Controllers can be ordered dependent on a few qualities as depicted in figure [7].

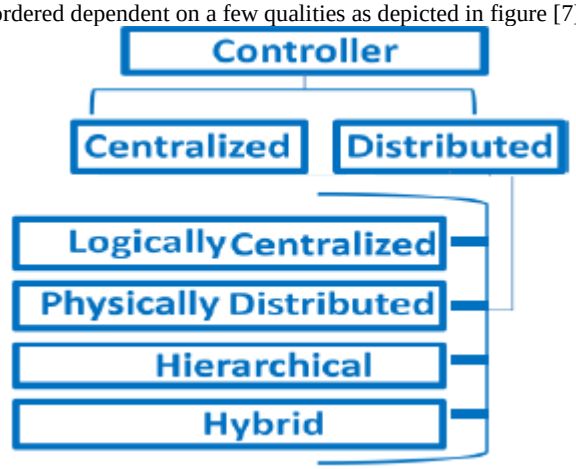

Figure 7: Classification of SDN controller

Table 4: Classification of SDN-Controllers platforms

Controller	Language	Created by	Open source	OpenFlow version	Description
NOX	Python, C++	Nicira	Yes	1.0, 1.3	Asynchronous, event-based programming model, component based framework. NOX-MT is multithreaded with improving throughput and response time ·
POX	Python (2.7)	Nicira	Yes	1.0	Component based framework, targets Linux, Mac OS, and Window ·
Beacon	Java	Stanford university	Yes	1.0.1	Event based and threaded Cross-platform, Dynamic, and Rapid Development ·
Maestro	Java	Rice university	Yes	1.0	Modular network control applications, multi-thread ·
Floodlight	Java	Big Switch Networks	Yes	1.0	Based on Beacon, core architecture is modular, open source agent (Indigo) ·
Floodlight-plus	Java	Big Switch Networks	Yes	1.3	New version of floodlight for supporting OF 1.3 ·
Ryu	Python	NTT Labs	Yes	1.0, 1.2, 1.3, 1.4	Component based, supporting components development in other languages, event management and reusable NETCONF library, sFlow/Netflow library ·
Open Daylight	Java	Linux Foundation	Yes	1.0 , 1.3	Modular, pluggable, and flexible controller platform, supporting multiple southbound protocols ·

3.2 Simulators and emulators

In this section available network simulators and emulators for SDNs such as Mininet, NS-3 and EstiNet technologies are described and compared as show in table 5.

- Mininet is a system emulator which makes a system of virtual hosts, switches, controllers, and connections. Mininet has run standard Linux arrange programming, and its switches bolster OF for profoundly adaptable custom steering and SDN[34]. Mininet is utilized generally in view of: quick to begin a straightforward system, supporting custom topologies and bundle sending, running genuine projects accessible on Linux, running on PCs, servers, virtual machines, having sharing and recreating capacity, simple to utilize, being in open source and dynamic advancement state. Conversely with these focal points, Mininet has a few weaknesses as well: having inability of exchanging tremendous measure of information in one single framework, non-accessible supporting discretionary OF controllers, supporting only one stage (Linux portion), doing NAT out of box, having sharing host document framework and PID space and virtual time idea nonappearance[35][36].
- NS-3 is a discrete occasion arrange test system which is suited for specialists and teachers. The NS-3 library is part crosswise over numerous modules composed under the modules tab. One of these modules is OF comparable to SDN. NS-3 has OF Switch Net Device protest acts as a switch and is OF perfect. This question executes a stream table for every single got bundle and furthermore an association with controller simply like SDN architecture. This test system has these favorable circumstances: including new conventions, deficiency of separation among genuine system and reproduced organize, having reconciliation and adaptable without redoing the center of test system. NS-3 hindrances are: loss of accessible models, nonappearance of visual interface for making topology and obvious capacity in trial level[37][38].
- EstiNet 8.0[39][40] test system and emulator underpins a large number OF switches. That compatible with Linux open source network application in SDN. EstiNet network simulation and emulation for SDN in virtual network platform. However, this can also use with several OF controller as RYU, NOX, and POX. Therefore, EstiNet OF supported the performance of SDN controller in real application programs. In the copying method of EstiNet, controllers can keep running up on an outer machine that is unique in relation to the machine which mimics switches. Additionally, in this mode usage of the controller as a devoted equipment gadget utilizing an Ethernet link is conceivable, coming about remote controlling. Alternate advantages of utilizing this test system/emulator are: exactness, briskness, redundancy and versatility.

Table 5: Classification of SDN-simulators/emulators

Simulator /emulator	Open source	Language	Platform	License	OpenFlow version	Docs
Mininet (Emulator)	Yes	Python	Ubuntu or experimentally Fedora	BSD open source	OF 1.3 of the reference user switch and NOX from CPqD and Ericsson	Good
NS-3 (simulator)	Yes	C++, Python	Linux, Mac, Free BSD	GNU GPLv2	Pre OF 1.0 and version of OF-SID that support MPLS	Good
EstiNet (emulator/ simulator)	NO	-	Linux	-	OF 1.3.2 and 1.0.0	Medium

3.3 Debuggers

As referenced before SDN controller is programmable and this element expands the likelihood of incidentally blunders. For the most part, discovering bugs is hard and tedious in this way debuggers have turned out to be one of the imperative segments OF on SDN. Debuggers are instruments that are utilized to test and analyze program and empower software engineers to interface with program while it is executing on PC. OF debuggers enable us to follow bundle stream conduct to check whether the system is working not surprisingly[41]. The fundamental data for troubleshooting is assembled from stream table, diverse traffic measurements and controller messages. Some extraordinary instruments were utilized for investigating reason and checking traffic among controller and switches utilizing Wireshark combination, which was expressed as the best methodology[42][43][44][45].

4. Conclusion

In this paper, we presented a survey for one of the most prominent methods in the application layer which is traffic engineering (TE) technology in SDN architecture .This can optimize the performance of networks and resource utilization, we show how to adaptable the performance of transmitted data in SDN environment .Therefore which can be achieved with TE mechanisms and discuss that related technologies from four aspects including traffic analysis, fault tolerance, flow management and topology update in current SDN. This structure can robotize the system arrangement to accomplish high QoS for the ideal applications.

References

[1] P. A. Morreale and J. M. Anderson, *Software defined networking: Design and deployment*: CRC Press, 2014.
[2] O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab, "SDN controllers: a comparative study," in *Electrotechnical Conference (MELECON), 2016 18th Mediterranean*, 2016, pp. 1-6.
[3] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using openflow: A survey," *IEEE*

communications surveys & tutorials, vol. 16, pp. 493-512, 2014.
[4] OpenFlow Protocol. http://www.openflow.org/wp/learnmore/
[5] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in SDN-OpenFlow networks," C*omputer Networks*, vol. 71, pp. 1-30, 2014.
[6] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional IP routing protocols," I*EEE communications Magazine*, vol. 40, pp. 118-124, 2002.
[7] A. Sánchez-Monge and K. G. Szarkowicz, M*PLS in the SDN Era:* O'Reilly Media, 2015.
[8] S. Agarwal, M. Kodialam, and T. Lakshman, "Traffic engineering in software defined networks," in I*NFOCOM, 2013 Proceedings IEEE*, 2013, pp. 2211-2219.
[9] Z. Shu, J. Wan, J. Lin, S. Wang, D. Li, S. Rho, *et al.*, "Traffic engineering in software-defined networking: Measurement and management," I*EEE Access*, vol. 4, pp. 3246-3256, 2016.
[10] B. N. S. Dr. Mohammed Najm Abdulla1, "Comprehensive Study on Software Defined Network for Energy Conservation," International Journal of Advanced Research in Computer and Communication Engineering, 2016.
[11] M. N. A. A. S. B. N. Shaker, "Energy Saving Based Routing Algorithms in SDN Environment," vol. 6, 2017.
[12] A. H. Moravejosharieh, M. J. Watts, and Y. Song, "Bandwidth Reservation Approach to Improve Quality of Service in Software-Defined Networking: A Performance Analysis," in 2018 *15th International Joint Conference on Computer Science and Software Engineering (JCSSE), 201*8, pp. 1-6.
[13] D. Sanvito, I. Filippini, A. Capone, S. Paris, and J. Leguay, "Adaptive Robust Traffic Engineering in Software Defined Networks," arXiv *preprint arXiv:1712.05651, 2017*.
[14] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, et *al., "B4*: Experience with a globally-deployed software defined WAN," in ACM S*IGCOMM Computer Communication Review, 201*3, pp. 3-14.
[15] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic Flow Scheduling for Data Center Networks", in Proc. 7th USENIX Symp. on Netw. Syst. Design & Implemen. NSDI'10, San Jose, CA, USA, 2010, vol. 10, pp. 19–19.
[16] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: scaling flow management for highperformance networks", ACM SIGCOMM Comp. Commun. Rev., vol. 41, no. 4, pp. 254–265, 2011.
[17] A. R. Curtis, W. Kim, and P. Yalagandula, "Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection", in Proc. 30th IEEE Int. Conf. Comp. Commun. IEEE INFO-COM 2011, Shanghai, China, 2011, pp. 1629–163.
[18] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: Fine grained traffic engineering for data centers", in Proc. 7th Conf. on Emerg. Networking Experim. & Technol. Co-NEXT'11, Tokyo, Japan, 2011, p. 8.
[19] R. Trestian, G.-M. Muntean, and K. Katrinis, "MiceTrap: Scalable traffic engineering of datacenter mice flows using OpenFlow", in IFIP/IEEE Int. Symp. on Integr. Netw. Managem. IM 2013, Ghent, Belgium, 2013, pp. 904–907.
[20] H. Farhadi and A. Nakao, "Rethinking flow classification in SDN", in Proc. IEEE Int. Conf. on Cloud Engin. IC2E 2014, Boston, MA, USA, 2014, pp. 598–603.
[21] Z. A. Qazi et al., "Application-awareness in SDN", ACM SIGCOMM Comp. Commun. Rev., vol. 43, no. 4, pp. 487–488, 2013.
[22] K. T. Dinh, S. Kukliński, W. Kujawa, and M. Ulaski, "MSDNTE: Multipath Based Traffic Engineering for SDN", in Intelligent Information and Database Systems. Asian Conference on Intelligent Information and Database Systems, N. T. Nguyen, B. Trawiński, and R. Kosala, Eds. Springer, 2016, pp. 630–639.
[23] L. Fratta, M. Gerla, L. Kleinrock, The flow deviation method: an approach to store-and-forward communication network design, Network, 3(2):97-133, 1973, John Wiley & Sons
[24] M. Gerla, L. Kleinrock, On the topological design of distributed computer networks, IEEE Transactions on Communications, 25(1):48-60, 1977
[25] Jackson network - http://en.wikipedia.org/wiki/Jackson_network
[26] NOX ,available online: http://www.noxrepo.org/nox/about-nox/, last visit:18.10.2014.
[27] POX , available online: http://www.noxrepo.org/pox/about-pox/, last visit:18.10.2014
[28] Beacon , available online: https://openflow.stanford.edu/display/Beacon/Home, last visit:18.10.2014.
[29] E. Ng, "Maestro: A System for Scalable OpenFlow Control," available online: www.cs.rice.edu/~eugeneng/papers/TR10-11.pdf, last visit:18.10.2014.
[30] Floodlight OpenFlow Controller - Project Floodlight, available online: http://www.projectfloodlight.org/floodlight/, last visit:18.10.2014.
[31] Announcing release of Floodlight with OF 1.3 support, available online: http://sdnhub.org/releases/floodlight-plus-openflow13-support/, last visit:18.10.2014.
[32] Ryu 3.9 documentation, available online: http://ryu.readthedocs.org/en/latest/getting_started.html#what-s-ryu, last visit:18.10.2014.
[33] Opendaylight. available online: http://www.opendaylight.org/, last visit:18.10.2014.
[34] Mininet: An Instant Virtual Network on your Laptop (or other PC), available online: http://mininet.org/, last visit:18.10.2014.
[35] Introduction to Mininet - mininet/ mininet wiki – GitHub, available online: https://github.com/mininet/mininet/wiki/Introduction-to-Mininet, last visit:18.10.2014.
[36] M. Q. M. N. A. TariqV, "S.D.N IMPLEMENTATION USING MININET," International Journal Of C*ore Engineering & Management, vol. 4, August-2017.*

[37] *n*s-3, available online: http://www.nsnam.org/, last visit:18.10.2014.
[38] ns-3: ns-3 Documentation, available online: http://www.nsnam.org/docs/release/3.19/doxygen/index.html, last visit:18.10.2014.
[39] EstiNet Technologies, available online: http://www.estinet.com/products.php, last visit:18.10.2014.
[40] S.-Y. Wang, C.-L. Chou, and C.-M. Yang, "EstiNet openflow network simulator and emulator," IEEE Communications Magazine*, vol. 51, pp. 110-117, 2013.*
[41] F. Hu, "Network Innovation through OpenFlow and SDN: Principles and Design", CRC Press, (2014). http://dx.doi.org/10.1201/b16521.
[42] N. Handigol, B. Heller, V. Jeyakumar, D. Mazières, and N. McKeown, "Where Is The Debugger for My Software-Defined Network?", Proceedings of the first workshop on Hot topics in software defined networks, (2012), pp:55-60, available online: http://dx.doi.org/10.1145/2342441.2342453, last visit:17.10.2014.
[43] A. Khurshid, W. Zhou, M. Caesar, and P. Godfrey, "Veriflow: Verifying Network-wide Invariants in Real Time", ACM SIGCOMM Computer Communication Review, vol.42, (2012), pp:467-472, available online:http://dx.doi.org/10.1145/2342441.2342452, last visit:17.10.2014.
[44] E. Al-Shaer and S. Al-Haj, "FlowChecker: Configuration Analysis and Verification of Federated OpenFlow Infrastructures", Proceedings of the 3rd ACM workshop on Assurable and usable security configuration, (2010), pp:37-44, available online: http://dx.doi.org/10.1145/1866898.1866905, last visit:17.10.2014.
[45] M. Kobayashi, S. Seetharaman, G. Parulkar, G. Appenzeller, J. Little, J. van Reijendam, et al., "Maturing of OpenFlow and Software-Defined Networking Through Deployments", Computer Networks, (2013), pp:151–175, available online: http://dx.doi.org/10.1016/j.bjp.2013.10.011, last visit:17.10.2014.