DESIGN AND IMPLEMENTATION OF A DATA CENTRE FOR SMART GRIDS BASED ON SOFTWARE DEFINED NETWORK TECHNOLOGY (SDN)

Saeed Alishahi
* MEEDC - Iran
s.alishahi@meedc.net

Mohammad Hossein Yaghmaee
** FUM - Iran
yaghmaee@ieee.org

Ahmadreza Montazerolghaeem
** FUM - Iran
ahmadreza.20012@gmail.com

Mehdi Zarif
Islamic Azad University - Iran
Zarif.mahdi.315@gmail.com

* Mashhad Electric Energy Distribution Company ** Ferdowsi University of Mashhad

ABSTRACT

Transmission of data in intelligent smart grids (SG) is definitely dependent on the communication infrastructure of the network and how it communicates. The data transfer that connects to the AMI in the infrastructure brings together data from smart meters for analysis and management to a data centre. In general, the data centre is storing data. Note that, with the expansion of SG, fundamental mechanisms have lost their significance, thus posing difficulties, such as data centre overload. Therefore, there is a vital need for the softwarization of SG. To this end, Software-Defined Networking (SDN) is quite handy. With SDN and the aid of software-based controllers alongside predesigned APIs (e.g. OpenFlow protocol), an entire network and its elements may be controlled and programmed as a unified network. Software-centric data centres (SDDCs) are a new generation of information bays that work more on virtualization and software-centric technologies, and are less reliant on hardware or physical infrastructure layers. The main objective of this paper is to implement a software-centric data centre (SDN) for the smart grid. In this regard, the proposed architectures of data centres are based on the concepts of SDN and placed in a real test environment, implemented and evaluated under different factors and scenarios. The evaluation results are also discussed in depth.

SDN-BASED DATA CENTER ARCHITECTURE DESIGN:

The deployment and development of smart power grids are undoubtedly dependent on strong communication infrastructure, reliable data centers, and databases. Investments in the smart grid from 2010 to 2020 have been projected at least $ 100 billion, with a significant proportion being allocated to communications infrastructure and data centers. Using SDN can optimize the functionality of data centers. In this paper, we design and implement such a data center.

Three-layer architecture

Three-layer architecture based on SDN technology is presented in Figure 1. This architecture consists of three layers of switches that establish server communications. These switches use OpenFlow technology and are controlled by an SDN controller. Flows between data centers or servers are also managed by commands sent from the controller to the switches.

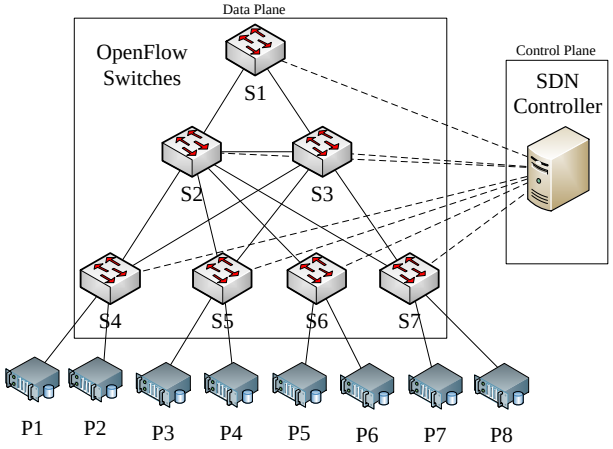


Figure 1- The proposed three-layer SDN-based data center architecture.

Fat-Tree architecture

The Fat-Tree architecture is presented in the following. The topology of this architecture is as a tree. This data center consists of 2 pods, each of which consists of 4 aggregation and edge switches. A total of 2 core switches establish communication between pods.

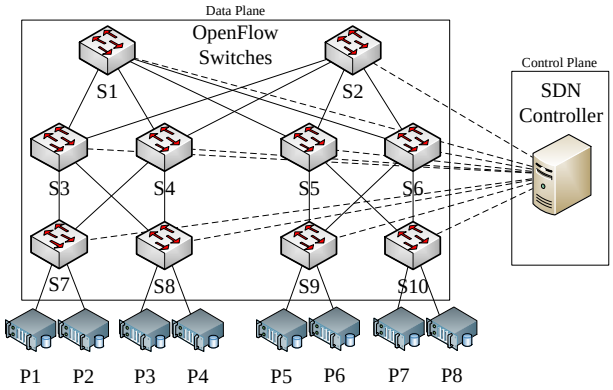


Figure 2- The proposed fat-tree SDN-based data center architecture.

BCube architecture

The proposed BCube architecture is presented in Figure 3. As you can see, there are two levels of communication switches between servers. Communications between all 4 adjacent servers are provided through low-level switches and communications between nonadjacent servers through high-level switches.

This architecture has two levels. At a low level, four 4-port switches establish communication between the 16

neighboring servers. At the top level, four 4-port switches set up communications between non-neighboring servers.

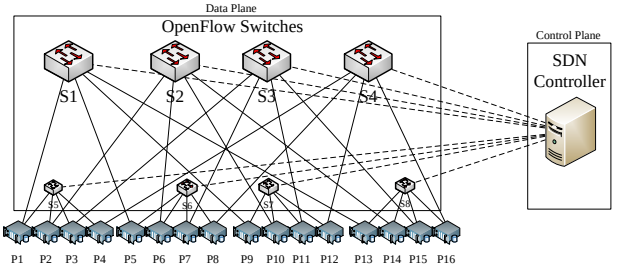


Figure 3- The proposed BCube SDN-based data center architecture.

Dcell architecture

The proposed Dcell architecture is presented in Figure 4. The framework of this architecture consists of several cells. Each cell contains a switch and four servers. Communications between different cell servers are provided through switches. The control of the flows and switches is also provided by the SDN controller.

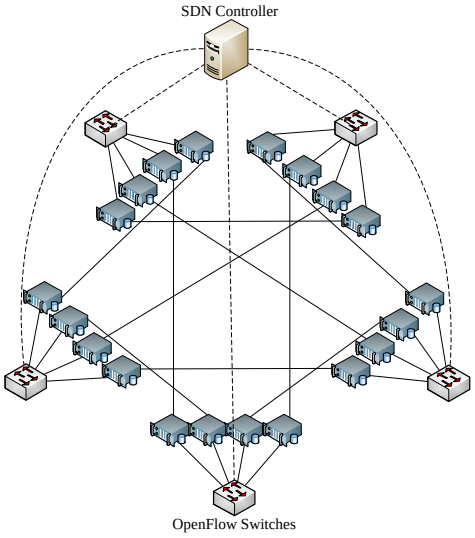


Figure 4- The proposed Dcell SDN-based data center architecture.

IMPLEMENTATION TEST BED:

In this section, we first explain the implementation details. Then, in a few succeeding sections, we evaluate performance and present the results. In the test bed, we use Open vSwitch v2.4.1 and v1.2 Floodlight to implement the OpenFlow switch and controller. Floodlight is a Java-based controller. Open vSwitch is also a software and virtual switch that supports OpenFlow protocol.

We used SIPp for traffic injections. Oprofile software was also used to monitor consumables resources. In case of the need to inject background traffic between servers, we used iperf to send packets with fixed rates. Network packet monitoring was also performed by Wireshark software.

We run each test three times and report the average as a result. The test bed is provided by the IPPBX Lab of Ferdowsi University of Mashhad for implementation of proposed architectures (Figure 5).

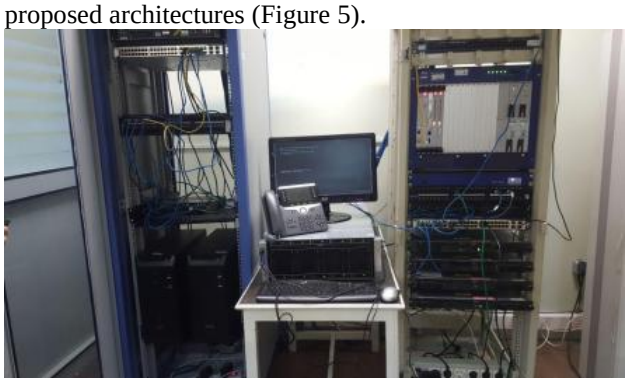


Figure 5- The overview of the test bed.

PERFORMANCE EVALUATION AND EXPERIMENTAL RESULTS:

The first experiment consists of two scenarios with different background traffic. In scenario 1, the background traffic of each server is 500 packets per second (pps). In Scenario 2, the background traffic for servers 1 to 4 (P1 to P4) is 1000 and servers 5 to 8 (P5 to P8) is 500 pps. Also, for a period of 400 seconds, a fixed rate of 1,500 requests per second is injected through the traffic generator.

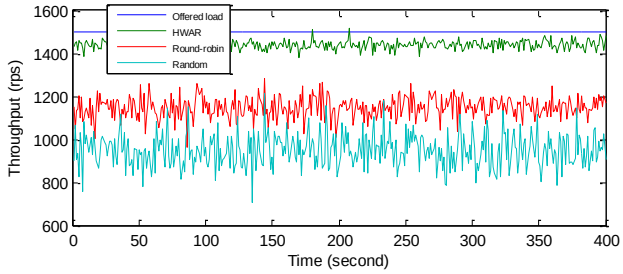
Figure 6 shows the performance of servers; the controller uses round-robin, Random, and HWAR [17] to select a server.

The HWAR method is based on the past response time of each server and estimates its available capacity and then loads it.

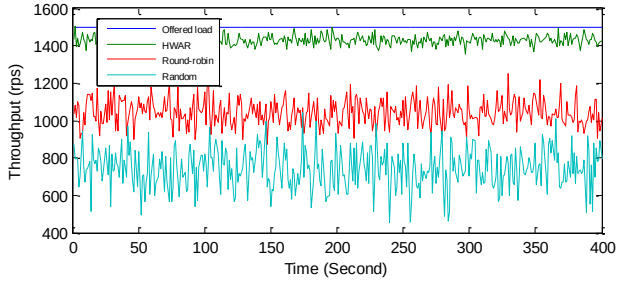
Server throughput (the number of service requests given by the servers per unit time), the average response time (the time between sending requests from the user agent to the server until receiving 200 OK from the server), and server resources are the evaluation criteria.

The goal is to achieve maximum throughput and the minimum response time without overload and with respect to resources.

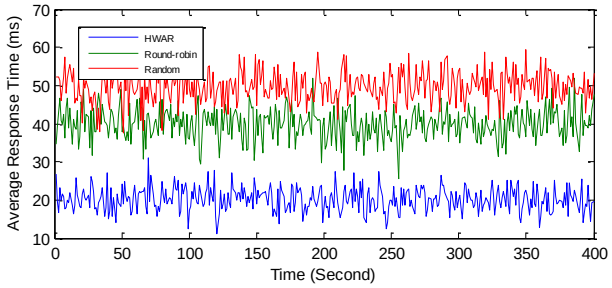
The HWAR method has achieved better results than Round-robin and Random methods in both scenarios. In addition, the results of HWAR are similar in both scenarios, but Raound-robin and Random results in Scenario 2 are worse than Scenario 1. This is because the different background traffic of servers in Scenario 2 has worsened the blind load distribution of these two methods over time. As can be seen from the comparison of Figure 6, the HWAR method, in both scenarios, has achieved throughput proximity to the offered load, since it can estimate the servers load by estimating the response time of the servers. In spite of more resource consumption in Raound-robin and Random methods than HWAR, the average response time of them is also higher. The resource utilization of both servers in the HWAR process is roughly equal, indicating a fairly distributed distribution of this method.



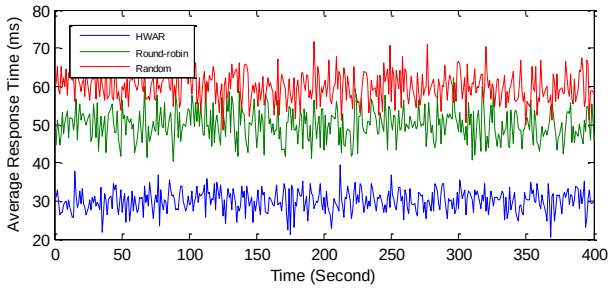
(a). Server throughputs in Scenario 1



(b). Server throughputs in Scenario 2



(c). Average response time in Scenario 1



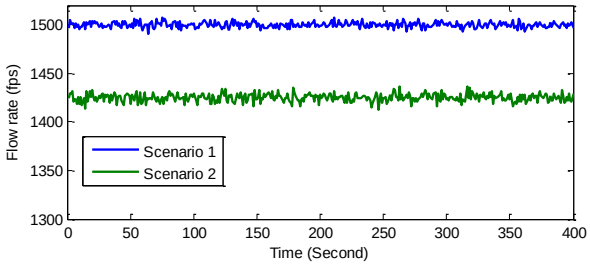
(d). Average response time in Scenario 2

Figure 6- Comparison of the performance of data center servers in two scenarios.

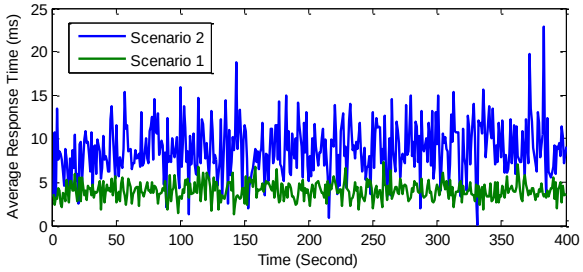
Figure 7 shows the controller efficiency. The controller throughput is the number of services flows given at the time unit. The average response time of the controller is the time between the transmissions of the Packet-In message from the switch until the Flow-mod message receives from the controller.

As shown in Figure 7 a and b, the controller performance is independent of the scenario and its average throughput is approximately 1450 fps and the average response time is approximately 7 ms. This indicates that the controller has been able to achieve a high throughput with very low delay.

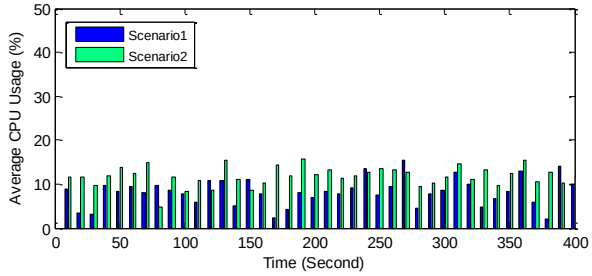
However, its modules do not add extra overhead. The controller consumes fewer resources than the servers. This is because the server is responsible for the establishment and termination of all requests, while the controller is responsible for managing a limited number of switches.



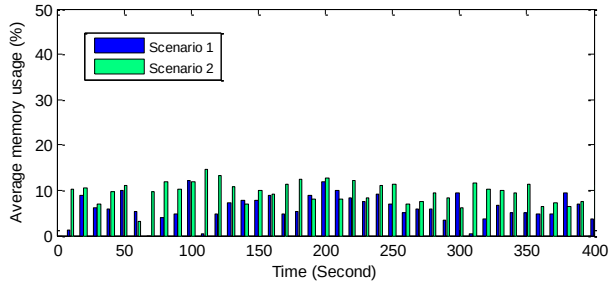
(a).Throughput of the controller



(b). Average response time of the controller



(c). Average CPU consumption of the controller



(d). Average memory consumption of the controller

Figure 7- Comparison of controller efficiency in two scenarios.

In the following, we perform another test. In the previous test, 1,500 requests per second (1,500 rps) were loaded into the system, but in this test we evaluate performance with variable load. In the previous test, servers were not overloaded because their capacity was more than 1500 rps (and thus did not encounter resource shortages). But in this test, with increasing load, the phenomenon of overload occurs.

The results are shown in Figure 8. The offered load starts from 1500 and increases in four stages to 6000 rps (up to 400 seconds). Then, with a sudden drop at the 400th second, it returns to 1500 rps. At the 500th second, a sudden jump increases to 6,000 rps, and in the last hundred seconds, it reaches 3,000 rps. Up to 200th second, servers and controllers throughput are very close to the offered load. At the 200th second servers are overloaded (continues up to 400th second). In these 200 seconds, the servers' average throughput is roughly 3000 rps, and the rest of the offered load is dropped.

Server overload occurs due to lack of resources, especially on the processor (Fig. 8 c and d). For example, Figure 8 c shows the processor saturation of servers in 250th, 350th and 550th seconds.

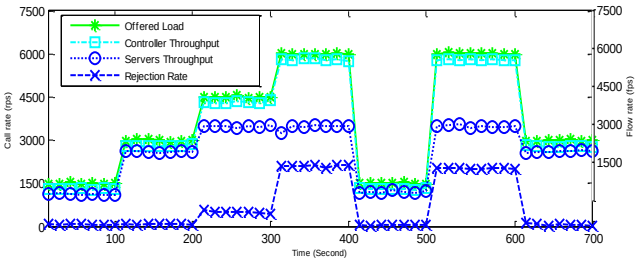
Usually, in a traditional architecture with a saturation of the server processor, throughput slows down to zero, but the SDN architecture has been able to prevent a severe drop in perceived overload and use the maximum capacity of servers.

Unlike servers, the controller is not overloaded at the peak time, and its throughput is always consistent with the offered load. For example, its average throughput during [300, 400]sec is approximately 5978 fps.

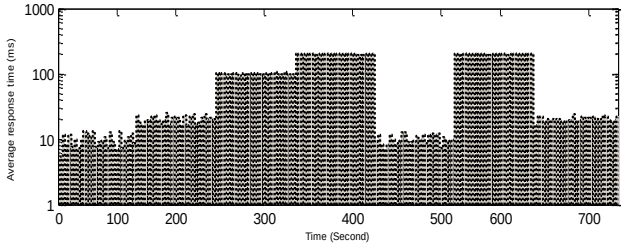
Proportionally, the server response time also varies (Fig. 8 b). With overload, response time is also multiplied. After returning the offered load to less than the capacity of the servers (returning to an under load state), the overload control algorithm is not capable of returning the server back to normal, but with the end of overload state (400th second), SDN-based architecture with the appropriate distribution of load once again has been able to bring the system to a offered load.

Contrary to the gradual increase in load in the first 400 seconds, an instantaneous swing occurs in 500th second. However, despite the complete saturation of the server processor (Figure 8 c), the SDN-based architecture has also been able to maximize servers throughput. Immediate instantaneous swing occurs when a large number of requests are given simultaneously. The high throughput in the sudden downsizing of the load represents the stability of the system.

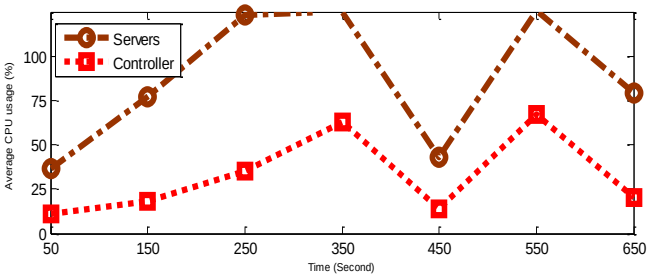
Finally, in the last 100 seconds, the rejection rate is negligible and the throughput is maximized and response time is approximately 10 ms.



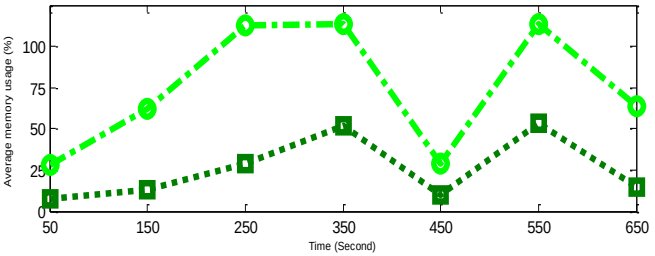
(a). Throughput



(b). Average response time



(c). Average CPU usage



(d). Average memory usage

Figure 8- Servers efficiency over time and with different offered loads.

In the remainder of this section, other test results are presented. In this new experiment, we compare the performance of SDN-based architecture with traditional architecture. The following table provides a comparison between latency and resource consumption between the traditional three-layer architecture and SDN-based three-layer architecture. As you can see, SDN technology has been able to greatly improve the quality of service, including throughput, latency and resource consumption.

Table 1- Comparison of traditional three-layer architecture with SDN.

Methods	1500 rps		2000 rps		2500 rps	
	Traditional	SDN	Traditional	SDN	Traditional	SDN
Throughput (rps)	956	1432	1498	1856	1998	2456
Delay (ms)	28	12	57	13	98	14
Average CPU Usage (%)	25	16	34	18	44	23
Average Memory Usage (%)	27	17	37	20	55	24

CONCLUSION

In this paper, we have proposed four SDN-based architectures for three-layer, Fat-Tree, BCube, and Dcell data centers, and implemented each one in a real test bed. In this regard, we use Open vSwitch v2.4.1 and Floodlight v1.2 to implement the OpenFlow switch and controller. In the first test, the traffic is fixed. In the second test, the traffic is variable, and the third experiment is to compare the traditional architecture with SDN-based architecture. Observations and experiments show that all four SDN-based architectures have been able to distribute loads between data center servers well with the use of algorithms such as HWAR or Round-Robin and data collected by the controller. Also, they achieve higher performance than traditional architecture and have significant improvements in parameters such as throughput, response time and resource consumption.

REFERENCES

[1] White paper, Software-Defined Networking: The New Norm for Networks, Open Networking Foundation, April 13, 2012. Retrieved August 22, 2013.

[2] David Strom, 100Gbps and beyond: What lies ahead in the world of networking, www.arstechnica.com, Feb 19 2013.

[3] D. Tennenhouse, J. Smith, W. Sincoskie, D. Wetherall, and G. Minden. A survey of active network research. Communications Magazine, IEEE, 35(1):80-86, 1997.

[4] D. Tennenhouse and D. Wetherall. Towards an active network architecture. In DARPA Active Networks Conference and Exposition, 2002. Proceedings, pages 2-15. IEEE, 2002.

[5] A. Campbell, I. Katzela, K. Miki, and J. Vicente. Open signaling for atm, internet and mobile networks (opensig'98). ACM SIGCOMM Computer Communication Review, 29(1):97-108, 1999.

[6] R. Enns. NETCONF Configuration Protocol. RFC 4741 (Proposed Standard), Dec. 2006. Obsoleted by RFC 6241.

[7] Scott Shenker, Gentle Introducton to Softwaer- Defined Networking, Technion lecture, The 2nd Annual International TCE Conference Networking, Cloud and Beyond, June 6-7 2012

[8] M. Casado, M. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker. Ethane: Taking control of the enterprise. ACM SIGCOMM Computer Communication Review, 37(4):1-12, 2007.

[9] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. Nox: towards an operating system for networks. ACM SIGCOMM Computer Communication Review, 38(3):105-110, 2008. The 8th Symposium on Advances in Science and Technology (8thSASTech), Mashhad, Iran. 8thSASTech.khi.ac.ir

[10] Brent Salisbury, The Northbound API- A Big Little Problem, www.networkstatic.net, June 12 2012

[11] G. Lu, R. Miao, Y. Xiong, and C. Guo. Using cpu as a traffic co-processing unit in commodity switches. In Proceedings of the first workshop on Hot topics in software defined networks, HotSDN '12, pages 31-36, New York, NY, USA, 2012. ACM.

[12] J. C. Mogul and P. Congdon. Hey, you darned counters!: get off my asic! In Proceedings of the first workshop on Hot topics in software defined networks, HotSDN '12, pages 25-30, New York, NY, USA, 2012. ACM.

[13] C. Tselios, C. Papageorgiou, K. Birkos, I. Politis, T. Dagiuklas, Real-Time Communications in Autonomic Networks: System Implementation and Performance Evaluation, Journal of Computer Networks and Communications, Volume 2012 (2012), 10 pages, Article ID 485875.

[14] John Markoff, Open Networking Foundation Pursues New Standards, The New York Times, www.nytimes.com/2011/03/22/technology, March 22, 2011

[15] Katherine Noyes, Google and other titans form Open Networking Foundation, www.computerworld.com.au, 23 March 2011

[16] OpenFlow Tutorial: Next-Gen Networking Has Much To Prove, www.techwe.com

[17] Montazerolghaem, Ahmadreza, et al. "A load scheduler for SIP proxy servers: design, implementation and evaluation of a history weighted window approach." International Journal of Communication Systems 30.3 (2017): e2980.