International Journal of Internet, Broadcasting and Communication Vol.12 No.3 9-17 (2020)
http://dx.doi.org/10.7236/IJIBC.2020.12.3.9

IJIBC 20-3-2

Simulator for Dynamic 2/3-Dimensional Switching of Computing Resources

Jang-Geun Ki*, Kee-Young Kwon*

**Professor, Div. of Electrical, Electronic and Control Engineering, Kongju Nat'l Univ., Korea*
E-mail: {kjg, kky}@kongju.ac.kr

Abstract

In this paper, as part of the research for the infrastructure of very high flexible and reconfigurable data center using very high speed crossbar switches, we developed a simulator that can model two and three dimensional connection structure of switches with an efficient control algorithm using software defined network and verified the functions and analyzed the performance accordingly. The simulator consists of a control module and a switch module that was coded using Python language based on the Mininet and Ryu Openflow frameworks. The control module dynamically controls the operation of switching cells using a shortest multipath algorithm to calculate efficient paths adaptively between configurable computing resources. Performance analysis by using the simulator shows that the three-dimensional switch architecture can accommodate more hosts per port and has about 1.5 times more successful 1:n connections per port with the same number of switches than the two-dimensional architecture. Also simulation results show that connection length in a 3-dimensional way is shorter than that of 2-dimensional way and the unused switch ratio in a 3-dimensional case is lower than that of 2-dimensional cases.

Keywords: *Switch; Modelling; Simulator; Resource Dynamic Connection*

1. Introduction

Unlike in the past when computer devices and information communication equipment were concentrated and managed in a certain space to efficiently provide diverse information and communication services, recent developments have been actively studied to dynamically allocate and link diverse computing resources that are distributed across distant regions by ultra-high speed dynamic networking.

In particular, looking at recent trends, many companies do not directly build their own computer and network resources on their premises. Instead, they are increasingly using the services of the IDC(Internet Data Center) to borrow computing and networking resources needed for various types of businesses over the Internet by on demand basis[1][2]. As a result, with the increasing data demand and the workload according to the development of new information communication technologies, also the data center traffic is dramatically increasing[3]. However, traditional data centers now operating all over the world face challenges such as large-scale geographical site construction, data management costs, and power consumption. Thus, cloud data centers

Manuscript Received: May. 3, 2020 / Revised: May. 10, 2020 / Accepted: May. 14, 2020
Corresponding Author: kjg@kongju.ac.kr
Tel: +82-41-521-9145, Fax: +82-41-522-9145
Professor, Div. of Electrical, Electronic and Control Engineering, Kongju Nat'l Univ., Korea

have recently come to the fore as a way to address this, to reduce power consumption and total costs in traditional data centers. According to the white paper by Cisco[4], the share of cloud workloads in the data center is expected to grow rapidly in the future to reach 94% by 2021.

Kumbhare[5] suggested a crossbar switching cell architecture to connect components in their JITA (Just in Time Architecture) system and Ki[6] described results of basic performance analysis for the architecture. In this paper, as part of the research for the infrastructure of the reconfigurable data center using very high speed crossbar switches, we developed a simulator that can model a 2-dimensional and 3-dimensional connection structure of switches with an efficient control algorithm using SDN(Software Defined Network) and analyze the performance accordingly.

The rest of this paper is organized as follows. Following the introduction in Section 1, Section 2 describes the switching structure and the resulting modelling, and Section 3 shows and analyses some simulation results. Section 4 concludes and discusses future research directions.

2. Switching Structure and Modelling

In order to meet requirements of future data centers, very high flexibly configurable switching architecture is needed. Components of the data center must be connected to an ultra-fast switching network such as a terabit network with very low latency. Kumbhare[5] suggested the optical interconnection switch structure in the JITA architecture in which each switching cell consists of two perpendicular passive ridge waveguides to guide input and output waves. Each cross-point has AVCs (Active Vertical Couplers) that are active waveguide couplers placed on top of input and output passive waveguides. Figure 1 shows Kumbhare's switch architecture.

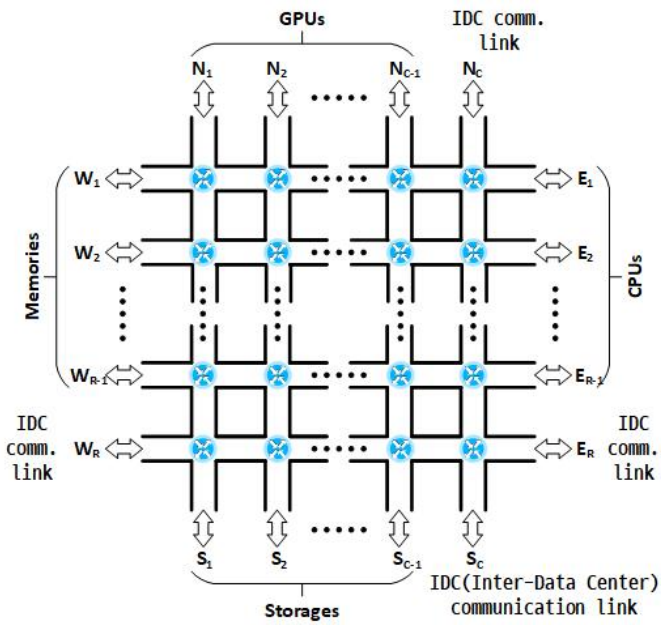


Figure 1. Kumbhare's switch structure[5]

In order to evaluate various switching architectures including the JITA system, this paper describes a simulator that is able to model and dynamically control various two or three dimensional connection states of each switching cell. The simulator developed in this paper is based on Mininet and Ryu controller. Mininet is

a kind of network emulator that provides virtual hosts, switches and controllers and links them together[7][8]. In order to develop the Mininet switch controllers, the Ryu Openflow[9] controller was used and programmed by using Python language[10]. The Ryu Openflow controller provides a component-based SDN(Software Defined Network) framework and supports many protocols and well defined APIs to manage network devices[9]. The simulator can model any 2D/3D states of the switch as well as all states of the Kumbhare's JITA switching cell.

The simulator consists of two modules, switch module and controller module. Switch module is based on the Mininet and programed by Python language. Switch module takes switch dimensioning parameters, row, column and depth, and then create switches and hosts and links to connect them in two or three dimension. Unique IP and MAC address are automatically allocated to each host and MAC address to each switch. Host can be considered as necessary resources such as CPU, memory or storage) to configure the data center. Switch module is set to use remote controller.

Pseudo code for the remote controller module is shown in Figure 2. Controller module in the simulator first creates graph object g and then adds host nodes and switch nodes in two or three-dimensional directions according to the simulation input parameters. Next, it adds links between switches and also links between switches and host nodes.

```
// create graph object g
g = Graph(row, col, dep);        // dep=0 in 2D case
// add hosts in the graph g in a 2 or 3-dimensional way
g.add_vertex(hostname);
// add row*col*dep switches in the graph g
g.add_vertex(swname);
// add links in the graph to connect hosts and switches
g.add_edge(swname1, swname2, 1);
g.add_edge(hostname, swname2, 1);
// calculate and shuffle the list of hosts
host_list = g.get_host_list();
random.shuffle(host_list);
// randomly select one master host and n slave hosts to connect to each
// other to perform simulations of an 1:n connection structure
loop
{       // randomly select a master host
        host_from = host_list.get_next_host(1);
        // select n slave hosts to be connected to the master host
        host_to_list = host_list.get_next_host(n);
        // calculate the shortest path from the master to each slave hosts
        from_to_path = g.search_shortest_path(host_from, host_to_list);
        if (len(from_to_path) > 0)       // if found
        {// update graph connection information, switch states, and statistics
                g.update_path(from_to_path);
                g.update_switch_status(from_to_path);
                g.update_statistics();
        }
}
```

Figure 2. Pseudo code for the controller module

The next step is to randomly select one master host and n slave hosts to connect to each other to perform simulations of an 1:n connection structure. For each combination of a master and slave hosts, the Dijkstra's shortest-path algorithm[11] is modified and used to establish an optimal path between 1:n nodes and the statistics are updated as a result. All switches on the path from the master host to the slaves maintain the list of output ports connected to each input port for routing, updating the output ports of that input port whenever

a new route is established. The established paths are verified by Ping operation during the simulation.

Figure 3 shows switch architectures which can be modelled in the simulator. Logical states of a switching cell are shown in Figure 4. Connection states can be two-dimensional or three-dimensional according to the simulated architecture. Each input port in a switching cell can support 1:1 or 1:n connection to output ports with or without intentional constraints to transmit the incoming data.

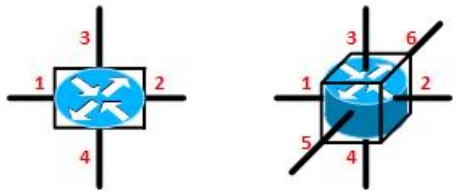


Figure 3. Two/three-dimensional switch architectures

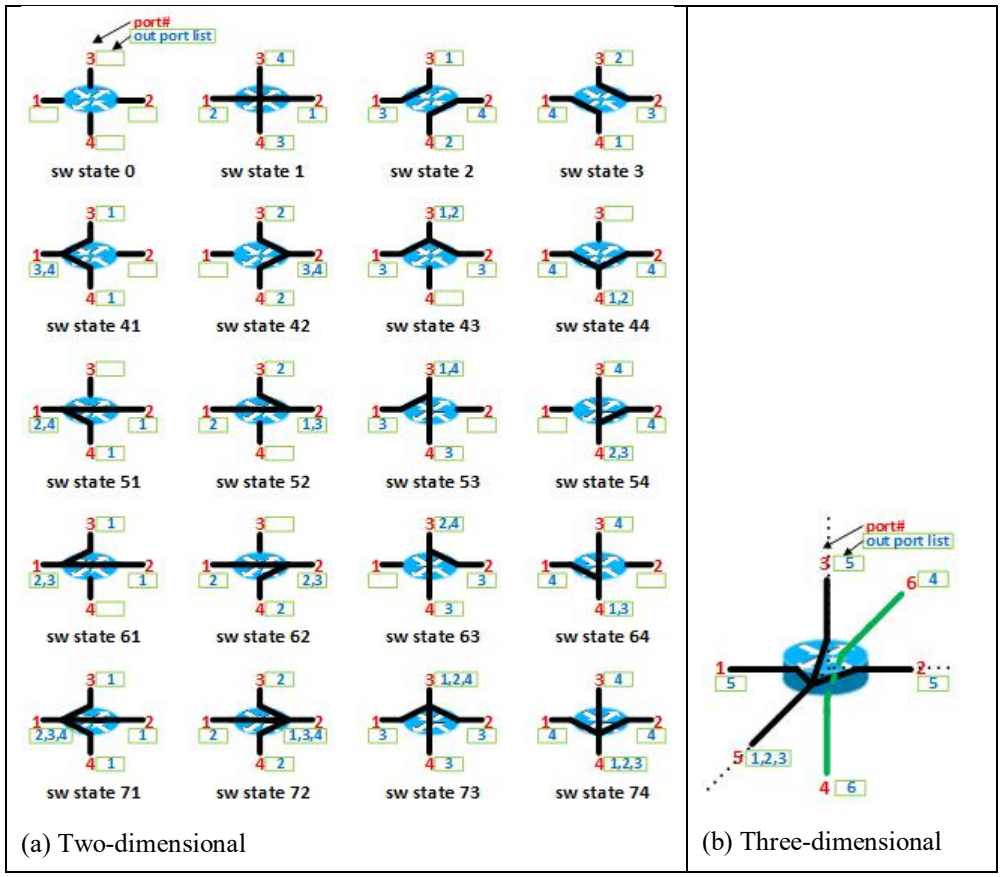


Figure 4. Examples of logical switching cell states

A two-dimensional switch has 4 ports and supports connections from 1:1 to 1:3, while a three-dimensional switch has 6 ports and supports connections from 1:1 to 1:5. Each port in a switch will dynamically provide proper 1:n connection function for routing according to real-time control of the central control center. Figure 4(a) shows some examples of two-dimensional switch states and black line in the figure is the path to connect each port number. Figure 4(b) is an example of three-dimensional switch states. In the figure, port 5 is connected to port 1, 2, 3 and port 4 to 6.

For a two-dimensional connection structure consisting of x*y switches, the number of acceptable hosts and the total number of links can be calculated as follows.

$H_2 = (y+x)*2$
$L_2 = (y-1)*x + y*(x-1) + (y+x)*2$

For a three-dimensional connection structure consisting of x*y*z switches, the number of acceptable hosts and the total number of links can be calculated as follows.

$H_3 = (x*y + y*z + z*x)*2$
$L_3 = \{(y-1)*x+y*(x-1)\}*z+x*y*(z-1)+(x*y+y*z+z*x)*2$

Figure 5 shows a comparison of the number of acceptable hosts with the number of switches used in the connectivity structure. As shown in the figure, connecting the switches into a three-dimensional structure can accommodate much more hosts compared to connecting in a two-dimensional structure.

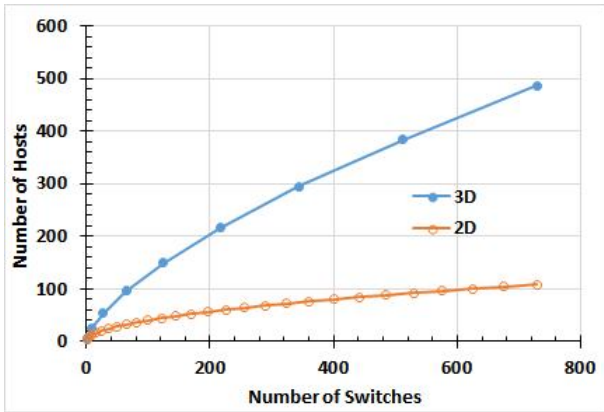


Figure 5. Number of affordable hosts in 3D/2D architecture

3. Verification and Performance Evaluation on Switching Architectures

In order to verify the function and evaluate the performance of the developed simulator, various simulations are conducted. Output results of the simulator includes 1:n successful connection probability, average path length, switch utilization, path trace and graphical representation, path verification, etc.

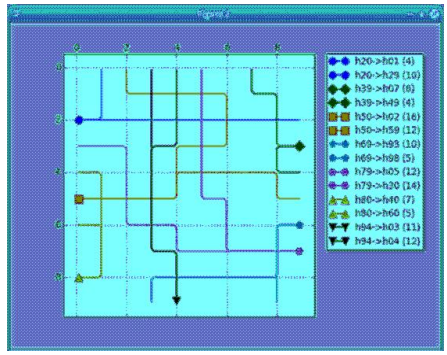
For verification of the simulator, an example of simulation results for two and three dimensional switch connection architecture are shown in Figure 6 and Figure 7.

Probability of successful 1:n connection in a 2-dimensional X*Y switch architecture and a 3-dimensional X*Y*Z switch architecture are defined as follows.

$$P_{2d} = \frac{number\ of\ successful\ 1{:}n\ connection}{number\ of\ possible\ 1{::}n\ connection} = \frac{number\ of\ successful\ 1{:}n\ connection}{floor\left[\frac{2(X+Y)}{n+1}\right]}$$

$$P_{3d} = \frac{number\ of\ successful\ 1{:}n\ connection}{number\ of\ possible\ 1{::}n\ connection} = \frac{number\ of\ successful\ 1{:}n\ connection}{floor\left[\frac{2(XY+YZ+ZX)}{n+1}\right]}$$

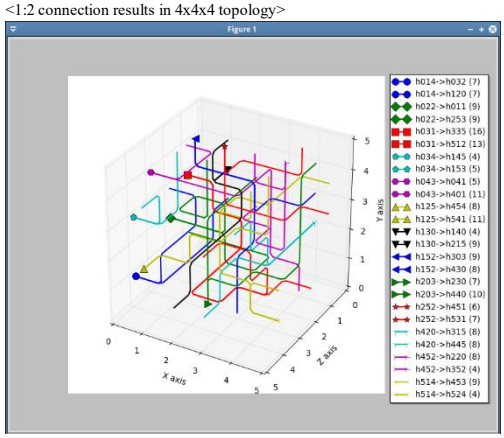
<1:2 connection results in 8x8 topology>



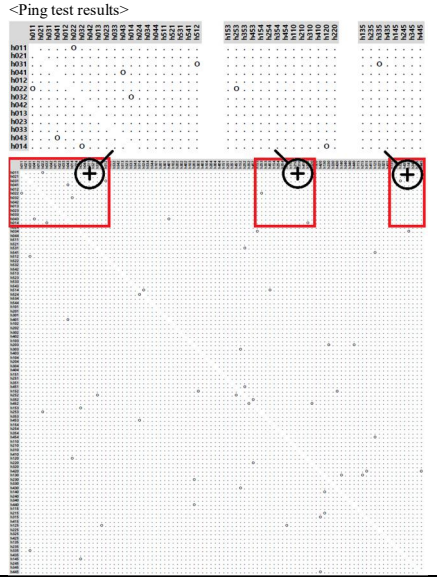

```
==========>>>>>> len(host_list)= 32 tot= 10 success= 7 fail= 3 prob= 0.7
==========>>>>>> num_of_fromtopath= 14 avg of sum_len_fromtopath= 9.14285714286
==========>>>>>> num_of_sw= 64 num_of_unused_sw= 6 unused_sw_ratio= 0.09375
######## sw_size_row, sw_size_col, one2n, T_way_only, numofrun= 8 8 2 False 1
######## final_avg_prob= 0.7
######## final_num_of_fromtopath= 14.0
######## final_sum_len_fromtopath= 9.14285714286
######## final_num_of_unused_sw= 6.0
######## final_unused_sw_ratio= 0.09375
(KJGdebug01)==========>>>>>> sim_probability= 0.7
```

Figure 6. Simulation result for two dimensional 8x8 switch architecture

<1:2 connection results in 4x4x4 topology>



<Ping test results>



```
==========>>>>>> len(host_list)= 96 tot= 32 success= 13 fail= 19 prob= 0.40625
==========>>>>>> num_of_fromtopath= 26 avg of sum_len_fromtopath= 7.92307692308
######## sw_size_row, sw_size_col, sw_size_dep, one2n, T_way_only, numofrun= 4 4 4 2 False 1
######## final_avg_prob= 0.40625
######## final_num_of_fromtopath= 26.0
######## final_sum_len_fromtopath= 7.92307692308
######## final_num_of_unused_sw= 1.0
######## final_unused_sw_ratio= 0.015625
(KJGdebug01)==========>>>>>> sim_probability= 0.40625
```

Figure 7. Simulation result for three dimensional 4x4x4 switch architecture

As an example of performance comparison between three-dimensional and two-dimensional switch structures, Figure 8 shows the number of successful 1:n connections per port. In a 3-dimensional structure, a total of 4x4x4=64 switches are connected in a three-directional way, x, y and z. In two 2-dimensional

structures, one(2D 8x8) has the same number of 8x8=64 switches and the other(2D 24x24) has the same number of 24x4=96 hosts as in the 3-dimensional case. Table 1 shows the total number of switches, hosts and ports in each structure. For fair comparison, number of successful 1:n connection per each port is shown in Figure 8. The three-dimensional case in the figure has about 1.5 times more successful 1:n connections per port with the same number of switches than the two-dimensional architecture.

Table 1. Number of switches, hosts, and ports in each structure

Structure	Number of Switches	Number of Hosts	Number of Ports
2D 8*8	64	32	256
3D 4*4*4	64	96	384
2D 24*24	576	96	2304

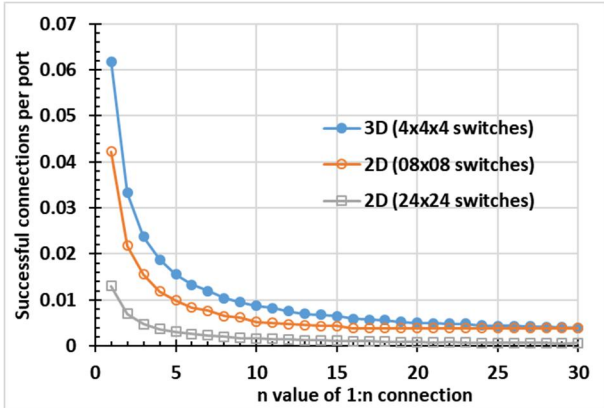


Figure 8. Number of successful connections per port

Figure 9 shows average path length from server node to client node in a 1:n connection. Path length is the total number of switch nodes that go through from server to client node including themselves. We can see that the 3-dimensional structure has shorter path length than 2-dimensional structures.

Unused switch ratio is shown in Figure 10. An unused switch is the switch not included in any path of successful 1:n connections during the simulation. From Figure 10, we can see that 3-dimensional structure has lower unused switch ratio and therefore uses switches more efficiently than 2-dimensional cases.

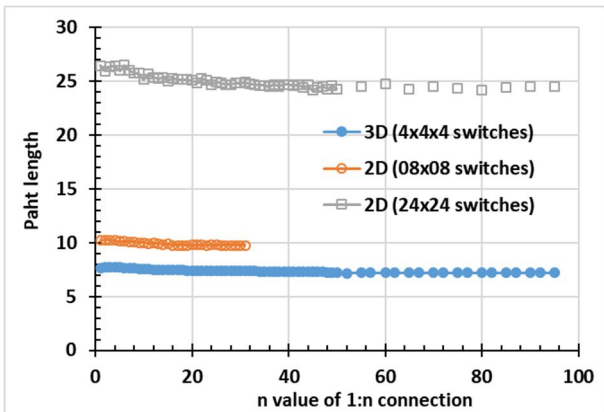


Figure 9. Average path length from server to client

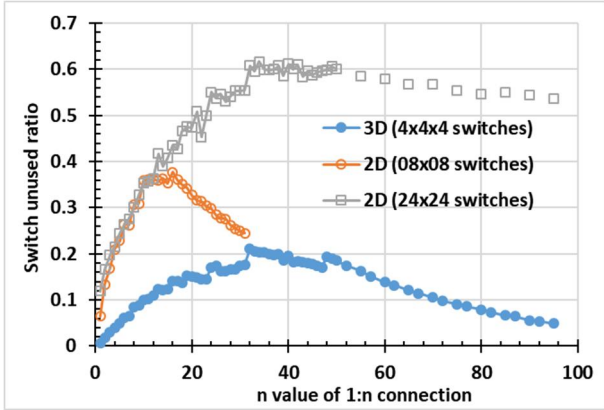


Figure 10. Unused switch ratio

4. Conclusion

In this paper, we have developed a SDN-based simulator that models two or three dimensional connection architecture and connects various computer and communication resources with a dynamic switch control algorithm. Operation of the simulator has verified by conducting performance analysis in various simulation environments. The simulator consists of a control module and a switch module that was coded using Python language based on the Mininet and Ryu Openflow frameworks. The control module dynamically controls the operation of switching cells using a shortest multipath algorithm to calculate efficient paths adaptively between configurable computing resources.

Performance analysis by using the simulator shows that the three-dimensional switch architecture can accommodate more hosts per port and has about 1.5 times more successful 1:n connections per port with the same number of switches than the two-dimensional architecture. Also simulation results show that connection length in a 3-dimensional way is shorter than that of 2-dimensional way and unused switch ratio in a 3-dimensional case is lower than that of 2- dimensional cases.

In the future work, we will introduce more diverse simulation parameters in the simulator and develop efficient path calculation algorithms accordingly.

References

[1] Adrian O'Connell, Forecast: Data Centers, Worldwide, 2015-2022, 2018 Update, Gartner, May 04, 2018. https://www.gartner.com/doc/3873992/forecast-data-centers-worldwide-

[2] Mohammad Alizadeh, Shuang Yang, Milad Sharif, Sachin Katti, Nick McKeown, Balaji Prabhakar, and Scott Shenker, "pFabric: Minimal Near-Optimal Datacenter Transport," ACM SIGCOMM Computer Communication Review, Aug. 2013. DOI: https://doi.org/10.1145/2534169.2486031

[3] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Holzle, Stephen Stuart, and Amin Vahdat, "Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network," ACM Special Interest Group on Data Communication (SIGCOMM) Review, Vol. 45, Issue 4, pp.183-197, Aug. 2015. DOI: https://doi.org/10.1145/2785956.2787508

[4] Cisco, "Cisco Annual Internet Report (2018-2023) White Paper," Cisco white paper, Mar. 9, 2020. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[5] Nirmal Kumbhare, Cihan Tunc, Salim Hariri, Ivan Djordjevic, Ali Akoglu, and Howard Jay Siegel, "Just In Time Architecture (JITA) for Dynamically Composable Data Centers," IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA), Nov.29-Dec.2, 2016.
DOI: https://doi.org/10.1109/AICCSA.2016.7945778

[6] Jang-Geun Ki and Kee-Young Kwon, "Development of Simulation Software for Switch Connection," Journal of Software Assessment and Valuation, Vol.14, No.1, pp.41-46, June, 2018. https://www.kci.go.kr/kciportal/ci/sereArticleSearch/ciSereArtiView.kci?sereArticleSearchBean.artiId=ART002395873

[7] Mininet Team, Mininet, http://mininet.org/

[8] Lindinkosi L. Zulu, Kingsley A. Ogudo, and Patrice O. Umenne, "Simulating Software Defined Networking Using Mininet to Optimize Host Communication in a Realistic Programmable Network," International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD 2018), Aug. 2018.
DOI: https://doi.org/10.1109/ICABCD.2018.8465433

[9] Ryu SDN Framework Community, Component-based software defined networking framework Build SDN Agilely, https://ryu-sdn.org/

[10] Python Software Foundation, Python, https://www.python.org/

[11] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, Introduction to Algorithms (2nd ed.), MIT Press and McGraw-Hill, pp.595-601, ISBN: 0-262-03293-7.