Journal of Innovative Engineering and Research (JIER)
Vol.- 7, Issue -1, April 2024, pp. 1-7 (7 pages)

A Review on Evaluation and Detection of requirement for Controller Placement in Software Define Data Center Networks

RITESH JAIN, DR. PRADNYA ASHISH VIKHAR, DR. AJAY R. RAUNDALE
Department of Computer Science & Engineering
, Dr. A. P. J. Abdul Kalam University, Indore 452010, India
Correspond Author Email: rit.rit1@gmail.com

Abstract— *Existing organizations are using very large networks and many different types of equipments are used. When a huge number of systems are added it becomes difficult to maintain the network infrastructure. We cannot store all of information on the local systems. Data centers contain many servers, cooling systems and many other devices, which makes it very tough task to maintain the data centers. Now cloud computing applications are increasing day by day. These applications require big data centers for processing and storage of their data. Data Center Network uses are high bandwidth, low latency and limited size buffer. Sizes of data centers may vary depending upon the need of the organizations. For data centers, traffic characteristics are very different from normal internet traffic characteristics. We can also categorize the data centers according to their usage such as university data centers which are useful for students and staff, private enterprises for corporate users, developers and small number of customers and commercial cloud data centers for external users and supports for many internet-facing services like webmail, search, and video, instant messaging, gaming. In this review paper we elaborate on Software Define Networking. We also discuss its working, process, available software in detail. We also try to find out some problems by explaining some literature in this field.*

Index terms: *OpenFlow, SDN, Designing, Control Functionalities, Control message.*

I. INTRODUCTION

Software defined networking contain three main part control plane, data plane and control channel. The main characteristics of DCN are high bandwidth, low latency and limited size buffer [1]. A tiered architecture is followed by the data centers and this organizes network devices into two or three layers [2]. Control plane is the functions in the network that controls behavior of the network like a brain (For example, network paths, forwarding behavior). Personal data like photos and videos are also being stored from personal devices to data centers, because of the ease of sharing, manageability and ease of access [3]. Typically, the control plane is instantiated as a single, high-level software controller [7]. It can be run separately and also computes logic of how traffic will be forwarded and their policies.

Data plane is the functions in the network that are responsible for forwarding (or not forwarding) the traffic. Data plane is instantiated as forwarding tables in routers, switches, firewalls and middle boxes. It is a typically programmable hardware which is controlled by the control plane. Control channel is the communication channel is an SDN controller over which communicates with the under lying network switches.

1.1 Tools, Platform of Software Defined Networking

For implementation, we need to understand some tools to implement controller and for simulating the network. For this purpose, we can use the floodlight controller and Mininet.

Floodlight is an open-source Java controller and supports OpenFlow protocol. It is maintained by Big Switch Organization. The Floodlight is written in Java and runs in a JVM. When we run Floodlight, operations of both the northbound and south bound APIs from the controller becomes active. The northbound REST APIs exposed by all the running modules becomes available via the specified REST port. Any application can interact (retrieve information and invoke services) with the controller by sending an http REST command. On the other hand, at the outbound, the provider module of Floodlight will start listening on the OpenFlow-specified TCP-port for connections from the OpenFlow switches. Floodlight currently supports OpenFlow 1.0. Versions of 1.3 and 1.4 are in the pipeline. With an extensible Java development environment, and enterprise-grade core engine, Floodlight is both an easy to use and robust SDN controller.

Mininet is a network emulation platform that has the ability to create a virtual Open Flow network, controllers, switches, hosts and links on a single real or virtual network. Mininet executes python under the hood. It is a launch scripts that executes specific python code. It is a virtual network environment that can run on a single computer that is particularly useful for experimentation and learning. Mininet runs real kernel, switch, it can run real software and it can run real application code on a single machine. It provides command line, graphical user interface, and python interface to interact with it. Many OpenFlow features are already built in so it is useful in developing, deploying and sharing various things which we do in Mininet programming environment.

There are three tiers in SDN architecture, the top tier contains the applications and higher level instructions, the middle tier contain the controller, which directs data traffic, the third and the last tier which resides at the bottom contains physical and virtual switches (As shown in figure 1). In software defined networking the responsibility of the northbound API interface on the controller is to enable applications and the overall management system to program the network and request services from it. This layer provides basic network functions such as data path computation, routing and security.

OpenFlow protocol also works as the southbound API which defines a set of open commands required to forward the data. With the help of these commands routers finds the network topology and decides the behavior of physical and virtual switches, and it all depends on the requests of applications from the north bound APIs [16]. A centralized controller performs the control operations. Every time when a packet from a new flow comes to a switch, it contacts to its controller for flow rules. Software Defined Networking gives hope to change the current network infrastructure limitations [19]. It decouples the data plane and control plane, i.e., the network devices that forward traffic forms the data plane and the software logic that controls how traffic should forwarded from the network is known as control plane [5].

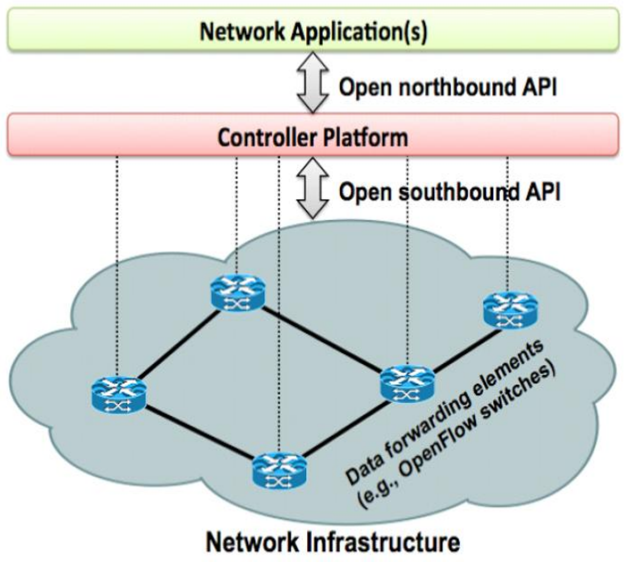

Figure 1: Simplified View of SDN Architecture

SDN is well suited for Data Centers, as SDN allows managing the traffic according to need. But data center has huge traffic and lots of servers so it becomes very tough to manage the whole network with single controller as single controller have the problem of single point of failure and the hardware limitations increases the flow set up time thus it degrades the performance. To solve this issue multi-controller approach is used where multiple controllers are used to manage the traffic and to provide the required services to the data centers. Multiple controllers can solve the problem of single point of failure and can increase the performance.

With the use of multiple controllers, we face some problems; in our work we are considering two main problems of multiple controller approach:

Cascading failure - In multi-controller approach if one controller fails, then its work and switches under this controller are assigned to another controller, but there is no mechanism for assigning these switches and when we assign switches to any random controller it is possible that these switches can increase the load on that controller beyond its capacity and causes the failure of that controller and this process can go on causing the failure of all other controllers. This situation can collapse the network; this kind of failure is known as cascading failure.

Dynamic assignment of switches - As the traffic changes rapidly in data center networks so the load on the controller also changes, thus it is required to dynamically distribute the load equally among all the controllers and to decide how many controllers can remain active at any given time so that we can achieve maximum utilization of the available resources and reduce the power consumption. In this work the possible approaches are proposed by which we can manage and maximize possible utilization of available resources of a multi-controller environment. Architecture is needed to be proposed to prevent the cascading failure of controllers in multi-controller environment, according to which we will assign a switch to any controller only when it is not exceeding the capacity of the controller.

II. LITERATURE SURVEY

In early days networked computer systems have used basic file sharing, peripheral sharing or hosting of company wide applications on a server. Organizations are now using highly advanced computing environments to meet their needs which include cloud based networks, desktop, virtual servers, remote data storage devices and technologies which requires more computing devices, labor and good planning to properly deploy and maintain [21]. With the large number of devices and with increasing requirements Computer networks becomes complex to manage; there are many kinds of equipments involved in computer networks like routers and switches, firewalls, network address translators [16]. Complex control functions and forwarding functions are performed switches and routers. When a large number of end systems added, it becomes difficult to adjust the network infrastructure. The solution to this is software defined networking (SDN).

Software defined networking is a new standard architecture which uses standardized application programming interfaces (APIs) by the help of which network programmers define and reconfigure the way data or resources can be handled in a network [18]. SDN separates the control plane from network devices. A centralized controller performs the control operations. Every time when a switch receives a new packet, it contacts to the controller. The Controller decides the rules to handle the packets and it gives instructions to the switches. And packets are forwarded by the switches based on the controller’s instructions. Current networks reduce the opportunities of innovation and evolution of the networking infrastructure.

Software defined networking gives hope to change the current network infrastructure's limitations. It separates the control plane and data plane and converts the network switches to simple forwarding elements and a logically centralized controller implements the control logic. Figure 2 is showing a simplified view of SDN architecture. There are three tiers in this architecture which are showing the different function of SDN. It is showing that Northbound API connects application programs to control plane and Southbound API connects controller to forwarding Switches.

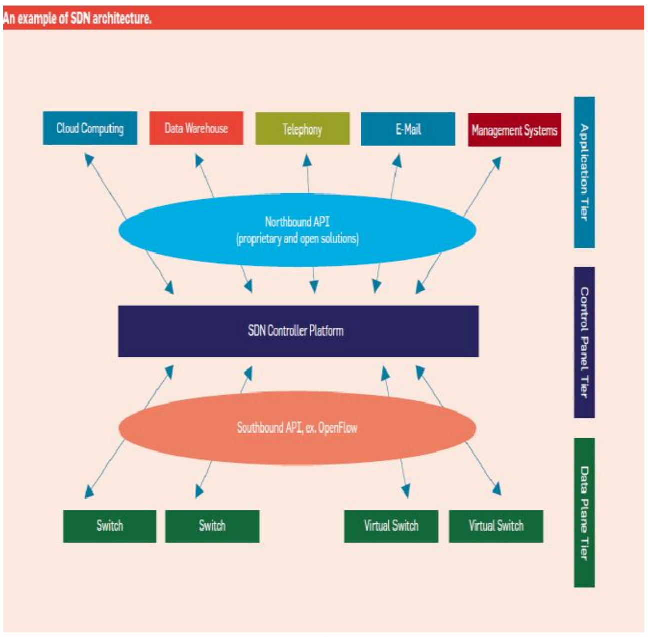

Figure 2: SDN Architecture

On Current network data is transferred using the dedicated switches or routers from source to destination using other connected devices. There are two layers in these switches, the forwarding plane (Data plane) which is responsible for routing the data packets from source to destination. The second layer is control plane, which generates and maintain routing tables. It is not possible for network administrator to individually configure each network device using configuration interfaces that vary between vendors; it can vary in different products supplied by the same vendor. It slows down the innovation, increases the complexity and cost of running a network [17]. Making computer networks more programmable makes innovation in network management possible and lowers the barrier to deploying new services.

2.1 OpenFlow

OpenFlow is an open standard that empowers software defined networking in IP networks. It is another network technique that enables many new applications and new ways of managing networks [20]. Figure 3 shows OpenFlow architecture. An OpenFlow switch contains one or more than one tables which have entries of packet handling rules and known as flow tables. After matching a flow table entry rule certain actions are performed on the traffic. An OpenFlow switch can work as switch, firewall, router, network address translator according to the rules instead by the controller.

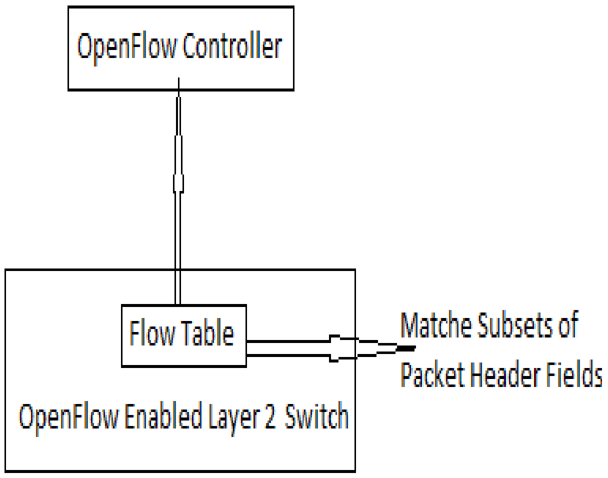
OpenFlow Controller
Flow Table
OpenFlow Enabled Layer 2 Switch
Matche Subsets of Packet Header Fields

Figure 3: OpenFlow Architecture

2.2 Management of allocated bandwidth

In a wide area network, bandwidth utilization is 30 percent. The bandwidth for "burst" times must be reserved by it. A system was developed which uses OpenFlow in which applications with requirement of mass data exchange could use the extra bandwidth. The main uses for the bulk data transfer can be database backups, daily replication of datasets and the exchange of log files. Source address, destination address and the amount of data need to transfer are attached to central service by consumers. Central service and applications use bidirectional communication that is service informed by the application about their need and the service responds on the availability of bandwidth, and the application informs service after its completion. At the same time routers feed real-time usage information to the central service.

2.3 Tenantized Networking:

Organizations which are holding substantial virtual machines have a system administration issue. If service is "tenanted" then it implies that each client is isolated from every other client similar to the situation like tenants are isolated to each other in a same building. When the virtual machine migrates, then the virtual LAN portions must be reconfigured. There is a software-emulated network inside every physical machine that connects the virtual machines.

There must be careful coordination between the virtual LANs, physical LANs and the software emulated world. The issue is that connections can be misconfigured by human error, and tenants have tenuous boundaries. OpenFlow permits new components to be added to the VM administration frame work so that it communicates with the network infrastructure as virtual and changed. This application guarantees that each tenant is disconnected from the others regardless of how the VMs migrate or where the physical machines are located.

2.4 Game Servers

A group of undergraduate students prepare a server on a laptop for conducting a closed competition. Latency of this server is critical for the purpose of game play and fairness, because the server is running on a laptop. While the game is going on, in between the game, owner of the laptop takes the laptop and unplugged it from the wired internet connection, as expected; it connects with the Wi-Fi. The individual then strolls with the portable PC to the canteen and comes after two hours. At the time of moving this path to cafeteria, IP address of laptop changes five times, and the bandwidth varies from 100 Mbps on wired internet connection, to 11Mbps on an 802.11 b connection on different places.

Today's routers are too critical, and it is very difficult to design, routing protocols, to design and implement the routing protocol highly programming skills is required and its verification is very expensive [20].Today's networks are constructed of endpoints, which are our computer and the server to which it is sharing information and some devices which connects them, such as routers, switches, etc. Every router has many numbers of ports and a routing table when a packet arrives at any one port, then router finds the destination address and then searches the routing table for the same and when the match found it simply forwards the packet to other port.

When our computer wants to forward a packet, it does not know how the packet should be forwarded to the destination. So, it simply puts the destination address in the packet and forwards it to the next hop and trusts on all the routers which will come in the route that packets will reach to its destination. Routers share their topological information to construct the routes; each router does the route calculation independently. If two routers are calculated the same route information or some of the route information is overlapping in between their route calculation, they do not exchange their overlapping results. Each router consumes some amount of power and time to calculate the route, thus this duplication of the route computation is not energy efficient.

The central compiler need not have to be vendor specific thus there will be a competition between vendors to provide good services and good route compiler. OpenFlow provides us the facilities to program the networks on per-flow basis. It means we can make it sure that the latency sensitive. The Traffic chooses the fastest path and the bulk flow takes the cheapest path. OpenFlow don't forward the packets on the basis of endpoints as normal networks do, instead of that it forwards packets on the basis of traffic coming from each endpoint.

An organization may use OpenFlow to control the network inside a huge data center and have an alternative OpenFlow space to deal with its WAN. In software defined networking the forwarding decisions are taken on the basis of flow instead of destination address. A group of packet field values, behaving as a matching criterion and a group of actions for forwarding packets is known as a flow [8]. Network Operating System (NOS) or the SDN controller is an external entity which has control over the network. The Network Operating System facilitates the required resources and abstractions which help to program the forwarding switches depending on the logically centralized network view. The network can be programmed with the help of software applications that are working on top of the Network Operating System (NOS) which communicates with the data plane devices [9].

Normally the software defined networking environment uses single controller, but as the devices increases in any network the size of the network also increases and becomes very tough to manage the network with the help of a single centralized controller for this the distributed controller or multi-controller approach is used [13]. The problem with distributed controller environment is consistency that means that updates performed on any controller should be performed on every other controller at the same time. System with weak consistency means that some controllers have the latest data, and some have the old data, and strong consistency means that all controllers have the most updated values [11].

For the load balancing the network traffic is monitored by the controller and it has to use some kind of threshold in the flow counters to redistribute the load or clients among the available controllers when bottlenecks are supposed to happen. Load balancing in SDN also simplifies the placement of the services provided by the network. Whenever a new system comes in the network, then the load balancing service can distribute the traffic among the available servers. SDN can provide a fully automated system to control the router configuration. It allows the reduction of replicated data on routing tables, which will reduce the size of the routingtables.SDN can be used to detect the abnormal behaviors of the data center network operations.

III. CONTRIBUTIONS IN THE FIELD OF PROPOSED WORK

Some ongoing research efforts and challenges are drawn from software defined data center networks based on frequently used paths in controller placement are given below

3.1 Switch Design

Currently available OpenFlow switches are very diverse and exhibit notable differences in terms of feature set (e.g., flow table size, optional actions), performance (e.g., fast vs. slow path, control channel latency/throughput), interpretation and adherence to the protocol specification (e.g., BARRIER command), and architecture (e.g., hardware vs. software designs).Flow entries (matching rules) are stored in flow tables inside network devices. One important and practical challenge is to make the availability of the switches with large and efficient flow tables to store the flow rules [19].

3.2 Controller Platforms

In SDN the controller platform is very important part of the architecture, so the efforts for developing fast, scalable, distributed controller are in process. While designing the controllers the most important part is a high availability of controllers. The Controller should be able to sustain healthy operating under the pressure of different operations from the applications they are hosting. As interoperability is important to forwarding devices, it is also important for controllers, it means that controllers should be capable to communicate in multi-controller environment.

3.3 Scalability

Scalability is one of the big concerns of Software defined networks. Scalability issues, mainly arises due to separation of data plane and control plane. As in SDN every new flow forwards its first packet to the controller, this additional traffic on controller increases load on the network and causes the bottleneck. In addition to this, flow table entries are configured in real-time by the controller, so the extra latency is introduced by the flow setup process. In large networks millions of flow requests come every second on the controller. So, the controller should be able to handle these requests without compromising the quality of service. Once we separate the control elements from the forwarding elements, it's likely that a control element may be responsible for many forwarding elements, sometimes thousands of forwarding elements.

3.4 Reliability/ Security

What happens when a controller fails? We should hope that forwarding elements continue forwarding traffic as usual, but once we have separated the brains of the network of the devices that are actually forwarding the traffic correct behavior is no longer guaranteed [Shin et al., 2012]. Some more problems like if a link fails the how the network will react? How it will recover from the link failure? How many controllers we require and where these controllers should be placed? There are three fault domains in the SDN-based networks. (1) Data plane, where a switch or link fails, (2) control plane, where the connection between the controller and switch fails, and (3) controller, where the controller machine fails [15].SDN is well suited for Data Centers, as SDN allows managing the traffic according to need.

For a single, centralized controller in data center networks, there are lots of limitations Such as data center has variable traffic, sometime it increases very high, so the number of flow requests on controller also increases and sometimes traffic reduces so the number of flow requests also reduces. Whenever the load on controller increase the flow setup time also increases and it increases the probability of a controller failure, as every controller has its physical limitations [6]. To avoid this problem a new approach is proposed which is multi-controller provisioning, in which multiple controllers are used in data center networks, and switches are divided in those multiple controllers. Every switch is connected to exactly one controller. So now the load is distributed among the controllers thus it reduces the flow set up time and reduces the probability of controller failure.

In data center networks the traffic varies frequently and thus the load on the controller also varies, so at any point of time when traffic decreases required number of controllers should also decrease to save energy and unnecessary synchronization delay. We should be able to decide the number of controllers dynamically in data center networks. For deploying SDN in Data Centers multiple controllers are required. So, the placement of this controller is a major problem.

The energy consumed by data center network is very high and most of the time the servers which are not being used also consumes the energy [12]. To save the energy we must be able to decide the number of controllers dynamically, if traffic increases, then we should be able to increase the number of controllers and if the load decreases then also we should be able to decrease the number of controllers.

II. Conclusion And Further Work

The most common SDN implementation today relies on a logically centralized controller that possesses a global view of the network. Whenever a switch receives a new flow, it requests the controller to install appropriate forwarding rules along the desired flow path. The time required to complete this operation is known as the flow setup time [2]. A single controller usually has a limited resource capacity and hence cannot handle large amounts of flows originating from all the infrastructure switches. In this case, the average flow setup time can rise significantly and degrade application and service performance.

In traditional multi-controller environment synchronization between controllers reduces the link utilization. To solve that problem, we can use tree-based architecture for controllers. The main problem here is to place this controller that is how many controllers are required and where these controllers should be placed [10]. If one controller fails, then its load is handled by other controllers which may cause the failure of other controllers also and causes the cascading failure of controllers.

Model for cascading failure in SDN

During the research firstly we will discuss the cascading failure and then after work on solution to prevent cascading failure.

Splitted Multicontroller Environment

During the research firstly we will discuss the cascading failure and then after work on solution for Different dynamic switches assignment. After discussing the entire problem, The Proposed solution is considered a Large Data center, which consist Open Flow switches to deploy the system. It is assumed that in this system controllers are always running on designated servers. We can divide controllers in “Active” and “Inactive” controllers.

If a controller has at least one switch assigned to it, then it is considered as an Active controller, otherwise it is considered as Inactive controller. Inactive controllers continuously listen on a particular port for “Hello” messages coming from newly assigned switches. In dynamic assignment an active controller will become an inactive controller if all the switches controlled by it are assigned to other controllers. Similarly, if a switch is assigned to any inactive controller then that controller will become an active controller.

This approach considers frequent communication between two switches as main parameter for deciding that which switch should go under which controller. When a controller gets overloaded it can decide to split its switches under two controllers. So, for splitting or division load is used as parameter. For this reason, every controller maintains a counter which counts how many requests a controller is serving per unit of time. Frequent Communication: Frequent communication between two switches S1 and S2 can be defined as how many times S1 is sending the first packet of its flow to controller and asking the route for S2, within a particular time interval.

Division: We will be having a path matrix at the controller which will store the number of times a link between two switches is used. In this way controller will come to know that how many times the switches communicates. So, when the condition for splitting arises the single controller can decide that after division most frequently communicating switches should go under the same controller. The previous controller becomes the parent controller. Two new controllers become active as child controllers. A controller can take a decision of the division on the basis of its load handling capacity, if the number of requests per time unit will increase beyond a threshold the division will take place.

Combining: Every child controller will share its communication matrix to its parent controller so that parent controller can have the idea of whole topology and communication or load between switches. Now Parent controller has the idea of the full topology of the network so if the load decreases or if any two switches under different controllers start communicating frequently then the parent node can decide to merge switches or reassignment of switches to controllers, so that the communication delay between two switches can be maintained.

Algorithms

We will work on to prepare algorithms to provide solution to prevent cascading failure and solution for dynamic switch assignments. To design an algorithm which can dynamically decide the number of switches that can be assigned to each controller and assign those switches to the controllers in multi controller environment.

V. CONCLUSION

The complete work expects an integrated solution based on product characteristics and customer behavior for product recommendation. For product suggestion, the entire effort anticipates an integrated solution based on product features and customer behaviour. The following list of points serves as an illustration of the expectations.

1. To create an algorithm for classifying customers for customer behaviour analysis
2. To determine sales trends and popular product lists by calculating product similarity and popularity indices.
3. Product recommendation based on the characteristics of the product and the customer.

The accuracy of the collaborative filtering method is improved by combining sentiment analysis with it, and the item cold start problem is then solved using a content-based approach.

REFERENCES

[1] Bari, M. F., Boutaba, R., Esteves, R., Granville, L. Z., Podlesny, M., Rabbani, M. G., Zhang, Q., and Zhani, M. F. (2013a). Data center network virtualization: A survey. Communications Surveys & Tutorials, IEEE, 15(2):909–928.

[2] Bari, M. F., Roy, A. R., Chowdhury, S. R., Zhang, Q., Zhani, M. F., Ahmed, R., and Boutaba, R. (2013b). Dynamic controller provisioning in software defined networks. In Network and Service Management (CNSM), 2013 9th International Conference on pages 18–25. IEEE.

[3] Barroso, L. A. and Ranganathan, P. (2010). Guest editors' introduction: Data center scale computing. Micro, IEEE, 30(4):6–7.

[4] Benson, T., Akella, A., and Maltz, D. A. (2010a). Network traffic characteristics of data centers in the wild. In Proceedings of the 10th ACM SIGCOMM conference on Internet measurement, pages 267–280. ACM.

[5] Benson, T., Anand, A., Akella, A., and Zhang, M. (2010b). Understanding data center traffic characteristics. ACM SIGCOMM Computer Communication Review, 40(1):92–99.

[6] Cervello-Pastor, C., Garcia, A. J., et al. (2014). On the controller placement for designing a distributed SDN control layer. In Networking Conference, 2014 IFIP, pages1–9. IEEE.

[7] Dhamecha, K. and Trivedi, B. (2013). SDN issues-a survey. International Journal of Computer Applications, 73(18):30–35.

[8] Feamster, N., Rexford, J., and Zegura, E. (2013). The road to SDN. Queue, 11(12):20.

[9] Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., and Shenker, S. (2008). Nox: towards an operating system for networks. ACM SIGCOMM Computer Communication Review, 38(3):105–110.

[10] Heller, B., Sherwood, R., and McKeown, N. (2012). The controller placement problem. In Proceedings of the first workshop on hot topics in software defined networks, pages 7–12. ACM.

[11] Hock, D., Hartmann, M., Gebert, S., Jarschel, M., Zinner, T., and Tran-Gia, P. (2013).Pareto-optimal resilient controller placement in SDN-based core networks. In Tele traffic Congress (ITC), 2013 25th International, pages 1–9. IEEE.

[12] Hoelzle, U. and Barroso, L. (2009). The datacenter as a computer. Morgan andClaypool.Hu, Y., Wang, W., Gong, X., Que, X., and Cheng, S. (2014). On reliability optimized controller placement for software-defined networks. Communications, China, 11(2):38–54.

[13] Jain, R. and Paul, S. (2013). Network virtualization and software defined networking for cloud computing: a survey. Communications Magazine, IEEE, 51(11):24–31.

[14] Kim, H. and Feamster, N. (2013). Improving network management with software defined networking. Communications Magazine, IEEE, 51(2):114–119.

[15] Kim, H., Santos, J. R., Turner, Y., Schlansker, M., Tourrilhes, J., and Feamster, N.(2012). Coronet: Fault tolerance for software defined networks. In Network Protocols (ICNP), 2012 20th IEEE International Conference on, pages 1–2. IEEE.

[16] Kirkpatrick, K. (2013). Software-defined networking. Communications of the ACM,56 (9):16–19.

[17] Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y.,

Inoue, H., Hama, T., et al. (2010). Onix: A distributed control platform for large-scale production networks. In OSDI, volume 10, pages 1–6.

[18] Kreutz, D., Ramos, F., and Verissimo, P. (2013). Towards secure and dependable software-defined networks. In Proceedings of the second ACM SIGCOMM workshop on hot topics in software defined networking, pages 55–60. ACM.

[19] Kreutz, D., Ramos, F. M., Esteves Verissimo, P., Esteve Rothenberg, C.,Azodolmolky,S., and Uhlig, S. (2015). Software-defined networking: A comprehensive survey Proceedings of the IEEE, 103(1):14–76.

[20] Limoncelli, T. A. (2012). OpenFlow: a radical new idea in networking. Queue,10(6):40

[21] Phemius, K., Bouet, M., and Leguay, J. (2014). Disco: Distributed multi-domain SDN controllers. In Network Operations and Management Symposium (NOMS), 2014IEEE, pages 1–4. IEEE.