Indonesian Journal of Electrical Engineering and Informatics (IJEEI)
Vol. 6, No. 2, June 2018, pp. 225~234
ISSN: 2089-3272, DOI: 10.11591/ijeei.v6i2.471

Using SDN as a Technology Enabler for Distance Learning Applications

Zahra Al-Abri[1], Ahmed Al Maashri[2], Dawood Al-Abri[3], Fahad Bait Shiginah[4]
[1]CGG, PDO Dedicated Subsurface Imaging Center–Oman, Oman
[2,3,4]Department of Electrical & Computer Engineering, Sultan Qaboos University, Oman

Article Info

Article history:

Received Apr 30, 2018
Revised May 12, 2018
Accepted May 26, 2018

Keyword:

Distance Learning
Mobile Networking
OpenDayLight Controller
Software-Defined Networking (SDN)

ABSTRACT

The number of students who obtained degrees via distance learning has grown considerably in the last few years. Services provided by distance learning systems are expected to be delivered in a fast and reliable way. However, as the number of users increases, so does the stress on the network. Software-Defined Networking, on the other hand, is a new technology that provides a rapid response to the ever-evolving requirements of today's businesses. The technology is expected to enhance the overall performance of cloud services, including those provided by distance learning. This paper investigates the benefits of employing such a technology by educational institutions to provide quality services to the users. The results of the experiments show an improvement in performance of up to 11%, when utilizing the technology. In addition, we show how resource reservation features can be utilized to provide quality service to users depending on their role in the distance learning system.

Corresponding Author:

Zahra Al-Abri,
CGG, PDO Dedicated Subsurface Imaging Center–Oman, Oman
Email: Zahra.AlAbri@cgg.com

1. INTRODUCTION

Distance Learning has flourished in the last few years, where students are no longer required to be physically present on campus. In the UK alone, there are more than 270,000 students who use distance learning to pursue undergraduate degrees; and another 108,000 pursuing postgraduate degrees [1]. Since the student's interest in distance learning has increased significantly, many universities are offering degrees through distance learning.

Distance learning offers multitude of services; including educational content, audio/video conferencing, chat, and email [2]. The architecture of distance learning system may vary from one implementation to another. One implementation of the system is based on the two-tier architecture model [2-4]. As shown in Figure 1, the model comprises layers that host system components. For instance, the bottom layer hosts the databases (e.g. lessons, users, media, etc.). These databases are created, updated, and administered through the Relational Database Management System (RDBMS). The top layer hosts servers that offer a variety of services. Web servers (e.g. Microsoft's IIS and Apache) allow the separation of the content from the presentation. In addition, they can be used for executing scripts and components on the server side. Similarly, media servers store and share multimedia content. The server delivers these content to clients upon request. Media servers are widely used to deliver video on demand (VOD) and live streaming. Furthermore, a load balancer provides the proper distribution of workloads between the servers.

In order to handle the communication between the databases and servers, a www-to-database interface is provided. Similarly, a graphical user interface provides the communication between servers one side, and users/administrators on the other side.

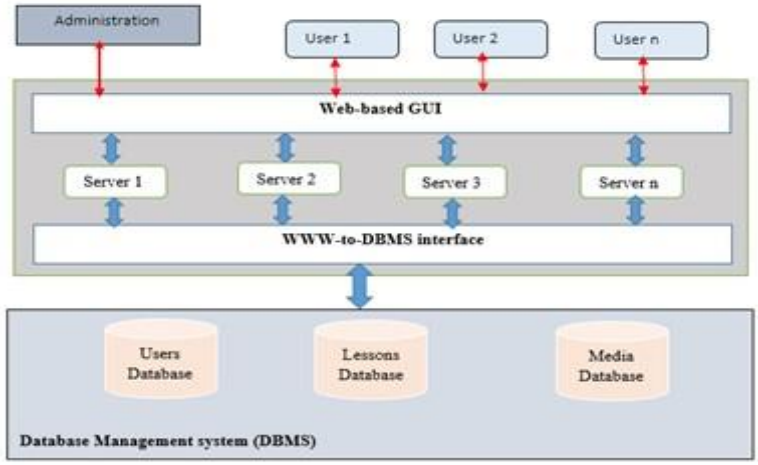

Figure 1. Two-tier architecture of the distance learning system

As the number of active users in the distance learning system increases, so is the demand on the online resources. Such a high demand puts an enormous pressure on the online resources such as web portals, video streaming, etc. These resources, which are usually hosted in a cloud, mandate an efficient and flexible management. In fact, cloud computing has a very dynamic environment, where users, resources, and applications are changing from time to time. Moreover, cloud computing needs a very large-scale infrastructure layer that can support this huge change. Nowadays, there is an increasing demand to expand the scale of cloud computing, which means that more stress to be put on control, management, and programmability.

However, traditional networks may not be able to cope with such requirements [5]. This is mainly attributed to the fact that in traditional networks the data plane and control plane coexist within the same layer. Therefore, when network administrators want to configure the network devices, they must configure each of these devices separately. This means that changing required policies is a very challenging task, especially when there is a large number of devices that need to be configured. As such, traditional networks may not be the most suitable environment for implementing distance learning systems. This is because the latter require capable control and management tools to secure and manage a large number of students accessing various resources.

On the other hand, Software-Defined Networking (SDN) is a new network architecture that provides a rapid response to the ever-evolving requirements of today's businesses. Unlike traditional network architectures, SDN is more flexible and agile to support the dynamic computing and the huge storage resources hosted in the cloud and data centers. In contrast to traditional networks, the control and data planes are decoupled in the SDN network. This separation of control and data planes offers direct programmability, less complexity, virtualization, security, and agility [6,7].

Therefore, deploying cloud computing using SDN could potentially help to scale up the size of cloud computing and enhance the utilization of network resources. As such, it is expected that employing SDN can help to deliver reliable distance learning services. Hence, SDN could potentially enable educational institutions to cope with the demand on distance education.

The main objective of this paper is to investigate the use of SDN technology for distance learning applications. This includes investigating the flexibility of SDN controller to provide distance learning services and measuring the performance gain compared to traditional networks. To the best of our knowledge, no prior work has done that.

This paper makes the following contributions: (1) a distance learning network is modeled in an emulation tool, (2) SDN technology is introduced to the distance learning model, (3) experiments are conducted on the model to quantify the benefits of introducing SDN using a number of realistic scenarios, and (4) comparative results are reported that clearly show the benefits of using SDN compared to traditional networks.

The rest of this paper is organized as follows: Section 2 describes the SDN technology, its architecture, and controllers. This is followed by a discussion of the emulator that was used for modeling the system, in Section 3. The experiment setup and results are discussed on Section 4, while Section 5 concludes the paper with highlights on future extensions to this work.

2. BACKGROUND

This section discusses the basics of (SDN), as a technology enabler for multiple application domains. This discussion includes the proposed architectures of SDN, Controllers, and using SDN in distance learning applications.

2.1. SDN

SDN has three main layers; namely, application layer (Application plane), control layer (Control plane), and infrastructure layer (Data plane). This is illustrated in Figure 2. In contrast to traditional networks, SDN decouples the control and data planes, such that each plane exists in a separate layer. That is why SDN architecture provides ease of administration, a more dynamic and adaptable solution, and is less expensive to implement compared to other architectures [6, 7].

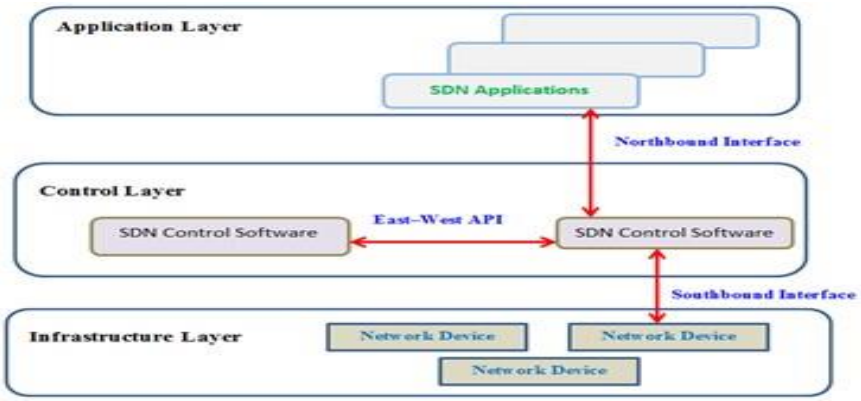

Figure 2. Layers and Interfaces of SDN

The Application layer includes SDN applications, which are programs that have application requirements. In contrast, the control layer includes multiple controllers, which are responsible for information management that is collected by the network devices at the infrastructure layer. These devices, at the infrastructure layer, are used for forwarding the data.

Three interfaces are used to provide the communication between those three layers. The first interface is called Northbound API, which is used to facilitate the communication between an application running in the application layer and SDN controllers in the control layer. The second interface is called Southbound API and it provides the link between controllers of the control plane and the network devices in the data plane. Moreover, there is a third interface, which is used for facilitating the communication between the neighboring controllers. This interface is referred to as East–West API [6, 7].

SDN has attracted the attention of the community because it can be integrated into several technologies. Examples of such technologies are Internet of Things (IoT), big data, cloud computing, telecom, etc. In addition, SDN is one of the most active topics in research and development.

Because it decouples the Control layer from the Infrastructure layer, SDN architecture reduces the complexity as it hides the infrastructure layer from the control layer. Isolation is another characteristic of SDN in which the organization can be split into different departments and each department can be logically separated from others using a single infrastructure layer. This separation leads to the implementation of virtualization, which can significantly reduce the cost [7].

SDN provides direct programmability. As such, when there is any change in SDN application requirements, the application can directly inform SDN controller using open APIs. Then, centralized SDN controller can change custom policies across the underlying network hardware using the Southbound Interface. Therefore, some consider SDN as an agile service, since the network is highly responsive to any change in the requirements of SDN's application. As a result of centralized SDN controller, network managers can quickly configure and manage the network.

2.2. SDN Architecture

There are a number of proposed approaches to build SDN architecture such as OpenFlow, SDN Overlay, and Hybrid SDN. Each one of these approaches has its own pros and cons. The most commonly used is the OpenFlow architecture.

OpenFlow, a SouthBound API, is an open standard communication protocol that is used to allow the communication between control layer and infrastructure layer of the network. OpenFlow uses a flow table and group table to perform packet look-up and forwarding. The administrator uses these tables to quickly change the network layout and traffic flow, where the SDN controller is responsible for updating these tables. The OpenFlow switch is connected to the controller using the OpenFlow channel as shown in Figure 3. A flow is a path for the packets to travel from the host source to the destination through an OpenFlow-compliant switch. Similarly, the flow entry is the instruction or rule that shows where the switch should forward data packets. Therefore, each flow should be matched with the flow entry. The instructions in the flow entries perform actions such as "forward", "drop", and "modify" on the incoming traffic. There is a counter inside the flow entry, which is responsible for recording the number of received packets, as well as the time taken to process each flow. By using OFStatisticsRequest messages, controllers request the above information from a switch, where the latter should respond to the request. The logically centralized controller of SDN can control and view all the forwarding packets between network devices using OpenFlow interface [6]. This is why SDNs are highly flexible compared to traditional networks, where in the latter the administrators need to make custom policies and protocols on each network device [8]. Furthermore, OpenFlow architecture supports security since the OpenFlow channel can encrypt traffic by using Transport Layer Security (TLS) [7].

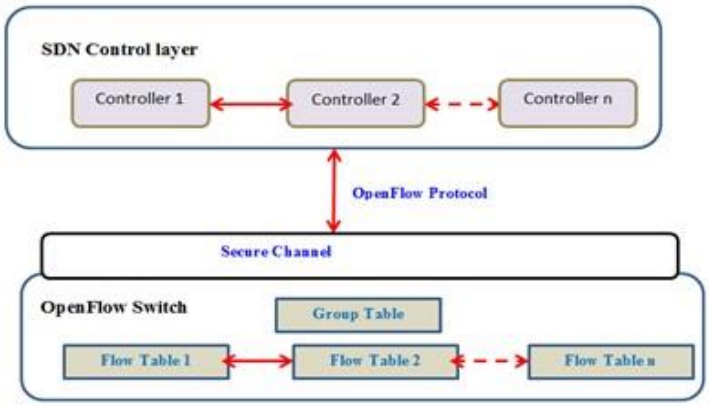

Figure 1. OpenFlow architecture

2.3. OpenDayLight (ODL) SDN Controller

On April 2013, the Linux Foundation [9] developed a platform named OpenDayLight (ODL). The objective of this project is to accelerate the adoption of SDN by providing the platform that allows network designers to create a network with any size and scale [10]. The Linux Foundation built a common platform, allowing users to manage applications and protocols. Moreover, connections between customers and providers can be implemented using this platform. This could be the reason why ODL has become the industry's de facto standard. It is worth noting here that ODL platform is updated every six months by the global community of ODL's development [11].

2.4. Deployment of SDN in Cloud Services

Cloud computing requires flexible and responsive control, management, and programmability due to its dynamic environment. The characteristics of SDN allow this technology to scale up the size of cloud computing and to make a better utilization of network resources. There are several proposals that suggest the integration of SDN with compute and storage systems in single environment that has a complete automated provisioning and orchestration of IT infrastructures for cloud computing [5, 12, 13]. Such integration is referred to as Software-Defined Cloud Environment (SDCE) [5] or SDN-based cloud [14]. To be able to build the architecture of SDCE, a Service-Oriented Architecture (SOA) is used. In essence, SOA is a mechanism that allows all the components of heterogeneous environment to communicate with each other using loose-coupling interactions. In fact, it is suggested that SOA can be adopted in any of the cloud computing's layers; namely, Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS). Furthermore, if SOA is applied in a network, it forms a paradigm known as Network-as-a-Service (NaaS).

3. SDN-BASED DISTANCE LEARNING SYSTEM

Figure 4 illustrates a conceptual view of a distance learning system powered by SDN. The system consists of Network Connectivity and the Cloud. Distance learning users request services from a Cloud that hosts a distance learning system.

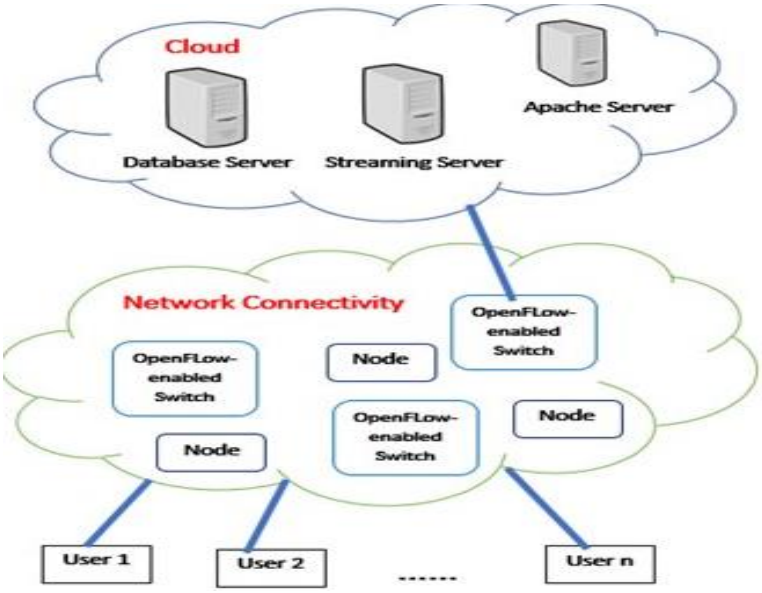

Figure 4. Overview of the system

The Network Connectivity represents the process of connecting various devices of the network to one another. These devices are connected via OpenFlow-enabled switches. These switches along with the participating nodes represent the infrastructure layer. Each of the OpenFlow-enabled switches should be connected to the logically centralized controller. The latter is configured to control the switches according to the requirements of the distance learning application. ODL has been selected in this work as the controller for the SDN network. Furthermore, the cloud comprises a database and web servers. The database server hosts the educational and administrative content such as lessons, media, and users' information.

The following subsections discuss the tools that were used to model each component in the system described above.

3.1. Mininet Emulator

Network simulators such as NS2, NS3, and OMNet++ have been heavily used in literature to model networks. While these tools provide flexibility in modeling, however, they may not be accurate in their output. In addition, new technologies, such as SDN, may not be adequately supported by these tools. For example, NS-3 supports OpenFlow version 0.8.9 only. Furthermore, the implementation of SDN controllers may not be as per the standard.

The Mininet emulator, on the other hand, accurately captures the network behavior as it relies on standard implementations of protocols and controllers. In fact, Mininet has a better support for SDN (e.g. it supports all OpenFlow versions). Additionally, Mininet supports spanning tree protocol, which is deemed necessary to remove cycles and loops in the network [15-17].

Mininet is proposed as a new emulator tool that allows implementing networks with large numbers of devices. By using Mininet, network designers can create SDN prototype easily. Mininet links all the network devices such as hosts, switches, and routers on a single Linux kernel. By using virtualization, all of these devices are perceived as individual systems. Network designers can easily create an SDN prototype using Mininet with huge capabilities of interacting, sharing, customizing the prototype and running it on hardware. Another important characteristic of Mininet is that controlling and managing the whole virtual network can be done from a single console using CLI. Most importantly, the created Virtual Machine can be shared across the developers, allowing them to evaluate and run the existing simulation framework, without the need to create them from scratch [18, 19].

In this work, a Python script was developed to capture the system as depicted in Figure 4. In addition, OpenDayLight Controller was used instead of Mininet's default controller

3.2. Cloud Computing Modelling

Implementing a cloud computing system can be done using many cloud platforms such as Rackspace, Amazon EC2, Microsoft Azure, Google Cloud, and OpenStack. Cloud management tools add complexity to the system that does not contribute significantly to the focus of this paper. Therefore, we opted for representing the cloud with a number of online services such as web, database, and streaming services. For that reason, LAMP sever was used, which is an open source software stack that consists of Linux operating system, Apache web server, MySQL database, with a support for PHP scripting language.

4. EXPERIMENTS

We have developed an end-to-end environment that emulates distance learners who are remotely accessing educational content. The main objective of building this environment is to identify the performance gains when using SDN-enabled network. First, we describe the experiment setup.

4.1. Experimental Setup

The Mininet emulator is hosted on a virtual machine, with a 32-bit Ubuntu OS running on 3.4GHz processor. Another virtual machine was used to initiate requests to the Distance Learning server. This machine is running on 64-bit Ubuntu operating system. A separate physical machine was used as a Cloud server, which hosts the distance learning services. The server runs Ubuntu 16.04 LTS with 11.6 GB system memory and quad-core 3.4GHz processor. The content in the server ranges in size according to the application (e.g. lecture notes, exams, streaming media, etc.).

Distance learning users are allowed to download content (e.g. accessing lecture slides), upload content (e.g. submitting an essay or a project report), or interactively download/upload content (e.g. solving an online exam). It is worth noting that all experiment were repeated a minimum of 10 times to obtain a reliable average for the results.

4.2. Test Scenarios and Results

In this subsection, we introduce 3 test scenarios to identify the performance gained by introducing SDN into distance learning network.

4.2.1. Traditional vs. SDN-based Networks (Ideal scenario)

In this scenario, we compare the performance of a traditional network (see Figure 5) and SDN-enabled network (see Figure 6), assuming ideal environment. In this context, an ideal environment is one in which all the links that connect nodes never fail and introduce no propagation delays. In addition, it is assumed that there is only one single path to reach the destination. Of course, such unrealistic environment doesn't exist. However, we use the outcomes of this scenario to highlight the superiority of SDN networks in later experiments.

Figure 6 illustrates the ODL-SDN Network Topology, where OpenFlow switches are configured dynamically by ODL. Red-dashed lines represent virtual channels between the controller and OpenFlow switches. Both networks connect multiple users to the distance learning services. However, in the traditional network both control and infrastructure layers coexist in the same device (using a legacy switch). In contrast, SDN network decouples the control layer (operated by ODL controller) and the infrastructure layer (operated by OpenFlow-enabled switches). All switches in the SDN network are logically connected to the ODL controller. In this scenario, we use the throughput as a performance metric.

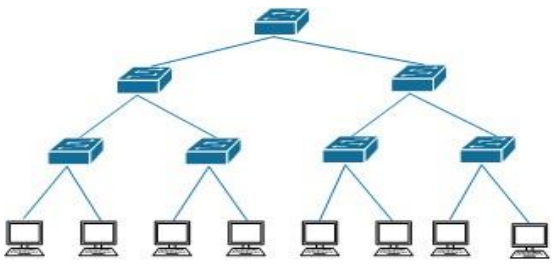

Figure 5. Traditional network topology. Switches are configured individually by network administrator. Both control and Infrastructure co-exist in the same device

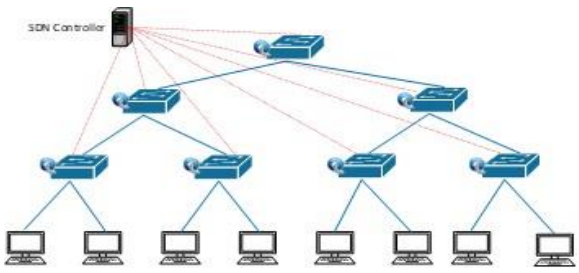

Figure 6. ODL-SDN network topology. OpenFlow switches are configured dynamically. Red-dashed lines represent virtual channels between the controller and OpenFlow switches

Figure 7 shows a throughput comparison between the two networks. As the size of the exchanged file increases, SDN networks exhibit better performance. However, the performance gain is insignificant. Since there is only one route to the destination, relay switches do not have to engage in smart decisions. Therefore, in such a scenario, using SDN networks does not yield much benefit to the users.

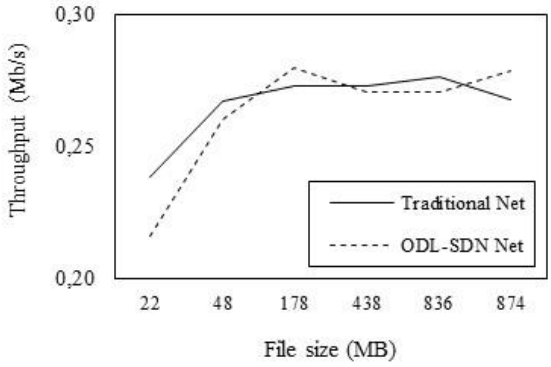

Figure 2. Throughput comparison between traditional and ODL-SDN assuming ideal environment

4.2.2. Traditional vs. SDN-based Networks (Realistic Scenario)

Unlike the previous scenario, this subsection compares the performance of the network in the presence of realistic network constraints. These constraints include varying bandwidth and delay of each link and introducing multiple paths to the destination. The SDN network, illustrated in Figure 8, is powered by the ODL controller. The traditional network uses the same topology as of the SDN network, expect that all nodes rely on legacy switches.

Figure 9 shows the performance gain when using ODL-SDN network. The figure shows that the ODL-SDN outperforms the traditional network with an average of 7%. The performance gain can be attributed to a number of factors. First of all, ODL controller has a complete view of what is happening inside the network and it can choose the best route. In addition, the controller has many features such as load balancing, traffic engineering, multi-layer network optimization, and traffic-rerouting. These features contribute to the performance improvement of the SDN network.

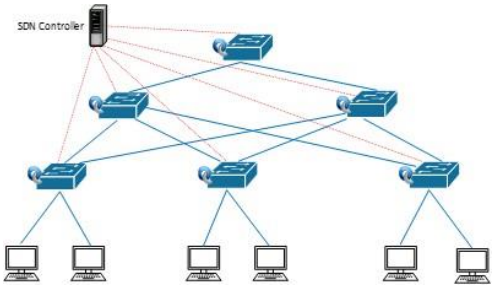

Figure 8. ODL-SDN Network Topology with Network constraints

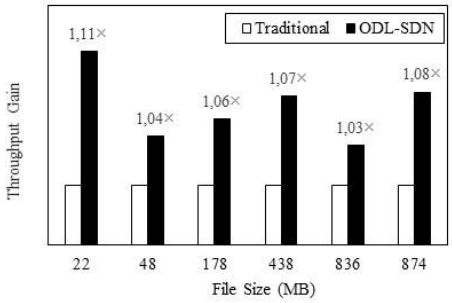

Figure9. Throughput comparison between Traditional and ODL-SDN Network. Values are normalized to the throughput of the Traditional network

4.2.3. Dynamic Resource Reservation

Educational content that is provided to students vary in their requirements and nature. For instance, lecture notes require sufficient download throughput. In contrast, video conferencing has more mandating requirements when it comes to download/upload speed and delay. In addition, students enrolled in distance learning login from different locations and are expected to vary in their connection speed. It makes sense to

provide the manager with the tools to perform dynamic resource reservation, which can be enforced on demand depending on the nature of the content and participating students. ODL controller offers provisioning tools that allow setting the speed and access to resources. Furthermore, these tools can be utilized to set a priority to distance learning users. This subsection, emulates a number of cases, where the requirements are different and thus the resource allocation would differ accordingly.

In the following cases, the network was modeled in such a way that links vary in their allocated bandwidths and delay. The number of users and generated flows, along with the traffic size vary according to the test case. Also, note that the status of OpenFlow switches are reset for each case. This helps in identifying the impact of the combination of traffic and settings on the overall performance.

Case I: Content Download

This scenario emulates students downloading contents from the distance learning cloud. The objective is to manage the resources allocated to students, by configuring the ODL controller such as that links are associated with varying throughputs. ODL allows administrators to create queues, which are utilized to segregate between different clients, where each queue is associated with a particular port in an OpenFlow switch. The queue has a predefined minimum and maximum throughput. Additionally, the ODL controller allows the network administrator to associate clients to queues using their Address (physical or virtual), provided services (e.g. TCP, FTP, etc.), or even protocol stack.

We declare two flows, each of which is set with a different maximum throughput. The IP addresses and TCP port number were used to identify users, and accordingly match them to the corresponding flow. These flows are as follows:

- Flow 1: This flow is assigned to campus-based students and it is capped to 2 Mb/s throughput.
- Flow 2: This flow is used by students who are enrolled as distance learning student. The maximum throughput set to 5 Mb/s.

Figure 10 shows a comparison in throughput gain between the two flows. In the absence of any resource reservations, the throughput of Flow 2 is insignificantly higher than that of Flow 1. By enforcing reservation of resources, the figure shows that the throughput of Flow 2 is 3.5× higher than Flow 1. This improvement in throughput gives an edge to the students who are accessing online content.

Case II: Content upload

In contrast to the previous scenario, students might be required to submit assignments, simulation traces, reports, etc. In such cases, the upstream throughput is more crucial than downstream. We identify three different groups of users who access the cloud service. Each group is assigned a flow (i.e. Flow 1, Flow 2, and Flow 3). The only difference between Flow 1 and the other two flows is that the former has a cap on the maximum upload speed (5Mb/s). The assumption here is that more priority should be given to users in the two other groups during peak times (e.g. deadline of report submission).

Results show that before enforcing speed limits on Flow 1, all flows had the same throughput, as illustrated in Figure 11. However, once the speed Flow 1 is limited, we can see that the users in the other flows were given an advantage, as now they can upload contents 20X faster that users in Flow 1.

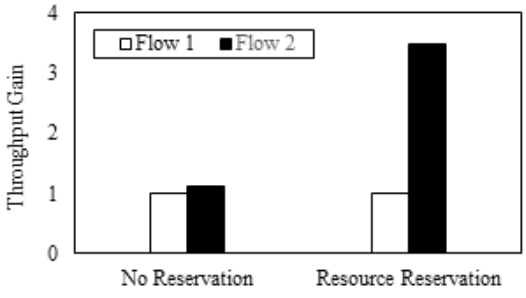

Figure 10. Throughput gain comparison for content download between two flows before and after enforcing resource reservation. Values are normalized to the throughput of Flow 1

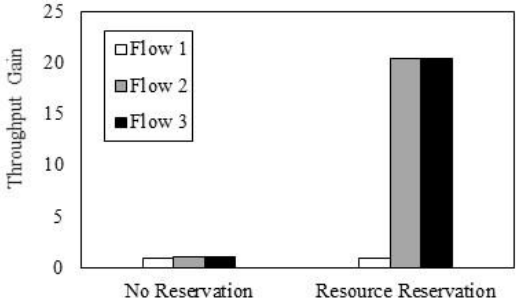

Figure 11. A comparison of throughput gain between three flows before and after enforcing maximum throughput on Flow1. Values are normalized to the throughput of Flow 1

Case III: Online Event Service

The two previous scenarios assume that users are either downloading or uploading content. However, distance learning involves much more than that. For instance, instructors can deliver webinars to students who join remotely. In essence, online events include exchange of information in both directions with very little tolerance to delay.

In this scenario we consider three group of users; namely, presenters, students who are joining the online event, and other users. Accordingly, we allocate flows to these groups as follows:

- Flow 1: This flow is allocated presenters of online events. This group has the highest priority, and therefore are provisioned more resources compared to the two other groups.
- Flow 2: This flow is allocated to students who are joining the online event. While this group has less priority than presenters, they definitely have higher priority compared to other users.
- Flow 3: This flow is assigned to other users. This group has the lowest priority.

Figure 12 shows the implications of prioritizing users based on their role in the online event. By enforcing highest priority, presenters are 5X faster than other users. Similarly, students who attend the online service are 2.2X faster than other users. The ability to dynamically enforce priorities whenever needed provides a flexible management of distance learning resources and users.

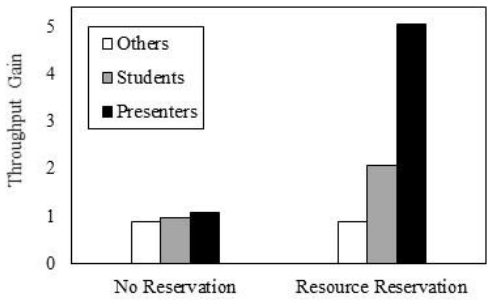

Figure 12. The effect of prioritizing groups of users on the overall speed of each group. All values are normalized to the throughput of users who are not presenting or attending online events. Values are normalized to the throughput of "Others"

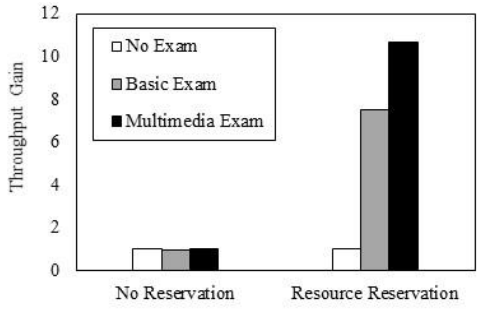

Figure 13. Throughput comparison between three groups of students. By enabling ODL-SDN flows, the students enrolled in online exam are favored and are allocated more resources. Values are normalized to the throughput of students who are not taking any online exams

Case IV: Online Exams

Several campuses allow students to do online exams. The type of questions in these exams include Multiple-Choice Questions (MCQ), True/False, short answers, essays, etc. Generally speaking, the size of download content is larger than upload content. This is because the questions may embed multimedia content (e.g. pictures, animation, or video). On the other hand, the content size of answers is much smaller compared to the questions themselves. This dissimilarity between download and upload size should be considered by network administrators. In addition, it is essential to provide a reliable access to the students to ensure that questions are delivered in a short amount of time, while answers are uploaded within a reasonable amount of time.

A set of experiments have been conducted to emulate a scenario of three (3) groups of students. The first group includes students who are not enrolled in any exam (No Exam). The second group represents students who are taking an exam that doesn't include questions with multimedia content (Basic Exam). This group requires more resources compared to the first group. The last group represents students who are taking an online exam with multimedia content (Multimedia Exam). Therefore, this group should be allocated more resources compared to the first two groups. Accordingly, three flows have been declared to match the requirements of each group.

As expected, the use of dynamic resource allocation has allowed students to experience better access to their online exam. Figure 13 shows that as soon as flows are enabled, the students who are taking online exams are allocated higher throughput compared to other students. In particular, students who are taking Basic (Multimedia) Exam, are having an average throughput that is 7.5X (10.7X) higher than the throughput of the students who are not currently taking any online exams.

5. CONCLUSIONS AND FUTURE WORK

Recent breakthroughs in computing systems and network gear have abstracted the complexity of the infrastructure and leveraged programmability to the administrators. Service providers need to take avail of new technologies such as SDN, which provide additional flexibility in relevant application domains. One such application, is distance learning. It is expected that SDN networks would potentially enable educational institutions to provide their services with quality and high performance.

This paper has investigated the use of SDN in provisioning distance learning services. SDN-based distance learning network has been modeled using Mininet emulator, where a set of experiments were conducted. Some of the experiments are used to compare the performance of the network for distance learning system using traditional network vs. ODL-SDN network. Other experiments were conducted to implement and analyze the provisioning of a variety of distance learning services using resource reservation. The results show that the SDN-based network outperforms traditional network by up to 11%. Furthermore, the use of resource reservation allows the service to distinguish between users, and hence allocating more resources to those who actually need them.

The work in this paper can be further extended to investigate the performance of other applications such as Mobile Learning (M-learning) and Videoconferencing. Some of the applications are streaming in nature, and hence have more stringent real-time requirements. Additionally, one can investigate the impact of larger client size on the quality of services provided by the distance learning systems.

REFERENCES

[1] What is Distance Learning?, The Complete University Guide [online] https://www.thecompleteuniversityguide.co.uk/distance-learning/what-is-distance-learning/
[2] Bouras C, Destounis P, Garofalakis J, Gkamas A, Sakalis G, Sakkopoulos E, Tsaknakis J, Tsiatsos T (2000) Efficient web-based open and distance learning services. Telematics and informatics 17 (3):213-237
[3] W. Li and Y. Wang, "Design and Implementation of Distance Learning Platform Based on Information Technology and Cloud Computing," JNW, vol. 9, pp. 2059-2065, 2014.
[4] Saad MNM, Selamat AW (2012) UPSI Learning Management System (MyGuru2) in the cloud computing environment. Procedia-Social and Behavioral Sciences 67:322-334
[5] Duan Q (2015) Modeling and performance analysis for composite network–compute service provisioning in software-defined cloud environments. Digital Communications and Networks 1 (3):181-190
[6] da Silva AS, Smith P, Mauthe A, Schaeffer-Filho A (2015) Resilience support in software-defined networking: A survey. Computer Networks 92:189-207
[7] Jammal M, Singh T, Shami A, Asal R, Li Y (2014) Software defined networking: State of the art and research challenges. Computer Networks 72:74-98
[8] Koulouzis S, Belloum AS, Bubak MT, Zhao Z, Živković M, de Laat CT (2016) SDN-aware federation of distributed data. Future Generation Computer Systems 56:64-76
[9] Linux.com (2014) Why Contribute to an Enterprise Open Source Project?
[10] Brooks M, Yang B A Man-in-the-Middle attack against OpenDayLight SDN controller. In: Proceedings of the 4th Annual ACM Conference on Research in Information Technology, 2015. ACM, pp 45-49
[11] "The OpenDaylight Platform | OpenDaylight". (2016). https://www.opendaylight.org/.
[12] P. Azodolmolky and R. Yahyapour, "SDN-based computing networking," in processdings of the 15th international Conference on Transparent Optical Networks (ICTON), 2013, pp. 1-4.
[13] E. Chirivella-Perez, J. Gutierrez-Aguado, J. M. A. Calero, and J. M. Claver, "Towards a SDN-based architecture for analyzing network traffic in cloud computing infrastructures," in proceedings of 23rd International Conference on Software, Telecommunications and Computer Networks (SoftCOM), 2015, pp. 295-299
[14] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges," IEEE Communications Surveys & Tutorials, vol. 18, pp. 602-622, 2016.
[15] Wang S-Y Comparison of SDN OpenFlow network simulator and emulators: EstiNet vs. Mininet. In: Computers and Communication (ISCC), 2014 IEEE Symposium on, 2014. IEEE, pp 1-6
[16] M.-C. Chan, C. Chen, J.-X. Huang, T. Kuo, L.-H. Yen, and C.-C. Tseng, "OpenNet: A simulator for software-defined wireless local area network," in Wireless Communications and Networking Conference (WCNC), 2014 IEEE, 2014, pp. 3332-3336.
[17] P. Pupatwibul, A. Banjar, A. Al Sabbagh, and R. Braun, "A Comparative Review: Accurate OpenFlow Simulation Tools for Prototyping," JNW, vol. 10, pp. 322-327, 2015.
[18] GitHub hgcmmwI-t-MA-O- mininet/mininet. https://github.com/mininet/mininet/wiki/Introduction-to-Mininet.
[19] Keti F, Askar S Emulation of Software Defined Networks Using Mininet in Different Simulation Environments. In: Intelligent Systems, Modelling and Simulation (ISMS), 2015 6th International Conference on, 2015. IEEE, pp 205-210