DOI: 10.2298/CSIS121114036H

Design and Implementation of an Efficient and Programmable Future Internet Testbed in Taiwan

Jen-Wei Hu[1, 2], Chu-Sing Yang[1], and Te-Lung Liu[2]

[1] Institute of Computer and Communication Engineering,
National Cheng Kung University, Tainan, Taiwan, R.O.C
{hujw, csyang}@ mail.ee.ncku.edu.tw
[2] National Center for High-Performance Computing, Tainan, Taiwan, R.O.C
{hujw, tlliu}@nchc.narl.org.tw

Abstract. Internet has played an important part in the success of information technologies. With the growing and changing demands, there are many limitations faced by current Internet. A number of network testbeds are created for solving a set of specific problems in Internet. Traditionally, these testbeds are lacking of large scale network and flexibility. Therefore, it is necessary to design and implement a testbed which can support wide range of experiments and has the ability of programmable network. Besides, there has been a big change enabled by cloud computing in recent years. Although networking technologies have lagged behind the advances in server virtualization, the networking is still an importance component to interconnect among virtual machines. There are also measurement issues with growing number of virtual machines in the same host. Therefore, we also propose integrating management functions of virtual network in our testbed. In this paper, we design and create a Future Internet testbed in Taiwan over TWAREN Research Network. This testbed evolves into an environment for programmable network and cloud computing. This paper also presents several finished and ongoing experiments on the testbed for multiple aspects including topology discovery, multimedia streaming, and virtual network integration. We will continue to extend our testbed and propose innovative applications for the next generation Internet.

Keywords: Future Internet, OpenFlow, Testbed, TWAREN.

1. Introduction

Internet has become the most important information exchange infrastructure that provides business transaction, personal communication, information sharing, etc. With wide range of applications and services applied to the Internet, some challenges are issued beyond its original design including scalability, security, mobility, flexibility, and so on [2], [10].

For resolving the increasing issues in current Internet, the U.S., E.U., Japan, and Korea have launched research projects for the Future Internet [5], [7], [13], [14], [20]. There were many issues discussed in these projects, especially on how to rethink and redesign decisions underlying current network architecture. Each project has its different aspects for Future Internet, but comes to the same conclusion, that is to provide an environment for performing research. Therefore, an experimental infrastructure on real networks is desirable to apply new protocols or develop new technologies. However, running experiments on the production network may be risky [4], and control-plane functions in most of network equipments are untouchable. There are some research projects focusing on eliminating the barriers of innovation, such as FEDERICA [8] and GENI [9]. The main goal is to develop a programmable network and enable multiple researchers to obtain a slice of resources by using network virtualization.

TaiWan Advanced Research & Education Network (TWAREN) [22] was established and managed by NCHC, which has been operating since Jan, 2004. It was developed using the latest network technologies and can offer users a variety of new services including IPv6, Multicast, and Light Path. The goals of TWAREN network design are:

- Hybrid technology: IP (routing) over optical Light Path (dark fiber, SDH, or Wavelength).
- Dual networks: production and research networks.
- Hierarchical topology: 3 tiers (cores, POPs, and end nodes).
- Multiple services.
- As shown in Fig. 1, TWAREN owns an island-wide network infrastructure in Taiwan. It plays an important role like Internet2 in the U.S. and GEANT in Europe. One mission of TWAREN is to continue developing and providing new technologies and environment for researchers. To meet this goal, we plan to deploy the Future Internet testbed in TWAREN and further extend into universities or research institutes.
- Besides, cloud computing has become a common word in IT industry. One key technology of cloud computing is hardware virtualization. A well-known of hardware virtualization techniques is the hypervisor (e.g., VMware, Xen, and KVM etc.) which allows multiple operating systems, called virtual machines (VMs), running concurrently on a same host machine. However, virtual networking technologies have lagged behind the advances in hardware virtualization [17]. The main reason is that cloud computing considers the service interaction more than network infrastructure. Each virtual machine shares same physical resources including network connection. Currently, most hypervisors use the existed network bridge to provide virtual machines connectivity [18]. As everything is virtualized in cloud environment, it gets even harder to manage. There still remains many research topics and open problems (e.g., traffic visibility, isolation, and security among VMs) in current cloud networking. In addition to supporting programmable network, we also expect our architecture to

provide a small cloud environment in which virtual switching services are enabled.

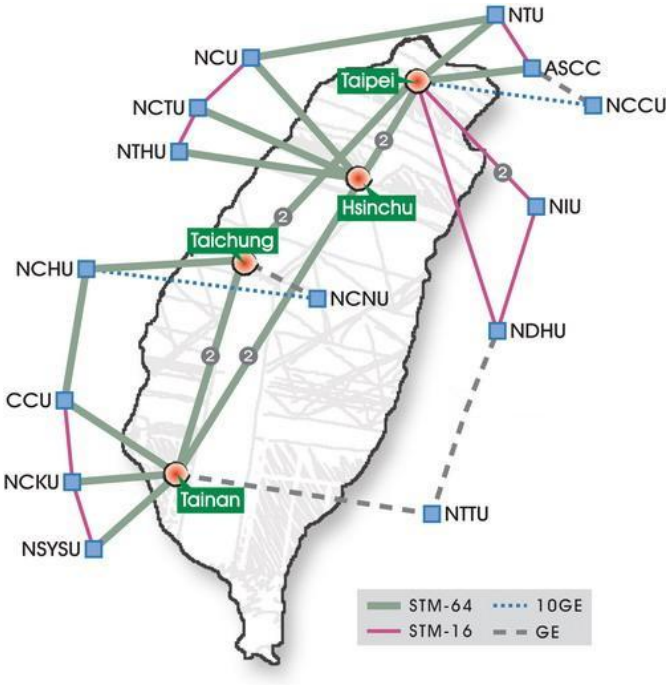

Fig. 1. TWAREN network structure [22]

The rest of this paper is organized as follows, In Section 2, we discuss related work in existing Future Internet testbed. In Section 3, we outline the implementation of our testbed and present current deployment status. Also, we briefly describe useful management modules that have been run on the testbed in Section 4. We present some performance results of our testbed in Section 5. Finally, we conclude this paper with a summary of this work in Section 6.

2. Background and Related Work

In this section, we present background information relevant to our work. We also survey related work and point out their relationship to our work. To design a future-proof testbed, there are some conditions that need to be considered. The network on this testbed should be programmable and isolable. Therefore, we first discuss Software defined network (SDN) and OpenFlow [15]. Then we introduce two famous testbeds based on the SDN architecture and discuss some similarities and differences between these existing testbeds and ours.

2.1. SDN and OpenFlow

The current Internet architecture is not sufficient to support the emerging applications in the future. One of the main reasons why new ideas cannot be tested on production networks is the closed support from the vendors. Legacy network devices, such as IP routers or Ethernet switches, run both data planes and control planes. All control functions are implemented by vendors and cannot be modified or touchable. To overcome these obstacles to testing innovative ideas and redesigning the Internet architecture, SDN approach was proposed. SDN separates data and control planes with well-defined protocol. The control functionalities are taken out of the equipment and given to a centralized or distributed system, while retaining only data plane functionality on the equipment.

OpenFlow is one of SDN implementations, which is an initiative by a group of people at Stanford University as part of their clean-slate program to redefine the Internet architecture. Processing packets decisions are moved to the OpenFlow controller. That means the network is programmable in OpenFlow. Each OpenFlow-enabled switch performs packets forwarding based on the flow table. The flow table contains a set of entries with packet header fields, an action, and flow statistics. Each flow entry is associated with actions that dictate how switch handles matching packets. Thus, OpenFlow uses distinct entries of flow tables to achieve isolation among experiments.

2.2. Future Internet Testbeds

Global Environment for Network Innovation (GENI) [3], [8] is a US program funded by the National Science Foundation (NSF). It is an experimental facility designed to form a federated environment to allow networking researchers to experiment on a wide variety of problems in communications, networking, distributed systems, cyber-security, and networked services and applications with emphasis on new ideas. GENI will provide an environment for evaluating new architectures and protocols, over fiber-optic networks equipped with optical switches, novel high-speed routers, radio networks and computational clusters [3].

The GENI architecture can be divided into three levels, Physical substrate, User services, and GENI Management Core (GMC). Physical substrate represents the set of physical resources, such as routers, switches; User services represent the set of services that are available for the users in order to fulfill their research goals; GMC defines a framework in order to bind user services with underlying physical substrate. In order to implement this, it includes a set of abstractions, interfaces and name spaces and provides an underlying messaging and remote operation invocation framework.

For constructing a topology of multiple substrates, GENI proposed the Aggregate Manager to control its own domain. Each Aggregate Manager has a unique RSpec which defines its Substrate resources. These RSpecs are

represented as a topology description of the individual substrate. However, how to automatically discover a global perspective of substrate topology is not mentioned.

The OpenFlow in Europe: Linking Infrastructure and Applications (OFELIA) is another famous testbed, which is funded by the European Union as part of its FP7 ICT work program. The OFELIA project consortium is made up several academic partners, commercial organizations, and telecom operators. Its infrastructure facility consists of five different islands spread across the Europe. Each island will host different capabilities to offer different functionalities to the researchers.

OFELIA architecture is still under development. However the architecture will be based on OpenFlow technology [3]. Currently, OpenFlow switches topology can be discovered when these reside in the single controller. With the growing OpenFlow domains, the environment of multiple controllers is needed for load balance. However, there does not have any mechanism which automatically retrieves the topology among OpenFlow switches controlled by multiple controllers.

3. Design and Implement Future Internet Testbed on TWAREN

We explain how to design and implement the future-proof testbed with OpenFlow in this section. As mentioned in Section 1, we expect the proposed architecture not only supporting OpenFlow but also providing virtual switching services for cloud networking research. To accomplish these goals, we propose the architecture as shown in Fig. 2. There are three parts in our design: Services layer, Networking layer, and Resources manager.

A number of controllers comprise the controller pool in the Services layer [1]. We provide different types of controllers (e.g., standalone, virtual machines) for researchers to request. If researchers would like to use their own host machine as a controller, binding a public IP is the only constrain. We use FlowVisor [19], a network virtualization layer of OpenFlow, to support these external controllers. Because FlowVisor contains a mapping table, we can maintain the relation between controllers from external users and our OpenFlow switches. There are several servers in the Services layer, some of them are classified as Virtualized Servers for concurrently running multiple virtual machines and the other are categorized into Bare-metal Servers for performance-concerned experiments.

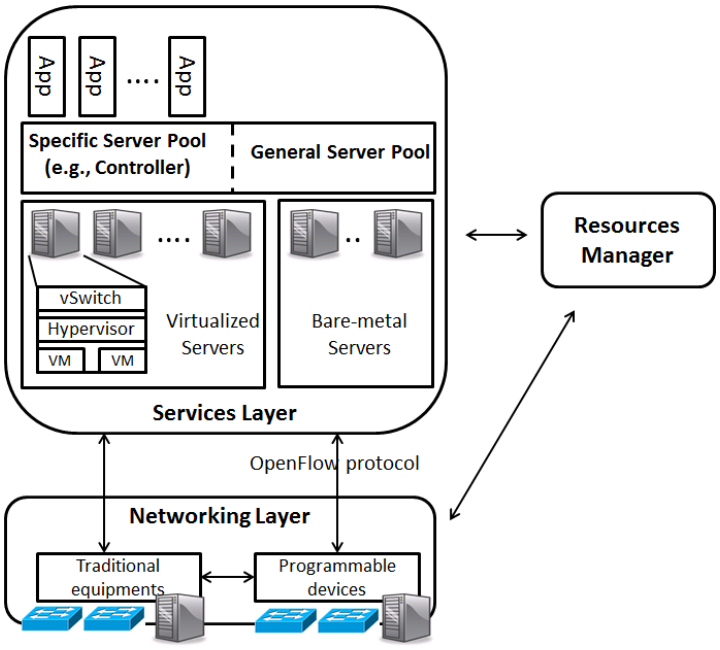

Fig. 2. Future Internet testbed architecture

For the Networking layer, we deploy legacy network equipments (e.g., switches, routers) and OpenFlow switches from different vendors including HP, Extreme, and PC with NetFPGA card. Since OpenFlow switches have to be operated at Layer 2 network, in this layer we provide hybrid solutions for extending our testbed smoothly. First, we use one of many TWAREN services, VPLS/VPN, which can connect multiple sites in the same local area network. This service is very useful for creating Layer 2 networks dynamically. However, there are some OpenFlow sites that cannot be applied directly to VPLS. For resolving this problem, we reserve several servers in the Service layer as tunneling servers in which software-based tunneling tools are installed (e.g., Capsulator [6]). About Resources manager, we use existing tools (e.g., OpenNebula, libvirt, virt-manager) to manage and control VMs in servers. It also maintains several services configurations, such as FlowVisor, tunneling, etc. We plan to develop a user interface for centralized management.

However, there are some OpenFlow sites that cannot be applied directly to VPLS. For resolving this problem, we reserve several servers in the Service layer as tunneling servers in which software-based tunneling tools are installed (e.g., Capsulator [6]). About Resources manager, we use existing tools (e.g., OpenNebula, libvirt, virt-manager) to manage and control VMs in servers. It also maintains several services configurations, such as FlowVisor, tunneling, etc. We plan to develop a user interface for centralized management.

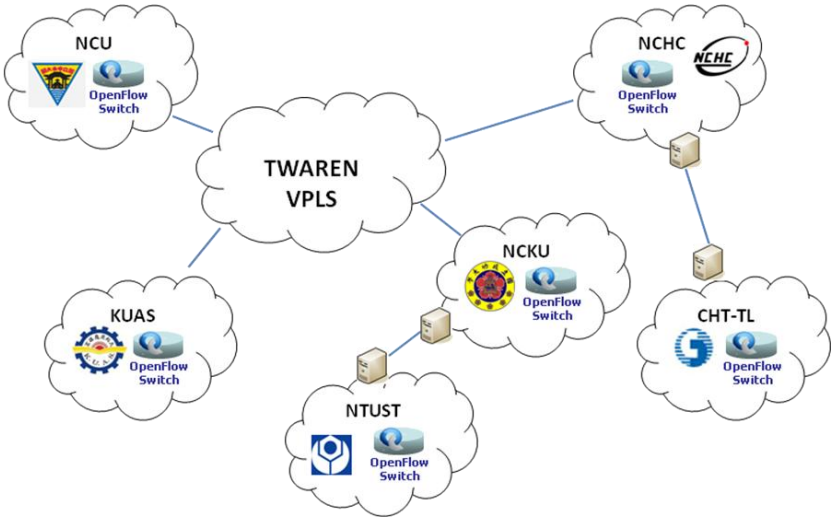

Fig. 3. Current OpenFlow connection in TWAREN research network

At the beginning of our project, two universities in Taiwan (e.g., NCKU, KUAS) participate in this Future Internet tesbed. Each site, including NCHC, is connected by Capsulator for operating at Layer 2 network. To deal with poor performance, we leverage VPLS service in TWAREN to provide a hardware-based tunneling. Many institutes that have interests in Future Internet research join our testbed, the current status is shown in Fig. 3.

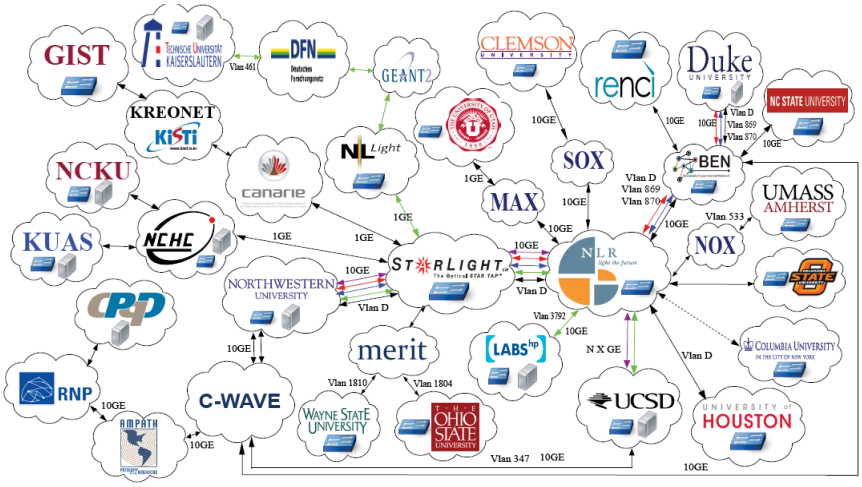

Fig. 4. Participating institutes of iGENI project [12].

In 2011, we joined iGENI [12] project by TWAREN international connection, as illustrated in Fig. 4. This will provide more real experiment network for our testbed.

4. Management Functionalities on TWAREN Testbed

In this section, we briefly describe some network management modules running on our testbed, which developed and resided in different aspects including inter-domain topology and virtual machines management.

4.1. Management of Inter-domain Connection

As mentioned previously, OpenFlow separates data and control plane. The only responsibility of OpenFlow switch is to forwarding received packets according to its flow table. Other complex works (e.g., routing decisions) are taken by controllers. Each OpenFlow switch has its own controller, so directly connected switches can be easily perceived by controllers. In addition, LLDP (Link Layer Discovery Protocol) packets are exchanged between any two OpenFlow switches to figure out neighbor switches. With these links information, controller can discover the topology in its controlled domain. As Fig. 5 shows, there are four OpenFlow switches (e.g., OF_A, OF_B, OF_C, and OF_D) residing in two different domains. $Controller_1$ for $Domain_1$ is responsible for OF_A and OF_B while OF_C, and OF_D are taken by $Controller_2$ in $Domain_2$. These two domains are directly connected by the link between OF_B and OF_C in Fig. 5. However, two controllers do not specify these links in their discovered object lists. That may cause the complexity of management and link provisioning.

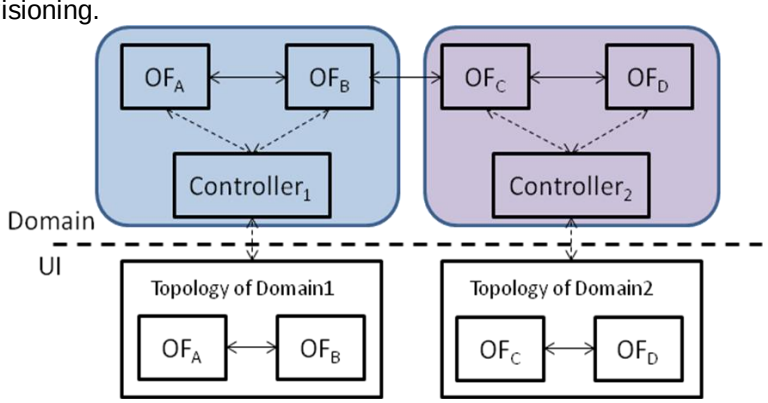

Fig. 5. Original inter-domain topology

For solving this problem, we proposed a mechanism to insert additional information into LLDP messages. In addition, we modify some applications in NOX for retrieving links among inter-domains. The full links information of our proposed solution is illustrated in Fig. 6.

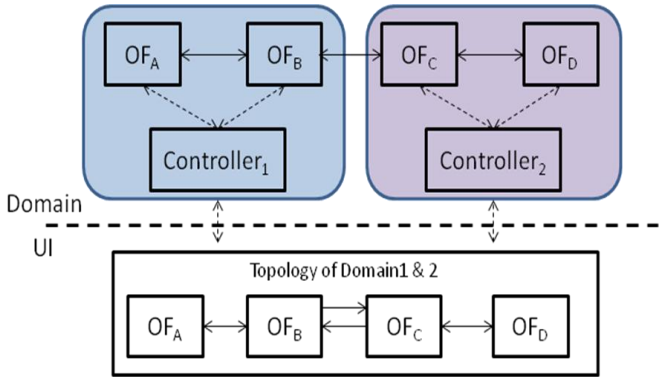

Fig. 6. Inter-domain information after applying proposed mechanism.

In general, LLDP information is sent by network devices from each of their interfaces periodically. A LLDP frame, as shown in Fig. 7, is composed by a series of LLDP Data Units (LLDPDUs). Each LLDPDU is a type-length-value (TLV) structure. There are four mandatory TLVs and zero or more optional TLVs in every LLDPDU.

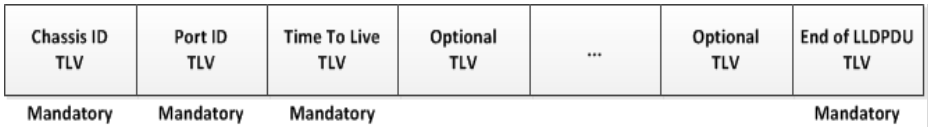

Fig. 7. LLDPDU format

As mentioned above, we can obtain topology information from devices but they must be resided in the same controller's domain. Hence, our main goal is to combine all topology information from different controllers. Through our experiments and observations, we found LLDP packets are also exchanged between any two directly connected devices. However, LLDP packets across different domains will be eventually dropped by receiving controller because they come from another domain. Since LLDP frame reserves optional TLVs to be extended by vendors or users, we add an optional TLV which contains controller information (e.g., IP and port) into generating application in NOX controller. Then, we modify the dropping policy and stored the received LLDP packets from different controller domains. Therefore, we aggregate all received information to build an overall topology. Fig. 8 shows the operations and relationship in modules of our mechanism.

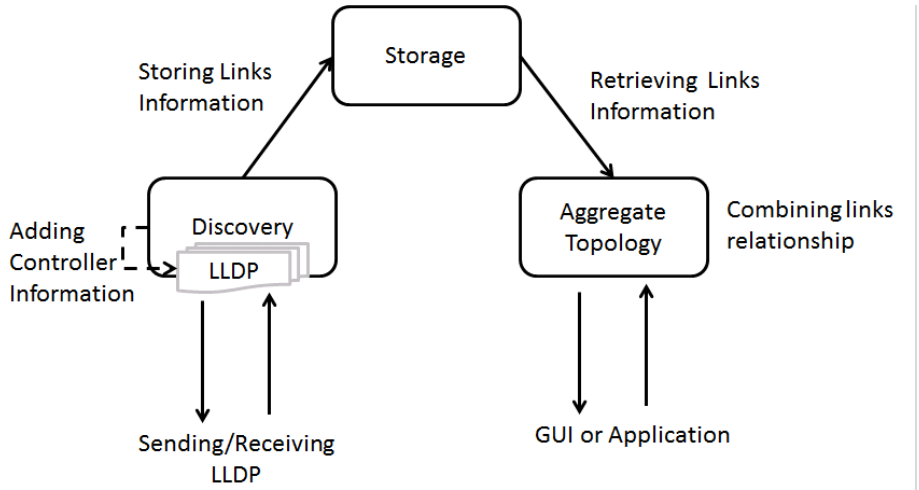

Fig. 8. Modules relationship in our mechanism

To verify the proposed mechanism in real OpenFlow network, we deploy it in three different domains including NCHC in Taiwan, NWU (Northwestern University) in the U.S., and CRC (Communications Research Centre) in Canada. Fig. 9 shows the links topologies of this experiment [11].

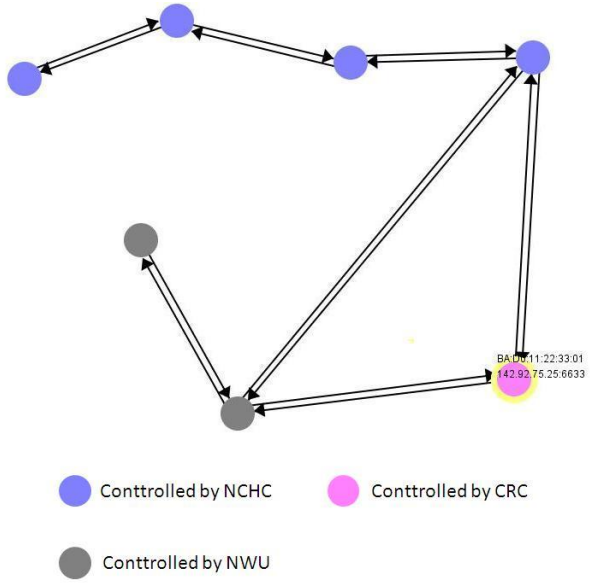

Fig. 9. Auto-discovery applies in a real multi-controller environment.

Since our proposed mechanism adds extra domain information in original LLDP packets, we quantified its processing overheads including CPU usage,

allocated memory, packet size, and processing time. We compare our proposal results against the original NOX controller. We setup two Linux hosts (Quad-code 2.53GHz, Xeon CPU, 4GB RAM, 1Gbps NIC), one to be an OpenFlow controller and the other uses Mininet to create the network topology which has 4 linear-connected OpenFlow switches.

Table 1. Comparison between original discovery application and our proposal

Mechanisms	CPU (%)	Memory (MBytes)	Packet size (Bytes)	Proc. Time (sec)
Original application	1%	23	60	1.5974
Our proposal (persistence version)	1%	23	60	7.5101
Our proposal (on the fly version)	1%	23	60	1.6102

Each application in OpenFlow controller is event-driven. When an OpenFlow switch receives packets, it will pass through all started applications and trigger their Packet_In event. For each mechanism, we generate 100 LLDP packets to measure the performance results shown in Table 1. There are no differences in CPU usage and allocated memory. The format of LLDP has only 14 bytes, but most network equipments will send it in 60-byte packet by padding the last few bytes. Although our mechanisms add extra information in original LLDP packet (e.g., Optional TLV), the size of modified LLDP packet is still less than 60 bytes. Therefore, the LLDP packet size is also no different from the original one. In order to discover multi-domain topology, we add a procedure to combine topology information received from each neighbor domain. For measuring overhead of our proposals, we define the processing time which represents a period starts from processing an incoming LLDP packet to storing its recognized information in controller. In our first proposal – persistence version, we had a poor performance than origin because it stored the topology information into persistent file for interoperating with multiple program languages application and recording current topology in our system. Furthermore, we developed another version (e.g., on the fly version) to solve this performance issue. It uses a compatible data structure instead of file and creates a thread to periodically write the topology information into file. There reduces much time when processing LLDP packets.

Considering scalability, in [21] they mentioned on an eight-core machine with 2GHz CPUs, NOX controller handles 1.6 million requests per second with an average response time of 2ms. We add additional topology information without affecting the original LLDP packet size and the time of processing LLDP packets is nearly same as the origin. Therefore, our proposals can have the same performance in real environment.

4.2. Management of Inter-domain Connection

In the past, standalone servers connect to physical switches directly. Many management functions, such as access control, port mirroring, and so on, are provided by network equipments. When moving to cloud, servers are replaced by VMs and reside in host machines. The network connections between servers and network devices have transferred to VMs and virtual switches. In this management module, we focus on integrating packet monitoring and network virtualization into our testbed, we call it VM manager module.

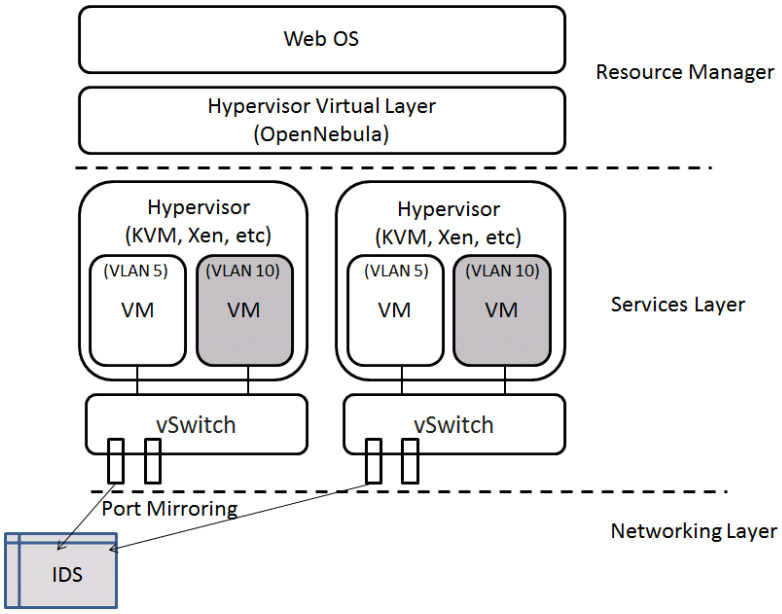

Fig. 10. VM Manager module

The common way for separating various users is to assign distinct ranges of private IP addresses. This mechanism can work properly in network connectivity, but all users will be resided in the same broadcast domain. That means users can access any virtual machines if they modify their own VM IP address to specific IP ranges. With increasing the number of VMs per host, this issue causes the difficulties of network management and security in cloud environment. Open vSwitch [16] is an open source tool fitting our requirements to resolve this problem. It implements 802.1q VLAN that can isolate different broadcast domain to keep inter-VM security. In addition to VLAN features, it also supports NetFlow, sFlow, and RSPAN for network visibility.

VM Manager module is shown in Fig. 10. It crosses the three layers of our testbed. In Resource Manager Layer, we use WebOS for our user interface and OpenNebula for hypervisor manager respectively. For Services Layer,

we set up several servers for deploying VMs. Each of them is installed Open vSwitch for virtual network and managed by OpenNebula. We implement some integrated programs to bind OpenNebula and Open vSwitch smoothly. Besides, we use Layer 2 technology, VLAN, to separate different VM users. But the valid VLAN range is from 1 to 4095, it is the limitation of our VM users in this status. We still develop and integrate other approaches to solve this limitation. Each of virtualized servers contains a management port which is connected to external analysis system for monitoring abnormal traffic among VMs. This integrated mechanism provides security capabilities in our VM users.

The VM resource allocation is an important issue for performance transmissions. In general, users require multiple VMs which are often arranged on the same host. Our VM manager module has different policies (e.g., Round-robin, Keep-in-one-host, and Random) to allocate multiple VMs requested from a single user. For suiting different types of VM services and allocating efficiency, we measure the performance by different packet sizes to provide allocating policy in our manager module. We setup two hosts (Quad-code 2.53GHz, Xeon CPU, 16GB RAM, 1Gbps NIC), each of them running 8 VMs and the measurement tool is “iperf”. Random choosing two VMs on each hosts (e.g., one is server and the other is client) to be the same host group. Then, we random choose one VM from other six VMs on each host respectively, and assign these two VMs as the different hosts group. Other VMs (e.g., five VMs in each host) run the same application which has ten megabytes in memory usage and one percent of CPU time. Our experiment results are shown in Fig. 11, larger packet size increases throughput because it generates less number of packets when transferring the same data frame. Each packet has fixed header, thus fewer packets will have less overhead (e.g., the source and destination addresses). We can also observe that the throughput for assigning two or more VMs on the different hosts is exceeds than arranging them on the same host as the packet size exceeds 32K. In this situation we find two hosts need using more memory to buffer and process packets when these two VMs on the same host. However, the memory usage will be shared and reduced if these two VMs are on different hosts. Therefore, we think packet size will be a factor in allocating VM resources. According to this experiment result, we extend the default VM allocation with a network-oriented policy which considers network factors (in currently, we just define default transferring packet size) to assign VM resources in our manager module.

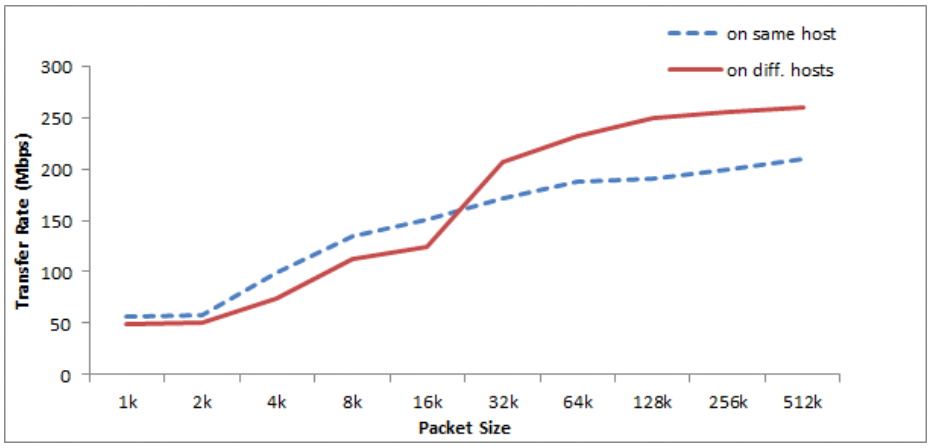

Fig. 11. Throughput of different VM assignment approaches.

5. Testbed Performance Result

As described the Networking layer of our testbed architecture in Section 3, we built two different mechanisms for network connection. In this section, we will do some performance experiments to measure the overhead of these two mechanisms in our testbed. In addition, VM Manager is also an important module in our testbed. We will compare its performance against original OpenNebula in this section.

In Table 2, NCHC-TN and NCHC-HC are the southern and the northern departments of our center respectively. The distance between NCHC-HC and NCHC-TN is around 230 km. Another site of our experiment, NCKU, is a university in southern Taiwan. The distance to NCKU from NCHC-TN is around 20 km while.

Our latency experiments used 100 64-byte packets. The first row in Table 2 shows the result. We also measured one-direction TCP throughput by different sizes of packets. For each case, we ran 20 30-second trials. The results show that VPLS technology is significantly faster and more efficient than the mechanism with tunneling software.

Table 2. Micro-benchmarks for TWAREN Future Internet testbed overheads

Cases	VPLS		Tunneling Software	
	NCHC-TN to NCHC-HC	NCHC-TN to NCKU	NCHC-TN to NCHC-HC	NCHC-TN to NCKU
RTT (ms)	3.512	0.895	5.822	2.873
Throughput (1M packet) (Mbps)	461	815	75.7	89.2
Throughput (10M packet) (Mbps)	473	831	76.5	89.3
Throughput (100M packet) (Mbps)	472	838	75.5	87.6

For comparing network throughput between VM Manager and original OpenNebula, we set up two Linux hosts which create four VMs in each host. Each VM has one-core 2.53GHz CPU, 512MB memory, and 1Gbps NIC. For the first trial, we compared the VM TCP throughput of VM Manager and original OpenNebula on the same host. We chose one host and divided its VMs into two groups. One VM of each group is running iperf server and the other is client. In this experiment, each group received a result and we chose minimum one of them to be TCP throughput. The Fig. 12 shows the first trial result. Clearly, VM Manager outperforms original OpenNebula at any sizes of transferring packets.

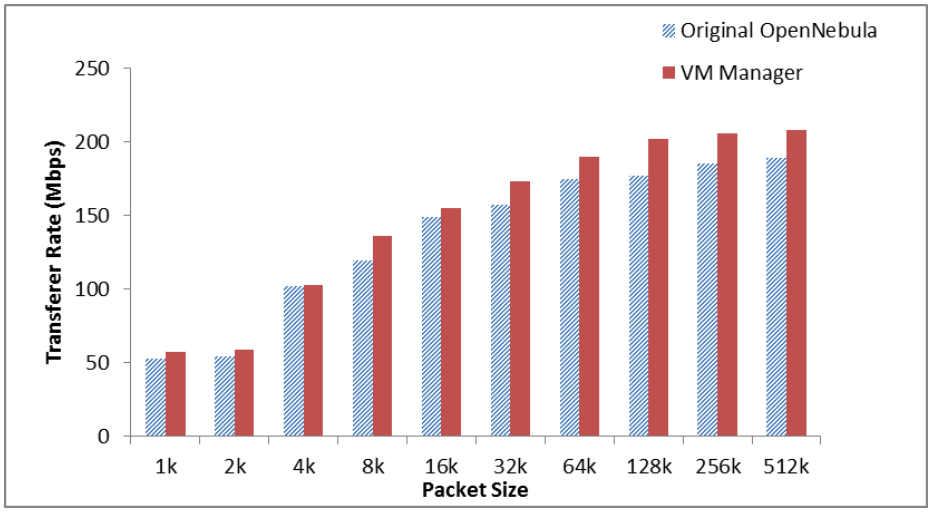

Fig. 12. Throughput with VMs on the same host

The second trial, we consider the network performance when VMs are arranged on different hosts. We classified two hosts into two groups, one host make its all VMs be iperf servers and all VMs of the other host are iperf clients. The experiment result is shown in Fig. 13, which appears VM Manager outperforms original OpenNebula by 19% in average.

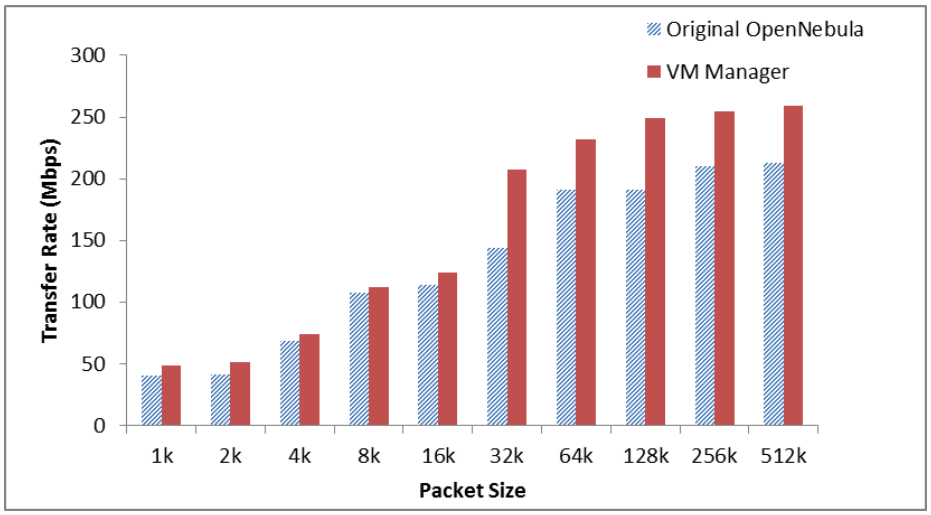

Fig. 13. Throughput with VMs on the different hosts

6. Conclusion

In this paper, we propose and create a Future Internet testbed which has the capabilities for programmable network and cloud. This testbed is deployed over TWAREN Research Network. We experiment and verify different research activities on this tesbed, including Future Internet and cloud. In our future work, we will keep developing more innovative functions for Future Internet. It will be useful to build and maintain a cross organization and a large scale multinational Future Internet platform. We believe the TWAREN Future Internet testbed opens up a new environment in Taiwan for network research. It enables us not only to design new thoughts, but also to solve and verify current issues in real network.

References

1. Bădică, C., Budimac, Z., Burkhard, H., Ivanović, M.: Software Agents: languages, tools, platforms. Computer Science and Information Systems, Vol. 8, No. 2, 255-296. (2011).

2. Bellovin, S. M., Clark, D. D., Perrig, A., and Song, D.: A Clean-Slate Design for the Next-Generation Secure Internet. GENI Design Document 05-05. (2005).
3. Belter, B., Campanella, M., Farina, F., Garcia-Espin, J., Jofre, J., Kaufman, P., Krzywania, R., Lechert, L., Loui, F., Nejabati, R., Reijs, V., Tziouvaras, C., Vlachogiannis, T., and Wilson, D.: Virtualisation Services and Framework – Study. Formal report from European Commission. (2012).
4. Bianco, A., Birke, R., Giraudo, L., and Palacin, M.: OpenFlow Switching: Data Plane Performance. Communications (ICC), 2010 IEEE International Conference, pp. 1-5. (2010).
5. Cameron, D.: Internet2: The Future of the Internet and Next-Generation Initiatives. Computer Technology Research Corp. (1999).
6. Capsulator, http://www.openflow.org/wk/index.php/Capsulator.
7. Fairhurst, G., Collini-Nocker, B., and Caviglione, L.: FIRST: Future Internet: A Role for Satellite Technology. IEEE International Workshop on Satellite and Space Communications (IWSSC). (2008).
8. FEDERICA: Federated E-infrastructure Dedicated to European Researchers Innovating in Computing network Architectures, http://www.fp7-federica.eu/.
9. GENI: Global Environment for Network Innovations, http://geni.net.
10. Greenberg, A., Hjalmtysson, G., Maltz, D. A., Myers, A., Rexford, J., Xie, G., Yan, H., Zhan, J., and Zhang, H.: A Clean Slate 4D Approach to Network Control and Management. ACM SIGCOMM Computer Communication Review, Vol. 35, Issue 5. (2005).
11. Huang, W. Y., Hu, J. W., Lin, S. C., Liu, T. L., Tsai, P. W., Yang, C. S., Yeh, F. I., Mambretti, J. J., and Chen, J. H.: The Implement of Automatic Network Topology Discovery System in Future Internet across Different Domains. 26th International Conference on Advanced Information Networking and Applications Workshops (WAINA). (2012).
12. iGENI: International Global Environment for. Network Innovations, http://groups.geni.net/geni/wiki/IGENI.
13. Kim, D. Y., Mathy, L., Campanella, M., Summerhill, R., Williams, J., Shimojo, S., Kitamura, Y., and Otsuki, H.: Future Internet: Challenges in Virtualization and Federation. Fifth Advanced International Conference on Telecommunications, (AICT), pp.1-8. (2009).
14. Lee, J., Kang, S., Lee, Y., and Lee, J.: A Study on the Future Internet Requirement and Strategy in Korea. 10th International Conference on Advanced Communication Technology (ICACT), Vol. 1, pp. 627-629. (2008).
15. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J.: Openflow: enabling innovation in campus networks. SIGCOMM CCR, Vol. 38, no. 2, pp. 69-74. (2008).
16. Open vSwitch, http://www.openvswitch.org/.
17. Pettit, J., Gross, J., Pfaff, B., and Casado, M.: Virtual Switching in an Era of Advanced Edges. 2nd Workshop on Data Center – Converged and Virtual Ethernet Switching (DC-CAVES), ITC 22. (2010).
18. Pfaff, B., Pettit, J., Koponen, K.A.T., Casado, M., and Shenker, S.: Extending networking into the virtualization layer. Proceedings of the ACM SIGCOMM HotNets. (2009).
19. Sherwood, R., Gibb, G., Yap, K. K., Apenzeller, G., Casado, M., McKeown, N., and Parulkar, G.: FlowVisor: A Network Virtualization Layer. Tech. Rep. OPENFLOWTR- 2009-1, OpenFlowSwitch.org. (2009).
20. Stuckmann, P. and Zimmermann, R.: European research on future Internet design. IEEE Wireless Communications, Vol. 16, Issue 5, pp. 14-22. (2009).

21. Tootoonchian, A., Gorbunov, S., Ganjali, Y., Casado, and M., Sherwood, R.: On Controller Performance in Software-Defined Networks. 2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE). (2012).
22. TWAREN Research Network, http://www.twaren.net/.

Jen-Wei Hu received the B.S. degree in Applied Mathematics from National Chung Hsing University, Taiwan, in 2001, and the M.S. degree in Computer Science and Engineering from National Sun Yat-sen University, Taiwan, in 2003. Currently, he works as an Assistant Engineer in the Network and Security Division of National Center for High-Performance Computing, Taiwan. His current research interests include Software-Defined Networking, Networking in data centers, and Multipath transmission.

Chu-Sing Yang is a Professor of Electrical Engineering in the Institute of Computer and Communication Engineering at National Cheng Kung University, Tainan, Taiwan. He received the B.Sc. degree in Engineering Science from National Cheng Kung University in 1976 and the M.Sc. and Ph.D. degrees in Electrical Engineering from National Cheng Kung University in 1984 and 1987, respectively. He joined the faculty of the Department of Electrical Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan, as an Associate Professor in 1988. Since 1993, he has been a Professor in the Department of Computer Science and Engineering, National Sun Yat-sen University. He was the chair of the Department of Computer Science and Engineering, National Sun Yat-sen University from August 1995 to July 1999, and the director of the Computer Center, National Sun Yat-sen University from August 1998 to October 2002. He was the Program Chair of ICS-96 and Program Co-Chair of ICPP-2003 and MTPP-2010. He joined the faculty of the Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, as a Professor in 2006. He participated in the design and deployment of Taiwan Advanced Research and Education Network and served as the deputy director of National Center for High-performance Computing, Taiwan from January 2007 to December 2008. His research interests include future classroom/meeting room, intelligent computing, network virtualization.

Te-Lung Liu received the B.S. and Ph.D. degrees in computer science from the National Tsing Hua University, Hsinchu, Taiwan, R.O.C., in 1995 and 2002, respecively. He is currently a Research Scientist in National Center for High-Performance Computing, Tainan, Taiwan, R.O.C. He is also a Team Member of the Taiwan Advanced Research and Education Network (TWAREN) and now working on OpenFlow Testbed in Taiwan. His current research interests include Software-Defined Networking, Future Internet, optical networks, and network design.

Received: November 14, 2012; Accepted: April 05, 2013