A Survey on Software-Defined Wide Area Networks

Xugang Wu, Kai Lu, and Guoliang Zhu
School of Computer, National University of Defense Technology, Hunan, 410073, P.R. China
Email: xugangwu95@gmail.com; kailu@nudt.edu.cn; zhuguoliangcs@gmail.com

Abstract—Better cost effectiveness of Software-Defined Network(SDN) is motivating enterprises to adopt it in Wide Area Network(WAN). The idea is then called Software-Defined Wide Area Network(SD-WAN). In this paper, first we review the architecture of SDN, which provide theoretical support for SD-WAN. Then we analyze three typical SD-WAN realizations, which are Google's B4, Microsoft's SWAN and Software-Defined Internet eXchange Point respectively. Finally, we make a conclusion and discuss future works.

Index Terms —SD-WAN, B4, SWAN, SDX, survey

I. Introduction

Nowadays, a shifting paradigm of data intensive applications such as social networks, video and image processing, cloud computing has been widely adopted. However, traditional WAN cannot fully saturate network bandwidth and is hard to deploy due to its operational complexity. Moreover, due to lack of knowledge of network traffic, traditional WAN often disobeys SLAs. Cloud computing is driving WAN towards new direction. For the first generation of WAN, connection is the priority. The user only concerned about whether the network is reachable. The second generation of WAN works on richness of network services, multi-service processing capabilities. At present, WAN focuses more on user experience, which leads to the next generation of network evolution. SD-WAN solution is a specific third-generation WAN solution, which is now leading the revolution of the traditional WAN architecture.

With the gradual migration of business to the cloud, WAN has a higher bandwidth demand so that each company's annual link rental costs continue to increase. However, the traditional WAN utilization is always maintained at 30% -40% according to Google. Therefore, companies need to release the potential of bandwidth. At the same time, simplifying the deployment process, dealing with the cloud era of fast on-line business trends and having a better support for Internet of Things require WAN to be more programmable, easier to deploy and more reliable.

Manuscript received November 29, 2017; revised May 2, 2018.
This work was supported by The National Key Research and Development Program of China (No. 2016YFB0200401).
Corresponding author email: xugangwu95@gmail.com
doi:10.12720/jcm.13.5.253-258

To explain the advantages that SD-WAN can bring to us, we must first understand what is SD-WAN. As Gibson suggested in "Software-Defined-WAN-for-Dummies" [1], "SD-WANs use centralized controllers without relying on interactions with underlying provider transport solutions. De-coupling the data plane from the control plane, replacing internetworking protocols with APIs, and building policies based on application metrics rather than network metrics make this possible".

In this paper we first give a hierarchical view of the SDN architecture. Then we introduce three typical deployments of SD-WAN in industry, including Google's B4, Microsoft's SWAN and the Software Defined Internet eXchange Point(IXP) called SDX, focusing on the design considerations and the spotlights for these SD-WANs. Finally, we discuss the future work of SD-WAN concludes.

II. SDN Architecture

Fig. 1 is a figure from Kreutz's paper called "Software-Defined Networking: A Comprehensive Survey". This figure can clearly describe the general architecture of SDN in three different perspectives, (a) planes, (b) layers, and (c) system design. Based on this figure, we will introduce the SDN Architecture layer by layer.

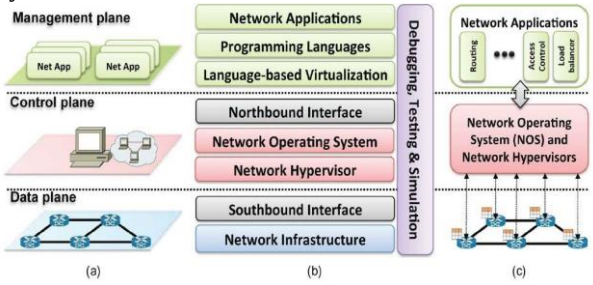


Fig. 1. Architecture of the system [2]

A. Data Plane

Packet forwarding is the basic functions of data plane. Therefore, researches on data plane focus on two fields: one is the utilization and organization of Network Infrastructure like OpenFlow switches, and the other is the Southbound Interface, which is the OpenFlow protocol in our discussion.

1) OpenFlow switch

Researches on OpenFlow switch can be divided into two parts. The first is the hardware switches, aiming at adapting the switches' hardware to achieve high performance.

Some researchers try to use CPU/FPGA in switches. Although they all try to add computing units into switch, the functionality of these units is different. In Lu *et al.*'s paper [3], they propose a new way to handle the traffic in CPU. They make use of the technique in handling traffic in control plane and put a CPU into the switches so that they can avoid limitations like limited forwarding table and shallow buffer. With the help of CPU unit, OpenFlow switches are enabled to tackle the explosive traffic situation. In another paper [4], the authors point out that we can try to make the rule-match records a stream and use it to replace traditional counters. With the help of this stream, we can use a programmed CPU to calculate the records since ASIC-based counters are not flexible enough to compute novel metrics. FPGA is also a nice accelerator due to its programmable feature. Sivaraman's team propose that it will be helpful to add a small FPGA to switches to make the switches more powerful and more flexible [5].

Others try to add some middle components to accelerate the process of match-up, which is one of the most time-consuming task in OpenFlow switches. In Jose et al.'s paper [6], they propose a compiler for OpenFlow switching chips, which tackle the mapping process from logical lookup tables to physical tables, while meeting dependencies between them. Yan and Katta use a different method. They try to make use of cache to accelerate the match-up process to promote the performance of switches [7].

The second part is software openflow switch. Since the protocol keeps improving and changing, researchers and engineers try to implement a software based, OpenFlow-support switches in virtualized server environments. There are two popular software Openflow switches. One is Open vSwitch [8], which is a multilayer virtual switch. The other one is a further version of Open vSwitch whose behavior is customized using P4 [9].

2) Southbound interface

Referred to Southbound Interface, we cannot avoid OpenFlow protocol, which is originated from a project named Clean Slate [10]. The original goal is to design a protocol to help switch separating the data plane and control plane as well as use a controller to manage and configure switches (data forwarding device). With the development of SDN, there are several problems researchers focus on.

One is the load balancing between the controller and switches. Since all the traffic will be calculated in the centralized controller, the controller can encounter heavy peak traffic at any time, which makes the network less expandable. Yu's solution, named DIFANE, puts selective load on switches so as to make the network more scalable [11].

Another is about the visibility and abstraction, which are the points most interface-concerned work should take into consideration and there is also much related work on these two fields [12].

B. Control Plane

Control plane is the bound between the application layer and the physical data forwarding layer. The role it plays in software-defined network is similar as the role Operating System plays in a traditional computer. Researchers and engineers attempt to design a network operating system to support SDN, which provides a high-level programmatic interface for those application programmers as well as manage the low-level configurations [13], [14]. When we do survey about control plane, we find out that Northbound interface and controller are usually taken as a whole. Most systems will consider about the state consistency, scalability, flexibility and security. In Table I, we introduce some of the control systems and their features.

TABLE I: EXISTING CONTROLLERS AND THEIR FEATURES

System name	Implementation	Feature
NOX [13]	C++	NOX is the first Openflow controller to provide a uniform and programmatic interface for the network.
Onix [14]	C++	Based on NOX, Onix tries to have a common and centralized control on the network and focus on the general API for those application programmers.
Hyperflow [15]	C++	Hyperflow can be considered as an application of NOX which enables the interconnect of two networks controlled by two different controllers.
Maestro [16]	Java	Maestro pays attention to the friendly interface for programmer and exploits parallelism to make the system more scalable.
Beacon [17]	Java	Based on Floodlight, it mainly focuses on the programmer interface, managing applications and efficiency.
Yanc [18]	Framework	Yanc expresses the network as a file system and use a standard I/O interface to communicate with application, which will act as a network operating system.
Covisor [19]	Framework	Covisor is the first network system to use multiple controllers to control the same network.
Beehive [20]	Framework	Beehive works on simplifying the control logic and making the management easier.

C. Application

1) Monitoring and management

As is discussed before, one of the SDN's benefits is that it facilitates network manager to monitor and measure the network, which are the base for traffic engineering. Researchers mainly discuss how to monitor the network state and how to make it efficient. In Yu's paper [21], he suggests a traffic measurement architecture called OpenSkecth, which can separate the measurement data plane from the control data plane to achieve a satisfactory trade-off between generality and efficiency. In another paper, he proposes a management tool named FlowSense

[22], which can monitor the network utilization with no extra cost. However, any measurement should face a problem: the trade-off between resources and accuracy, which in this case is discussed by Moshref *et al*. [23].

2) Traffic engineering

Traffic engineering is the most important part of application layer since it can promote the traffic efficiency, which can show the power of SDN. Mohammad et al. propose a schedule system called Hedera [24], which can utilize the network resource dynamically. The power consumption is also unavoidable problem and in Heller's paper, their manager aims at power saving through switch on/off the switches based on the network load [25].

3) Middle box

With the help of SDN, the implementation of middle boxes become much easier. For example, in his paper "OpenFlow-Based Server Load Balancing Gone Wild" [26], Wang points out that they can adjust the load balancing policy automatically according to the network traffic condition. As far as security is concerned, Hu et al. propose a robust firewall for SDN called FLOWGUARD, which is an intelligent firewall making full use of the granularity, visibility and flexibility brought by SDN [27].

III. Analysis of Typical SD-WAN

As discussed above, SD-WAN can bring lots of profits such as high efficiency, good flexibility and robust security for the enterprise. Therefore, not only those companies with large-scale data center network but also those Internet Service Providers(ISP) are trying to make use of SD-WAN and proposing their solution. In the following section, we will first introduce two SD-WAN networks for inter-data center. One is Google's B4, the other is Microsoft's SWAN, after which we will introduce how SD-WAN is used in Internet Exchange Point.

A. Google's B4

1) Motivation

Google pioneers in leveraging SDN technique in inter-data center network. They find out that in some case the utilization of network is just about 30% to 40%, which means that there is a huge waste and unnecessary payment for their network resources. Therefore, Google presents B4 [28], which is a SD-WAN that can make full use of their network resources.

2) Architecture

The architecture contains three layers: The first layer is Central Traffic Engineering (TE) Sever, which runs the Traffic Engineering Algorithm to control the network. The second layer is Network Control Server(NCS), which is the controller for every data center. OpenFlow Controller (OFC) runs on it, using Paxos to elect a master and make the others remain as hot standby. The bottom layer is those physical switches, running OpenFlow Agent (OFA), which receives instruction from OFC and write the rules into the flow-table of the switches.

3) Traffic engineering

The highlight of B4 is its TE Algorithm. The ideal way to optimize the traffic is to control the traffic according to the type of services. However, it will require a huge table to mark the information. Therefore, Google makes use of QoS to identify the priority of each service and make a tuple {source site, dest site, QoS} to build the Flow Group (FG). After the flow is divided into groups, B4 uses its bandwidth function to calculate the bandwidth allocated to this flow. The function is based on max-min objective function and the fair share dimension. The controller will take the result of the bandwidth function and the topology of the network into consideration and then send instruction to OFA to reconfigure the flow-table.

4) Composing traditional routing and TE

Since deploying TE in WAN is an innovative attempt for them, Google make it a gradual progress. They run TE and traditional routing system parallel and make the priority of TE higher. In this way, SD-WAN can take effect gradually and more and more flow will transfer from traditional routing to TE. At the same time, if the TE system down, traditional routing will keep the network running normally.

B. Microsoft's SWAN

Considered as one of the most critical infrastructure for providers of online services, Inter-data center wide area network shows need of high capacity and low latency. However, the efficiency of traditional WAN is not high enough due to the lack of coordination of services and distributed resource allocation. To solve this problem, Microsoft presents their SD-WAN called SWAN [29], which can globally coordinate the sending rates of services as well as centrally allocating the network path. In this section we will introduce what SWAN has done and the main feature of this network system.

1) Service-based traffic control

Traffic Engineering is the most important algorithm for SD-WAN. SWAN proposes a solution that can control the sending rates based on the types of the services.

They divide the applications into three kinds: Interactive, Elastic and Background. Interactive services are sensitive to loss and delay since they are in the critical path of end user experience. Elastic services require less in latency but still need timely delivery. Background services work for maintenance and provisioning activities, which will be bandwidth hungry and latency insensitive.

Based on this classification, SWAN allocates bandwidth for services while preferring the shorter latency for services with higher priority. Interactive services will be sent like today while the other services' traffic will send their request to the controller and the controller will compute and update the forwarding state of switches.

2) Atomic reconfiguration of the switches

To control the traffic or adapt to the changes in topology of the network, controller needs to update the network forwarding state. However, lacking WAN-wide atomic

reconfiguration will lead to transient congestion even if the initial and final states are compatible. To avoid this kind of congestion, SWAN propose a congestion-free plan by leaving a scratch capacity (0-50%) and used its strategy to guarantee no congestion with minimal steps.

Furthermore, to avoid waste of the scratch capacity, SWAN allocates it to background traffic, which requires high bandwidth but not sensitive to latency.

3) Making full use of the limited number of rules

Since the number of rules in switch is limited, the network capacity is restricted. As is the scalability of the network. Motivated by the phenomenon that working set is always a subset of the total tunnels, SWAN automatically works out the best tunnel set using their allocation method based on linear program (LP). Hence it can configure the rules of the switches dynamically, which makes the best of the limited rules.

C. Software Defined Internet Exchange Point-SDX

Internet Exchange Point(IXP) is physical points where multiple network meet to exchange traffic and BGP routes. Different from Inter-data center WAN, IXP focuses more on high scalability as well as the ability to become a flexible network with a variety of interface speeds, advanced management capabilities, and high levels of automation. In this section we will introduce a Software Defined IXP called SDX presented by Gupta et al. [30].

1) Existing BGP problems

Traditional way to exchange traffic between multiple network is BGP network. However, with the development of network and the flourishing of cloud computing, the shortage of traditional wide area network emerges. First of all, BGP routing depends only on the IP prefix of destination, which makes it lacking in flexibility. What's worse, BGP routing can only view the network in neighbor and influence over direct neighbor, which make it difficult for manager to achieve global optimization. Last but not lease, BGP routing has indirect expression of policy, which will make management complicated.

2) Four challenges to implement SDX

On their way to build an IXP based on Software Defined Network(SDN), SDX faces four challenges.

The first challenge is to fit for different applications. In case of SDX, they propose a strategy called Application-specific peering, which can let two neighbor ASes exchange the traffic of certain applications.

Moreover, SDX's ASes have inbound policy to decide how the traffic enters it should be dealt and how can other ASes reach them, which is much more flexible than traditional BGP protocol. Besides these, SDX can also provide Wide-area server load balancing simply by rewriting the destination location based on fields of the packet headers to realize load balancing. There is another functionality make SDX powerful. Its software-defined feature will make it easy for network constructors to build their middle box without applying complicated hardware.

The second challenge is to design a northbound interface to provide a friendly abstraction to those application programmers. SDX creates virtual switches at all ASes' border and use these virtual switches to implement the traffic control and middle box functions. In this way they create a uniform programing interface which can interact directly with the virtual switch. This interface also makes it possible for the switches define policies similar to BGP protocol, which enable SDX to work along with the existing BGP WAN.

The third challenge is to make sure that SDX scales with the network expansion. To improve network efficiency, they propose three methods. First, based on equivalent replacement rules, SDX transforms the policy and make it simpler to lessen the burden of controller. Also, SDX groups the packet with same forwarding behavior into a Forwarding Equivalence Class (FEC) and install a minimum set of forwarding rules for each FEC. Finally, by optimizing the initial compilation and incremental updates, SDX can minimize the computation for the controller, which is of vital importance for scalability of SDX.

The last challenge is to make the process of deployment easy. To solve it, SDX provides a version that can be deployed using virtual containers in Mininet so that it can be deployed easily by using the virtual technique.

IV. Future Work

As is introduced in Section III, companies are making effort to bring SD-WAN into use since it is a promising future of WAN. In this section, we will summarize their contributions and propose the future work to do.

A. Scalability

Different from LAN, the burden of the WAN's controllers will be much heavier. This implies that the controller should have the ability to handle huge computation amount in order to make the network scalable. Distributed controller and parallel computing can be a possible solution for it.

B. Efficiency

One of the most important advantage of SD-WAN is to improve the utilization and efficiency of the network. To achieve this, there are three methods. The first one is to optimize the Traffic Engineering Algorithm. The second one is to optimize the logical design of the control plane. The third one is to design better and more suitable hardware for data plane.

C. Update Congestion

One existing problem in SD-WAN is that when controller updates the flow-table of the switches, sometimes it will cause a congestion due to the limitation of a typical link. Microsoft's SWAN proposes a solution, but we believe there should be more solutions to resolve the congestion such as some logical restriction and so on.

D. Unification of North API

As is discussed in Section 2.2, now there is a large variety of control plane solution and most of them have their own Northbound API. Too many APIs will make the migration of application complicated and costly. Therefore, Unification of Northbound API can not only guide the design of control plane solution but also prompt developing process.

E. Middle-box Development

The feature of SD-WAN makes it easier for enterprise to deploy all kinds of middle box. However, middle box that is designed specifically for SD-WAN is limited. Therefore, the development of middle box for SD-WAN has a huge market.

F. The Optimization of OpenFlow

Although OpenFlow is used as a standard Southbound API nowadays, there are still some limitations in it. For example, the expression ability of OpenFlow is limited, which makes it difficult for some function to express in OpenFlow logic. Moreover, the load assignment between data plane and control plane still has a long way to go.

V. Conclusion

In this paper, we first give a hierarchical view of SDN architecture, which is the foundation of SD-WAN. Then by analyzing three typical SD-WAN implementations, we explicit how SDN technique is incorporated into the design of SD-WAN.

In comparison with traditional WAN, the central controller in SD-WAN makes the implementation of customized traffic engineering easier, which enables flexible strategy on flow control. Secondly, the hierarchical abstraction of SD-WAN eases the developing of middle box. Enterprises can apply a firewall or a load-balancer by simply adding some rules in the control plane. Thirdly, due to the soft-defined nature of SD-WAN, the reorganization and the expansion of the network become much easier.

All in all, SD-WAN has quite a lot of advantages compared to traditional WAN, such as its agility, low cost, security, reliability and high performance. However, as pointed out in Section Ⅳ, it should be noted that there are still some challenges in the deployment of SD-WAN.

Acknowledgment

This work is supported by The National Key Research and Development Program of China (No. 2016YFB0200401).

References

[1] D. Gibson, *SD-WAN for Dummies*, John Wiley & Sons, Inc., 2015.

[2] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[3] G. Lu, R. Miao, Y. Xiong, and C. Guo, "Using cpu as a traffic co-processing unit in commodity switches," in *Proc. First Workshop on Hot topics in Software Defined Networks*, 2012, pp. 31–36.

[4] J. C. Mogul and P. Congdon, "Hey, you darned counters!: get off my asic!" in *Proc. First Workshop on Hot Topics in Software Defined Networks*, 2012, pp. 25–30.

[5] A. Sivaraman, K. Winstein, S. Subramanian, and H. Balakrishnan, No silver bullet: Extending sdn to the data plane," in *Proc. Twelfth ACM Workshop on Hot Topics in Networks*, 2013, p. 19.

[6] L. Jose, L. Yan, G. Varghese, and N. McKeown, "Compiling packet programs to reconfigurable switches," in *NSDI*, 2015, pp. 103–115.

[7] B. Yan, Y. Xu, H. Xing, K. Xi, and H. J. Chao, "Cab: A reactive wildcard rule caching system for software-defined networks," in *Proc. Third Workshop on Hot Topics in Software Defined Networking*, 2014, pp. 163–168.

[8] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, *et al*., "The design and implementation of open vswitch," 2015

[9] M. Shahbaz, S. Choi, B. Pfaff, C. Kim, N. Feamster, N. McKeown, and J. Rexford, "Pisces: A programmable, protocol-independent software switch," in *Proc. Conference on ACM SIGCOMM 2016 Conference*, 2016, pp. 525–538.

[10] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," in *Proc. ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[11] M. Yu, J. Rexford, M. J Freedman, and J. Wang, "Scalable flow-based networking with difane." *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, pp. 351–362, 2010.

[12] P. Bosshart, G. Gibb, H. Kim, G. Varghese, N. McKeown, M. Izzard, F. Mujica, and M. Horowitz, "Forwarding metamorphosis: Fast programmable match-action processing in hardware for sdn," *ACM SIGCOMM Computer Communication Review*, vol. 43, pp. 99–110, 2013.

[13] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S.t Shenker, "Nox: towards an operating system for networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, 2008.

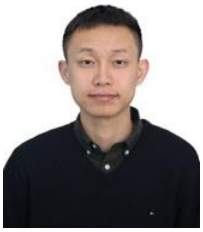
[14] T. Koponen, M. Casado, N. Gude, *et al*., "Onix: A distributed control platform for large-scale production networks," *OSDI*, vol. 10, pp. 1–6, 2010.

[15] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for openflow," in *Proc. Internet Network Management Conference on Research on Enterprise Networking*, 2010, pp. 3–3.

[16] E. Ng, Z. Cai, and A. L. Cox, "Maestro: A system for scalable openflow control," Rice University, Houston, TX, USA, TSEN Maestro-Techn. Rep, TR10-08, 2010.
[17] D. Erickson, "The beacon openflow controller," in *Proc. Second ACM SIGCOMM workshop on Hot Topics in Software Defined Networking*, 2013, pp. 13–18.
[18] M. Monaco, O. Michel, and E. Keller, "Applying operating system principles to sdn controller design," in *Proc. Twelfth ACM Workshop on Hot Topics in Networks*, 2013, p. 2.
[19] X. Jin, J. Gossels, J. Rexford, and D. Walker, "Covisor: A compositional hypervisor for software-defined networks," *NSDI*, vol. 15, pp. 87–101, 2015.
[20] S. H. Yeganeh and Y. Ganjali, "Beehive: Towards a simple abstraction for scalable software-defined networking," in *Proc. 13th ACM Workshop on Hot Topics in Networks*, 2014, p. 13.
[21] M. Yu, L. Jose, and R. Miao, "Software defined traffic measurement with opensketch," *NSDI*, vol. 13, pp. 29–42, 2013.
[22] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. V. Madhyastha, "Flowsense: Monitoring network utilization with zero measurement cost," in *Proc. International Conference on Passive and Active Network Measurement*, 2013, pp. 31–41.
[23] M. Moshref, M. Yu, and R. Govindan, "Resource/accuracy tradeoffs in software-defined measurement," in *Proc. Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, 2013, pp. 73–78.
[24] M. Al-Fares, *et al*., "Hedera: Dynamic flow scheduling for data center networks," *NSDI*, vol. 10. 2010.
[25] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "Elastictree: Saving energy in data center networks," *Nsdi*, vol. 10, pp. 249–264, 2010.
[26] R. Wang, D. Butnariu, J. Rexford, *et al*., "Openflow-based server load balancing gone wild," *Hot-ICE*, vol. 11, p. 12, 2011.
[27] H. Hu, W. Han, G. Ahn, and Z. Zhao, "Flowguard: Building robust firewalls for software-defined networks," in *Proc. Third Workshop on Hot Topics in Software Defined Networking*, 2014, pp. 97–102.
[28] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, *et al*., "B4: Experience with a globally-deployed software defined wan," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3–14, 2013.
[29] C. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," *ACM SIGCOMM Computer Communication Review*, vol. 43, pp. 15–26, 2013.
[30] A. Gupta, L. Vanbever, M. Shahbaz, S. P. Donovan, B. Schlinker, N. Feamster, J. Rexford, S. Shenker, R. Clark, and E. Katz-Bassett, "Sdx: A software defined internet exchange," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 551–562, 2015

Xugang Wu received the B.S. degree from the School of Computer Science, National University of Defense Technology in 2017. He is currently pursuing the M.S degree. His research interests include parallel and distributed computing as well as operating system.

Kai Lu, Ph.D, Professor, Deputy dean of College of computer science, National University of Defense Technology, China. His research interests include parallel and distributed system software, operating system, parallel tool suites and fault-tolerant computing technology.



Guoliang Zhu received the BS and Master degrees in computer science from the National University of Defense Technology, China, in 2011 and 2013, respectively. He is currently an PhD candidate in National University of Defense Technology, China. His research interests include cloud computing, Non-volatile memory and Operating System.