A Software Defined Approach to Unified IPv6 Transition

Wenfeng Xia
University of Science and Technology of China
fengwen@mail.ustc.edu.cn

Tina Tsou
Huawei
tena@huawei.com

Diego Lopez
Telefónica I+D
diego@tid.es

Qiong Sun
China Telecom
sunqiong@ctbri.com.cn

Felix Lu
Huawei
felix.lu@huawei.com

Haiyong Xie
USTC / Huawei
haiyong.xie@ustc.edu

ABSTRACT

The IPv6 transition has been an ongoing process throughout the world due to the exhaustion of the IPv4 address space. However, this transition leads to costly end-to-end network upgrades and poses new challenges of managing a large number of devices with a variety of transitioning protocols. Recognizing these difficulties, we propose an software defined approach to unifying the deployment of IPv6 in a cost-effective, flexible manner. Our deployment and experiments demonstrate significant benefits of this approach, including low complexity, low cost and high flexibility of adopting different existing transition mechanisms.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Centralized networks; C.2.4 [**Distributed Systems**]: Network operating systems

Keywords

IPv6 Transition; Software Defined Network

1. INTRODUCTION

The exhaustion of the IPv4 address space has been a practical problem that network carriers are facing today. Existing solutions such as IPv4 re-addressing and address reusing fail to fundamentally solve this problem. Instead, IPv6 is regarded as a complete and thorough solution to this problem.

The transition and evolution to IPv6 are expected to complete in three stages: (1) the deployment of IPv6 is prepared and begun on the IPv4-based network, (2) IPv4 and IPv6 coexist, and (3) IPv6 plays a leading role on the network and the IPv4 network is gradually retired. A variety of mechanisms and equipment are introduced for different transition stages. Such transition mechanisms (see, *e.g.*, [1, 2, 3, 4]) can be largely divided into three types: dual stack, tunneling, and translation. Among these mechanisms, dual stack is the simplest and easiest to deploy, and the remaining two are applicable only to specific scenarios.

To date, the adoption of IPv6 is progressing slowly mainly for numerous reasons. First, IPv6 has not brought new business opportunities for network carriers, and this is not likely to change in the foreseeable future. Second, IPv6 serves as a solution only to the depletion of the IPv4 address space, and there are alternative solutions for carriers to choose. Last but not least, the adoption of IPv6 by three major players, *i.e.*, the end users, the network carriers and the Internet applications, is not developing in a balanced manner.

In particular, currently there exists a deadlock on the adoption of IPv6 between end users and Internet applications. More specifically, on one hand, IPv6 lacks support from applications; in other words, almost no application is implemented technically only based on IPv6; as a result, end users are reluctant to transition to IPv6 due to lack of attractive applications and competitive prices on IPv6. On the other hand, for Internet applications, a large-scale IPv6 network as well as a stable and large IPv6 user group are the fundamental driving force for evolving to IPv6. While users are waiting for IPv6 applications, network carriers and application are also waiting for IPv6 users. The situation is essentially a chicken and egg problem. However, as network carriers provide Internet connectivity to both end users and applications, we believe that the key to the above deadlock is that network carriers should take the initiative in constructing and developing an IPv6-friendly infrastructure, thus providing IPv6-based service access capabilities and actively nurturing the IPv6 adoption.

Additionally, the introduction of various transition mechanisms and corresponding equipment also pose a new challenge to the IPv6 transition, *i.e.*, there lacks a unified scheme that supports all of these mechanisms. It becomes more challenging when considering the fact that there are a large number of devices on the current IPv4 networks; upgrading and replacing these devices can incur tremendous investments. More importantly, it is not acceptable to interrupt the existing services during the long-lasting transition process.

Recently the emergence of the software defined networking (SDN) architecture decouples the data plane and control plane, which are historically tightly coupled in the past few decades. The decoupling of the data/control planes allow unprecedented flexibility and programmability in the networks. We see great potentials of applying SDN to the deployment of IPv6. Recognizing the challenges in the adoption of IPv6 as well as the great flexibility provided by SDN, we propose a software defined approach to flexibly unify the existing solutions to the IPv6 transition problem.

2. SOFTWARE DEFINED IPV6 TRANSITION

The existing IPv6 transition mechanisms require costly end-to-end network upgrades and managing a large number of devices with a variety of transitioning protocols. We believe that a unified, flexible approach is necessary for the success of IPv6 transition. SDN provides a perfect supporting mechanism to achieve the desired unification and flexibility.

2.1 Overview

We leverage SDN as a programmable platform for deploying var-

SIGCOMM'13, August 12–16, 2013, Hong Kong, China.
ACM 978-1-4503-2056-6/13/08.

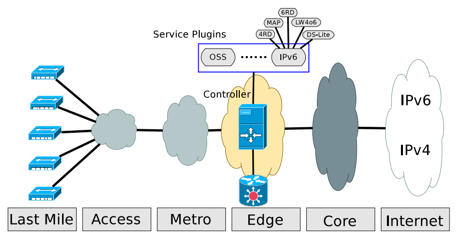

Figure 1: An overview of the SDN-based approach.

ious IPv6 transition services, thus we unify the available transition protocols and effectively address the aforementioned challenges.

In our unifying approach, we deploy OpenFlow switches located at the edge of network and the IPv6 transition service module (ITSM) as a service plug-in for the OpenFlow controller, as shown in Figure 1. In order to cope with IPv6 packets, we also extend the switches to support IP-in-IP tunnels. Note that this is a simple and straightforward extension which can be cost-efficiently implemented in hardware. The switches with such an extension process the incoming packets based on the flow tables delivered by the ITSM via the SDN controller.

The controller provides a northbound interface (NBI) that enables the ITSM to manipulate the traffic via OpenFlow. More specifically, the controller provides an OpenFlow driver allowing the ITSM to instruct SDN-enabled equipment to treat traffic using the ONF-Controller-to-Application interface. This interface is used to send and receive all specified OpenFlow messages (*e.g.*, `packet_out`, `flow_mod`, *etc.*) between the controller and the ITSM.

2.2 Software Defined Unification

The above SDN-based approach enables the ITSM to program SDN-enabled equipment to tunnel IPv6 traffic across an IPv4 data plane (the controller translates the commands issued by the ITSM into a form that can be executed by the SDN-enabled equipment). Figure 2 illustrates an example of how packets are processed in the software define unifying approach. Note that in this approach, a flow can be identified by components in the L2-to-L4 packet header (*e.g.*, all packets to a subscriber can be treated as one single flow, which will greatly reduce the number of flows).

More specifically, when an SDN device receives the first packet of a flow the packet is forwarded to the controller (steps 1–2), because this flow is a new one, and the flow table does not have a matching rule to cope with it. The controller then sends it to the proper service module (*i.e.*, the IPv6 Transition Service Module) via the NBI (step 3). The ITSM generates policies (*e.g.*, `packet_out` and `flow_mod`) for this flow, and these policies are sent to the controller via the NBI and then to the SDN-enabled equipment. On receiving these policies, the equipment adds new flow and the corresponding policies to the flow table (steps 4–6). Finally, the subsequent packets of the flow will be processed and forwarded by following the policies defined in the flow table; as a result, the SDN-enabled equipment forwards tunnelled IPv6 packets to their proper destinations.

3. EVALUATIONS AND DEPLOYMENT

We implement the the software defined transition approach based on the Open vSwitch using commodity hardware and quantify the performance of our implementation via extensive evaluations. More specifically, we run single-threaded IPv6 transition applications on Intel Xeon E5-2407 with four CPU cores and 32 GB memory. The data path is implemented by running the Open vSwitch on commodity hardware with an Intel Core i5-2400 CPU with 4GB memory. We install 64-bit CentOS as the operating system on all hardware platforms.

We vary the number of concurrent flows and quantify the processing latencies of key components. Figure 3 summarizes the results. We observe that the overall total processing latency is about

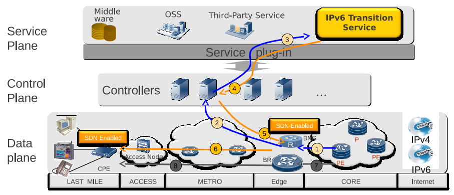

Figure 2: Packet processing in the unified approach.

0.6–0.9 millisecond, and that the latencies incurred by the controller and the forwarding element (*i.e.*, Open vSwitch) are largely constant, which is what we expect (recall that only the first packet of a flow is processed by the controller and applications). We also observe that IPv6 transition applications incur a significant portion (approximately 50%) of the overall processing latency. We find that this high latency is mainly caused by the transition applications' constructing the flow table for the data path. This problem could be addressed by leveraging the CPU's multi-core and multi-thread capability, which we leave as a part of our future work. Additionally, our prototype implementation can cope with 4Gbps traffic when fully utilizing the CPU on the data path.

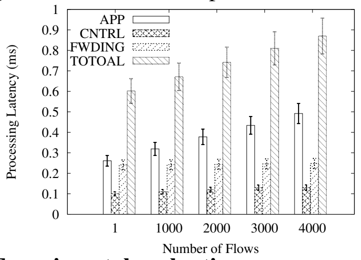

Figure 3: Experimental evaluations on processing latencies.

We also deployed the above approach on an enterprise campus in Santa Clara, CA. The deployment consists of 14 WiFi access points, one OpenFlow controller and one IPv6 Transition Service Module. The same deployment had been used to provide IPv6 Internet access for the ETSI Network Function Virtualization Workshop on April 22–23, 2013. The workshop alone had more than 270 participants who had used the transition service. Our experiments and deployment demonstrate that our approach is significantly more cost-effective and flexible than existing approaches.

4. CONCLUSION

We proposed a novel, software defined approach to address the challenges of IPv6 transition. Our approach unifies the variety of IPv6 transition mechanisms in a cost-effective, flexible manner. We deployed the approach on an enterprise campus to provide IPv6 Internet access. Our experiments suggest significant benefits of this approach, including low complexity, high flexibility and low cost.

5. ACKNOWLEDGMENTS

The corresponding authors are Tina Tsou and Haiyong Xie. Haiyong Xie is supported in part by, and Wenfeng Xia is supported by, NSFC Grant No. 61073192, 973 Program Grant No. 2011CB302905, NCET Program grant No. NCET-09-0921, and USTC Grant No. WK0110000014.

6. REFERENCES

[1] R. Despres. IPv6 rapid deployment on IPv4 infrastructures (6RD). RFC 5569, 2010.
[2] A. Durand, R. Droms, J. Woodyatt, and Y. Lee. Dual-stack lite broadband deployments following IPv4 exhaustion. RFC 6333, 2011.
[3] I. Farrer and A. Durand. lw4over6 deterministic architecture. IETF Internet Draft, Jul. 2012.
[4] O. Troan, W. Dec, X. Li, C. Bao, Y. Zhai, S. Matsushima, and T. Murakami. Mapping of address and port (MAP). IETF Internet Draft, Jun. 2012.