Designing a Virtualization Cloud Platform for Network Applications in SDN Intra-domain Production Network

Pingping Lin, Jun Bi, Hongyu Hu
Network Research Center, Department of Computer Science, Tsinghua University
Tsinghua National Laboratory for Information Science and Technology
4-204, FIT Building, Beijing, 100084, China
linpp@netarchlab.tsinghua.edu.cn, junbi@tsinghua.edu.cn, huhongyu@cernet.edu.cn

***Abstract*—SDN is considered to be a promising way to re-architect the Internet. Currently, Network operating system (NOS) of SDN only provides the basic NetAPI, and the programs for the network protocols are mixed together. This is feasible in experimentation. However, in the production environment, network protocols running on SDN controller need isolated network environments with isolated network resources. Thus, this paper designs a practical virtualization cloud platform upon the NOS.**

Keywords- Software Defined Networking, Virtualization Cloud Platform

I. INTRODUCTION

Currently, the architecture of the network device is closed. This is not favorable for the network innovation, and protocols related to the core network lay or core network device are hard to apply. Software defined networking (SDN) [1] decouples the vertically coupled architecture by separating the control plane and data plane, and moves the network intelligence in control plane to a logically centralized controller. At the same time, it opens up the control plane and the protocol implementation in control plane. Then the architecture in network device is no longer closed.

The idea of Software defined networking to promote the rapid network innovation is well received by both academic researchers and industry researchers and is considered to be a promising way to re-architect the Internet. So far, SDN itself is not mature enough in many aspects and the research on how to properly support the network application (APP)(the applications in this paper refer to network protocols like an interior gateway protocol, network applications like topology calculation, and new network architectures) just started. Currently, NOS such as NOX [2] Maestro [6], Beacon [7], and SNAC [8] only provide the most basic APP programming interface (NetAPI), and the programs for the APPs are mixed together. This is feasible in experimentation but forbidden in production network: APPs might conflict with each other in flowtable items (Traffic Engineering APP X requires a switch to forward a certain flow to port 2 while Multipath Routing APP Y requires forwarding the same flow to port 5). In practice, Each APP needs an isolated environment with isolated network resource to make sure that APPs are not affected by each other (if APP X occupies too much computing resource, other APPs on the same NOS could not work as normal). Besides, APPs always need a specific or virtual network [1] view such as the access network. Thus, this paper proposes building a virtualization cloud platform located upon NOS as the running environment for APPs.

II. VIRTUALIZATION CLOUD PLATFORM

As shown in Figure 1, NOS hides heterogeneity from the underlying forwarding hardware and has the ability to perceive network state. Thus, NOS should collect physical network state and deliver it to VCP above. VCP runs as a cloud service model to provide the APP hosting service, and provides all the services to the entire life cycle (deduction phase, emulation phase, operation phase) of the APP. VCP executes system calls southward and provides the standardized NetAPI to APPs northward. For intra-domain, the SDN is a centralized control model, and for inter-domain, NOSes in different ASes are independent and can be heterogeneous. Thus, APPs located in different ASes should communicate in a negotiation manner. Furthermore, VCP adopts the Extensible Markup Language (XML) which is NOS-independent and APP-independent for information expression and policy negotiation. Inter-domain APPs such as DIA [3], Pathlet routing [4], and Identifier-Locator network protocol can define their own XML labels.

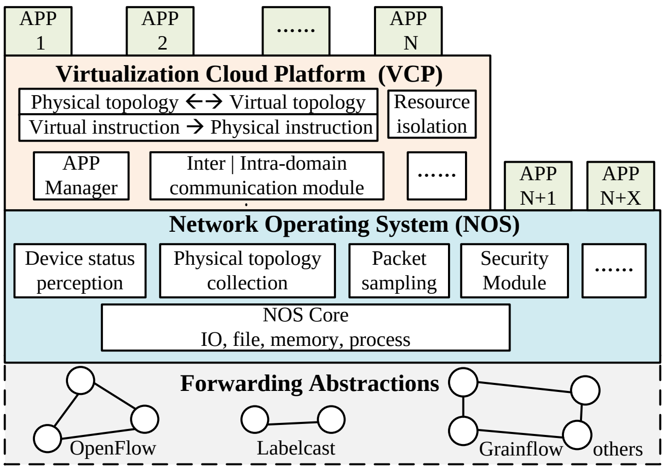
APP 1
APP 2
……
APP N
Virtualization Cloud Platform (VCP)
Physical topology ←→ Virtual topology
Virtual instruction → Physical instruction
Resource isolation
APP Manager
Inter | Intra-domain communication module
……
APP N+1
APP N+X
Network Operating System (NOS)
Device status perception
Physical topology collection
Packet sampling
Security Module
……
NOS Core
IO, file, memory, process
Forwarding Abstractions
OpenFlow
Labelcast
Grainflow
others

Figure 1. High level architecture of SDN.

A. *Network Resource Virtualization*

The physical network resource in SDN mainly refers to topology, flowtable, link bandwidth, and the system resources on VCP servers. The virtualization technologies of link bandwidth and the system resources are mature. This paper focuses on the virtualization of the topology and flowtable. **1) Topology abstraction and virtualization:** The VCP uses directed graph to describe the physical topology. Besides the full physical topology, network APPs always need specific topologies [1] such as the access network and the edge network shown in Figure 2. VCP is responsible for calculating and maintaining virtual topologies for APPs. **2) Flowtable virtualization:** Different specific networks might use the same forwording devices. So the flowtable in forwarding layer such as the OpenFlow [5] and Grainflow [11] flowtables should be virtualized. OpenFlow, a representative SDN instance, uses the VLAN (Virtual Local Area Network) technology for network virtualization. However, the length of VLAN-ID is only 12 bits, which means the OpenFlow can support only 4096 virtual networks simultaneously. Besides, each port of the forwording device can only be divided into one fixed VLAN and the port cannot be shared by different virtual networks, which means a forwording device with 24 ports can only run 24 virtual networks simultaneously at most. Thus, we define an APP-ID (a 24bits label) and extend the flowtable by adding the APP-ID field. This way, different specific networks can share the same port, and all the flowtable items with the same APP-ID compose one virtual flowtable.

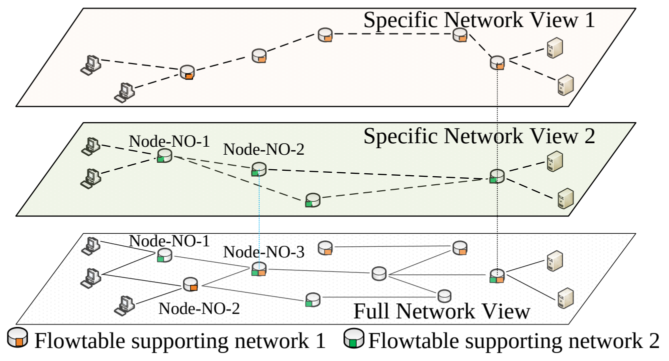

Figure 2. Topology Virtualization.

B. Mapping Specific Network to Physical Network

The mapping of the specific network mainly includes two aspects: **1) Network view mapping:** when the VCP receives a virtual network request, VCP should generate the required network topology and the corresponding network nodes resource according to a certain virtual network embedding algorithm (Different algorithms can be designed). At the same time, the VCP should maintain the mapping from specific topology to physical topology. **2) Network operation mapping**: APP directly operates the specific network. Then the VCP should translate the command from against the specific network to against the physical network. For example, a command to the virtual node {drop packets in node-no-2 with source IP address 16.11.32.20} will be translated into {drop packet in node-no-3 with source IP address 16.11.32.20} for physical node in Figure 2.

TABLE I. VIRTUAL NETWORK EMBEDDING

Symbol	Meaning
G^p	Physical Network
NE^p	Physical Network entity(switch, host, controller)
NC^p	Physical Network connector(link)
G^v	Virtual Network
NE^v	Virtual Network entity
NC^v	Virtual Network connector
CS	Constrains, the relationship between NE and NC
NE_a	Network entity attributes (computing, flowtable…)
NC_a	Network connector attributes (bandwidth, hop, distance)

The virtual network embedding is mapping the G^v to a subset of the full network view G^p:

$$Request\ (G^v)\ =\ (NE^v, NC^v,\ NE_a,\ NC_a, CS)$$
$$G^v = F_{\text{embed}}\ \{Request\ (G^v)\ \rightarrow\ (NE^p, NC^p, NE_a,\ NC_a, CS)\}$$

The objective of the embedding algorithm in production network is the economic revenue, and it is an optimization problem. VCP assumes the price of NE^v is α and the price of NC^v is β. Then the objective of $R(G^p)$ is:

$$\max R\{\sum_{NE^v(\mathrm{i})\in NE^p} \alpha[NE^v(\mathrm{i})] + \sum_{NC^v(\mathrm{j})\in NC^p} \beta[NC^v(\mathrm{j})]\}$$

Furthermore, to maximize the revenue, sometimes the VCP may need virtual network migration. VCP assumes the number of the virtual networks which need to be migrated is *N*. Then the $R(G^p)$ changes to the following format:

$$R^{'}(G^p) = \max\{\, R(G^p) - \textstyle\sum_{\mathrm{k}\in\mathrm{N}} \text{migration cost}[G^v(\mathrm{k})]\}$$

C. VCP Components Cooperation Mechanism

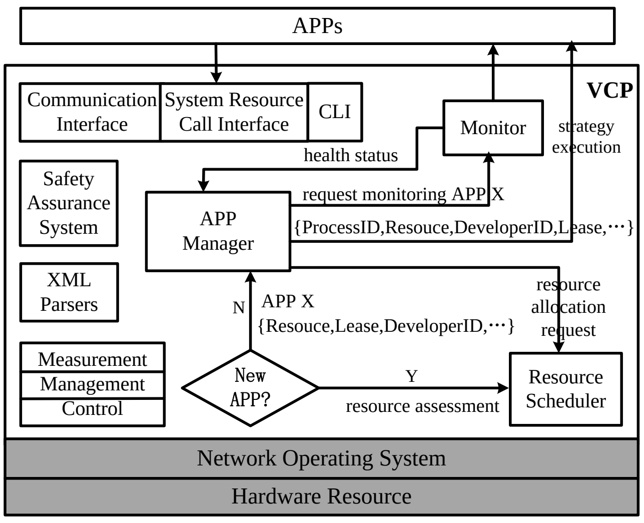

Figure 3. Components cooperation mechanism.

APPs could run directly on the NOS or on the VCP. For the latter scenario, the VCP provides a component mechanism shown in Figure 3.

- Communication Interface Set: APPs in different locations may need to communicate with each other. The communication interface similar to the communication socket is essential to APP developers.

- System Resource Call Interface: VCP should provide the general system resource like the memory, storage, and network view. Thus, VCP should provide the corresponding system calling interfaces. Then new APPs proposed by researchers could run directly on the VCP interfaces.

- XML parsers: for information expression and policy negotiation process, VCP provides the public XML parsers such as SAX [9] or DOM [10].
- APP Manager: responsible for the access control, management, withdrawing of APP and maintaining records {resource size, lease length, developer, APP-ID …}.
- Monitor: monitoring the health status of APPs and feedback the results to APP developers.
- Resource Scheduler: This is responsible for resource allocation, scheduling and recycling.
- Safety Assurance System: protecting the safety of the whole VCP environment. In addition, some APPs may need to communicate with each other and deliver message in secure channels.
- Command-line Interface (CLI): for VCP administrators to configure the network equipment.
- Measurement, Management, Control module and its Interface: for APP administrators to operate their APPs. The three functions running as three APPs are independent of the user data flows to guarantee their effectiveness.

D. Entity and Mapping Storage

All of the entities information and mapping relationships are stored in databases (DBs): 1) APP DB: stores APP-ID, life cycle, corresponding specific network and so on. 2) Full physical network information DB. 3) Specific networks information DB. 4) Mapping DB: storing the mapping relationships from specific networks to their corresponding physical network (The node-no-2 in virtual topology 2 is correspondent to node-no-3 in the full physical topology shown in Figure 2).

III. Experiment Results

We implemented a prototype and installed it in controller 1 and 2. The controllers in each AS ran IBGP while the EBGP still runs in inter-domain. Two cases were carried out on the VCP prototype: 1) APP - DIA source address verification [3] with an edge network view; APP-DIA plays the role of DIA intra-domain central controller. We transformed new packet and fields defined in DIA into XML DIA-labels. 2) APP – Pathlet Routing [4] with a full network view. A pathlet APP in each AS asks the registration center for the location of the other APPs. Then, APPs in different ASes exchanged the pathlets information and performed the pathlet routings. The experiment results shown in Figure 4 verified the feasibility of the VCP.

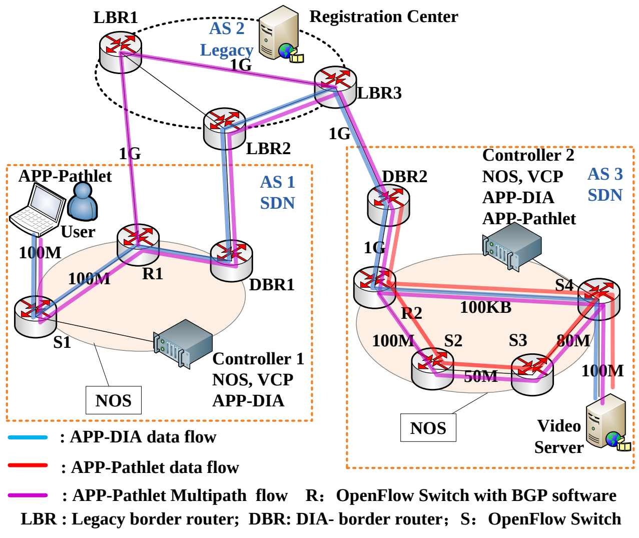

Figure 4. Experiment topology.

Acknowledgment

Supported by National Science Foundation of China under Grant 61073172, Program for New Century Excellent Talents in University, Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant 200800030034, and National Basic Research Program ("973" Program) of China under Grant 2009CB320501.

References

[1] N. McKeown. “Keynote talk: Software-defined networking”. In Proc. of IEEE INFOCOM’09, April 2009.

[2] NOX: http://www.noxrepo.org/. Accessed April 2012.

[3] Bingyang Liu, Jun Bi, Yu Zhu. “A Deployable Approach for Inter-AS Anti-spoofing”. ICNP, October 2011.

[4] P. B. Godfrey, I. Ganichev, S. Shenker, and I. Stoica. “Pathlet routing”. SIGCOMM, August 2009.

[5] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. “OpenFlow: enabling innovation in campus networks”. ACM SIGCOMM Computer Communication Review, 38(2), 2008.

[6] Z. Cai, A. L. Cox, and T. S. E. Ng. “Maestro: A System for Scalable OpenFlow Control”. Tech. Rep. TR10-08, Rice University, 2010.

[7] Beacon: http://beaconcontroller.net/.

[8] Simple network access control:

http://www.openflow.org/wp/snac/.

[9] SAX: http://www.saxproject.org/.

[10] W3C, Document Object Model (DOM) Level 3 Core Specification. W3C Recommendation, April 2004.

[11] Zhongjin Liu, Yong Li, Li Su, Depeng Jin, Lieguang Zeng, “Grainflow: a per-bit customizable scheme for data plane innovation on programmable hardware”.Proceedings of The ACM CoNEXT Student Workshop, 2011.