No More Middlebox: Integrate Processing into Network

Jeongkeun Lee, Jean Tourrilhes, Puneet Sharma, Sujata Banerjee
Hewlett-Packard Laboratories, Palo Alto, USA
{jklee,jean.tourrilhes,puneet.sharma,sujata.banerjee}@hp.com

ABSTRACT

Traditionally, in-network services like firewall, proxy, cache, and transcoders have been provided by dedicated hardware middleboxes. A recent trend has been to remove the middleboxes by deploying the network services into switch/router-integrated computing modules or separate server/blade machines. In this abstract, by using a web Ad-insertion application as an example, we demonstrate our in-network processing (INP) framework that orchestrates various computing resources and network devices and enables seamless and efficient deployments of network services.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

General Terms

Algorithms, Design, Reliability

Keywords

In-network processing, middlebox, controller

1. BACKGROUND

The Internet has grown to support a plenty of applications over an increasingly heterogeneous underlying network with varying characteristics. However, lack of co-located processing and protocol inflexibility has led to the mushrooming of middle-boxes as the primary approach to deploy (new) network services. This leads to increased cost in maintaining separate devices and also a substantial increase in management complexity and waste of middlebox resources. For example, many middlebox appliances are deployed using an inline 'bump-in-the-wire' configuration. Traffic from one side of the middle-box is received and processed, then forwarded to the other side. If the middle-box fails, we lose all the traffic go through the wire. It is very hard or impossible to use the appliance for the traffic originally not going through the specific wire. Joseph et al. [1] proposed a mechanism to explicitly forward traffic flows through different sequence of middleboxes but the additional space, power, cost and management complexity of using dedicated hardware middleboxes still exist.

SIGCOMM'10, August 30–September 3, 2010, New Delhi, India.
ACM 978-1-4503-0201-2/10/08.

An interesting recent trend has been to make the computing/processing co-located with network devices available to third-party software developers [2, 3]. Another trend is the use of software switches/routers in enterprise and datacenter networks [4]. Software-based network OS run on standard x86-based hardware and thus network services can be easily deployed on the software switches/routers as virtual machines (VMs) or software modules. Finally, there are efforts to use standard x86-based servers/blades as in-network processing resources [5, 6]. All these efforts enable us to remove dedicated hardware middleboxes and place the network services directly on network platform and/or standard servers.

In order to provide seamless and efficient in-network services on top of various processing modules available across the network, we need a flexible *control framework* that orchestrates processing modules and networking devices in an automated way and copes with various performance requirements of network services. We describe our approach in designing and building such a flexible in-network processing (INP) framework and demonstrate a prototype solution by using an example of a web advertising insertion scenario.

2. INP FRAMEWORK

The INP framework is composed of INP switches, processing modules (stand-alone servers and/or switch-integrated ones) and an INP controller.

2.1 Switch

In order to dynamically steer packet flows between various points in the data path and multiple processing modules, we leverage OpenFlow protocol [7], that provides a rich set of API enabling a controller (or a network operator) to control the flow of packets in OpenFlow-enabled network devices. Hence, in our prototype system, an INP switch implements 1) OpenFlow APIs for flow steering and 2) additional features such as tunneling, load-balancing and flow rate limiting as will be discussed later.

When a new switch joins the INP framework, the new switch registers itself to the INP controller, which checks the connectivity between the new switch and other INP switches. If there is no direct link from the new switch to any other INP switches, the controller sets up a tunnel between the new switch and the closest INP switch to use the L2 forwarding feature of OpenFlow.

Some INP switches perform load-balancing between multiple processing modules. QoS APIs like rate limiter and prioritization are also needed in INP switches to throttle or

block suspicious flows identified by a firewall (or other security services) and to provide QoS guarantees to important service flows.

2.2 Processing module

All available INP processing modules are registered to the controller. We use virtual machine as a basic mechanism to install, remove, and migrate network services on the processing modules. Each processing module reports its specifications - number of VMs it can host, computing power of each VM, network bandwidth allocated to each VM - to the controller. Each module also tells the controller the INP switch that the module is connected to.

Currently, we use Xen hypervisor (http://www.xen.org/) and Open vSwitch (http://openvswitch.org/) to control flow paths between VMs and between VMs and outside of the box. Open vSwitch replaces the default Linux bridge of Xen hypervisor and it also implements OpenFlow and tunneling. We also put the additional features of load balancing and QoS APIs into Open vSwitch. Hence, every INP processing module has an INP-enabled virtual switch in its host OS between VMs and NIC(s).

2.3 Controller

The key to the success of the INP framework is on the controller that automates the deployment of network services and the configuration of the switches and processing modules. As described in the previous sections, the controller collects the network topology/bandwidth information and each processing module's location and spec. Another important input to the controller is the *service template* that specifies resource requirement and processing capability of a network service that will be deployed. The controller also takes a high-level user policy like *"apply a firewall to the TCP flows from client* C*"* together with the client's performance requirement as inputs.

Based on the collected information, the controller makes the optimal decisions on the number of VMs (service instances), the locations to deploy the VMs, and the flow steering paths to satisfy the client's SLA agreement and also to minimize the waste of networking and computing resources. The decisions are implemented on the involved switches and processing modules through OpenFlow, INP and VM management APIs. For efficient management of resources and high availability, the controller periodically monitors the network and processing modules and dynamically performs needed actions like addition/removal of VMs, balancing loads over VMs and changing flow routing paths.

The controller should be able to check the integrity of user policy inputs and audit the derived INP constraints/actions so to make them correct by design before being implemented on the network. In addition, the INP controller should interact with other network/server controllers such as a datacenter multi-path controller and a cloud resource controller. Providing these controller features are on-going work in our roadmap.

3. TESTBED DEMONSTRATION

We implement the prototype of the proposed framework in the testbed in HP Labs, Palo alto. The testbed consists of OpenFlow-enabled HP ProCurve switches with ONE processing blades, x86-based servers running Open vSwitch and/or Xen hypervisor. To support the discussed load balancing and QoS features in our prototype, we implement a hash-based load balancing [8] on Open vSwitch software switches and built per-flow rate limiting and prioritization APIs on both HP hardware switches and Open vSwitch.

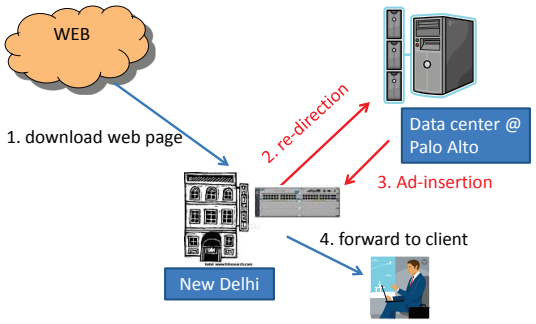

Figure 1: Ad-insertion application scenario.

In the demonstration, we will use an Ad-insertion application as an example scenario as shown in Fig.1. We will conceive the switches and servers in Palo Alto as a datacenter and use a laptop in the demo site as a client downloading web pages from the public Internet. Another laptop in the demo site will act as an (enterprise) INP switch that redirects the client's web traffic to the datacenter for Ad-insertion. The INP switch itself can also run an Ad-insertion VM instead of forwarding the traffic to the datacenter.

We will present the testbed operation via a GUI controller. The GUI controller displays the testbed topology of the switches, processing modules and the client. The GUI controller takes the user requirements and policies and will show the deployed locations of the Ad-insertion VMs and also show dynamic re-routing and VM-switching in response to a VM/network failure.

4. REFERENCES

[1] D. A. Joseph, A. Tavakoli, and I. Stoica. A policy-aware switching layer for data centers. In *ACM SIGCOMM*, pages 51–62, 2008.

[2] HP ProCurve Open Network Ecosystem (ONE). `http://www.procurve.com/one/index.htm`.

[3] J. Lee et al. Network integrated transparent tcp accelerator. In *IEEE AINA*, Perth, Australia, 2010.

[4] Vyatta Open Networking - Software-based Routing & Security. `http://www.vyatta.com/`.

[5] G. Gibb et al. Openpipes: Prototyping high-speed networking systems. In *ACM SIGCOMM, Demo session*, 2009.

[6] A. Gorti and V. Pandey. An integrated high-speed load balancing and flow steering framework. In *Workshop on Data Center - Converged and Virtual Ethernet Switching (DC CAVES)*, Issy-les-Moulineaux, France, September 2009.

[7] OpenFlow: Enabling Innovation in Campus Networks. `http://openflow.org`.

[8] M. Schlansker et al. Killer Fabrics for Scalable Datacenters. In *IEEE ICC*, Cape Town, South Africa, 2010.