RC25195 (C1107-011) July 27, 2011
Computer Science

IBM Research Report

Scalable and Agile Data Center Networking via Software Defined Network and Hybrid Control

Yue Zhang[1], Kai Zheng[1], Chengchen Hu[2], Kai Chen[3], Hang Liu[1], Athanasios V. Vasilakos[4]

[1]IBM Research Division
China Research Laboratory
Building 19, Zhouguancun Software Park
8 Dongbeiwang West Road, Haidian District
Beijing, 100193
P.R. China

[2]Xi'an Jiaotong University

[3]Northwestern University

[4]National Technical University of Athens

Scalable and Agile Data Center Networking via Software Defined Network and Hybrid Control

Yue Zhang[1], Kai Zheng[1], Chengchen Hu[2], Kai Chen[3], Hang Liu[1], Athanasios V. Vasilakos[4]
[1]IBM China Research Lab [2]Xi'an Jiaotong University [3]Northwestern University [4]National Technical University of Athens

Abstract

Data center networks today begin to confront the scalability problem, especially for the multi-tenant data centers sharing the flat address space for different users. Classical Ethernet protocols have fundamental limitations at scale, due to broadcast flooding and the inefficient Spanning Tree Protocol (STP). Software Defined Network (SDN)/OpenFlow provides enough agility to leverage network forwarding plane to be more efficient, however, it suffers from considerable control plane overhead and the risk of single-point-of-failure in the control plane. In this paper, we present CNA (*Cloud Network Architecture*), a scalable and agile data center network architecture, which achieves flat layer-2 network virtualization and agile user-defined network topology in a cost-effective way. The innovations cover through both data plane and control plane. For the data plane, benefiting from Internet experience, CNA adopts core-edge separation architecture to cost-effectively support overlay and SDN/OpenFlow, with virtual VLAN support. For the control plane, a novel hybrid control model is proposed to greatly mitigate control plane overhead and enhance the reliability. Through evaluation, CNA is proven to achieve fast data plane forwarding with very low control overhead.

***Index Terms*—Network Virtualization, Overlay, SDN/OpenFlow, Core-edge, Hybrid control**

I. Introduction

The increasing demand of cloud services, especially high performance computing applications, such as Map-Reduce, Dryad, etc., are easy to saturate data center resources by the compute-bound applications which require to exchange data among lots of server nodes. These has stressed the ability of the network to scale in previously unimagined ways, and led to the rise of the mega multi-tenant data center, consisting of tens of thousands of servers, or even much more in the future. Amazon Elastic Compute Cloud (EC2) and IBM's Blue Cloud are the examples of public cloud offered by data center providers for massive multi-tenancy.

Infrastructure as a Service (IaaS) is a widely accepted model to provide an overall virtualization for multi-tenant data centers which is considered to be cost effective, as well as better for availability and performance. Server and storage virtualization has been commercially applied, with quite a few sophisticated solutions, such as VMware, KVM, XEN. However, server virtualization on its own is not enough in today's mega multi-tenant data center, since ensuring network performance and security for highly mobile virtual machines (VMs) will quickly become an operations nightmare [1]. A few recent literatures [2, 3] proposed to use non-blocking multi-root tree topology and centric controlled overlay (e.g. MAC-in-MAC or IP-o-IP) to support tens of thousands of nodes in the mega data centers and provide a subset of capability of network virtualization, such as live VM migration, etc.

Agile controlled network virtualization, as a superset of the agility provided by [2, 3], is attracting more attention recently. On one side, as the number of VMs grows with server amount rapidly, current layer-2 network protocols reach significant limitations, such as broadcast and VLAN tag number, on providing large scale flat layer-2 network. Meticulous designed network virtualization is able to overcome these problems, benefiting multi-tenant data center in the following features: "Plug-and-play" deployment could be achieved to minimize management cost; Live VM migration is enabled, meaning that the services and security policies over the VM are preserved without any reconfiguration inside the VM, and the network changes can converge fast enough without incurring exception of the applications; Other value added services such as Quality of Service (QoS) and Service Level Agreement (SLA) are easy to deploy. On the other side, though [2, 3] achieve considerable achievement in providing agility to data center network (DCN), it is still quite insufficient to sustain the emerging multi-tenant cloud-based services and business applications which require on-demand network topology and user-defined flow control policy. Since different tenants may have various expectations of the network topology, sharing the network infrastructure with overlay technology is not enough.

In order to provide the capability of user-defined tenant network topology for the DCN users, more agile flow path control should be involved to build hierarchical logical topology over the flat layer-2 network. For example, two tenants may have VMs in the same physical server. Tanent#1 wants the traffic to pass through an IDS/FW before comes into the VMs, while Tanent#2 does not. The traffic forwarding behavior for each tenant needs to be defined individually. Unfortunately, these requirements are hard to be fulfilled efficiently by today's loosely structured control plane logic, which fragments network configuration into separate network elements.

Software-Defined Networking (SDN), promoted by the Open Networking Foundation (ONF), has the ability to provide agile flow based control for data center, the notion of which is to run global software based control and management functions independent with underlying switching/routing infrastructure. As a brilliant implementation of SDN sprit, the value of OpenFlow [4] has been recognized recently. It is incredibly important to define and build sophisticated SDN-based control plane architecture in large-scale data centers, not only because the data plane could be simpler and more efficient, separated from complex control logic, but also the network devices could be configured under unified management to achieve agile functionalities. However, there are still several challenges in deploying SDN/OpenFlow in DCN. First, the controller [5] might easily become the performance bottleneck, when the software implementation got overloaded with bursty requests. Second, centralizing control plane into one global controller would lead to the risk of single-point-of-failure and reliability degradation. Finally, it is unpractical for existing data centers to replace the network devices to support SDN/OpenFlow all in once.

Therefore, according to what have been discussed above, to build an agile and cost-efficient DCN, there are four objectives should be met:

Objective 1: Layer-2 network virtualization: To assign any service to any one or a few servers, and to realize "plug-and-play", e.g. via DHCP, BOOTP, etc, usually require a flat networking scheme with layer-2 broadcast/multicast support. Furthermore, multi-tenant clouds require "VLAN support" to achieve secure virtual network isolation. Layer-3 network virtualization, such as IP-in-IP encapsulation, has obvious challenge to provide this.

Objective 2: Agile and reliable network control: SDN provides high level network agility of making virtual connection inside network hardware, to construct user-defined logical topology over different kinds of tenants' needs. However, the agility should not be achieved at the cost or risk of sacrificing reliability and/or efficiency.

Objective 3: Universal compatibility: To sustain rapid growth of network scale and limit the capital/upgrading expense, a flexible DCN should be built with the physical devices which are universally backward compatible with the legacy or nowadays commodity.

Objective 4: Realizable deployment: Considering the deployment cost, it is impractical to replace all existing network devices to achieve the first two objectives, although, minimum amount of necessary hardware replacement could be tolerated.

To meet all of the four objectives, we present CNA (Cloud Network Architecture), a scalable network architecture for multi-tenant cloud computing oriented data centers. Generally speaking, the contributions of this paper are three-fold. Firstly, borrowing the idea from the success of the Internet, the CNA data plane is implemented and optimized with two separated parts, the core and the edge, respectively. The division inspires from the observation that date center users care mostly about end-to-end features such as network virtualization support, agile and user-defined flow sequence, etc., which typically is nothing to do with the network core. In contrast, data center switching fabric designers focus on network connectivity performance in the core side. The proposed core-edge separated network topology decouples the complex and intelligence from the design of performance oriented components.

Secondly, the problem in providing network virtualization in cloud-computing oriented mega data centers is revisited. Generally, besides the requirement on the capability of infrastructure sharing and isolation for multi-tenant users as have been addressed in the prior-arts [2, 3, 6, 7], we argue that on-demand/workload-optimized logical network topology for individual tenant users is also essential. In CNA, we propose solutions on this new identified requirement, as well as optimizations for the other requirements.

Thirdly, a hybrid architecture is proposed for the control plane to mitigate the design complexity increase and reliability degradation upon the adoption of SDN/OpenFlow. For the sake of seeking high reliability, we firstly remain a subset of static, simple but vital regular control functions, such as topology discovery and basic routing/forwarding, to the distributed control mechanism on the router/switch side. Further, to guarantee high scalability of the network, a hierarchical domain-based control scheme is adopted for the complicated control tasks, e.g. network virtualization and dynamic load balancing, etc. We propose to turn the control of individual switches into the control of groups of switches, which significantly lower down centric control overhead. Distributed real-time controls are deployed within a few pre-defined domains, respectively, while the centralized control setups per domain policies and coordinates among the domains in a loosely coupled manner.

Before stepping into the design details, in Section II, let's firstly go through the observations and design principles to get the overview and the primitive idea of the proposed work.

II. Design Concepts

Network planes are divided into forwarding plane (data plane) and control plane. With the appearance of SDN/OpenFlow, data plane is enlarged to the devices that perform operations according to rules, which are disseminated by control elements. Though the division of planes simplifies network design without losing reliability and scalability, it also needs elaborate design for DCN, because an unnatural division of control/data plane functionality causes complexity and inefficiency. To make DCN network more efficient and enhance agility by SDN/OpenFlow supporting with low cost, we explain our design philosophy for data plane and control plane, respectively.

A. Core-edge architecture in data plane

As mentioned before, flat layer-2 networks bring many advantages for DCN. It is not an exaggerated metaphor that considers DCN as a "big switch" from the perspectives of the resources being interconnected inside the data center. However, the "big switch" has several fundamental problems overwhelming current Ethernet devices and protocols. One is the limited memory and embedded CPU capacity which makes commodity switch hard to handle hundreds of thousands flat MAC addresses in a data center. Another problem is the performance and security degradation caused by the Ethernet flooding traffic. The third is the very inefficient STP protocol, which avoids network loop by disabling redundant links. To achieve Objective 1, these problems need to be solved. Some recent researches [8, 9] modify network protocols among the switches, losing protocol compatibility stated in Objective 3. Some other works try to build new DCN architecture, in research [2, 3, 10] or industry area such as QFabric, FabricPath, however, all of which have relatively strong hardware and/or topology assumption, failed to meet Objective 3&4.

In this paper, borrowing the idea from the hierarchical model of Internet, which is still under use today, we divide the "big switch" into two parts, the edge and the core. The core is designed to be simple and efficient which care only about connectivity and efficiency. It will be a forward-only network where the packets will not be modified or encapsulated. In the proposed experimental implementation, the core is an efficient underlay IP network, without any other pre-assumptions. Either the commodity Layer 3 switches (OSPF/ECMP supported) or the emerging efficient switch fabric such as QFabric/FabricPath can be leveraged. The core benefits the overall performance by providing, natively, topology discovery and loop-free forwarding mechanisms, as well as the standard packet interface and link-level load balancing solutions, which has universal compatibility with existing technologies (Objective 3). Meanwhile, the edge contains all the intelligent components required for the cloud-computing oriented DCN (Objective 1&2). The observation is that, the intelligential network operations (such as those related to network virtualization, fine-grained flow-control or server load balance, etc.,) can be achieved only with the control of network edge, without the participant of the network core. Hence the construction costs are greatly reduced by deploying simpler and performance oriented network core, or continue using the legacy devices (Objective 4).

B. *Hybrid control plane*

In several recent literatures, to achieve Objective 2, the powerful centralized control plane model is revisited [11, 12]. Researchers claim that with the progress of the supporting technique, single-node performance becomes strong and powerful enough to handle the network control and make decision for every details of the network, especially DCN; and only when it is with a centric control mechanism can the network provides the agility and flexibility that required for the modern cloud computing applications. Everything tends to be centralized, including topology discovery, routing decision, flow-control, traffic engineering etc. The centric controllers are thought to be powerful and overwhelming.

We argue that, completely rely on centralized control to meet the agility needs might not be scalable or efficient enough. Meanwhile, having every decision made centralized may lead to either performance or reliability issues, as well. As in the case of the Openflow-Nox [6] architecture which falls into the "powerful center" category, the switches will have to forward all unknown packets to the controller, which makes the architecture eventually not scalable. As the network scale goes up, the switches can hardly keep all rules (including basic routes), installed by the controller, in the expensive flow-table on the switch (implemented with TCAM usually), so that the controller will become performance bottleneck upon a certain portion of the traffic being considered unknown by the dump switches. And also note that the centric approaches will always face the single-point-of-failure problem.

Actually, a "high priority centralized control" is good enough to dynamically control the network on demand. In this paper, we propose a hybrid control plane model which consists of a high priority centralized control panel and a low priority distributed control mechanism. We argue that, the network activities and traffic should be treated differently. The observation is that basic control functions such as topology discovery and basic routing are not only simple but also vital and static. The majority of the traffic which is considered none-interesting can be handled by the low priority distributed routing mechanism on the switches when no pre-defined rules, installed by the controller, are matched (which is under the awareness of the centric controller). This gets rid of the reliability risks while remains the full control of the centric controller, since it always has higher priority control. Meanwhile, from performance and scalability perspective, the spirit of P2P/DHT can also be leveraged in DCN control plane [13]. The control targets, e.g. traffic or devices, can be grouped and managed autonomously according to a few policies dynamically generated by the centric controller, which also acts as the coordinator among the groups. This prevents the centric controller from being a bottleneck against the growing load with the network scale. And this also remains the full control of the centric controller, since the distributed P2P control mechanisms simply follow the policies pushed by the centric controller. The controller can deploy either fine-grained policies (e.g. for a specific switch) or coarse-grained ones (e.g. for a specific group of switches) on demand.

Let's take the human biologic system for a simile. The brain is a centric power of the body which can take the control of most of the body actions so as to express our emotion or reflect our internal idea; however meanwhile, we also have several auto-sub-systems, such as the circulation system, respiratory system, alimentary system etc, which do not rely on the conscious brain decision and can function well automatically even when we loss our conscious or become a human vegetable. Imagine the case when we have to "think" about every details of our breath or have to be conscious about the details of how the food are digested, how can we be possible to come up with the innovative and creative idea?

III. CNA ARCHITECTURE AND IMPLEMENTATIONS

A. *Core-edge separated topology*

The widely adopted multi-tenant data center infrastructure consists of three parts: host servers provide computing resource, which is shared with several VMs by hypervisor running on each server; storage devices provide fast and reliable data store for VMs; hierarchical network devices interconnect servers and storages. In the CNA architecture, the three parts are grouped into network edge and network core, as depict in Figure 1. Network edge consists of all the servers, storages and the "last-mile" edge switches, which is defined as the enhanced "smart switches" close to the server side (think the *Tag Edge Routers/TER*s in MPLS [14]). Network core consists of all the interconnect devices except the "last-mile" switches. For the sake of simplicity, we denote the smart "last-mile" switches as *Edge switch* in the following contents.

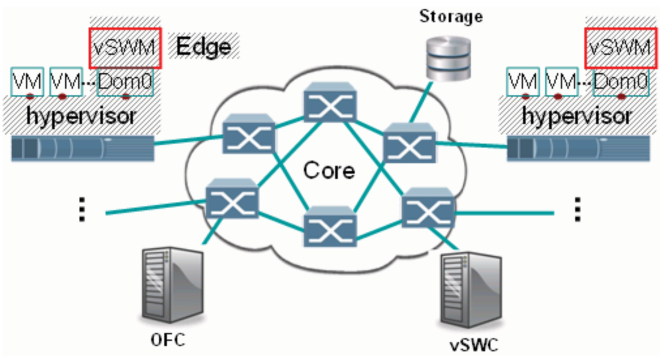


Figure 1. Physical infrastructure of CNA.

In the prototyping of CNA implementation, the intelligent edge switches are deployed on special virtual appliances, more specifically, the hypervisor domain (Dom0) [15] similar to Xen, which have privileges to access the hardware and forward traffic pass through the physical servers. Note that the edge switch could be also deployed either in the physical *Top of Rack* (ToR) switches or the virtual switches inside the hypervisor. The reason for deploy in Dom0 is that such an approach is independent with network devices and hypervisor, having the most universal compatibility and lowest realizing cost to meet Objective 3&4.

The network intelligence is provided by the 1) *virtual SWitch Module* (vSWM), following rules either from a local DHT (Distributed Hash Table) based mechanism or a centralized virtual SWitch Controller (vSWC), and 2) the OpenFlow enabling mechanism, following rules from OpenFlow Controller (OFC). In the next subsection, the hybrid control model of CNA is introduced.

B. *Hybrid control model*

Classical network control is distributed into individual network device via management interfaces. SDN/OpenFlow uses a generic centralized controller, with global view of the whole network. It pushes rules into the dump network devices, which only operate according to the rules without any intelligence. However, as for the data center scale, neither the pure distributed control nor the pure centralized control model is practical enough. As the number of devices growth, distribute control causes considerable performance/management overhead in synchronizing all network devices; centralized control may also become performance bottleneck to handle rules for huge amount of network devices which require real-time interactions with the controller.

In CNA, a hybrid control architecture is proposed, which combines the distributed control and centralized control to take benefits from both. The control plane is divided into many small *control domains*, in which distributed control protocols are deployed to make each domain autonomous. A *centralized controller* is deployed on top of the distributed control domains, which had the authority to install/update/delete policies to the *distributed control mechanism*, on demand, and this is called "hyper-control". For each control task, the two types of control elements are coordinated in a hierarchical model as follows: The distributed control mechanism is only responsible for the inner-domain control tasks and only share states among the peers within the domain, avoiding significant synchronization overhead. The centralized controller takes charge of the inter-domain coordination. It also has the privilege to control the interesting flows with the highest priority against the distributed control mechanism.

In the next two subsections, a functional oriented control mechanism of agile layer-2 network virtualization and a performance oriented control mechanism of dynamical network load balancing are explained in details to demonstrate how the hybrid control architecture is deployed for practical control tasks.

C. *Agile Layer-2 Network Virtualization via Hybrid Control*

On top of the physical infrastructure mentioned in Section III.A, CNA provides agile controlled network virtualization for multiple tenants through three phrases. First, flatten and formalize the network with the MAC-o-IP overlay. Second, isolate the logical tenant network and slice their subnets respectively using a *virtual VLAN* service realized on top of the overlay. Third, further define the corresponding logical topology using *virtual connection*. Figure 2 depicts the 3 phrases through which two example tenants define different network topologies on the same hardware infrastructure, respectively. In this example, Tenant#1 and Tenant#2 require different logical topology based on the same network infrastructure. In the following parts, the three steps are explained in details, respectively.

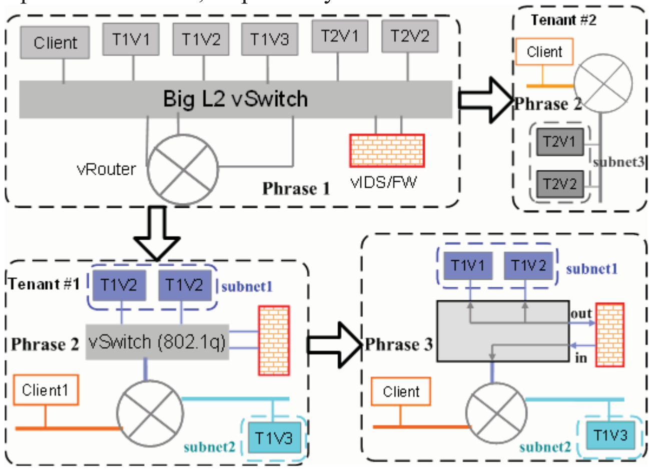


Figure 2. The 3 phrases of agile layer-2 network virtualization

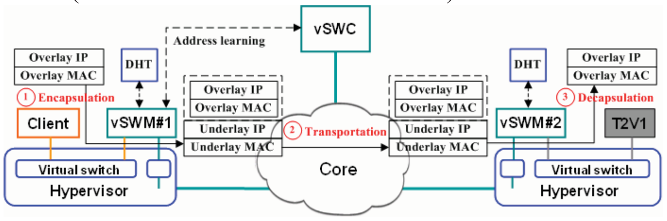
1) *Flatten & Formalize the network with MAC-o-IP overlay*

CNA flattens the network address by deploying a MAC-o-IP overlay network, through which the network edge is connected with a "big switch". To adopt hybrid control, the network edge is grouped into control domains, containing one or more Edge switches. For a given VM *h*, the corresponding mapping between the logical overlay address and the physical underlay address, i.e. <underlay_IP, overlay_MAC> is stored in both the centralized controller (i.e. vSWC) and the DHT of one of the distributed controllers in the same control domain with *h*.

The centralized controller vSWC learns the address information for all the VMs in DCN. When a newcomer VM broadcasts an ARP request, the corresponding vSWM will intercept that request, and then send out an update message to vSWC, telling it the mapping <vSWM IP, VM MAC>. As for the distributed controller, DHT learns address information in the same way, but for each DHT, it follows DHT protocols as distributed control mechanism to only store the partition of the mapping used in its control domain.

As for the traffic forwarding through MAC-o-IP overlay, the hybrid control model is implemented as follows: The edge switch searches the DHT within the corresponding domain first. In the case when the destination is inside the control domain, the address information should be found in the DHT, so the edge switch (vSWM) do not need to query the centralized controller (vSWC). In the case when the destination is outside the control domain, the edge switch will query vSWC, encountering a DHT searching miss. In this model, the control domain can be defined intentionally and re-defined dynamically to avoid the centralized controller (vSWC) from becoming a performance bottleneck.

Figure 3 depicts an example, in which 3 operations are performed for the layer-2 packet forwarding from Client to T2V1 (i.e. VM#1 in the Tenant#2 network).



Operation 1. Encapsulation.
1-1) Client sends packets to T2V1, through vSWM#1.
1-2) If vSWM#1 do not cache the mapping of (vSWM#2 IP, T2V1 MAC), it requests to DHT for the mapping, first.
1-3) If not found in DHT, vSWM#1 requests to vSWC, then.
1-4) vSWM#1 caches the mapping of (vSWM#2 IP, T2V1 MAC).
1-5) vSWM#1 encapsulates each layer-2 packets with vSWM#2 IP.
1-6) vSWM#1 sends the packets into the underlay network.
Operation 2. Transportation.
2-1) Network core transports the packets to destination vSWM#2, through legacy IP layer protocols.
Operation 3. Decapsulation.
3-1) vSWM#2 decapsulates the underlay packets.
3-2) vSWM#2 sends the original overlay packets to T2V1.

Figure 3. Layer-2 packet forwarding

2) *Subnet slicing by virtual VLAN*

After "connecting" all related devices or virtual devices to the "big switch", it is also necessary to isolate different tenants for security reasons and provide subnet partitioning within a tenant. In the example shown in Figure 3., Tenant#1 and Tenant#2 should be invisible to each other, though they are sharing the same underlying network and may even share the same overlay address space. Furthermore, the two subnets of Tenant#1 are also need to be layer-2 isolated. Classical VLAN provides layer-2 isolation in the switch level. However, scale current VLAN into data center will meet fundamental problems. For example, the total VLAN tag number (4096) is too limited comparing with the amount of tenants and servers required. And it was designed under the assumption that the network configuration is mostly static. Although some extended technologies such as QinQ, PVLAN, etc., try to mitigate these problems, they have strong hypervisor dependency, suffering from even more management overhead in the large scale data center containing different types of hypervisors.

In this phrase, CNA realizes the global virtual VLAN service, based on the layer-3 IP multicasting mechanism (on the underlay) and layer-2 802.1q VLAN (on the overlay). First, 802.1q compatible VLAN service is provided by the smart Edge switches for the local VMs interconnecting or isolating with each other, which is called the *Local virtual VLAN* (LvVLAN), the scope of which is limited within the hypervisor. Second, to further accommodate the need to include VMs with different physical locations to the same virtual VLAN, the term *Global virtual VLAN* (GvVLAN) is introduced. A dynamic mapping will be therefore setup between LvVLANs and GvVLANs, as Figure 4 depicts. Further, to realize the transition between the overlay virtual network and the underlay physical network, dynamic mappings are built between the underlay IP multicasting groups and GvVLAN IDs. vSWC is responsible for maintaining the mapping and replying to the queries to resolve the mappings.

LvVLAN IDs←→GvVLAN ID←→Multicast IP Group

Figure 4. Dynamic virtual VLAN mapping.

Once a VM belong to a certain tenant is attached to the hypervisor with LvVLAN ID configured, the related vSWM queries its GvVLAN ID from vSWC and attend the corresponding underlay multicast group by sending IGMP message to the core layer-3 switch it is attached. The multicasting routing protocol, e.g. RIM, among core switches ensure the multicast packet can be delivered to each vSWM attending the same multicast group. At the same time, vSWM searches for the existing LvVLAN ID for the attending VM, or allocates from the available pool if no existing LvVLAN is found. The destination vSWM also maintains the mapping of GvVLAN ID to the LvVLAN IDs on its side, so it can mark the incoming packet with correct destination LvVLAN ID. With the mechanism introduced above, Global VLAN-wise services, such as broadcasting, can be achieved with backward compatibility to the IEEE 802.1q standard.

Note that, it is important to provide the isolation between different virtual VLANs. On one side, the packets from different tenants need to be restricted within the GvVLAN. This can be achieved by defining restrict rules on the vSWC. The query request includes the source address of the packet as well as the destination, from which vSWC could verify whether the source and destination are in the same Global VLAN. If not, the deny response is replied, and the vSWM would block the traffic.

On the other side, the isolation is also needed within a tenant for agile subnet partitioning. This can be achieved by the user-defined mapping between LvVLAN and GvVLAN. For each subnet, LvVLAN ID is used to divide subnet within each hypervisor, which is allocated automatically by the virtual switch in the hypervisor.

The proposed virtual VLAN solution has the following advantages: 1) Total amount of GvVLANs supported by a data center is extended to the number of multicast groups supported by the underlying network (IPv4 28bits, IPv6 112bits), which is much more than classical VLAN tag capacity of 4096 (14bits); 2) Compatible with most existing hypervisors with only basic VLAN support, making it easy and efficient to implement; 3) The centralized VLAN configuration in vSWC reduces the management overhead for socializing Local VLAN IDs (on different location and belong to the same GvVLAN).

3) *Define logical topology through virtual connection*

With the overlay and VLAN configuration, multi tenants share the same network infrastructure without aware of each other. However, it is also an important requirement that tenants need to define the network logical topology as they want. It is hard to define hierarchical topology under the big switch scheme, because all the ports are flatly connected by the physical hardware. Fortunately, SDN/OpenFlow enabled switch has the ability to build logical topologies by setting up virtual connection between end hosts. Virtual connection is a user-defined end-to-end traffic path, customized in every passing switch through flow-based control policies.

Return to the example, Tenant#2 requires the flat topology, with no need for flow based control. For Tenant#1, the subnet1 and IDS/FW require a hierarchical topology that all traffic of subnet1 must go through the IDS/FW. The OpenFlow rules in this example are as follows:

```
For in_port=ouside, dst=subnet1, action=output:IDS/FW port_in
For in_port=subnet1, dst=outside, action=output:IDS/FW port_in
```

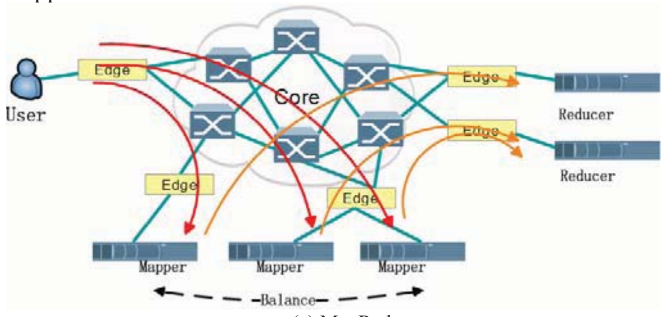
Through the three steps mentioned above, agile layer-2 network virtualization is fulfilled. The network infrastructure is flattened via MAC-o-IP, isolated for multi-tenants and then subnet partitioned inside each tenants, respectively. By the virtual VLAN service, it is also equipped with the distinct support for user-defined network topology.

D. Dynamical Network Load-Balance via Hybrid Control

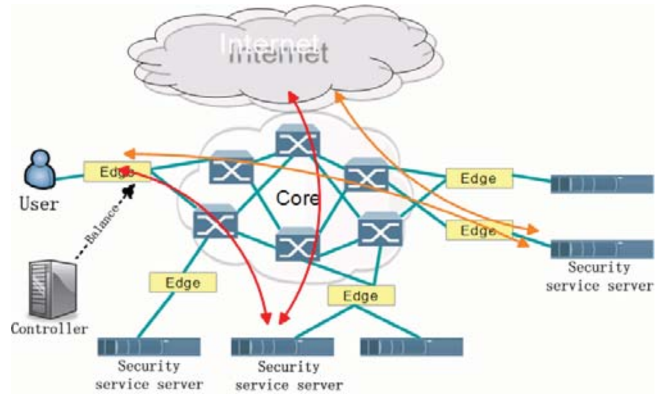
In this subsection, we take Dynamical Network Load Balancing (DNLB) as an example of performance oriented hybrid control model.

1) Scenario description and existing solution

For most applications in data center, there are usually many service elements which provide the same service, e.g., MapReduce application such as web search, as is shown in Figure 5 (a). A user query is decomposed into several meta-tasks and then sent to a set of *mapper*s to process, respectively. After that all the search records (e.g. web pages) are sent to corresponding *reducer* to further proceed and aggregate. Here the challenges we focus on is how to design a scalable and efficient control plane of the DCN which can co-operate with the MapReduce master node to balance the workload among the mappers and the reducers.



(a) MapReduce



(b) Security check

Figure 5. Examples of LB in data center

Another representative example of load balancing in DCN is interactive policy-enforcing and dynamic load-balancing, as mentioned in LiveSec [16]. As shown in Figure 5 (b), in LiveSec the controller forwards the packets to one of the security service elements to have security check when a user tries to access the Internet. There are usually a certain number of servers providing such security service, and, a centric controller takes charge of balancing the security checking load, e.g. policy check, deep packet inspection (DPI) in LiveSec.

In order to provide the intuition of the DNLB problem and have quantifiable discussion, here we model the DNLB problem with formal terms as following.

Given a set with n computing nodes which perform the same functions (e.g. in the case of MapReduce, it is the set of mappers, and in LiveSec, it is the set of security service elements), term $G=(g_1, g_2 \cdots g_n)$ is used to denote the set, where g_i denotes the i^{th} node (server) in the set. The resource (computational and/or communicational) of g_i is abstracted and defined as R_i, and we use L_i to refer to the current workload in g_i (In MapReduce applications L_i represents the computation or search workloads which are being processed or queuing for processing, and in LiveSec it represents the traffic loads of security service element). Then we use u_i to refer to the utilization ratio of g_i, which is defined as $u_i = L_i / R_i$. The average utilization ratio of the whole set is denoted as $\overline{u}$, which can be computed with ease as,

$$\overline{u} = \sum_{g_i \in G} L_i / \sum_{g_i \in G} R_i = \sum_{g_i \in G} R_i \times u_i / \sum_{g_i \in G} R_i \cdot$$

Traditional load balance mechanism adopts either distributed or centralized control manner. In the case of distributed control, typical schemes, e.g. OSPF-ECMP and VLB [2], usually adopt simple and efficient algorithm such as simple hashing or random node selection. In general, they have O(1) temporal complexity. But the traditional distributed control mechanism usually has no feedback information u_i because the communication cost will be significantly insufferable when the data center scales up. From this point of view, existing distributed control algorithms are based on statistical regularity, so they may be inefficient when working on unpredictable workload or in a highly dynamic environment.

In the case of centralized control, as described in LiveSec, the controller makes decision for every traffic flow which tries to access the Internet. Let's take the simple *Shortest Queue First* (SQF) algorithm, which is usually adopted by the centric controller, as an example. The temporal complexity of SQF is $\mathrm{O(n)}$ (where n is the number of nodes). Therefore, as the size of the data center scales up, e.g. to tens of thousands of nodes, the controller may become the performance bottleneck, and on the other hand might become a single-point-of-failure.

2) CNA dynamical load-balance using hybrid control

As mentioned in section III.B, the motivation of the hybrid control model is to leverage the advantages of both centralized and distributed control while compensate the inadequacy of each other.

Generally speaking, in CNA, the network will be partitioned into a few load-balancing control domains in the initiation process. Each control domain includes one or more edge switches which act as the distributed controllers for the computing resources (e.g. servers) connected to the network through them, respectively. The distributed control mechanism contains: 1) designated algorithm deployed on the distributed controllers to re-compute the forward path for the sake of load balancing. 2) traffic control rules installed by the centralized controller acting as the coordinator among the domains, which tries to perform long term load balance on a large scale. The DNLB design is described as the following.

a) Initialization

As mentioned above, the centralized controller partitions the data center edge (e.g. all the nodes and the edge switches they connect to) into several domains during the initialization phrase, denoted as $G=\bigcup_{G_i \subset G} G_i$,where $G_i \cup G_j = \phi, i \neq j$. Each domain contains m edge switches, e.g. $G_i=(g_{i1}, g_{i2} \cdots g_{im})$. And then the centric controller will designate one of the edge switches randomly as the designated edge switch (DEdge) in each domain.

b) The distributed part in hybrid control

The edge switches act as the distributed controllers within the corresponding domain. As shown in Figure 6, all the edge switches in the same domain share a workload table (WLT), via a dedicated distributed protocol, *Workload Table Update Protocol* (WTUP). The workload table keeps the utilization ratio of each node in domain.

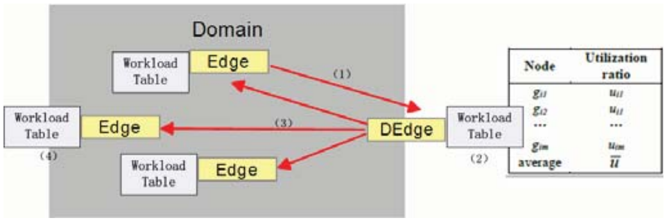


Figure 6. Distributed control intra-domain

The protocol procedure of synchronizing WLT is as follows:

Step1. Edge switch: for every T_{update}
update massage (g_{ik}, u_{ik}) is sent to DEdge;
Step2. DEdge switch: for every (g_{ik}, u_{ik}) received
update workload table WLT ;
Step3. DEdge switch: for every $T_{broadcast}$
broadcast WLT in the domain;
Step4. Edge switch: for every WLT received
update workload table WLT .

Based on the workload table, the essential part of the algorithm is to find out a load-balanced forward path for arriving workload L (meta-task for the mappers in MapReduce or traffic flows to access Internet in LiveSec. Note that, to avoid the out-of-order issue, the load balancing control is flow-grained.). As we know, the ideal balancing results of utilization ratio of each node are,

$$E(u_1)=E(u_2)=\cdots=E(u_m)=\bar{u}=(\sum L_i + L) / \sum R_i \qquad (1)$$

However, in the practical cases, there may be some elephant flows or monolithic computing tasks with which the overall workload can hardly be divided evenly. So usually only approximation result can be obtained. For example, the traditional SQF algorithms always look for the minimal load node, and in these cases, the utilization ratio after balancing is given by

$$\begin{cases} E(u_i) = u_i + L / R_i ......u_i = u_{\min} \\ E(u_i) = u_i ....others \end{cases} . \qquad (2)$$

Here in the proposed load balance scheme, the intuition is to reduce the complexity of the distributed algorithms by limiting the distributed control within small domains, while avoiding the convergence issues by letting the centric controller act as merely a loosely-couple coordinator among the domains (i.e. it handles only long term balancing goal and works in a loosely-coupled way with the distributed controllers, so that does not required to be responsive). Meanwhile, unlike the VLB algorithm which always chooses a forwarding path randomly, the forwarding path will be computed with feedback. To compensate to disadvantages of existing algorithms, a distributed LB algorithm is proposed based on a load-aware probability model, which is called *Load-aware Probability based Load Balance* (*LPLB*). The primitive observation is that, as the distributed control mechanisms are restricted within small domains respectively, the overhead to synchronize load information and utilization of the resources among the distributed is well bounded, not sacrificing scalability.

In the proposed algorithm, the forwarding path computation is based on a probability model which is related to the current utilization ratio in workload table. The detail of algorithm is described as following.

For each Edge switch g_i , we design a transition probability $P_i = (p_{i1}, p_{i2}, \cdots, p_{im})$ in which p_{ij} refer to the probability of forwarding workload from g_i to g_j . The computation of P_i is shown below.

$$p_{ii} = \begin{cases} 1................u_{\mathrm{i}} \le \bar{u} \\ (1-u_i) / (1-\bar{u}).........u_{\mathrm{i}} > \bar{u} \end{cases} \qquad (3)$$

$$p_{ij} = \begin{cases} 0...u_j \ge \bar{u} \\ (1-p_{ii}) \times (\bar{u} - u_j) / \sum_{k}^{u_k < \bar{u}} (\bar{u} - u_k).......u_j < \bar{u} \end{cases} \qquad (4)$$

Because the factor such as $(\bar{u} - u_j) / \sum_{k}^{u_k < \bar{u}} (\bar{u} - u_k)$ can be pre-computed according to the workload table before the arrival of packets, the time complexity is $\mathrm{O(logm)}$. The optimization result, i.e the mathematical expectation of the utilization ratio, is

$$E(u_i)=u_i + \sum_k (p_{ki} \times L) / mR_i \cdot \qquad (5)$$

The formula (5) indicates that the expectation in distributed controller is much closer to the ideal case of formula (1) than that of traditional distributed balancing algorithm such as simple hashing or random forwarding. And the complexity is reduced to the scale of the domain to compare with the traditional centralized control. The pseudo-code of the *LPLB* algorithm is described as the following, in Figure 7.

```
Algorithm LPLB
Input: $L$ (workload)
Output: $g_k$ forwarding path

01:while(true)
02:    when receive $L$
03:        foreach $g_j$ in domain
04:            $p_{ij} \leftarrow$ compute( $P$ );
05:        end for
06:        temp $\leftarrow$ generated random number(0,1);
07:        foreach $g_j$ in domain
08:            if temp is in $p_{ik}$
09:                forward( $L$ ) to $g_k$ ;
10:            end if
10:            break
11:        end for
12:
13:        if $\bar{u} > threshold$
14:            forward ( $L$ ) to centralized controller;
15:        end if
16: end while
```

Figure 7 Pseudo-code of Algorithm LPLB

c) The centralized part in hybrid control

The centralized controller is in charge of balancing the traffic load among domains based on a domain table (DT). As shown in Figure 8, generally the processing proceed can be summarized as the following steps via *Domain Table Update Protocol* (DTUP).

Step1. DEdge switch: for every $T_{updateDomain}$ $(G_i, \bar{u}_i)$ is sent to centric controller;
Step2. Centric controller: for every $(G_i, \bar{u}_i)$ received update domain table DT ;
Step3. Edge switch: for every L when overload L is sent to centric controller;
Step4. Centric controller: for every WLT received forward L to another domain.

The centralized controller runs merely a simple SQF algorithm, but as it is coarse-grained (i.e. balancing among domains), the temporal complexity of which is $O(n/m)$.

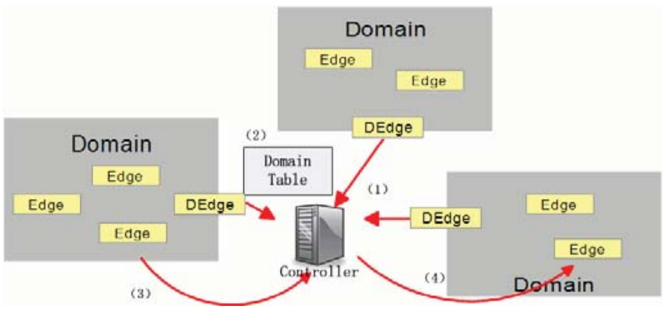


Figure 8. Centralized control inter-domain

IV. Evaluation

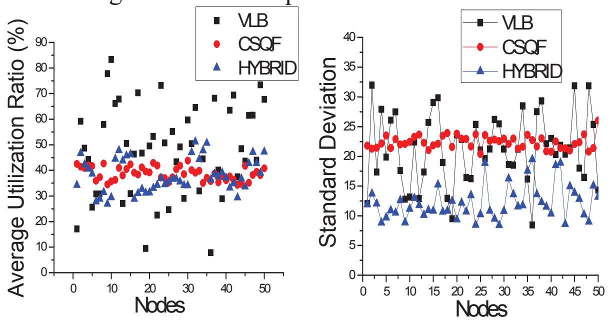
In this section, we evaluate the key performance indexes of CNA through both simulation and experimental tests. The first part focuses on evaluating the scalability of the proposed architecture, while the second part stresses on investigating the implementation related advantages with it.

A. Simulation results

C++ based simulations are developed to evaluate the scalability of CNA under the hybrid control model in large scale environment. The DNLB use-case introduced in Section VI is deployed, in which $n = 5000$ edge switches are used. The performance and overhead of three LB control models are compared. The workloads (the traffic flows which need security check mentioned in LiveSec) are generated by 50 users concurrently. Each flow arrives at a random edge switch. In centralized control model, the flows are scheduled by the centralized controller following the SQF principle; in distributed control model, the VLB scheme is deployed, that is the flow are forwarded randomly to all candidate resources; in hybrid control model, the forwarding is based on the scheme described in Section IV where the 5000 nodes (edge switches) network is divided into $m = 100$ domains and $n/m = 50$ edge switches in each domain. For the sake of simplicity and providing intuition, 50 out of the 5000 edge switches are sampled to present the results. The 50 edge switches are sampled from 10 domains. In the following subsections, *CSQF*, *VLB* and *HYBRID* are used to denote the centric control scheme, distributed control scheme and the hybrid scheme, respectively.

1) Balancing results

The balancing result is shown in Figure 9 (a) in terms of average utilization ratio of the sampled edge switches. The result of *HYBRID* is very close to that of *CSQF*, and both are significantly better than that of *VLB*. On one hand, the balancing results in the same domain are remarkable. This is because the workloads may loss balance in the *VLB* scheme, due to the uncertainty of the bursty traffic. On the other hand, the distributed control mechanism in *HYBRID* balances the workloads much better within the domains, respectively, thanks to feedback information collected in real-time. Moreover, the balancing results of HYBRID among domains are close to *CSQF*. This is because the centric controller running SQF algorithm in *HYBRID* will reassign the imbalance workloads among the domains. We will see later that, though CSQF achieve slightly better balancing result, the advantage of is at the cost of significant control plane overhead.



a) Average utilization ratio b) Standard deviation
Figure 9. Balancing results of three LB models

2) Encountering Overload

Another advantage of the hybrid control model is that workload can be reassigned by both distributed controller and centric controller according to the variable distribution of the workload and the utilization of resources. Figure 9 (b) shows the mathematic standard deviation value of each edge switch's utilization ratio which represents the unstable degree of workload. Note that, in *CSQF* the workload will be forwarded only once. In some extreme case, when the feedback of the utilization info can not converge in time, bursty workload will be assigned to a certain "idle" nodes before it manages to report the change of utilization in time, which leads to overload.

However, in *HYBRID*, the workload can be re-assign to other edge switch many times "automatically" and avoid being overload. We also measure the overload occurrence (by discarded workloads ratio) in the whole DCN, as is shown in Table 1, where we can see obvious advantages of *HYBRID.*

Table 1. Discarded workloads when overload occurrence

Balancing mode	*VLB*	*CSQF*	*HYBRID*
Overload (%)	6.20	1.75	0.07

3) Control overhead

Table 2 shows the temporal complexity of each LB model as previous analysis (in the table, DC refer to *Distributed Controller* and CC refer to *Centralized Controller*). To verify this, the average computing time (measured by CPU ticks) of the algorithms are also shown in Table 2, where we can see obvious higher control overhead for the centralized scheme.

Table 2. Temporal complexity and simulation results

Model	*VLB*	*CSQF*	*HYBRID*	
Controller	DC	CC	DC	CC
Complexity	O(1)	O(n)	O(logm)	O(n/m)
Computing time(CPU ticks)	4.22	96.10	10.82	17.87

To dedicatedly compare the cost of controller in each model, the experimental computation times are shown in Table 3. We can see that *CSQF* takes about 2 orders of magnitudes more time than the centric controller in *HYBRID*.

Table 3. Computation times of centric controller

Balancing model	*VLB*	*CSQF*	*HYBRID*
Computation times	0	175541	1672

B. Experimental forwarding performance

This experiment is based on a real test-bed, consisting three IBM system x3550 servers, in which VMware ESXi is deployed as hypervisor. VM1, Dom0#1 is deployed in server1, VM2, Dom0#2 is deployed in server2, vSWC and NOX are deployed in server3 for control plane. CNA adopts vSWM to achieve MAC-o-IP overlay, which is deployed in Dom0#1 and Dom0#2. As Figure 10 depicts, the capsulation overhead is very small, the ping RTT (Round-Trip Time) from VM1 to VM2 is less than 1ms, because for most case the capsulation rules could be cached in distributed DHT of Dom0#1. Centralized controller vSWC generates longer forwarding latency up to 4ms or even longer.

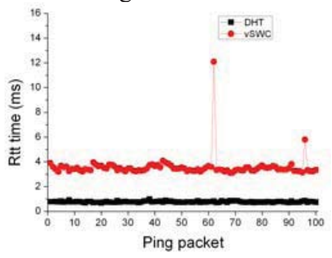


Figure 10. Packet forwarding Rtt.

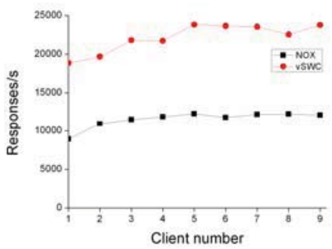


Figure 11. Throughput of vSWC and NOX

Figure 11 depicts the control plane throughput comparison between vSWC and NOX, both implemented by C/C++. Up to 9 clients (implemented in VMs) are used to flood control messages to the controllers, for stress performance test. The average response rate of vSWC is up to 20,000 Responses per second, which is twice more than that of NOX. The reason is that NOX is complex and could not be optimized for particular tasks. Instead, vSWC is implemented fast dedicatedly for mapping table search and easy to be put into the kernel space.

V. CONCLUSION

CNA is a scalable network architecture for multi-tenants data centers. The Network infrastructure is divided into the smart edge and a performance oriented core. This allows the network intelligence to be deployed in the most cost-effective way, without any assumption of network topology. CNA provides an agile network virtualization scheme which allows the multi-tenant users to define their own logical networks on demand, respectively, on top of the same physical infrastructure. A hybrid control model is proposed, in which, the distributed auto-control domains are "hyper-controlled" by the centralized controller. This not only remains the fully agile centralized control, but also avoids the reliability issues and scalability issues.

REFERENCES

[1] Bob Laliberte. Enabling VM Mobility via Intelligent, Automated, Virtual Machine Aware Networking Solutions. March, 2009

[2] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel and S. Sengupta. VL2: a scalable and flexible data center network. In Proc. of ACM SIGCOMM, 2009

[3] R. N. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya and A. Vahdat. PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric. In Proc. of ACM SIGCOMM, 2009

[4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner. OpenFlow: enabling innovation in campus networks. In Newsletter, ACM SIGCOMM Computer Communication Review, Volume 38 Issue 2, April 2008

[5] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, S. Shenker. NOX: towards an operating system for networks. In Newsletter, ACM SIGCOMM Computer Communication Review, Volume 38 Issue 3, July 2008

[6] J. Mudigonda, P. Yalagandula, J. Mogul, B. Stiekes, Y. Pouffary. NetLord: A Scalable Multi-Tenant Network Architecture for Virtualized Datacenters. In Proc. of ACM SIGCOMM, 2011

[7] F. Hao, T. V. Lakshman, S. Mukherjee, H. Song. Secure cloud computing with a virtualized network infrastructure. In Proc. of the 2nd USENIX conference on Hot topics in cloud computing, 2010

[8] C. Kim and J. Rexford, "Revisiting Ethernet: Plug-and-play made scalable and efficient," in Proc. of IEEE LANMAN, 2007.

[9] R. Perlman, "Rbridges: Transparent routing," in Proc. of IEEE INFOCOM, 2004.

[10] M. Yu, J. Rexford, M. J. Freedman, J. Wang. Scalable flow-based networking with DIFANE. In Proc. of ACM SIGCOMM, 2010

[11] M. Casado, T. Koponen, R. Ramanathan, S. Shenker. Virtualizing the network forwarding plane. In Proc. of the Workshop on Programmable Routers for Extensible Services of Tomorrow, 2010

[12] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker. Onix: a distributed control platform for large-scale production networks. In Proc. of the 9th USENIX conference on OSDI 2010.

[13] C. Kim, M. Caesar, J. Rexford. Floodless in seattle: a scalable ethernet architecture for large enterprises. In Proc. of ACM SIGCOMM, 2008

[14] http://www.ietf.org/rfc/rfc3031.txt

[15] http://wiki.xensource.com/xenwiki/Dom0

[16] Y. Qi, F. He, K. Wang, X. Chen, J. Fong, F. Xie,Y. Shao, Y. Gao, Y. Xue, J. Li. LiveSec: OpenFlow-based. Security management for production networks. In Proc. of the IEEE INFOCOM, 2011