Multi-tenant Network Monitoring Based on Software Defined Networking

Aryan TaheriMonfared and Chunming Rong

Department of Electrical Engineering and Computer Science
University of Stavanger, Norway
{aryan.taherimonfared,chunming.rong}@uis.no

Abstract. In any cloud service model multiple stakeholders are involved with different roles in the service provider-customer relationship. One service provider can use other services at the same time. It is vital to distinguish between stakeholders activities such as their communication in the network. Thus, there should be a monitoring system, which knows about stakeholders, their characteristics, and provisioned resources by them at any point of time. In most cases, traditional monitoring solutions does not have the capability to understand the difference between involved parties, while they're orchestrated to achieve a desired result (e.g. delivering a service for the end user).

In this study an improved architecture of the networking service for a cloud platform is introduced. Software Defined Networking (SDN), Network Virtualization, and traditional methods are adapted for gathering evidence and auditing activities on a per-tenant basis. Four approaches are investigated for monitoring and identifying tenants' traffic in an infrastructure. They are discussed and implemented to fulfil the cloud's network monitoring requirements. The focus is on the effectiveness of software defined networking for monitoring tenants' virtual networks.

Keywords: Software Defined Networking, Network Virtualization, OpenFlow, Network Monitoring, Multi-Tenant.

1 Introduction

In a virtualized environment, physical network resources can provide several virtual networks (VN). Traditional Internet Service Providers (ISPs) have two major roles, consisting of: physical infrastructure management and end-to-end service delivery. Network virtualization is realized by separating these roles and assigning them to different entities. In such an environment, an Infrastructure Provider (InP) is responsible for the physical infrastructure management, and a Service Provide (SP) is responsible for delivering the end-to-end network service [1]. Functionality decoupling leads to co-existence of multiple virtual networks which can be created by different SPs. These virtual networks are isolated and may span multiple InPs [2].

Network virtualization can be seen as a utility for studying new networking architecture or an important attribute of the new architecture [3]. The main idea

R. Meersman et al. (Eds.): OTM 2013, LNCS 8185, pp. 327–341, 2013.

behind network virtualization can be implemented using multiple technologies [1], such as: Virtual Local Area Network (VLAN) [4], Virtual Private Network (VPN) [5], active and programmable networks [6], overlay networks, and Software Defined Networking (SDN) [7]. SDN is a platform for network virtualization [8] that abstracts the hardware from the programming interfaces [9].

In the Infrastructure as a Service (IaaS) model, a customer has a set of virtual machine instances distributed over geographical regions. Instances' connectivity can be realized using Network Virtualization. A proper network monitoring solution must distinguish between customers' activities in the infrastructure's network. This is challenging due to the inherited properties of virtualization (e.g. utilizing shared pool of resources). Observation points should be placed in multiple layers with the functionality to monitor in different granularities. This study investigates applicability of existing monitoring techniques to the new computing model. Traditional techniques are improved using a network controller (e.g. an OpenFlow controller), which provides a unified view of networking substrates as well as the information about provisioned resources by each tenant.

1.1 Related Works

There has been a few efforts on the monitoring aspect of SDN, either using SDN for monitoring an infrastructure, or evaluating new characteristics of SDN enabled networks. Jose et al. [10] propose a low overhead measurement framework for commodity network switches that support OpenFlow. The solution is tuned to identify large traffic aggregates. Yu et al. [11] designed a push-based monitoring system using control messages sent to the OpenFlow controller by switches. It estimates links utilization in a passive way by analysing the stream of PacketIn and FlowRemoved messages. Ballard et al. [12] introduce OpenSAFE as a reliable and high performance approach of routing traffic for security monitoring purposes. In addition, a language (ALARMS [12]) is defined for management of network monitoring equipments. Braga et al. [13] presented a low overhead detection method for DDoS flooding attacks, which uses self organizing maps. The method periodically polls flow records from all OpenFlow enabled switches, and extract DDoS related features. Tootoonchian et al. [14] designed an OpenTM for traffic matrix (TM) estimation using capabilities in OpenFlow switches. TMs are crucial for anomaly detection in a network. Based on the routing information provided by the OpenFlow controller, OpenTM chooses a set of switches for querying. An efficient querying strategy lowers the overhead in network equipment.

The rest of the paper is structured as follows: Section 2 explains a common cloud platform architecture. Section 3 gives an overview of the network monitoring in a cloud platform with visibility into tenants' activities. Section 4 discusses challenges of designing such an awareness and monitoring system, as well as shortcomings of existing techniques. Then, Section 5 proposes four techniques to overcome challenges and shortcomings. Finally, Section 7 concludes the paper and compares proposed techniques.

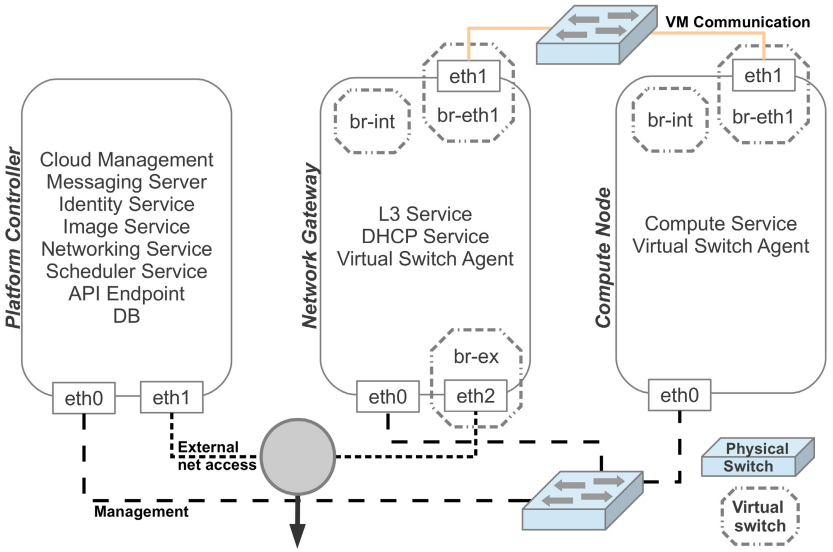

Fig. 1. Abstract Architecture

2 Abstract Cloud Platform Architecture

In the following, an abstract architecture of a cloud environment is explained. The architecture is the basis for our testbed. Later, the architecture is improved by delivering the networking service using SDN.

Three types of nodes are introduced in the architecture (Figure 1), comprising: Platform Controller, Network Gateway, and Compute Node. The platform controller runs core services, such as: inter-module message passing, authentication and authorization, virtual machine image management, basic networking service, job scheduling, and database service. Although core services are centralized in a single node, each service can be replicated on multiple nodes. The network gateway provides core networking services, including: DHCP service, L3 connectivity to the external network. The compute node hosts virtual machine instances and handles their network connectivity through a dedicated virtual switch.

From the networking perspective, four types of logical network are available that serves diverse requirements: infrastructure management, VM's communications, API access, and VMs access to the external networks.

Figure 2 gives more details about the network gateway and the compute node. The compute node has two and the network gateway has three virtual switches. Virtual switches are responsible for inter-VM connectivity. For the sake of simplicity one tenant (i.e. *Tenant A*) in a single compute node is depicted. Each tenant in the cloud platform has a dedicated virtual domain in the network gateway. The domain consists of virtual routers and networking services required for VMs connectivity and operation.

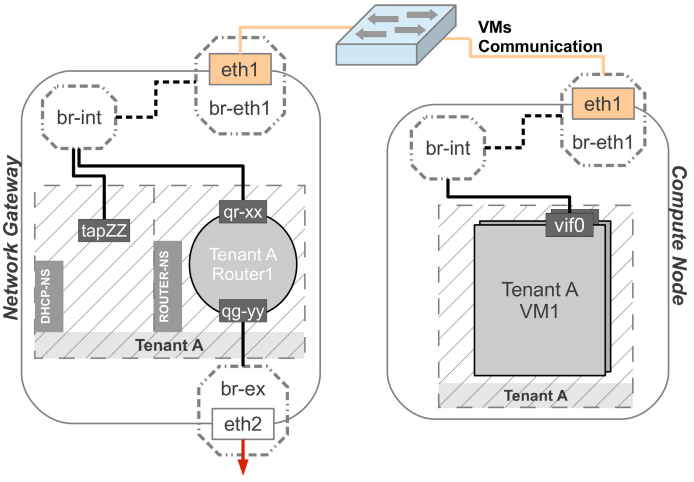

Fig. 2. Architecture Networking Details

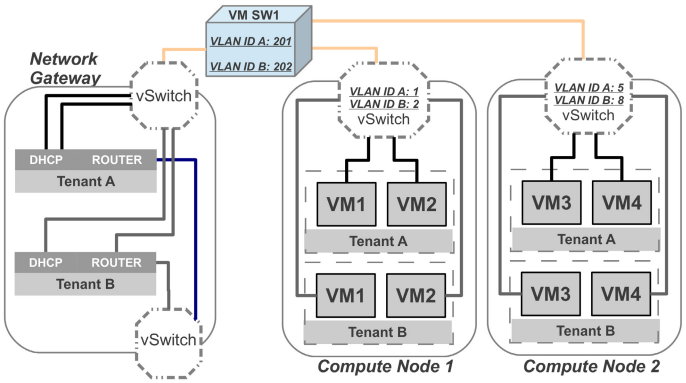

Fig. 3. VLAN Configuration Details

As depicted in Figure 3, *Tenant A* has several VMs distributed over multiple compute nodes. It has dedicated VLAN IDs (VIDs) on the physical switch (i.e. VM Switch 1), and virtual switches. VIDs on the physical and virtual switches may not be identical, since they're in different domains. The virtual switch in the compute node is responsible for translating VIDs between the physical and virtual domain. The mapping information is provided by the controller node, and is dependent on the tenant. All VMs, belonging to a particular tenant, are in the same broadcast domain, although they might be in multiple compute nodes.

In the testbed in Figure 1, the OpenStack Compute project (Nova[1]) provides the compute virtualization platform, and the networking project (Quantum[2])

[1] `http://www.openstack.org/software/openstack-compute/`
[2] `http://www.openstack.org/software/openstack-networking/`

is responsible for the networking services. Virtual switches are Open vSwitch[3] instances controlled by the Quantum server and they support the OpenFlow standard [15]. This feature is enabled to build a unified view of our network and fulfil the requirements of our monitoring solution (5.2). In addition, virtual switches are capable of exporting NetFlow[16], and sFlow[17] data, which are used for maintaining network awareness.

3 Tenant Based Network Monitoring Overview

In a distributed environment such as a cloud several stakeholders are involved. Stakeholders can be both service providers and consumers at the same time. Complexity of these interactions and the new characteristics of the networking architecture hinder the adaptation of traditional monitoring approaches. The following section explains our approach for monitoring a networking environment that supports network virtualization.

Monitoring can be done in different layers. By separating the layers based on the provider type, there will be three layers: service provider layer, platform provider layer, and infrastructure provider layer. There is no clear cut between these types and a single provider can cover several responsibilities.

- **The Infrastructure provider** is responsible for providing underlying substrates, such as physical compute nodes, networking equipment, cooling facilities, etc. (e.g. a data center provider). These components might be heterogeneous and distributed over a variety of geographic locations, under multiple jurisdictions.
- **The Platform provider** delivers functionalities which utilizes underlying components. It builds a unified view of distributed substrates for the upper layer entities. As an example a platform provider is responsible for deployment, maintenance, and operation of a cloud platform (e.g. OpenStack[4]), or a network virtualization platform (e.g. an SDN environment using standards such as OpenFlow [15])
- **The Service provider** uses functionalities exposed by the platform provider to build a variety of services. The provider can deliver one or more services from the cloud service models (e.g. Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS) [18]).

However, in a cloud environment where several stakeholders are involved, fine-grained monitoring capability is crucial. It should be possible to monitor all resources used by a specific tenant (e.g. service customer). As an example, in a security incident the responsible Computer Emergency Response Team (CERT) should be able to monitor resources which are or have been provisioned by a tenant. Switches, routers, and interfaces (either physical or virtual) are considered to be resources, and a proper set of observation points should be identified for fine-grained monitoring.

[3] `http://openvswitch.org/`
[4] `http://www.openstack.org`

This study concentrates on the Infrastructure as a Service model of Cloud. Therefore, network monitoring approaches for auditing low level resources (e.g. virtual machines, block storages) are investigated.

4 Tenant Based Network Monitoring Challenges

New generations of cyber-attacks use sophisticated communication methods. The communication period is short and the traffic may not be significant. As an example Stuxnet uses five methods to propagate itself: peer-to-peer communication, infecting WinCC machine, network shares, MS10-061 Printer Spooler, and MS08-067 Windows Server Service [19]. None of these methods employ heavy scanning or long period communication. Thus, it is crucial to have a complete view of flows, and active services. Berthier et al. [20] proposes a solution for maintaining such a view, and we will investigate its application in a virtualized environment.

NetFlow services [16] provide information on IP flows from network elements, such as routers and switches. Flow data are sent from data networks to collectors. The data has detailed information about resource usage and can be helpful for obtaining network awareness. According to [16], a flow is considered to be unidirectional and bidirectional flows (Biflows) is a sequence of packets that pass through a network in both directions between two endpoints [21]. Thus, Biflows are more informative for security purposes. Although it is possible to determine Biflows in the collector using NetFlow data, it's more efficient to have it as an extension in the protocol [21]. The IP Flow Information Export (IPFIX) protocol [22] exports Biflows, though it is not widely used in substrates and is not supported by Open vSwitch. Thus, analytic tools are developed for identifying Biflows.

NetFlow is considered to be the most suitable protocol because of its accuracy. Virtual switches are configured to export their NetFlow data to a collector for further analysis. Nfcapd, nfdump[5], nfsen[6], and NFSight [20] are used for the initial collection and processing of data.

Monitoring the virtual switch in each node provides better visibility of the VMs communication. There are two virtual switches in each compute node, `br-eth1` and `br-int`. `br-eth1` is our observation point, since it is connected to the Top of the Rack (ToR) switch and handles all VMs' traffic inside the physical machine. On the network gateway, three virtual switches are configured, and two of them are monitored, `br-eth1`, and `br-ex` (external network access for VMs).

As depicted in Figure 4, the aggregated traffic on a virtual switch is visible. However, we can not differentiate tenants' traffic yet. Since tenants may have used overlapping IP addresses, traffic classification based on network addresses is not sufficient. Based on the current architecture, tenants can be distinguished from each other using their VIDs, which have been assigned by virtual and physical switches. These IDs are not essentially identical among switches. Moreover,

[5] `http://nfdump.sourceforge.net/`
[6] `http://nfsen.sourceforge.net/`

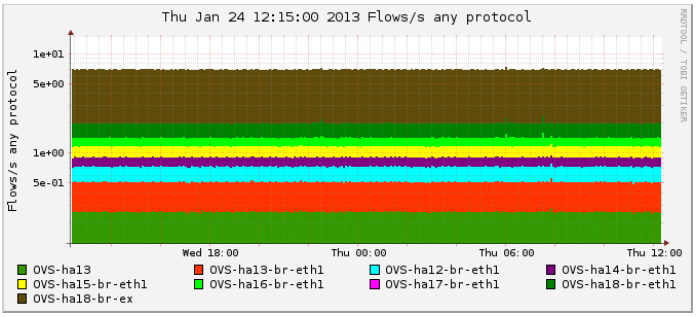

Fig. 4. Network traffic observed by a monitoring tool without the knowledge about tenants' co-existence

this technique is useful when VLAN tagging has been employed for tenants' isolation. VLAN tagging may not be implemented everywhere, due to its disadvantages (e.g. VLAN table scalability) for an environment with a large number of customers. For instance, the FloodLight OpenFlow controller does not use VLAN tagging for creating isolated logical layer 2 networks.

The following part summarize challenges of network awareness in a cloud platform, when multiple tenants coexist.

Elasticity and Dynamicity of The Environment: Tenants provision resources on demand and release them when there is no more need. Provisioned resources can scale up or down to meet the requirements, and can be migrated to other physical substrates or regions. Scalability, elasticity, and migration of virtual resources in the cloud, make monitoring much harder. A tenant is not bounded to a particular region or hardware. Therefore, the monitoring solution must be able to provide the tenant based statistics, independent of the physical location.

Complex Stakeholder Relations: In the new computing model, several stakeholders are involved and their interactions are complicated. Traditional monitoring solutions are not adapted to this model, and they can not logically distinguish between tenants in a cloud environment.

Incident Handling and Network Forensics: In an incident, the forensic team asks for relevant information. If the incident happens in a tenant domain, only data related to that specific tenant should be exposed to the forensic team. Proper clustering and anonymization of data become crucial, due to the privacy and security of other tenants.

Reliable Monitoring and Non-Repudiability: A malicious tenant may try to forge its networking specifications. The motivation can be concealing its own identity

or impersonating other tenants. Monitoring approaches should not solely rely on the information provided by tenants, but provide mechanisms for assessing the originality of data. For instance, IP addresses can be spoofed by a tenant, however, VIDs are assigned by switches out of tenants' domains.

Hardware and Software Limitations: Existing hardware and software may not fully support new protocols and standards (e.g. NetFlow v9 and OpenFlow are not supported widely). Proposed techniques for the new model should not rely on standards or protocols, which are not mature yet.

Invisibility of tenants' internal communications: Instances in a tenant communicate with each other or with external entities (e.g. other tenants, off-premises entities). Observation points must be chosen such that they cover all types of traffic.

Tenants' Interests for Monitoring Services: Some companies are interested in using external services and moving their resources off-premises. A big obstacle on the migration is the lack of visibility and control over resources. Offering tenant based monitoring services can relax the issue, and satisfies more customers.

5 Tenant Based Network Monitoring Solutions

There should be multiple techniques to fulfil requirements of a heterogeneous infrastructure. Four approaches in two categories are proposed for a tenant's network monitoring in a cloud infrastructure:

- Non-OpenFlow enabled components
  - Tenants monitoring using IP datagram header.
  - Tenants monitoring using data link frame header.
  - Tenants monitoring using virtual components in the network gateway.
- OpenFlow enabled components

Following sections discuss each approach and the advantages and disadvantages.

5.1 Non-OpenFlow Enabled Components

SDN is in its early stages, and OpenFlow capable components are the realization tools. Thus, we can not assume that all equipments in an infrastructure will support the standard in a near future. Three techniques are explained for answering the network monitoring demands. Although the technology behind these techniques are not new, they should be adapted to the new model.

Date flow start	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Packets	Bytes	Flows
2013-01-30 12:09:41.225	243.430	TCP	10.10.11.2:44511	->	10.10.11.9:22	286	19316	19
2013-01-30 12:09:41.225	243.430	TCP	10.10.11.9:22	->	10.10.11.2:44511	280	44144	19
2013-01-30 12:10:12.770	248.509	UDP	10.10.11.4:68	->	10.10.11.2:67	12	4152	12
2013-01-30 12:10:36.606	240.280	UDP	10.10.11.2:67	->	10.10.11.9:68	12	4392	12
2013-01-30 12:10:12.824	248.487	UDP	10.10.11.2:67	->	10.10.11.4:68	12	4392	12
2013-01-30 12:10:36.581	240.251	UDP	10.10.11.9:68	->	10.10.11.2:67	12	4152	12
2013-01-30 12:10:25.732	240.458	ICMP	10.10.11.3:0	->	10.10.11.2:3.3	10	3940	10
2013-01-30 12:10:18.020	240.494	ICMP	10.10.11.5:0	->	10.10.11.2:3.3	10	3940	10
2013-01-30 12:10:18.020	240.493	UDP	10.10.11.2:67	->	10.10.11.5:68	10	3660	10
2013-01-30 12:10:48.041	240.536	ICMP	10.10.11.7:0	->	10.10.11.2:3.3	10	3940	10

Fig. 5. Tenants monitoring using IP datagram header fields

Tenants Monitoring Using IP Datagram Header. Each tenant in a cloud platform can have several networks and subnets. The tenant's traffic can be identified using the IP addresses assigned to its subnets. The virtual switch (i.e. Open vSwitch) supports both sFlow and NetFlow v5 monitoring protocols. Most physical switches supports at least one of these protocols as well. Given that NetFlow is supported by all switches in an infrastructure, we can collect raw data and extract tenant's statistics from them.

Tenant's networks and subnets list can be retrieved from the networking service (e.g. OpenStack Quantum). Assigned IP addresses to the tenant is in the subnets information. The information can be used for processing the monitoring data that are collected from observation points. The analysis output shows the statistics for all instances which belongs to a particular tenant. For illustration, *Tenant A* is attached to two subnets, and we are interested in the subnet 10.10.11.0/24. Figure 5 shows *Tenant A*'s flows in all switches.

The implementation is simple, and requires several queries to the networking service and the monitoring data collector. Despite its simplicity, it's quite useful as it will preserve all capabilities provided by sFlow or NetFlow, as well as the visibility into tenants' traffic and behaviour.

However, the statistics may not be accurate or reliable. Because tenants may have overlapping IP address or a nefarious tenant can forge IP addresses. Thus, a tenant can impersonate another one during an attack. Auditing mechanisms fail to distinguish between the attacker and the impersonated tenant. It leads to accountability issues and contradicts non-repudiability principle. The same scenario can be imagined for billing issues, when tenants are charged according to their traffic. Overlapping IP addresses lead to inconsistent billing information and traffic measurement. So, the solution should be extended to cover these shortcomings.

Tenant Monitoring Using Data Link Frame Header. Monitoring tenants' traffic using VLAN information is more robust. VLAN tags are located in the data link layer frame header, and tenants can not forge them. Since, the virtual switch maintains a binding between instances in a particular tenant and the VID. In addition, using data link layer frame information makes the solution independent of network layer protocols. Thus, it will not be limited to IP.

```
Date flow start          Duration Proto      Src IP Addr:Port          Dst IP Addr:Port  Flows SVlan DVlan
2013-01-30 17:16:14.757   215.000 TCP         10.10.11.2:44649 <->       10.10.11.9:22        8     3   200
2013-01-30 17:16:48.757    23.000 TCP         10.10.11.9:42626 <->    91.189.92.200:80      177     3   200
2013-01-30 17:15:09.757     0.000 TCP         10.10.11.9:34536 <->    91.189.92.202:80        1     3   200
```

Fig. 6. Tenant monitoring using data link frame header fields

Table 1. NetFlow version 5 Flow Record Format [23]

Bytes	Contents
0-3	srcaddr
4-7	dstaddr
8-11	nexthop
12-13	input
14-15	output
16-19	dPkts
20-23	dOctets
24-27	First
28-31	Last
32-33	srcport
34-35	dstport
36	pad1
37	tcp_flags
38	prot
39	tos
40-41	src_as
42-43	dst_as
44	src_mask
45	dst_mask
46-47	pad2

As it was explained earlier, the virtual switch supports sFlow and NetFlow v5. NetFlow v5 is not suitable, as it doesn't deliver VLAN information, as shown in Table 1. For this approach sFlow or NetFlow v9/IPFIX must be used.

The networking service stores the mapping of tenants and physical switches VIDs, but tenants' VIDs in virtual switches are not known to it. An agent in each compute node should expose the VID mapping (i.e. tenant : VID : virtual switch ID). Then, the mapping can be used for processing sFlow monitoring data. The implementation is not trivial, because tenants' VIDs may not be identical among switches. First, we identify tenants' VIDs in each switch. Then, a query is created for aggregating the statistics from all virtual switches. As an example, *Tenant A* is using VID 3 in the virtual switch OVS-ha13 and VID 200 on the physical switch (Figure 6).

Tenant Monitoring Using Virtual Components in the Network Gateway. According to the architecture, tenants have their own dedicated network name-spaces in the network gateway. Tenant's virtual routers and interfaces are

visible in the its name-space. Components in the tenant's name-space are responsible for delivering networking services such as DHCP and layer 3 connectivity.

These components (in the hatched area inside the network gateway in Figure 2) can be considered as observation points (i.e. `Route1, tapZZ, qr-xx, qg-yy`). But only a part of the tenant traffic is passing through these points, including: DHCP traffic, ingress/egress traffic from/to tenant's network. In other words, internal communication of instances inside a tenant is not observable. Therefore, it has several disadvantages: *a)* monitoring these points will not provide complete visibility of the tenant's traffic, *b)* tenant's traffic in each virtual switch is not distinguishable, *c)* deploying multiple redundant network gateways can disrupt the visibility and will require additional data correlations.

Although the approach has multiple drawbacks, its benefits make it interesting to be offered as a service. The implementation is simple and scales well. It has low overhead and the resource consumption is considerably less than other solutions. A tenant can use the service for monitoring communications with other networks. Networks can be either other tenants' networks or destinations outside the cloud platform. This is an efficient solution when all internal entities inside a tenant are trusted, or when there is no interest in the internal communication.

5.2 OpenFlow Enabled Components

The networking service is provided by the Quantum project, that means all virtual switches are configured by the Quantum. Although Open vSwitch supports OpenFlow, activating the controller in the virtual switch overwrites the previous configuration. Thus, we need to commit Quantum's command through an OpenFlow controller. In this architecture, the controller is connected to all capable switches and has a unified view of the network. Therefore, the monitoring node communicates with the controller to build per-tenant view of the network and generates monitoring information for each tenant, Figure 7.

There are multiple Quantum plug-ins providing this functionality, such as: Ryu OpenFlow Controller plugin[7], NEC OpenFlow plugin[8], Floodlight OpenFlow Controller[9] plugin. The most suitable controller, for our use-case in this period, is Floodlight controller. Floodlight has an application module called Virtual Network Filter (VNF) that provides virtualized layer 2 networks over a single layer 2 domain. VNF listens for PacketIn messages, and verifies the source and destination MAC addresses. If they belongs to the same virtual network, the packet will pass; otherwise it will be dropped. This is a simple approach with some limitations, such as: virtual networks can only be defined in a single physical layer 2 domain, and limited number of gateways in a tenant network[24].

The list of tenants are retrieved from the identity service (Figure 8 (1)), and their networks and subnets are loaded from the networking service (Figure 8 (4, 5)). Then, we need to find devices in the network and their attachment

[7] `http://www.osrg.net/ryu/`
[8] `http://wiki.openstack.org/Quantum-NEC-OpenFlow-Plugin`
[9] `http://floodlight.openflowhub.org/`

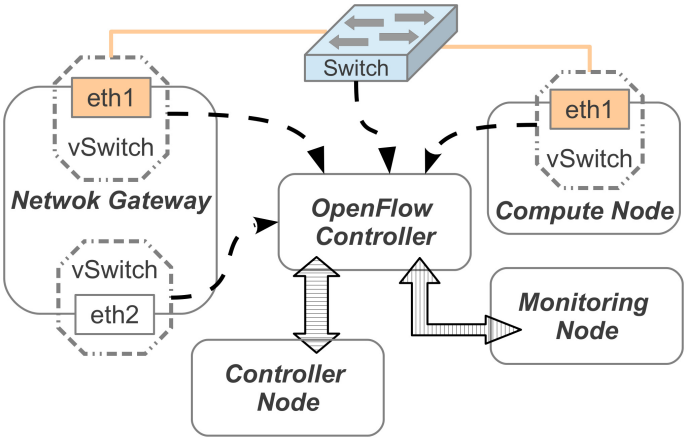

Fig. 7. OpenFlow enabled Lab Architecture

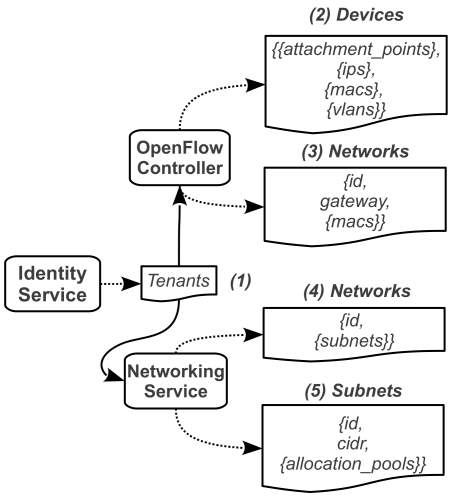

Fig. 8. Mapping switch ports to tenants

points (Figure 8 (2, 3)). Finally, the information required for monitoring tenants' network activity is ready. The information is in form of:

$$tenant_a : \{device_x : \{(mac_\alpha, ip_\alpha, port_\alpha, switch_\alpha)\}\}$$

Multiple types of report can be generated, including: *a)* a simple view of a tenant's traffic in the network by aggregating statistics of ports belonging to the same tenant, *b)* a sophisticated tenant's flow information by analysing NetFlow data using the information provided by the controller. The proof of concept code is developed and results are satisfactory.

6 Discussion

There are several shortcomings in the design of existing monitoring tools, such as: *a)* lack of tenant logic with complex stakeholder relationships, *b)* lack of flexibility for the new characteristics of the cloud model (e.g. elasticity, on-demand provisioning/release), and *c)* lack of robustness for a heterogeneous infrastructure. Four approaches in two category are discussed to cover requirements imposed by the model's criteria: Non-OpenFlow approaches, and the OpenFlow enabled approach. Non-OpenFlow enabled solutions use tenants' internet layer attributes (e.g. IPs), data link frame attributes (e.g. VLAN IDs), virtual devices in the Linux network stack namespace.

The first category uses core cloud platform APIs and standard monitoring data. Required parameters for building a tenant-based view are gathered, and filters are created. Parameters define networking attributes of a tenant, such as IP subnets, VIDs, and MAC addresses. Then, monitoring data are analysed using these parameters, and a tenant-based view of the network activities is created. The second category benefits from the OpenFlow controller's capabilities. The controller is responsible for the control plane of switches in the network, and it is aware of the tenant logic in the platform. Tenant awareness of the controller helps in developing a robust solution for recognizing and distinguishing tenants' networking behaviours.

Advantages, disadvantages, and requirements of of proposed solutions are briefly explained in Table 2.

7 Conclusion

In a multi-tenant model like the cloud computing, several stakeholders are involved. Distinguishing tenants' activities and provisioned resources, in the time domain, is the key factor for accountability, anomaly detection, as well as load-balancing. We have started with analysing the IaaS service model, and several stakeholders in the provider layers are identified. At each layer, monitoring can be performed with different granularities. Depending on the coarseness, the result exposes various characteristics of the system.

The architecture is powered by virtual switches and OpenFlow controllers. Two types of solutions are introduced, addressing a heterogeneous environment. The first type introduces methods for monitoring tenants' activities, when an OpenFlow controller is not available. In these methods, tenants' specifications are retrieved from the networking and identity service. Then, raw monitoring data are processed for building per-tenant traffic statistics. The second type benefits from an OpenFlow controller to build the per-tenant view. In this approach, the controller provides the unified view of the network, and is aware of the tenant logic.

An obstacle for deploying these mechanisms in a large scale production environment, is the storage and processing of large data sets. We will continue to enhance our monitoring solution by collecting flow information in a distributed

Table 2. Solution comparison (NS: Networking Service, IS: Identity Service)

Solution	Additional Requirements	Advantages	Disadvantages
Tenants monitoring using IP datagram header	* NetFlow data * NS Access * IS Access	+ Ease of implementation + Complete view of tenants IP activity + Per vSwitch statistics	- Prone to IP spoofing - Lack of accuracy
Tenants monitoring using data link frame header	* sFlow data * Monitoring agent in each vSwitch * NS access * IS access	+ Network protocol independence + Accuracy + Reliability against IP spoofing + Complete view of tenants network activities + Per vSwitch statistics	- Not working with NetFlow v5 - An agent should be deployed in each compute node
Tenants monitoring using virtual components in the network gateway	* NetFlow data * Monitoring agent in each network gateway * NS access * IS access	+ Simplicity + Complete view of ingress/egress traffic from/to the tenant network + Network protocol independence + Reliability against IP spoofing	- No per vSwitch statistics - No visibility to the internal traffic of a tenant - An agent should be deployed in each network gateway
Tenants monitoring using OpenFlow controller	* OpenFlow controller * NS OpenFlow agent * NetFlow data * NS access * IS access	+ Transparent integration + Network protocol independence + Complete view of tenants network activities + Accuracy + Flexibility + Reliability against IP spoofing	- Complicated implementation - Monitoring solution implementation depends on the controller

data-intensive processing framework. Thus, we will be able to divide a query task and execute it over a cluster of commodity servers. This leads to efficient statistical analysis of monitoring data. The result will be used for providing real-time feedback to the OpenFlow controller for updating networking decisions.

Acknowledgment. The authors would like to thank Martin Gilje Jaatun from SINTEF ICT, who provided valuable comments and assistance to the undertaking of this research.

References

1. Chowdhury, N.M.K., Boutaba, R.: A survey of network virtualization. Computer Networks 54(5), 862–876 (2010)
2. Chowdhury, N.M.K., Boutaba, R.: Network virtualization: state of the art and research challenges. IEEE Communications Magazine 47(7), 20–26 (2009)
3. Anderson, T., Peterson, L., Shenker, S., Turner, J.: Overcoming the internet impasse through virtualization. Computer 38(4), 34–41 (2005)

4. L.S. Committee: IEEE standard for local and metropolitan area networks virtual bridged local area networks. IEEE Std 802.1Q-2005 (Incorporates IEEE Std 802.1Q1998, IEEE Std 802.1u-2001, IEEE Std 802.1v-2001, and IEEE Std 802.1s-2002), 1–285 (2006)
5. Ferguson, P., Huston, G.: What is a VPN?. The Internet Protocol Journal 1(1) (1998)
6. Campbell, A.T., De Meer, H.G., Kounavis, M.E., Miki, K., Vicente, J.B., Villela, D.: A survey of programmable networks. ACM SIGCOMM Computer Communication Review 29(2), 7 (1999)
7. Lantz, B., Heller, B., McKeown, N.: A network in a laptop, pp. 1–6. ACM Press (2010)
8. Drutskoy, D., Keller, E., Rexford, J.: Scalable network virtualization in software-defined networks. IEEE Internet Computing, pp. 99, 1 (2012)
9. McKeown, N.: Software defined networking. In: IEEE InfoCom (2009)
10. Jose, L., Yu, M., Rexford, J.: Online measurement of large traffic aggregates on commodity switches. In: Proceedings of the 11th USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services, Hot-ICE 2011, p. 13. USENIX Association, Berkeley (2011)
11. Yu, C., Lumezanu, C., Zhang, Y., Singh, V., Jiang, G., Madhyastha, H.: FlowSense: monitoring network utilization with zero measurement cost (2013)
12. Ballard, J.R., Rae, I., Akella, A.: Extensible and scalable network monitoring using OpenSAFE. In: Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking, INM/WREN 2010, p. 8. USENIX Association, Berkeley (2010)
13. Braga, R., Mota, E., Passito, A.: Lightweight DDoS flooding attack detection using NOX/OpenFlow, pp. 408–415. IEEE (October 2010)
14. Tootoonchian, A., Ghobadi, M., Ganjali, Y.: OpenTM: traffic matrix estimator for OpenFlow networks. In: Krishnamurthy, A., Plattner, B. (eds.) PAM 2010. LNCS, vol. 6032, pp. 201–210. Springer, Heidelberg (2010)
15. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: OpenFlow. ACM SIGCOMM Computer Communication Review 38(2), 69 (2008)
16. Claise, B.: Cisco Systems NetFlow Services Export Version 9. RFC 3954 (Informational) (October 2004)
17. Phaal, P., Lavine, M.: sFlow version 5. Technical report (July 2004)
18. Armbrust, M., Stoica, I., Zaharia, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A.: A view of cloud computing. Communications of the ACM 53(4), 50 (2010)
19. Falliere, N., Murchu, L.O., Chien, E.: W32.Stuxnet dossier. Technical, Symantec, Version 1.4 (February 2011)
20. Berthier, R., Cukier, M., Hiltunen, M., Kormann, D., Vesonder, G., Sheleheda, D.: Nfsight: netflow-based network awareness tool. In: Proceedings of the 24th International Conference on Large Installation System Administration, LISA 2010, pp. 1–8. USENIX Association, Berkeley (2010)
21. Trammell, B., Boschi, E.: Bidirectional Flow Export Using IP Flow Information Export (IPFIX). RFC 5103 (Proposed Standard) (January 2008)
22. Claise, B.: Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information. RFC 5101 (Proposed Standard) (January 2008)
23. Cisco Systems: NetFlow FlowCollector installation and user guide. Technical Report OL-2587-01 (1999)
24. Kanui, B.: FloodLight Virtual Network Filter (October 2012)