Network-Wide Traffic Visibility in OF@TEIN SDN Testbed using sFlow

Shafqat Ur Rehman
Dept. of Computer Sci. and Eng.
Air University
Islamabad, Pakistan.
shafqat.rehman@gmail.com

Wang-Cheol Song*
Dept. of Computer Eng.
Jeju National University
Jeju, Korea
kingiron@gmail.com

Mingoo Kang
Division of Info. and Telecom.
Hanshin University
OSAN-SI, South Korea
kangmg@hs.ac.kr

***Abstract*—This paper provides insights into the traffic flow monitoring system of OF@TEIN (OpenFlow@Trans Eurasia Information Network) testbed. OF@TEIN is software defined networking (SDN) testbed adapted by KOREN (KOrea advanced REsearch Network) and integrated with the research and education networks of several Asian nations including Japan, Malaysia, Thailand, Vietnam, Philippine, etc. Traditional traffic monitoring solutions such as NetFlow, RSPAN ports, Network Packet Brokers (NPBs) can't provide network-wide visibility in OF@TEIN because OF@TEIN is a large multi-tenant testbed deployed over high speed research networks across several countries. Therefore, we have implemented a new sFlow-based flow monitoring system that is tailored to OF@TEIN requirements and can provide real-time L2 to L7 network wide visibility. OF@TEIN uses SDN-based network virtualization to slice the network among multiple concurrent experimenters. Machines in a network slice (or VLAN) communicate with each other using GRE tunnels. Our traffic monitoring system enables monitoring flow spaces of VLANs as well as physical provider network. It utilizes *northbound* interfaces (NBIs) exposed by sFlow-RT analytics engine and FloodLight SDN Controller. It periodically fetches flows statistics from sFlow-RT and stores them in time-series format in Whisper RRD database. Graphite real-time charting tool is used to plot the statistics stored in Whisper.**

***Index Terms*—SDN, OF@TEIN, OpenFlow, sFlow, sFlow-RT, Visibility.**

I. Introduction

Lack of visibility into today's high speed business critical networks is a major issue. Network engineers have struggled with monitoring traffic flows through their networks. They have suffered from the limitations of conventional traffic visibility methods such as port mirroring/SPAN (switch port for analysis) and TAP (Test Access Point). Since early 90s, SPAN(mirror) ports have been widely used to direct packets from any or all ports of a switch/router to a collector (through a single port) for analysis. However, typical SPAN ports can't handle full duplex (FDX) and VLAN monitoring is problematic. There is no guarantee that all the traffic required for proper analysis would be captured. Under specific load conditions, the Ethernet switch treats SPAN data with lower priority than regular traffic and the mirrored frames are arbitrarily discarded [1]. Moreover, SPAN port technology does not scale to today's Gigabit Ethernet (GbE) networks e.g., FDX Gigabit and 10 Gigabit networks. Routers/switches cannot replicate this amount of traffic to the SPAN port at line-rate in real time. SPAN ports can be useful only in low throughput scenarios. An alternative to SPAN ports are network taps. A tap installed between two points A and B in the network enables a third party to get a copy of all the traffic between points A and B passing through the tap.

In traditional systems, a monitoring appliance must be directly connected to a tap or SPAN port. This method makes provisioning network-wide traffic costly, cumbersome and inflexible as it would require installing monitoring appliances at each network segment. Typically, only a small segment of the traffic in a large network can be monitored this way. Network packet brokers (NPBs) are usually used to aggregate traffic from multiple monitoring devices and send it to analysis tools. However, this solution is costly and inflexible especially for high-speed 10/40 Gbps or future 100 Gbps networks as it makes almost impossible for tools to ingest resulting humongous data [8].

Netflow can provide high-level visibility into traffic flow but lacks the details for deep packet analysis. It is an aggregation technology and a single NetFlow v5 UDP datagram can contain up to 30 flows. However, because of its aggregation, we can get only a few details such as source and destination interfaces/ports, source and destination addresses, protocol, total packets, total bytes, flow start and end times, etc. We require high-end routers and switches to send NetFlow data. Sending NetFlow data can overload network resources already stretched thin. It does not support monitoring LAN and VLAN traffic.

Because of the limitations of traditional traffic monitoring technologies, there is a lot of interest in SDN as an alternative and far more flexible and scalable way of monitoring high-bandwidth networks especially data center networks [2] [7]. SDN systems use OpenFlow and sFlow enabled commodity switches with a sophisticated software-based centralized controller [5] which enables network engineers to monitor and engineer network traffic in new ways. These OpenFlow switches are connected to traffic analysis tools such as sFlow-RT [3], sFlowTrend, Ganglia, etc. The network of sFlow enabled switches exports sFlow measurement datagrams to one or more collectors. sFlow collectors enable an SDN application to gain visibility into the traffic across the network. sFlow and

* Corresponding Author

OpenFlow [4] together provide complementary functions for software defined network environments.

Rest of the paper is organized as follows. In Sections II, III and IV, SDN and sFlow technologies in the context of flow monitoring are described. In section V, we provide details of OF@TEIN flow monitoring system. In section VI, we discuss demonstration of the system. Section VII concludes the paper.

II. Background

Traditional network devices perform two types of functions.

- The data plane of the device looks up routing table to decide where to forward packets. It carries user data and uses dedicated ASICSs forward data packets.
- The control plane of the device carries control traffic originated from (or destined to) a network device. It makes decision about where user traffic should be sent. It takes care of tasks such as building routing table/spanning tree, exporting flow statistics, and more. It is the brains of the network and is implemented in software.

Software defined networking (SDN) separates the network Data Plane and Control Plane, permitting external software to monitor and control network resources. Open Southbound APIs like sFlow and OpenFlow are an essential part of this separation, connecting network devices to external controllers, which in turn present high level Open Northbound APIs to SDN applications as shown in Figure 2 [2]. This makes network management and control much more flexible because there are one or few control planes to configure. Network engineers have complete control over the network traffic and can automate orchestration of network services such as bandwidth reservation, load balancing, traffic differentiation, DDoS mitigation, etc.

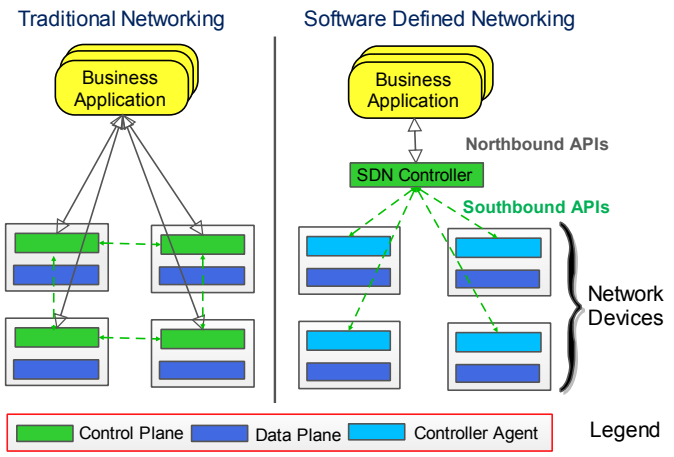

Fig. 1. Traditional vs. Software Defined Networking

As shown in Figure 1, in traditional network architecture, data plane and control plane both reside on the network device. Traffic forwarding rules are set on each device. Network devices do not have visibility of the entire network. In SDN architecture, control plane is taken away from the network device and moved to a centralized controller. The controller has visibility to the entire network and enables network engineers to make forwarding rules based on information about the entire network. Controller communicates policies/rules to the devices via controller agent (which resides in each network device) using a standard protocol usually OpenFlow. Controller exposes northbound APIs that allow business applications to communicate with the controller. In short, SDN consists of four components: controller agent, OpenFlow based southbound APIs, SDN controller and RESTful northbound APIs. Control layer communicates with data layer using OpenFlow and business applications communicate with the control plane using northbound APIs.

The key benefit of SDN is that it allows business applications to program the network behavior to optimize its performance. SDN is a disruptive technology with far reaching impact on network management and control.

III. Flow Monitoring using sFlow

Switches are configured to use sFlow and OpenFlow protocols to communicate with the sFlow Analytics engine (e.g., sFlow-RT [3]) and OpenFlow controller (e.g., Floodlight [4]) respectively in the control plane. Control plane software such as sFlow and OpenFlow controller use Open Northbound APIs to provide summary statistics and control functionality to SDN applications such as Load Balancer, DDoS, etc.

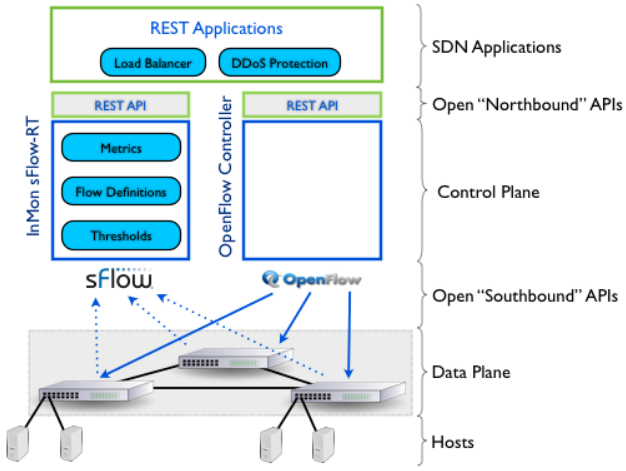

Fig. 2. SDN (Software Defined Networking) Stack

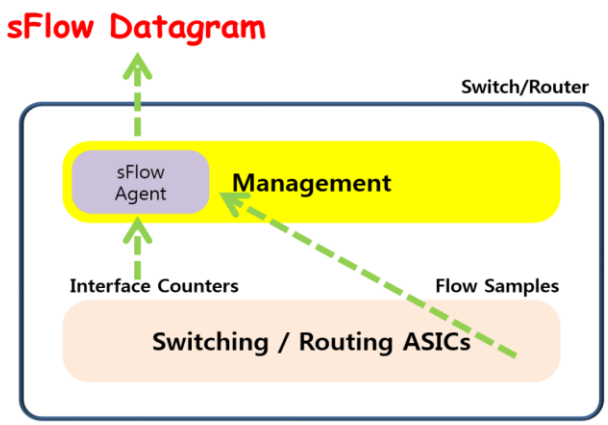

Fig. 3. sFlow Agent embedded in a switch/router

The OpenFlow protocol enables SDN Controller running on a server to gather topology information of a network of

switches and configure the forwarding behavior of these switches. SDN controller builds a graph based model of the network and runs a sophisticated routing algorithm to decide the path of flows through the network. Flow routes decided by the controller are added to the forwarding tables of the switches using the OpenFlow protocol. The sFlow standard is implemented in the switches using a separate ASIC (Application Specific IC) which allows for real time network-wide visibility in the traffic flows. Together, sFlow and Openflow can be used to provide an integrated flow monitoring system where OpenFlow controller can be used to define flows to be monitored by sFlow. Furthermore, metrics from sFlow can be used as feedback by an SDN application to control the forwarding behavior in the switches.

IV. sFlow Traffic Sampling Technology

Before sFlow is a multi-vendor sampling technology embedded within switches and routers. It provides the ability to continuously monitor application level traffic flows at wire speed on all interfaces simultaneously. The sFlow Agent is a software process that runs as part of the network management software within a device as shown in Figure 4. sFlow agent combines interface counters and flow samples into sFlow datagrams that are sent across the network to an sFlow Collector. Packet sampling is typically performed by the switching/routing ASICs, providing wire-speed performance. The state of the forwarding/routing table entries associated with each sampled packet is also recorded.

Using sFlow, traffic samples can be collected from wide range of network devices such as open virtual switches (OVS), physical switches, hosts, etc. sFlow monitoring can be configured on all interfaces of the device with little overhead and sampling rate for each link can be decided according to the monitoring policy.

sFlow agents in network devices use random sampling according to the defined sampling rate and therefore, can be used to monitor high speed networks (Gbps speeds and higher) with quantifiable accuracy. The Sampled data is sent as UDP packets to the specified host and port where sFlow collector software computes summary statistics and possibly display the results graphically.

Figure 4 shows the basic elements of the sFlow system. sFlow Agents throughout the network continuously send a stream of sFlow Datagrams to a central sFlow Collector where they are analyzed by an analytics engine to produce a rich, real-time, network-wide view of traffic flows.

A widely used tool to process sFlow packets received from the network devices is sFlow-RT. It enables real-time visibility into software defined networks. sFlow-RT sits in the control plane of the SDN stack. It converts the received datagrams into actionable metrics or summary statistics based on the flows as defined by the user. A flow of traffic is a set of packets with a common property, known as the flow key, observed within a period of time. The flow key is usually specified by fields from the packet header, such as the IP source and destination address and TCP/UDP port numbers. Flow names are usually used as metrics which are programmatically accessible through RESTful Northbound APIs. Any language that supports HTTP request messages (Perl, Python, Java, Java script, bash, etc.) can be used to retrieve metrics from sFlow-RT. sFlow-RT statistics can be retrieved in JSON format. JSON encoded text based results are easy to read and widely supported by programming tools. Following URL is used to retrieve JSON encoded metrics from sFlow-RT:

http://Server:8008/metric/agents/metrics/json?filter

where server is the host running sFlow-RT, agents are semicolon separated list of host addresses or names, or ALL to include all hosts, metrics are comma separated list of metrics to retrieve and filter is a filter to further restrict the hosts to include in the query.

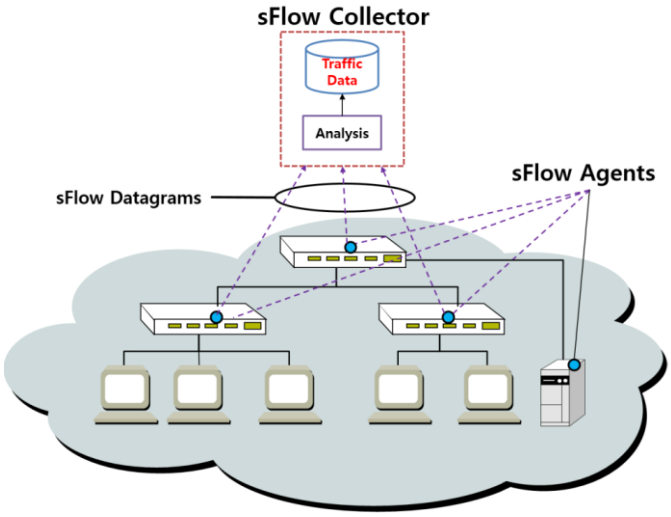

Fig. 4. sFlow Agents and Collector

V. Traffic Visibility in OF@TEIN Testbed

In OF@TEIN, sFlow agent is embedded in OpenFlow enabled physical and virtual network devices. sFlow is configured to capture packets according to a specified sampling rate. It sends samples in the form of measurement datagrams to sFlow-RT. sFlow-RT is real-time analytics engine. sFlow-RT processes the stream of measurement datagram from sFlow in real-time and, therefore, provides real-time summary statistics to application via northbound REST APIs. OF@TEIN employs FlowVisor proxy to create per user slices or virtual networks. FlowVisor acts as a proxy between physical switches and multiple OpenFlow controllers. Each OpenFlow controller controls flows within its own slice. FlowVisor slice the network resources such as link bandwidth, maximum number of forwarding rules, topology and fraction of switch/router CPU. OF@TEIN flow monitoring system supports monitoring of per slice FlowSpace. An experimenter can monitor her own FlowSpace while network administrator can monitor all flowspaces.

Figure 5 demonstrates the data, control and application layers of the current version of OF@TEIN flow monitoring system. We use sFlow-RT as sFlow collector and analytics engine. It receives a continuous stream of sFlow datagrams from network devices and converts them into actionable metrics that are accessible through REST APIs. REST APIs makes it easy for each application to configure flows, retrieve metrics, set thresholds, and receive notifications.

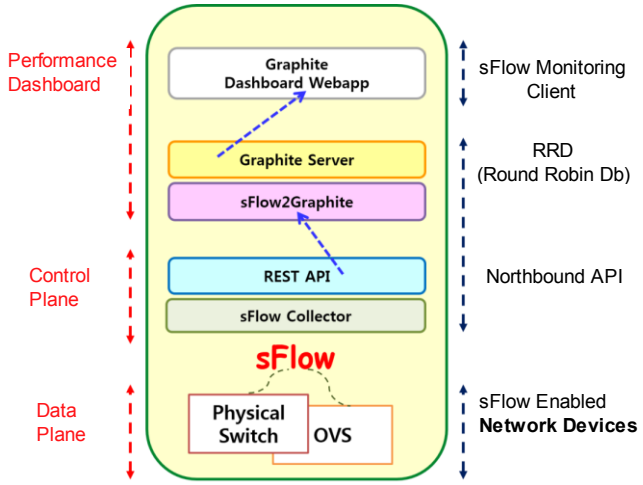

Fig. 5. sFlow monitoring system in OF@TEIN Testbed

We developed following python programs: **parse_flows**, **submit_flows**, **parse_metrics** and **statistics**. Users define traffic flows using xml according to a schema. **parse_flows** parses theses flows, converts xml flow definitions to JSON flow definitions and builds an index with host URL as key and JSON flow definition as value. Then **submit_flows** program is invoked to push the flow definitions to sFlow-RT one by one using the REST commands stored in the index. Summary statistics i.e., metrics against the flow definitions are specified again using xml. **parse_metrics** parses the xml description and builds a REST command that is used to fetch statistics in JSON format from the sFlow-RT Analytics Engine. **statistics** is used internally by parse_metrics to make the REST API call using 'requests' python package and fetch the metrics from sFlow-RT. **statistics** also translates metrics results into {name, value, time} tuples and feeds them to Graphite's whisper database [6]. Whisper is a fixed-sized Round Robin Database similar in design to RRDtool and only stores time-series numeric data. statistics.py corresponds to sFlow2Graphite in Figure 5. Graphite Webapp is used to render plots using the time-series data from whisper. Graphite Webapp provides a dashboard for retrieval and visualization of our metrics.

A. Multi-tenancy and FlowSpace Monitoring

OF@TEIN is a multi-tenant testbed which allows data plane resource to be shared among multiple experimenters. This is achieved by flowspace-level virtualization using FlowVisor. VMs in tenant networks can reside on physical servers deployed across various sites in Korea as well as sites in Japan, Philippine, Thailand, Malaysia, Indonesia, etc. Nodes distributed at various sites are connected via L3 networks. To allow L2 connectivity between these nodes, we have used GRE tunneling. GRE tunnels are created using software-based OVS (Open vSwitch) as well as hardware-based NetFPGA solutions. Figure 6 shows multiple tenant/virtual networks typical of real-world concurrent experiment slices on top of OF@TEIN physical substrate. An experimenter gets topology visualization and flowspace visibility of her slice(s). Testbed administrators on the other hand can monitor all the overlay tenant networks as well as underlay physical network. In Figure 6, topology visualization is provisioned using NetOpen RA UI developed by GIST, South Korea. However, a new visualization UI is in progress.

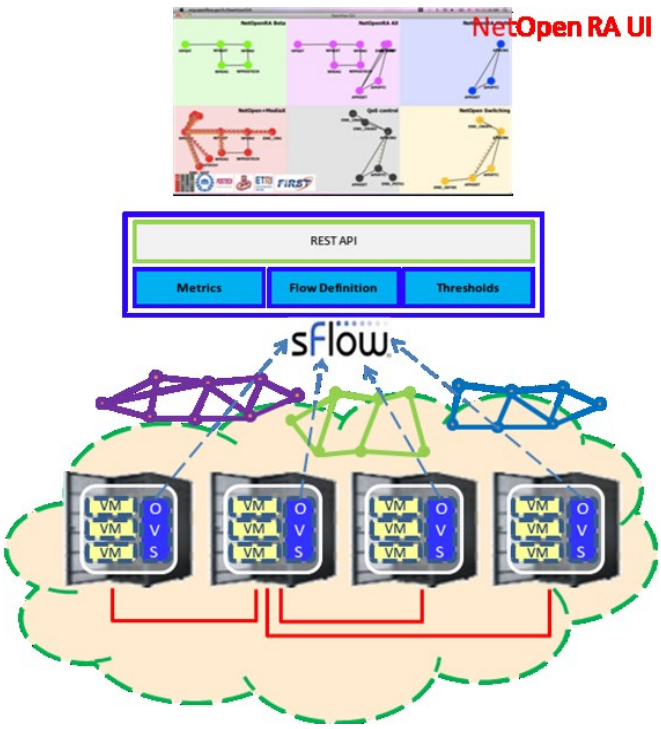

Fig.6.Virtual playgrounds (or tenant networks) in OF@TEIN

B. Visibility into GRE Tunnels

In OF@TEIN, VMs in a tenant network (or virtual playground) are spread across public WAN. However, these VMs are mostly assigned private IP addresses that are not routable over the Internet. To provide a LAN connectivity illusion between these VMs, we use GRE (Generic Routing Encapsulation) to encapsulate tenant network host packets in routable IP packets. We use OVS (Open vSwitch) to encapsulate traffic between hosts via GRE tunnels and hence create overlay user networks. Note that OF@TEIN is managed and orchestrated as a cloud using OpenStack as the cloud OS. OpenStack uses GRE tunneling functionality of OVS to isolate different tenant networks from one another.

sFlow enables us to monitor traffic flows through GRE tunnels. Any networking device, virtual or physical, installed between endpoints of a GRE tunnel can capture tunneled packets using sFlow. However, OVS that hosts a tunnel

endpoint cannot capture GRE encapsulated packets because packets are captured before they enter the tunnel.

C. Flows Definition

The Users can define the flows using xml data model as given below:

```
<?xml version="1.0"?>
<flows>
 <!-- Flow keys are a set of comma delimited packet attributes -->
 <flow name="tcp1"
      keys = "ipsource,ipdestination,tcpsourceport,tcpdestinationport"
      value="frames"
      filter="ipsource=192.168.1.1&amp;ipdestination=192.168.1.2"
/>
</flows>
```

A flow is defined using **name**, **keys**, **value** and optionally **filter** attributes.

D. Visualization o f flow statistics

We use Graphite Dashboard to provide real time visibility into flows across OF@TEIN testbed. When the network engineer opens the graphite web interface, he shall see two panels as shown in Fig. 7.

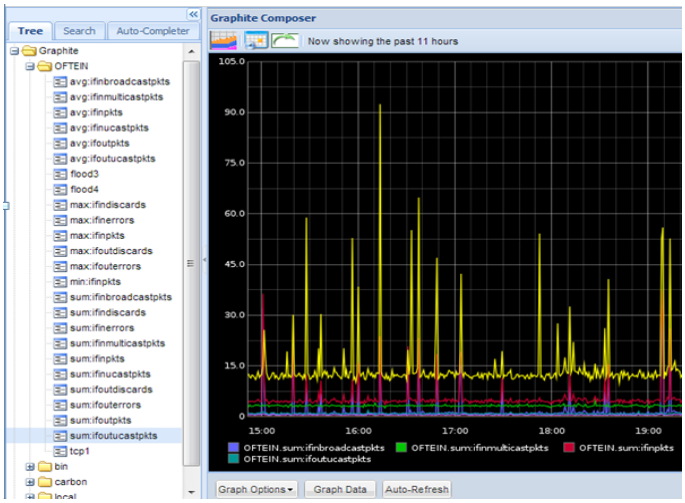

Fig. 7. Flow visualization using Graphite

Left panel is taken up by the **browser tree**. Browser tree allows her to select metrics to show on a real-time graph in the right panel called **composer**. To add a new graph directly, she selects a metric series in the browser tree, and a graph for that value is added to the graph panel. Alternatively, if a graph for that metric series already exists, it will be removed.

E. sFlow Monitoring Methodology

The flow monitoring application in OF@TEIN software defined tested goes through following steps to deliver real time network visibility. The workflow is also demonstrated in Fig. 8.

- **Get Topology Info:** SDN Application gets topology information based on LLDP (Link Layer Discovery Protocols). The information comprises both DIRECT and TUNNEL links.
- **Build Topology:** SDN Application uses Avior GUI to show a graph of the network topology.
- **Define Flows:** Create, delete or modify flow rules in OpenFlow switches. Create, delete or modify flow definitions in sFlow-RT Analytics Engine.
- **Get Metrics:** Fetch metrics (i.e., flow statistics and interface counters) periodically from sFlow-RT.

  Plot Metrics: Feed metrics to Graphite for generating graphs of the metrics being monitored.

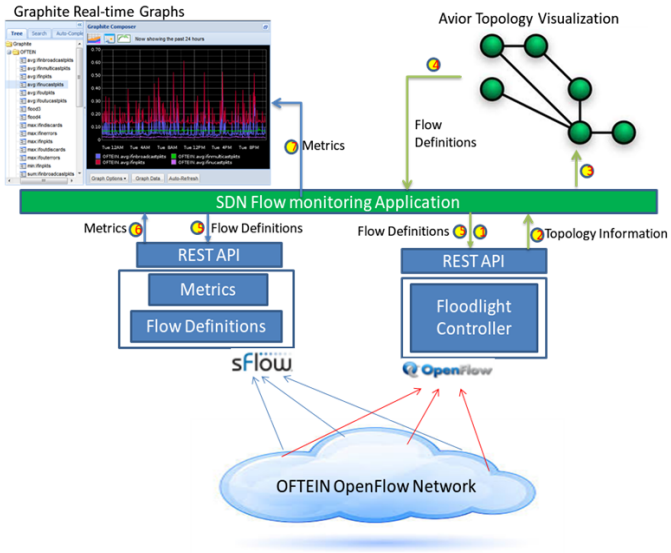

Fig. 8. Flow-space monitoring methodology using SDN

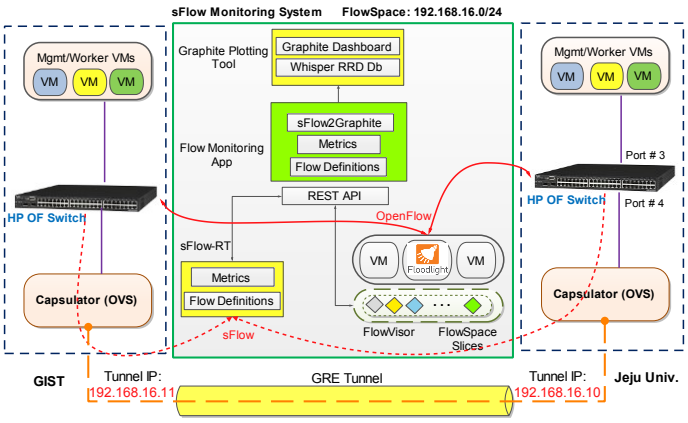

Fig. 9. OF@TEIN Monitoring System Demo Setup

VI. System Evaluation

Using the methodology described in Figure 8, we setup a demo to demonstrate OF@TEIN sFlow traffic monitoring system. Our Demo setup consisted of VMs at Jeju National University and GIST as shown in Figure 9. Jeju and GIST VMs are connected to each other via HP OpenFlow Switches. These switches are connected through a GRE tunnel. The Tunnel is created using OVS (Open vSwitch). The slice that VMs belong to is assigned the network address 192.168.16.0.

Tunnel endpoints are assigned 192.168.16.10 and 192.168.16.11 IP addresses. We generate traffic between Jeju and GIST VMs using IPerf. Both Jeju OpenFlow Switch and GIST OpenFlow switch are controlled by Floodlight controller. sFlow-RT analytics engine gets sFlow datagrams from both the switches.

sFlow monitoring application communicates with both Floodlight controller and sFlow-RT using REST API. sFlow monitoring application gets metrics from sFlow-RT and feeds them to Graphite to render the plots in real-time.

Two of the plots and corresponding flow definitions are shown in Figures 10 and 11. Plots show real-time traffic stats corresponding to the two flows.

```
1.  <flow name="testflood3" keys="ipsource" value
="bytes" />
```

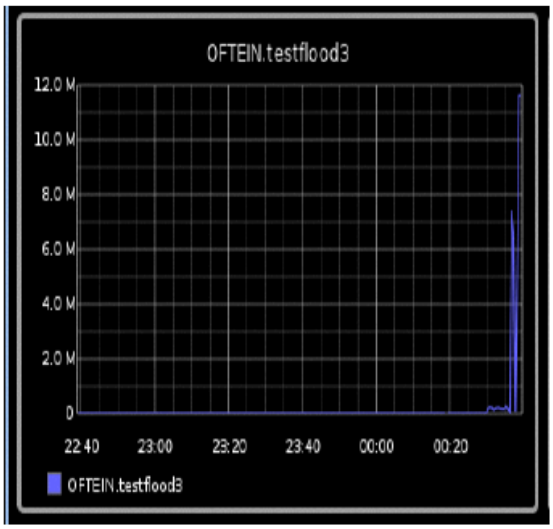

Fig. 10. Real-time visualization of flow named testflood3

```
2. <flow name="test3" keys="ipsource,ipdestination"
   value="frames" filter="ipsource=192.168.16.10&amp;
   ipdestination=192.168.16.11" />
```

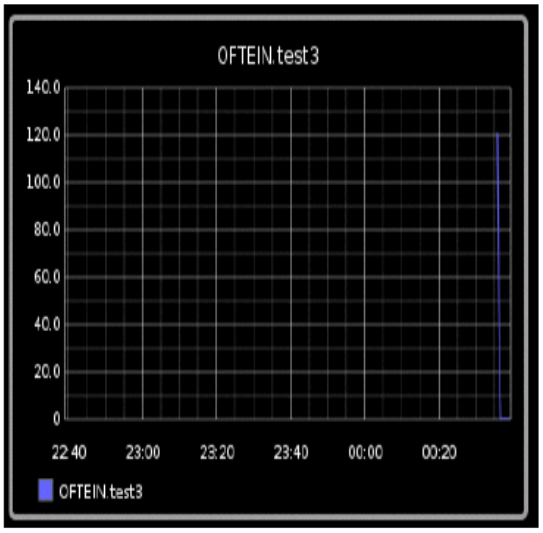

Fig.11. Real-time visualization of flow named test3.

VII. Conclusion

Software defined networking (SDN) is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable making it ideal for today's dynamic and high bandwidth applications. This architecture decouples control plane of networking devices from their data plane enabling applications to directly program the network control logic. This enables multitude of SDN use cases. Packet flow monitoring is a major use case of SDN. It enables the network engineer to guide network management and control applications. A key use case of SDN enabled flow monitoring in OF@TEIN is FlowSpace monitoring. SDN flow monitoring application gets slice flow definitions from OpenFlow controller, loads them into sFlow-RT, fetches summary statistics and feeds them to Graphite real-time charting tool. Our monitoring system also enables us to monitor GRE tunnels which are used to isolate traffic of tenant networks.

In the future, we will extend OF@TEIN flow monitoring system to monitor slice flows and overall (i.e., provider network flows side by side. Furthermore, we intend to identify small (mouse) and large (elephant) flows and study the performance of the system at scale.

VIII. ACKNOWLEDGMENT

This work was partially supported by one of KOREN projects of National Information Society Agency (13-951-00-001).

References

[1] Big Switch Networks, “Open SDN for Network Visibility,” April 2013.

[2] Rich Groves, "Microsoft's DEMon (Distributed Ethernet Monitoring)," http://blog.sflow.com/2013/04/sdn-packet-broker.html, April 2013.

[3] sFlow-RT, http://www.inmon.com, May 20, 2014.

[4] N. McKeown, Keynote talk: Software Defined Networking. In Proc. of IEEE Conference on Computer Communications (INFOCOM'09), Apr. 2009.

[5] Floodlight Controller, http://www.projectfloodlight.org, May 14, 2014.

[6] Graphite - Scalable Realtime Graphing, http://graphite.wikidot.com/, May 15, 2014.

[7] Google, "OpenFlow@Google," Open Networking Summit, April 12, 2014.

[8] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hlzle, S. Stuart, and A. Vahdat. "B4: Experience with a globally deployed software defined WAN," In ACM SIGCOMM, Aug. 2013.