K-ONE Technical Document #33

A Prototype Implementation of Multi-Belt Onion-Ring Visualization for SmartX MultiView Visibility

Document No.	K-ONE #33
Version	1.0
Date	2018-12-26
Author(s)	Usman, Muhammad

■**History**

Version	Date	Author(s)	Contents
Draft - 0.1	2018. 11. 20	Usman, Muhammad	Chapter 1
0.2	2018. 11. 28	Usman, Muhammad	Chapter 2
0.3	2018. 12. 15	Usman, Muhammad	Chapter 3
0.4	2018. 12. 26	Usman, Muhammad	Chapter 4

본 문서는 2015년도 정부(미래창조과학부)의 재원으로 정보통신 기술진흥센터의 지원을 받아 수행된 연구임 (No. B0190-15-2012, 글로벌 SDN/NFV 공개소프트웨어 핵심 모듈/기능 개발)

This work was supported by Institute for Information & communications Technology Promotion(IITP) grants funded by the Korea government(MSIT) (No. 2015-0-00575, Global SDN/NFV Open-Source Software Core Module/Function Development)

Summary

In this document, we present the preliminary prototype of Multi-Belt Onion-Ring Visualization for SmartX MultiView Visibility. In Chapter 1, we provide an overview of SDN-enabled Multisite Cloud Playground, playground visibility, and associated visualization challenges. In Chapter 2, we briefly discuss SmartX MultiView Visibility Framework, limitation of existing visibility visualization support tools, and requirements of Multi-Belt Onion-Ring Visualization. In Chapter 3, we provide design and preliminary prototype implementation of Multi-Belt Onion-Ring Visualization for SmartX MultiView Visibility. Finally, we show how to deploy and verify Multi-Belt Onion-Ring Visualization functionality into SDN-enabled multi-site cloud playground for delivering interactive visualizations.

Contents

K-ONE 기술 문서 #33 A Prototype Implementation of Multi-Belt Onion-Ring Visualization for SmartX MultiView Visibility

List of Figures

List of Tables

K-ONE #33. A Prototype Implementation of Multi-Belt Onion-Ring Visualization for SmartX MultiView Visibility

1. Introduction

In this chapter, we briefly discuss emerging ICT technologies, OF@TEIN Playground, and associated operational challenges of the playground.

Recently, Software-Defined Networking (SDN) has captivated the interest of both industry and research community since it proposes to decouple data-plane functionality from control-plane functionality. Data-plane functionality of packet forwarding is built into switching fabric, while the control-plane functionality of managing network devices is placed in logically-centralized software controller(s). This separation simplifies the network management tasks and minimizes the associated complexities of distributed configurations. Because of the cost-effectiveness of SDN approach, it enables researchers to perform various experiments, which were too costly or difficult with legacy network devices. Cloud computing is another emerging paradigm, which attracts the interest of its service providers and consumer communities with its pay-per-use service model since it removes the upfront cost of purchasing physical hardware and hides network connectivity complexities. That is, cloud computing delivers on-demand IT resources by dynamically scaling its provisioning power.

Lately, various research work has concentrated on establishing Future Internet testbeds to provide advanced experimental networking facilities for evaluating emerging software-defined technologies. Aligned with Future Internet testbed projects like GENI and FIRE, we launched OF@TEIN Playground. Mainly, serving as SDN-enabled multi-site clouds to facilitate academic miniaturized experiments. OF@TEIN Playground is composed of 10 distributed sites, spread over nine countries (i.e., Korea, Malaysia, Thailand, Indonesia, Philippines, Pakistan, Taiwan, Vietnam, and India) as depicted in Fig 1.

OF@TEIN Playground consists of multiple types of resources: physical, virtual, and container types. Generally, physical servers and switches, and physical interconnects are referred as physical resources. Virtual machine instances (via the support of hypervisors), virtual switches, and virtual interconnects are referred as virtual resources. Container instances recently entered the scene to provide a new light-weight category of resource for flexible workload deployment. In order to effectively operate OF@TEIN Playground, it is truly imperative to recognize where physical, virtual, and container resources are located and how they are inter-related to each other. Therefore, in this paper, we propose multi-belt onion-ring style visualization, as a helper solution to the above challenge of single view unified visualization. To effectively realize onion-ring visualization, diverse types and volume of visibility data are required to be collected, processed, and interpreted. However, to the best of our knowledge, existing open-source solutions do not provide the required onion-ring visualization capability to directly match the multisite resource infrastructure of OF@TEIN Playground.

Lately, we have been developing an integrated framework for multi-layered visibility workflow, denoted as 'SmartX Multi-View Visibility Framework (MVF)' [1]. By applying SmartX MVF to OF@TEIN Playground, we are able to monitor the various dynamic visibility metrics from various measurement points across both physical and virtualized resources and associated flows of the playground. We leverage the already collected playground visibility data with slight modifications followed by careful integration (i.e., organized and interpreted) to enable the required Multi-Belt Onion-Ring Visualization.

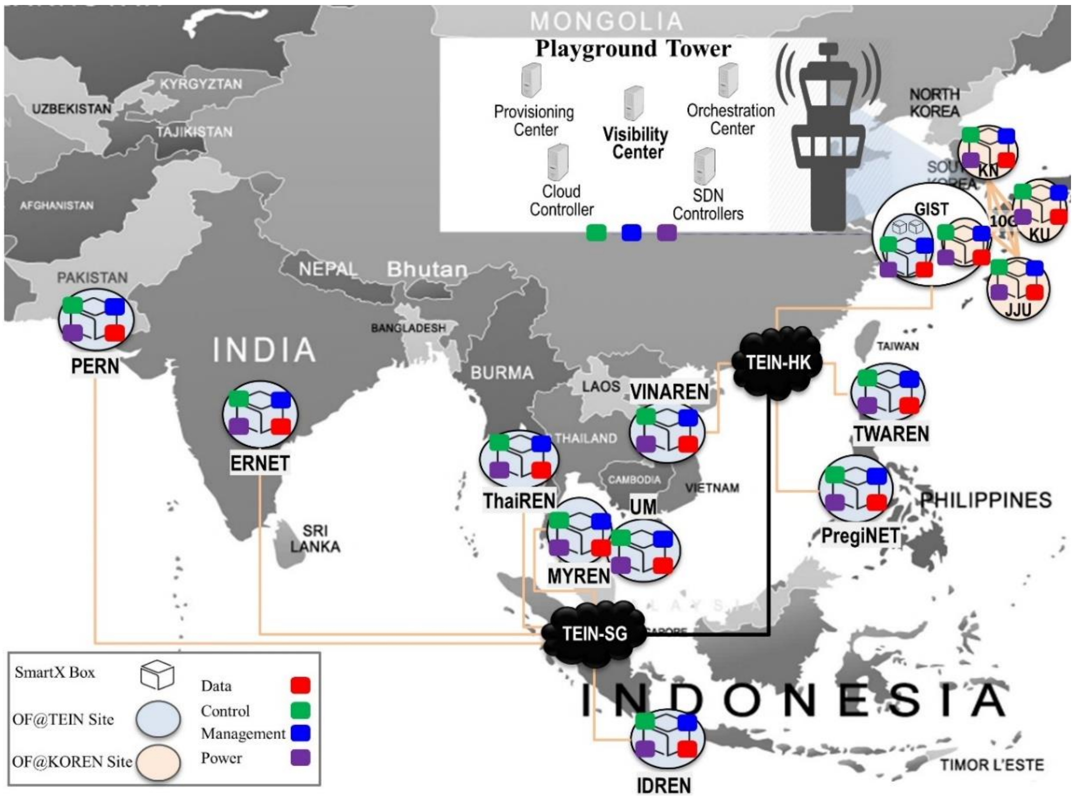


Figure 1: OF@TEIN playground as multi-site SDN-enabled Cloud.

The remainder of this document is organized as follows. In Chapter 2, we briefly discuss SmartX MVF and issues with its current visualization support followed by Multi-Belt Onion-Ring Visualization requirements. In Chapter 3, we collectively cover both design and preliminary prototype implementation of Multi-Belt Onion-Ring Visualization. Finally, we discuss prototype deployment and verification results for proposed Multi-Belt Onion-Ring Visualization in Chapter 4.

2. Requirements of Multi-Belt Onion-Ring Visualization for SmartX MultiView Visibility

In this chapter, we will discuss the SmartX MultiView Visibility Framework, limitation of existing visualization tools, and requirements of Multi-Belt Onion-Ring Visualization.

2.1. SmartX Multi-View Visibility Framework

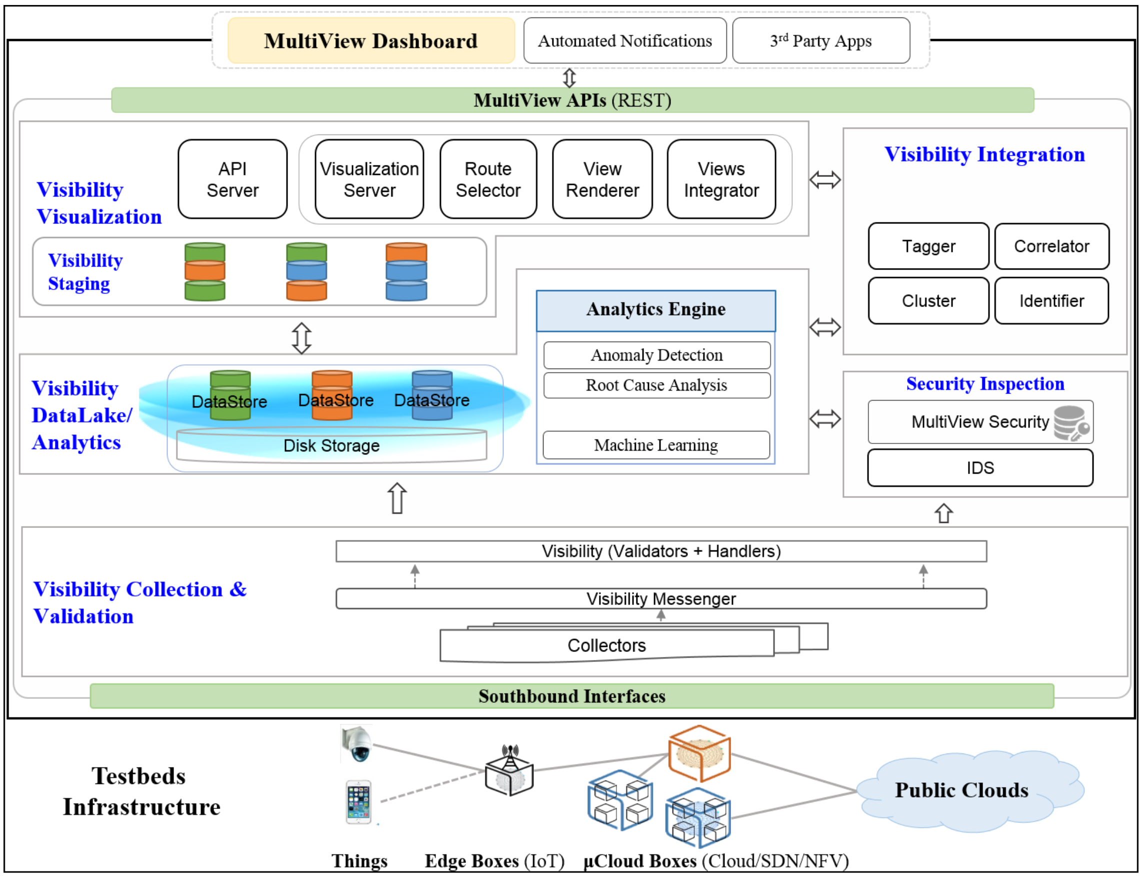


Figure 2: Design of SmartX MultiView Visibility Framework.

SmartX Multi-View Visibility Framework is proposed to deal with the multiple layers of visibility and interactive visualization of OF@TEIN Playground. That is visibility framework is mainly designed to deal with four layers such as underlay resource-layer, physical resource-layer, slice-layer, flow-layer, and workload-layer. Underlay resource-layer is responsible for monitoring underlay network operational status and identifying any potential failures. Physical resource-layer visibility is responsible for monitoring and visualization of playground physical resources and inter-connects (e.g. paths and links). Slice-layer visibility covers monitoring and visualization of isolated virtualized resources and their virtual interconnects that are configured on a physical resource. Flow-layer visibility is responsible for providing different levels of flow-centric information by utilizing flow collection, clustering, and tagging schemes to assist SDN-based flexible networking (i.e. steering/ mapping). Lastly, workload-layer visibility is responsible for the monitoring and visualization of inter-connected functions and their placements for tenant-based applications. Also, for visibility data processing, SmartX MVF has four main stages. Visibility Collection and Validation stage collects and validates data based on selected visibility metrics. Cleaned data is sent to Visibility Storage and Staging stage that is responsible for data storage. After that, Visibility Integration stage fuses collected data for identifying key patterns and identifies any anomalies. Finally, Visualization stage accesses processed data and transforms it into graphical views. The overall design of the SmartX Multi-View Visibility Framework is shown in Fig. 2.

2.2. Limitation of Existing Visualization Tools

A number of open-source visibility monitoring and visualization tools are already available; but, they do not directly match unique single-box-virtualized deployment style of OF@TEIN Playground. In order to solve this shortcoming, we have developed p+v topological visualization for SmartX Multi-View Visibility [2]. This p+v topological visualization is in a tree-style view, focusing on the interconnections among distributed physical and virtual resources. A snapshot of p+v topological visualization is shown in Fig. 3. However, this flat tree-style topological visualization is proving insufficient to effectively demonstrate multi-layer visibility of multisite resource infrastructure of OF@TEIN Playground. Therefore, a more systematic and fine-grained visualization with clear separation between multiple layers and multiple sites is essential for SmartX Multi-View Visibility to effectively operate OF@TEIN Playground.

2.3. Requirements of Multi-Belt Onion-Ring Visualization

Multi-Belt Onion-Ring Visualization should meet the following requirements in order to fully qualify as a capable solution for OF@TEIN Playground. Firstly, Multi-Belt Onion-Ring Visualization should be innovative to offer a single unified view for SDN-enabled multisite clouds by covering various p+v+c resource components and their associated flows of OF@TEIN Playground. Also, for faster troubleshooting, Multi-Belt Onion-Ring Visualization must define several dedicated belts to flexibly incorporate multi-layer visibility data that is collected from multiple sites. Furthermore, multi-tenancy support is another key requirement that Multi-Belt Onion-Ring Visualization should support.

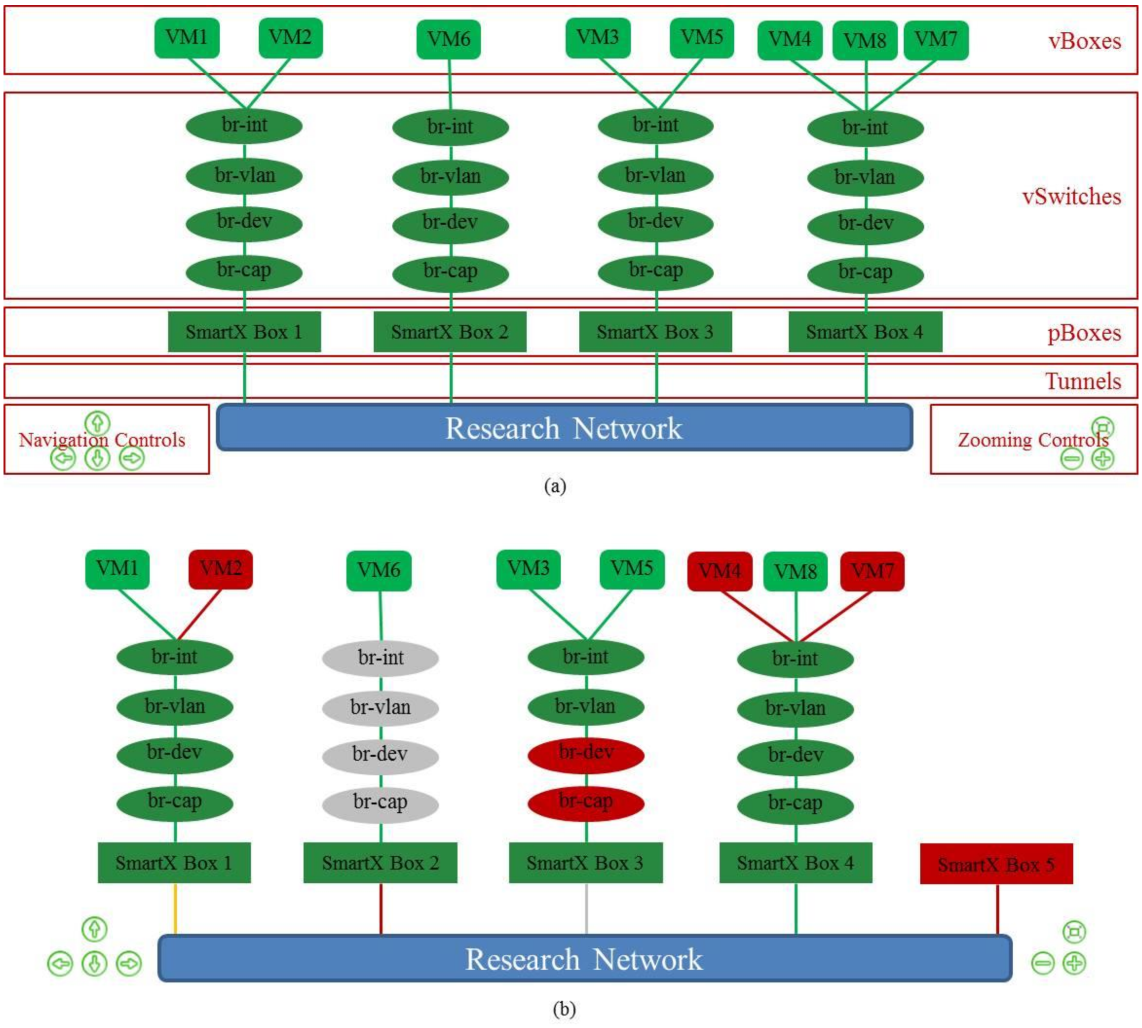


Figure 3: p+v Topological Visualization for OF@TEIN Playground.

3. Prototype Design and Implementation of Multi-Belt Onion-Ring Visualization for SmartX MultiView Visibility

In this chapter, we provide the details of overall design and preliminary prototype implementation of Multi-Belt Onion-Ring Visualization for SmartX MultiView Visibility.

3.1. Selected Tools for Playground Visibility Measurements

Initially, the operation of visibility collection requires the identification of key performance metrics that affirm a major impact on the operational states of the playground. Currently, for underlay resource-layer visibility measurement, we utilize perfSONAR [3]. For physical and virtual resource-layers visibilities measurements, we extensively utilize Intel Snap monitoring framework [4]. Besides, for flow-layer visibility measurement, we use sFlow-based sampling and eBPF-based tracing tools [5]. In addition, to diversify the visibility measurements of resource, slice, and flow layers, we collect visibility metrics from both the cloud and SDN controller(s) as well.

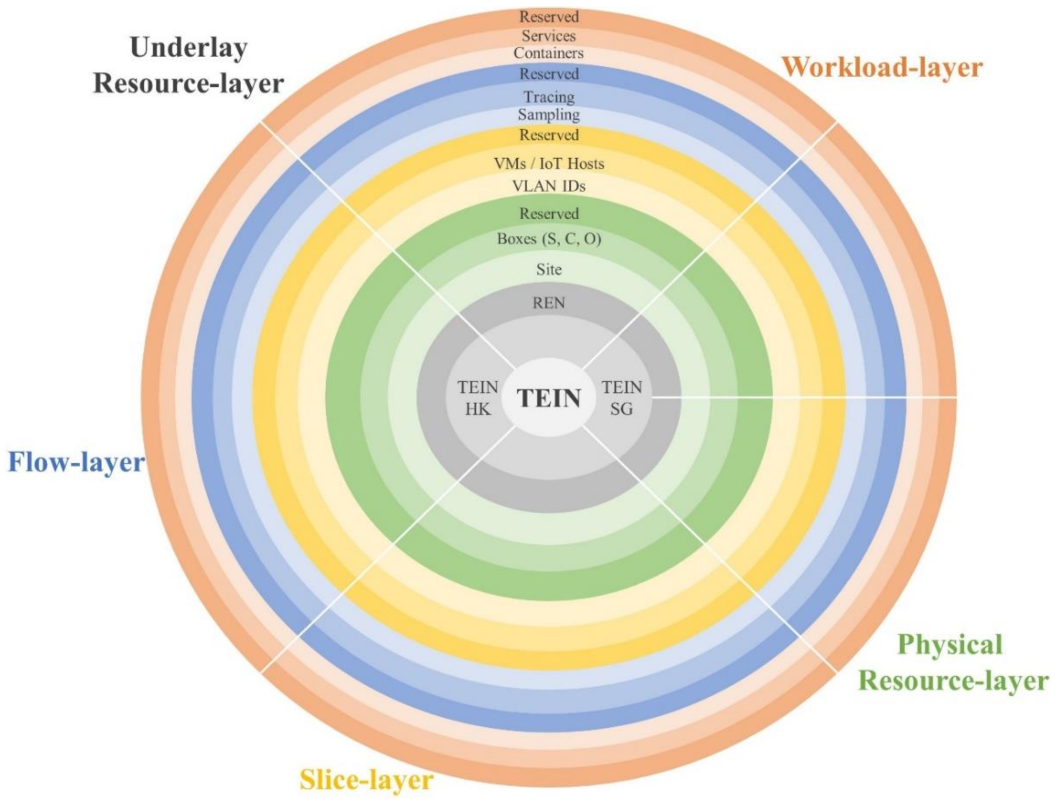


Figure 4: Design of Multi-Belt Onion-Ring Visualization for SmartX Multi-View Visibility.

3.2. Design and Prototype Implementation

To address the above-highlighted difficulties of playground visualization, we focus on Multi-Belt Onion-Ring style visualization as shown in Fig. 4. Tiered belts on the surface partly cover the multiple layer visualization requirements of SmartX Multi-View Visibility. Primarily there are five main belts for each layer of visibility and then they are further divided into three sub-belts to allow incorporation of additional information for display in a particular belt. Also, the color-coded areas that are separated with white lines correspond to distributed sites that host physical, virtualized, or containerized resources (e.g., boxes, switches, and service functions).

In order to realize the multi-belt onion-ring visualization, we leverage open-source visualization library called psd3 [6]. That is, psd3 is primarily based on D3.js [7] and supports multi-level pie charts. A snapshot of implemented code in JADE template language for view generation is shown below. To easily visualize performance metrics data and observe various trends, we utilize open-source software named Grafana latest version [8]. Finally, for onion-ring visualization deployment, Node.js JavaScript runtime latest version is selected due to its non-blocking I/O and event-driven features [9]. For semi-automated provisioning of all these tools, a script called Visibility_Center_Install.sh is developed.

// Jade source code for Multi-Belt Onion-Ring Visualization.

```
doctype html
html
head
meta(charset='UTF-8')
title Onion-ring-style Visualization (Operator)
style(type='text/css').
.sdncontrollerdiv{
position: absolute;
top: 40px;
right: 30px;
width: 240px;
height: 135px;
font-size: 18px;
opacity: 0.7;
border: 1px solid #bfbfbf;
background: #f5f5ef;
}
script(type='text/javascript', src='javascripts/d3.v3.min.js')
script(type='text/javascript', src='//cdnjs.cloudflare.com/ajax/libs/canvasjs/1.7.0/canvasjs.js')
script(type='text/javascript', src='javascripts/psd3.js')
link(type='text/css', rel='stylesheet', href='opentip/css/opentip.css')
link(rel='stylesheet', type='text/css', href='stylesheets/psd3.css')
body(style='margin: 15px;')
.container
#chartContainer.well
.sdncontrollerdiv
.sdncontrollerheaddiv(style='color:white;font-family:verdana;text-align:center;')          Plaground
```

```
Controllers
.sdncontrollerbodydiv
table#controllertable(style='width:100%;color:black;text-align:center;')
tbody
script(type='text/javascript').
//var session_username = sessionStorage.getItem('ss_user_name');
//if(session_username === null){
//              window.location.replace("http://103.22.221.56:3011/login");
//};
var data =   !{data};
var controllerList      = !{controllerList};
//* Important Data is the data of psd3
var config = {
containerId: "chartContainer",
width: 850,
height: 850,
data: data,
label: function(d) {
return d.data.label;
},
textBoder : function(d){
return d.data.colorBoder;
},
value: "resource",
inner: "drilldown",
tooltip: function(d) {
return "<div style='background-color: #4a4; color: white; padding: 15px; text-align: middle;
border: dotted 1px black;'><strong>" + d.info;
},
textColor: function(d){
return d.data.color;
},
transition: "linear",
transitionDuration: 20,
donutRadius: 50,
gradient: true,
colors: d3.scale.category20(),
labelColor: "black",
stroke: "#eee",
strokeWidth: 2,
drilldownTransition: "linear",
drilldownTransitionDuration: 0,
highlightColor: "#c00",
rotateLabel: false
};
    ////////////////////////////////////////JADE Code Ends///////////////////////////////////////
```

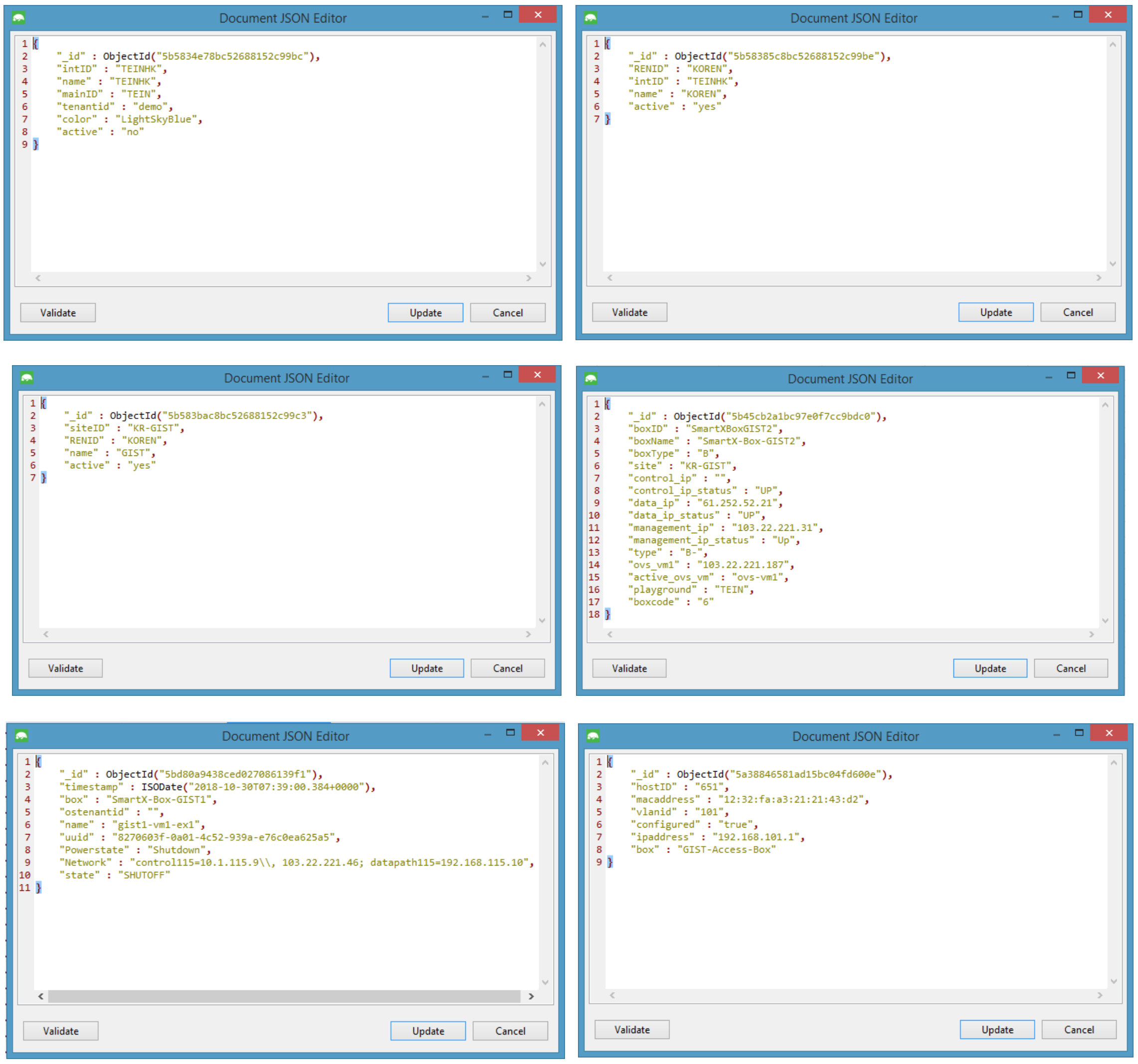
Npm packages for Multi-Belt Onion-Ring Visualization are defined in a file called package.json that is located inside OpenStack-Multi-View/pvcT-Visualization.

// Contents of package.json.

```
{
  "name": "resource-visualization",
  "version": "0.0.1",
  "private": true,
  "dependencies": {
"body-parser": "^1.18.2",
  "canvas": "*",
      "dateformat": "*",
      "errorhandler": "^1.5.0",
      "express": "^4.8.0",
      "express-session": "^1.15.6",
      "html5shiv": "^3.7.3",
      "jquery": "^3.3.1",
      "leaflet-headless": "*",
      "method-override": "^2.3.10",
      "metismenu": "^2.7.7",
      "mongodb": "*",
    "morgan": "^1.9.0",
    "multer": "^1.3.0",
    "node-font-awesome": "^1.0.2",
    "opentip": "^2.4.3",
    "popper.js": "^1.14.3",
    "pug": ">=2.0.0",
    "responder": "^0.1.6",
    "serve-favicon": "^2.5.0",
    "startbootstrap-sb-admin-2": "^3.3.7",
    "stylus": ">= 0.0.1",
    "vis": "^4.21.0"
  }
}
    ///////////////////////////////////Package.json Ends//////////////////////////////////////
```

3.3. Visibility Data Preparation for Multi-Belt Onion-Ring Visualization

The collected multi-layer visibility data is stored into so-called Visibility DataLake. At the moment, we extensively utilize MongoDB (version 3.2) to store playground inventory data while Elasticsearch (version 6.5.1) is used to store near-real-time metrics data and flows data respectively. Please note that MongoDB (document-oriented) and Elasticsearch (index-oriented) are all special-purpose NoSQL DataStores. More specifically, to enable Multi-Belt Onion-Ring Visualization, we rely on separate database that stores updated configuration and status data for the different entities of OF@TEIN Playground as shown in Fig. 5. This configuration or status database is managed via MongoDB collections. For example, each layer of the onion-ring has an associated and dedicated collection. Then to facilitate data integration across different collections, we define specific keys (e.g., physical box identifier). At the backend, Java-based plugins are used to update playground entities states. That is, Java-based plugins are explicitly developed for collecting and updating data in MongoDB collections on regular time intervals.



```
{
    "_id" : ObjectId("5b5834e78bc52688152c99bc"),
    "intID" : "TEINHK",
    "name" : "TEINHK",
    "mainID" : "TEIN",
    "tenantid" : "demo",
    "color" : "LightSkyBlue",
    "active" : "no"
}
```

```
{
    "_id" : ObjectId("5b58385c8bc52688152c99be"),
    "RENID" : "KOREN",
    "intID" : "TEINHK",
    "name" : "KOREN",
    "active" : "yes"
}
```

```
{
    "_id" : ObjectId("5b583bac8bc52688152c99c3"),
    "siteID" : "KR-GIST",
    "RENID" : "KOREN",
    "name" : "GIST",
    "active" : "yes"
}
```

```
{
    "_id" : ObjectId("5b45cb2a1bc97e0f7cc9bdc0"),
    "boxID" : "SmartXBoxGIST2",
    "boxName" : "SmartX-Box-GIST2",
    "boxType" : "B",
    "site" : "KR-GIST",
    "control_ip" : "",
    "control_ip_status" : "UP",
    "data_ip" : "61.252.52.21",
    "data_ip_status" : "UP",
    "management_ip" : "103.22.221.31",
    "management_ip_status" : "Up",
    "type" : "B-",
    "ovs_vm1" : "103.22.221.187",
    "active_ovs_vm" : "ovs-vm1",
    "playground" : "TEIN",
    "boxcode" : "6"
}
```

```
{
    "_id" : ObjectId("5bd80a9438ced027086139f1"),
    "timestamp" : ISODate("2018-10-30T07:39:00.384+0000"),
    "box" : "SmartX-Box-GIST1",
    "ostenantid" : "",
    "name" : "gist1-vm1-ex1",
    "uuid" : "8270603f-0a01-4c52-939a-e76c0ea625a5",
    "Powerstate" : "Shutdown",
    "Network" : "control115=10.1.115.9\\, 103.22.221.46; datapath115=192.168.115.10",
    "state" : "SHUTOFF"
}
```

```
{
    "_id" : ObjectId("5a38846581ad15bc04fd600e"),
    "hostID" : "651",
    "macaddress" : "12:32:fa:a3:21:21:43:d2",
    "vlanid" : "101",
    "configured" : "true",
    "ipaddress" : "192.168.101.1",
    "box" : "GIST-Access-Box"
}
```

Figure 5: Playground Inventory database arranged as MongoDB collections.

4. Deployment and Verification of Multi-Belt Onion-Ring Visualization for SmartX MultiView Visibility

This chapter first shows how to deploy Multi-Belt Onion-Ring Visualization for SmartX MultiView Visibility by targeting OF@TEIN Playground. Following in this chapter, we also verify the overall working of the deployed software. The preliminary prototype Implementation of Multi-Belt Onion-Ring Visualization is available at https://github.com/K-OpenNet/OpenStack-MultiView.

In order to deploy SmartX Multi-View Visibility software packages, first we have to download it from the aforementioned GitHub repository to SmartX Visibility Center /opt/ directory and change owner of the directory to a normal user.

```
# cd /opt/
```

```
# sudo git clone https://github.com/K-OpenNet/OpenStack-MultiView.git
```

```
# sudo chown netcs:netcs –R OpenStack-MultiView
```

Install the required dependencies on SmartX Visibility Center by executing Visibility_Center_Install.sh script.

```
# sudo ./Visibility_Center_Install.sh
```

Install required npm packages by executing below command, which reads required list of dependencies from the package.json.

```
# npm install package.json
```

After the successful installation of required npm packages, we have to restart the multi-view visualization service, which was already created by Visibility_Center_Install.sh script.

```
# sudo systemctl restart multi-view-web.service
```

Open Firefox or Chrome web browser and redirect to http://103.22.221.56:3000 to access Multi-Belt Onion-Ring Visualization from SmartX Visibility Center. Once, successfully logged in to the dashboard, you will be provided Multi-Belt Onion-Ring Visualization view as shown in Fig. 6. This dashboard is effective in presenting underlay resource-layer, physical resource-layer, slice-layer, and flow-layer visibilities together in a single unified view. At the underlay resource layer, that is from the center to outside, we can see international network PoPs (TEINSG and TEINHK), national PoPs (e.g., MYREN, KOREN, ...) and sites (e.g., GIST, CHULA, ...). On top of the distributed sites, physical boxes are shown with their respective type (B, C, S). Next belt shows the VLAN IDs configured for particular tenants on a physical box. Few physical boxes also have a number of virtual boxes (VMs) within them, respectively. For example, type B physical box in MYREN site have two virtual boxes and type O box in KN site have 4 virtual boxes. Next two belts in blue color show sampled flows and traced packets of each individual virtual box. Please note that, with the help of associated box IDs, we can easily visualize resource usage graphs in the boxes. Thus, with the support of visualization dashboard, the operators and developers of the playground can easily grasp the overlay-based playground topology and check its relationship with the underlay WAN networks that belong to separate network operators.

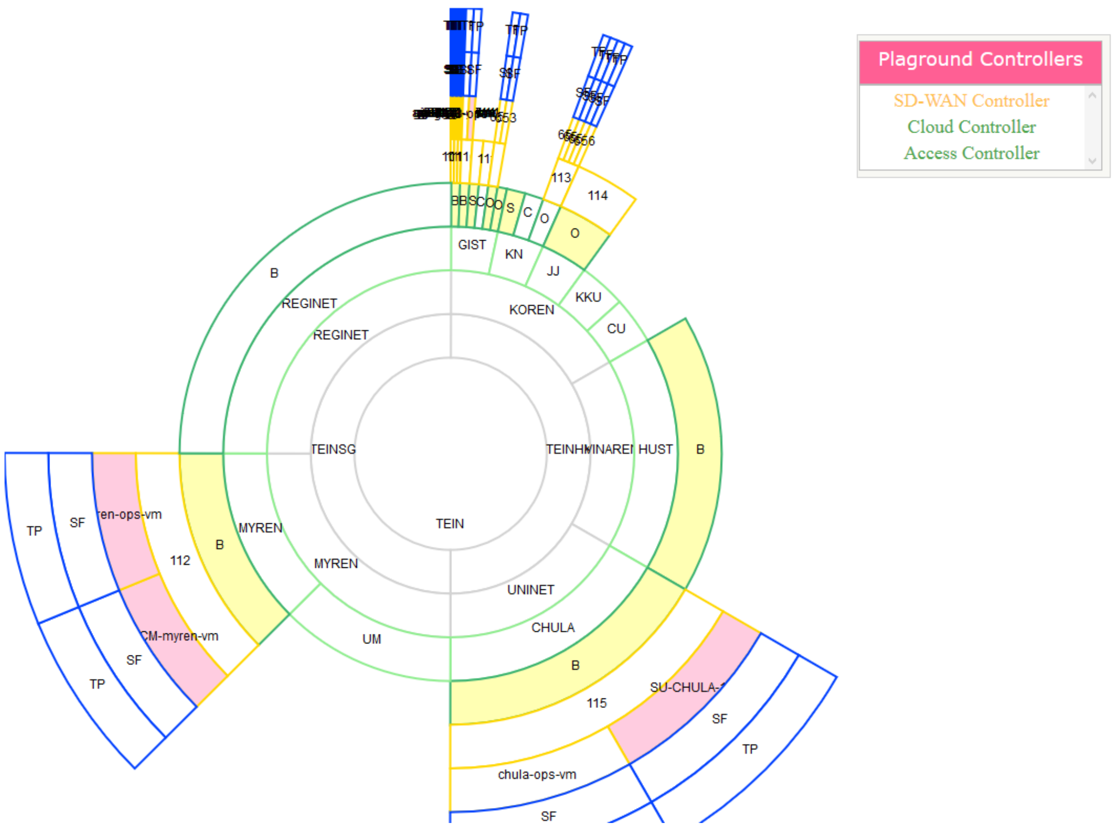


Figure 6: Multi-Belt Onion-Ring visualization for OF@TEIN Playground.

Additionally, we can drill down any particular belt by double-clicking on it for the more transparent view. For example, Fig. 7 shows a drill down view for GIST site. Also, on the top right corner, we can see the list of playground controllers with their online working status.

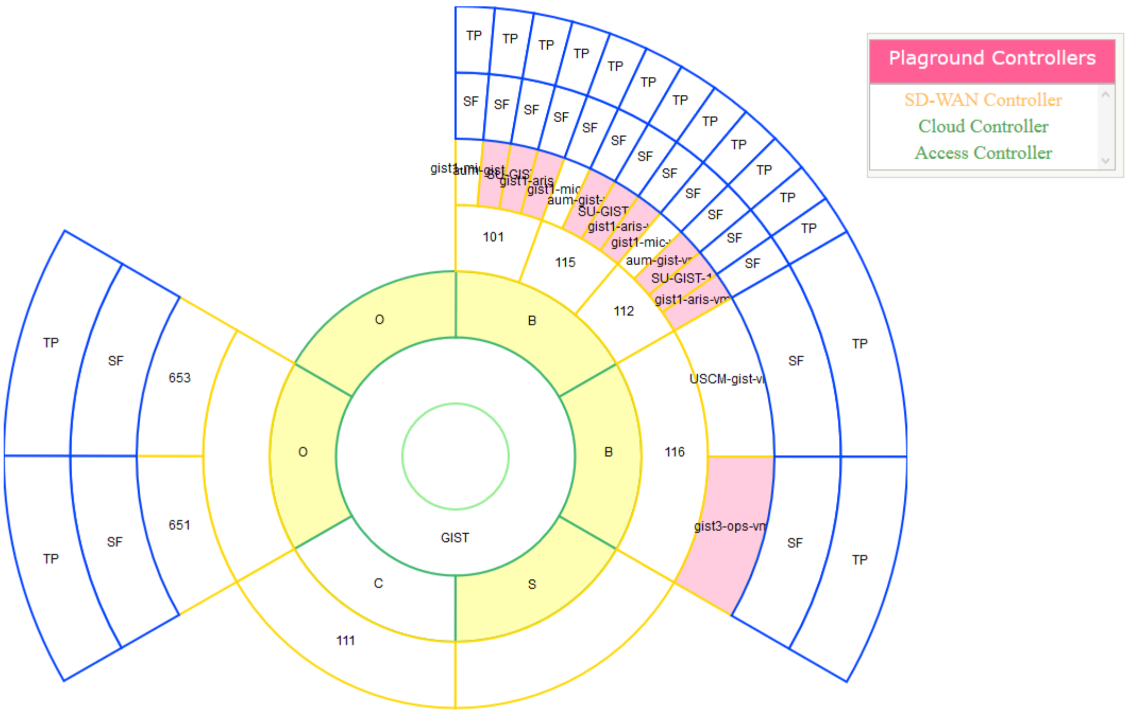


Figure 7: Multi-Belt Onion-Ring visualization drill down for the GIST site of OF@TEIN Playground.

References

[1] M. Usman, A. C. Risdianto, J. Han, M. Kang, and J. Kim. SmartX MultiView visibility framework leveraging open-source software for SDN-Cloud playground. In: Proc. of *IEEE NetSoft Conference and Workshops (NetSoft)*, Bologna, 2017, pp. 1-4.

[2] M. Usman, A. C. Risdianto, J. Han, J. Kim, and Nguyen Van Huynh. Physical-virtual topological visualization of OF@TEIN SDN-enabled multi-site cloud. In: Proc. of *International Conference on Information Networking (ICOIN)*, Da Nang, 2017, pp. 622-624.

[3] perfSONAR, https://www.perfsonar.net/.

[4] Intel Snap Telemetry Framework, https://github.com/intelsdi-x/snap.

[5] IO Visor Project, https://www.iovisor.org/.

[6] psd3, https://github.com/pshivale/psd3.

[7] Data-Driven Documents, https://d3js.org/.

[8] Grafana, https://grafana.com/.

[9] Node.JS, https://nodejs.org/en/.

K-ONE Technical Document

Written by K-ONE Consortium

Written in 2017/05