Computer Science and Information Systems 18(1):169–188 https://doi.org/10.2298/CSIS200317034R

Spatio-temporal Summarized Visualization of SmartX Multi-View Visibility in Cloud-native Edge Boxes

Muhammad Ahmad Rathore[1] and JongWon Kim[1]

[1] School of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology, Gwangju 61005, South Korea
Room No. 207, EECS Building C, Gwangju Institute of Science and Technology, 123 Cheomdangwagi-ro, Buk-gu, Gwangju 61005, South Korea
ahmadrathore@gist.ac.kr
[2] jongwon@gist.ac.kr

Abstract. The existing data summarization (and archival) techniques are generic and are not designed to leverage the unique characteristics of the spatio-temporal visualization at multi-resource level. In this paper, we propose and explore a family of data summaries that take advantage of the multiple layers i.e. physical/virtual resources with temporal and spatial correlation among distributed edge boxes. Significant challenges in measuring spatio-temporal data, however, contribute to both a tendency towards identifying efficient metrics with summarizing functionalities and effective verification methods. In this paper, we present our idea of maintaining summarized spatio-temporal data and verify through visualization of gathered operational data.

Keywords: spatio-temporal, edge, visualization, cloud-native.

1. Introduction

With the continuous development and popularization of IoT devices, high-performance computing, and storage technology, a type of time series data with space information, called spatio-temporal data, has emerged. This spatio-temporal data consists of useful and meaningful information that needs to be automatically mined, leading to the continuous development of spatio-temporal data processing technologies [1]. Maintaining such data has hard limits on service quality constraints that must be maintained, despite network fluctuations and varying peaks of load [2]. Furthermore, equipping cloud-native distributed data platform for edge devices with effective spatio-temporal query processing capability, well established in a centralized database is, therefore, a challenge for smooth operations.

Aligned with Future Internet testbeds, we launched "OF@TEIN+: Open/Federated Playgrounds for Future Networks" in 2017, to further enhance, extend and expand OF@TEIN collaboration [3]. OF@TEIN+ multi-site cloud (denoted as OF@TEIN+ Playground) connects around 10 international sites in 10 countries (Korea, Malaysia, Thailand, Indonesia, Laos, Cambodia, Vietnam, Myanmar, Bhutan, and India) spread across 14 research institutes and interconnected via OF@TEIN+ network. As depicted in Fig 1, to automatically build, operate, and utilize OF@TEIN+ Playground resources, we deployed Playground Tower as a logical space in a centralized location. It systematically covers various functional requirements of operating multi-site Playground by employing Provisioning / Visibility / Orchestration Centers. For example, the Visibility Center covers playground visibility and provides visualization support. The underlay network in the playground spread across many research and educational networks (REN) under distinct administrative domain. Previously from late 2013, a hyper-converged SmartX Box [4] was introduced to virtualize and merge the functionalities of four devices (i.e. Management Worker node, Capsulator node, OpenFlow switch, and Remote power device.) into a single box. Multiple SmartX Boxes are deployed and interconnected through SDN in the distributed environments. In OF@TEIN+ Playground, a multi-site affordable cloud-native version of the SmartX Playground [5] is established that consists of software-defined (i.e., composable) Playground with hyper-converged Box-style resources [6] named *"SmartX Micro-Box"*. These Boxes consist of interfaces for management/control and data, IoT devices, and remote power control. The SmartX Micro-Box is configured to support Cloud-native computing having a software stack that is microservices oriented with virtualized/-containerized IoT-SDN-Cloud functionalities. In addition, capabilities of edge computing in these resources enable technologies that allow computation to be performed at the network edge near data sources.

Identifying and maintaining collected temporal knowledge of distributed resources through monitoring and visualization are important operational activities to ensure smooth operations of the OF@TEIN+ Playground. However, burdens of large volume generated by collecting spatio-temporal multi-layer visibility over distributed multi-site cloud with scalable edge devices at regular intervals and transferred to a centralized cloud, lead to inefficient utilization of bandwidth, storage, and computing resources. To effectively operate OF@TEIN+ Playground, it is truly requisite to recognize how physical, virtual, and container resources are running in a steady-state over a concerned time period. Spatio-temporal knowledge is useful to analyze and troubleshoot server and network issues before they affect the developers. Lately, we have been developing tools for monitoring and measurement of OF@TEIN+ Playground resources enabled with cloud-native edge computing. Leveraging the 'SmartX Multi-View Visibility Framework (MVF)' we can monitor various dynamic visibility metrics from multiple measurement points across both physical and virtualized resources and associated flows of the playground [7]. The time-series multi-view metrics collected over multi-layer resources termed as *"Spatio-temporal visibility"* delivers a rich understanding of operations that can provide deep insights into current operations as well as the possibility of supporting continuous, agile delivery of applications to end-users.

In this paper, we propose a data spatio-temporal visibility technique that introduces a synchronized and timely collection of monitoring metrics over physical, virtual, and flow layers from distributed multi-site cloud-native edge Boxes.

- First, we summarized a multi-view visibility data scheme at a centralized location collected persistently over six months. By keeping useful visibility data without hurting the time-line we summarize and align temporal multi-view visibility in a much smaller size, i.e., the summarized visibility is 10% of original data.
- Second, based on a large-scale multi-view visibility data collected from the OF@TEIN+ Playground, our proposed approach can reduce the storage requirements up to 80% while maintaining high accuracy (with bounded-errors) on the query results.

- Third, we evaluate the multi-layer spatio-temporal data through effective visualization schemes, i.e. as multi-parameter, single site, and multi-site summarization view. In addition, analysis schemes are employed to extract useful patterns and trends from visibility data.

The remainder of this paper is organized as follows: Section II elaborates on the requirements together with the overall approach and challenges entailed by the inclusion of visibility for cloud-native edge Boxes. The design and implementation are described in Section III. Verification results are detailed in Section IV. Finally, we conclude the paper in the conclusion section.

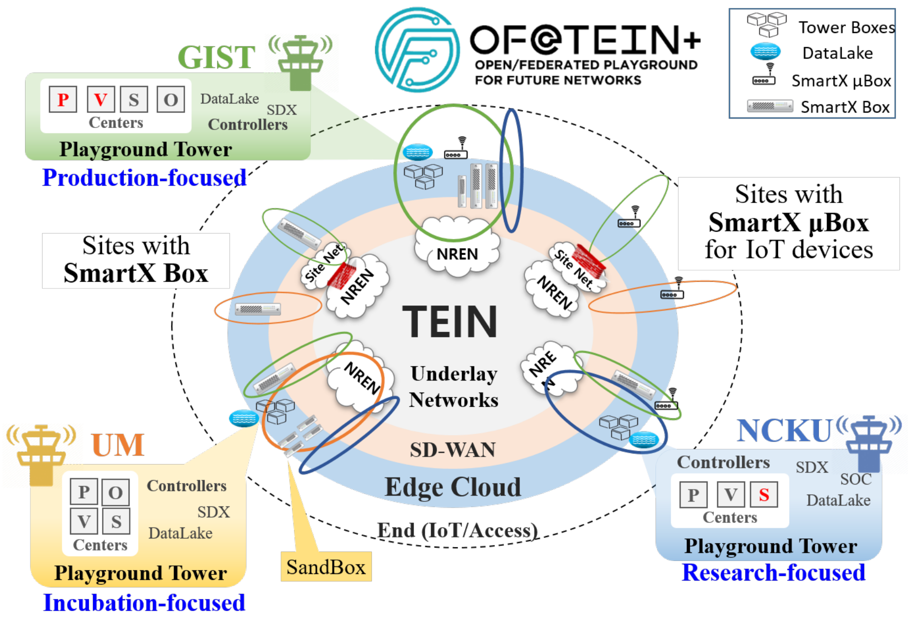


Fig. 1. OF@TEIN+ Playground as multi-site clouds.

2. Background and Requirements of Spatio-temporal Visibility

In this section, we briefly highlight the related work. Next, we mention the concept of the OF@TEIN+ Playground. In the end, we mention the requirements for spatio-temporal summarized visibility at distributed (multi-site) edge clouds by leveraging SmartX MVF.

2.1. Related Work

One of the most important development is handling a time series data based on various concepts such as visibility [8][9], correlations [10][11], recurrence analysis [12], phase-space reconstructions [13] and transition probabilities [14]. Studies have shown that researchers can summarize the characteristics of a time series into compact metrics, which can then be used to understand the dynamics or learn how the system will evolve with time. [15]. This data summarization can be employed to find a compact representation of a dataset [16]. It is important for data compression as well as for making pattern analysis more convenient. Summarization can be done on classical data, spatial data, as well as spatio-temporal data [17]. Spatio-temporal data contains the timestamp of a spatial object, a raster layer as well as the duration of a process. Spatio-temporal summarization is often performed after or in conjunction with spatio-temporal partitioning so that objects in each partition can be summarized by aggregated statistics or representative objects.

Automated identification of interesting patterns is developed in time series [18]. A clustering technique is adopted to summarize and refine the description of silent features and their relations. Ahmed et al. analyze non-stationary, volatile, and high-frequency time series data to present summarization [19]. Multi-scale wavelet analysis is employed for separating the trend, cyclical fluctuations, and auto-correlation effects. It is useful to summarize the data about the 'chief features' of the data.

A solution for flexible co-programming architecture is offered to support the life cycle of time-critical cloud-native applications [2]. Similarly, mobility-driven cloud-fog-edge offered collaborative real-time framework solution, which has IoT, Edge, Fog and Cloud layers [20].

In this research, multi-view visibility data is collected leveraging SmartX MVF. However, SmartX MVF has not been tested in an operational environment as time-series data. SmartX MVF was designed with minimal consideration given at the requirements needed to conduct spatio-temporal data mining research including space-time summarization. In addition, MVF was not implemented on cloud-native edge Boxes having IoT devices. These mentioned challenges limit the capabilities of SmartX MVF for maintaining multi-view visibility both at distributed resources (Micro-Box) and centralized collection center (Visibility-Center). To overcome these limitations, our solution added time-series summarization with an improved visualization scheme in an updated environment of cloud-native edge Boxes.

2.2. OF@TEIN+ Playground with multi-site cloud-native Edge Boxes

OF@TEIN+ Playground is a miniaturized overlay-interconnected, multi-site Playground over heterogeneous underlay networks with hyper-converged Box-style resources. To facilitate developers in learning operational and development issues as well as perform various experiments, OF@TEIN+ Playground supports multiple resource types: physical, virtual, and container types [21]. Generally, physical resources consist of physical servers and switches, and physical interconnects. Virtual resources, mainly constitute virtual machine instances (via the support of hypervisors), virtual switches, and virtual interconnects. In the given environment, we distributed cloud-native edge enabled Boxes i.e. Micro-Box, at multi-site edge locations of OF@TEIN+ Playground. Container instances recently entered the scene to provide a new lightweight category of resources for flexible workload deployment. Unlike cloud-based infrastructure that hosts virtual machines and has fixed allocated resources for users, in cloud-native, we adapt for containers to deploy applications over these Boxes. As mentioned in the Introduction section, Micro-Boxes are commodity server-based hyper-converged resources (compute /storage/networking) to allow experiments (Cloud/Network Function Virtualization) over OF@TEIN+ Playground.

Micro-Boxes have combined interfaces for management/control traffic as one and data, IoT devices, and remote power control.

2.3. Requirements of Spatio-temporal Visibility for monitoring

This research is focused on identifying requirements of summarized spatio-temporal multi-view visibility to understand the long-term operations of the Playground. Maintaining summarized spatio-temporal visibility in cloud-native infrastructure poses several challenges. First Boxes may join and leave the network (e.g. shutdown, connection failure). The operators for troubleshooting purposes require temporal Information on the on-going status of these resources. Secondly, job scheduling, resource provisioning, and allocation mechanism require monitoring metrics with a real-time collection and low latency. However, edge Boxes, in a geographically distributed infrastructure could induce delay. This creates the need for keeping the monitoring data at the edge and keeping the resource collection at a reasonable volume. Thirdly visualization of collected visibility data with analysis is a requirement to verify the experiment and provide an environment for decision-making. To address these challenges, requirements of maintaining persistence multi-view visibility are as follow,

R1: To design, develop, and implement infrastructure that is capable of collecting visibility metrics as spatio-temporal multi-view visibility on cloud-native edge Boxes.

R2: Reducing the data size of flow-layer with spatio-temporal compression. Specifically, a compression algorithm is developed based on spatial similarities between multiple streams of data over a specific time.

R3: Aggregate the time-series visibility metrics to identify spatio-temporal summarized visibility. Leveraging summarized visibility facilitates in identifying meaningful information such as missing collection, mean value for a day and percentage uptime, etc.

R4: To verify spatio-temporal visibility using multiple visualization schemes. Provide visualization support to enable support for time-series based visibility data analysis.

3. Spatio-Temporal SmartX Multi-View Visibility: Design and Implementation

In this section, we discuss the proposed design of spatio-temporal summarized multi-view visibility in cloud-native edge Boxes and identify key terminologies. Afterward, we discuss detailed components design and implementation as functionalities responsible for summarizing spatio-temporal multi-view visibility from the measurement phase at the Micro-Box up-to-the visualization phase at the Visibility-Center.

3.1. Design for Spatio-Temporal SmartX Multi-View Visibility

Spatio-temporal summarized monitoring and measurement leveraging SmartX MVF is proposed to deal with the multiple layers such as resource layer (underlay, physical), flow-layer, and workload-layer [22]. In resource-layer visibility, monitoring and visualization of Playground physical resources and inter-connects (e.g. paths and links) are considered. Whereas flow-layer visibility monitors overlay network traffic in near real-time through packet tracing. Flow-layer visibility deals with different levels of flow information (i.e.,

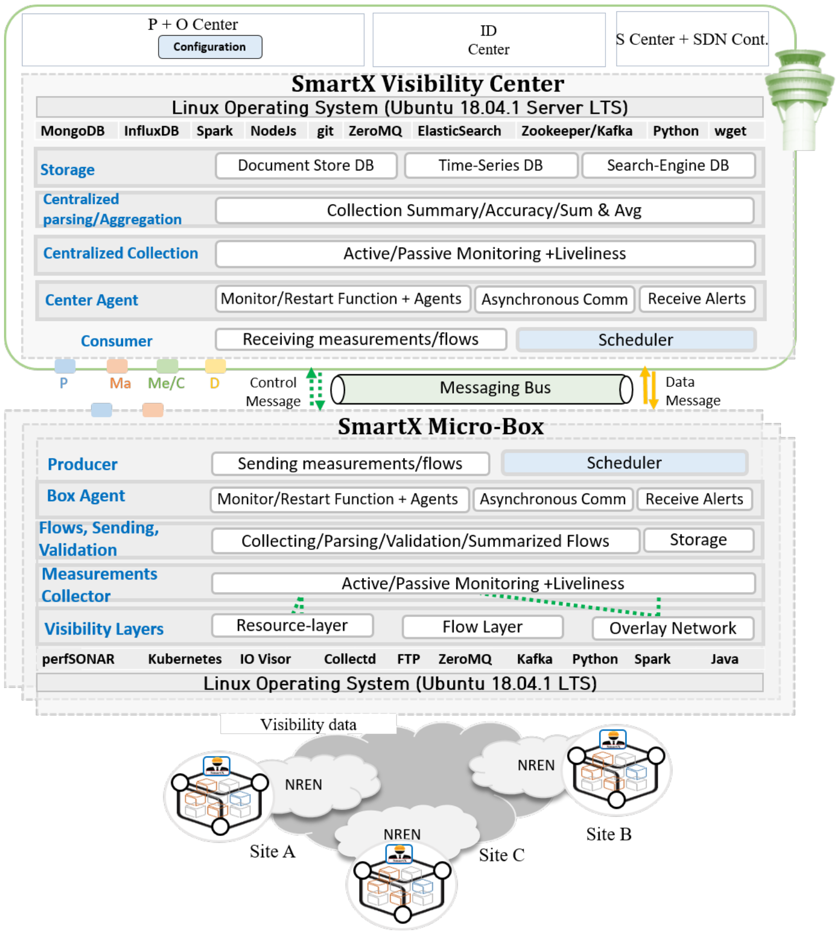

P + O Center
Configuration
ID Center
S Center + SDN Cont.
SmartX Visibility Center
Linux Operating System (Ubuntu 18.04.1 Server LTS)
MongoDB InfluxDB Spark NodeJs git ZeroMQ ElasticSearch Zookeeper/Kafka Python wget
Storage
Document Store DB
Time-Series DB
Search-Engine DB
Centralized parsing/Aggregation
Collection Summary/Accuracy/Sum & Avg
Centralized Collection
Active/Passive Monitoring +Liveliness
Center Agent
Monitor/Restart Function + Agents
Asynchronous Comm
Receive Alerts
Consumer
Receiving measurements/flows
Scheduler
P
Ma
Me/C
D
Control Message
Messaging Bus
Data Message
SmartX Micro-Box
Producer
Sending measurements/flows
Scheduler
Box Agent
Monitor/Restart Function + Agents
Asynchronous Comm
Receive Alerts
Flows, Sending, Validation
Collecting/Parsing/Validation/Summarized Flows
Storage
Measurements Collector
Active/Passive Monitoring +Liveliness
Visibility Layers
Resource-layer
Flow Layer
Overlay Network
perfSONAR Kubernetes IO Visor Collectd FTP ZeroMQ Kafka Python Spark Java
Linux Operating System (Ubuntu 18.04.1 LTS)
Visibility data
NREN
NREN
NREN
Site A
Site B
Site C

Fig. 2. Design for spatio-temporal summarized functionalities at Micro-Box and Visibility-center.

collected, clustered, identified, and un-clustered flow) by utilizing a balanced flow collection, clustering, and tagging. A network flow is typically a sequence of network packets that belongs to a certain network session between two endpoints. Next, workload-layer visibility is responsible for monitoring and visualization of inter-connected containerized functions (e.g. Web, App, and DB) and their deployments for tenant-based applications.

To handle the challenges of spatio-temporal operations, we leverage SmartX Micro-Box distributed at multi-site edge locations. These Boxes support cloud-native (containerized) IoT-Gateway with DataPond functionality (with IoT-Cloud Hub and other application functions) and prepared as Kubernetes-orchestrated workers with SDN-coordinated special connectivity to other SmartX Boxes. As shown in Fig 2, SmartX-Micro-Box is deployed at multiple sites in OF@TEIN+ Playground with capabilities of sending control/data messages at the Visibility-Center employing multiple tools. At each stage, multiple functionalities are deployed to ensure smooth operations. In the Micro-Box *"Vis-*

ibility Collection and Validation" stage collects and validates monitoring and measurement data based on selected visibility metrics. Formatted visibility data is sent to *"Visibility Integration and Storage"* stage where data is stored and integrated to accommodate historical/latest records. *"Aggregation and Summarization stage"* deals with generating consolidated reports and to prepare for data analysis as well as identify any anomalies. Finally, *"Spatio-temporal Visualization"* stage accesses processed data and auto-generates graphical views. Following in this section, we discuss the aforementioned four main stages of spatio-temporal SmartX Multi-View Visibility.

3.1.1 Multi-layer Visibility Measurement At the Micro-Box the visibility collection and validation stage collects and validates monitoring and measurement data based on selected visibility metrics. For reliable and resilient operation network measurement for all traffic passing through, Micro-Boxes are enabled with end-to-end link quality monitoring.

Table 1. A selected list of multilayer visibility metrics for OF@TEIN SDN-enabled multi-site clouds

Visibility-Layer	Metric type	Measured Metrics	Measurement Interval
Underlay-Resource	Active monitoring (connectivity status)	Ping (Between Site-to-site and site-visibility centre)	10 minutes
	Active Monitoring (Network performance)	Latency (Between Site-to-site)	10 minutes
	Active Monitoring (Network capacity)	Bandwidth (tcp, udp)	2 time/day
Flow	Overlay network traffic as passive monitoring	Packet tracing	30 seconds
	Summarized Flows	Flows from Packet tracing	5 minutes
Physical-Resource	Traffic on Interface	Bytes received/sec, Bytes sent/sec	10 seconds
	System performance stats	CPU/Load/memory/disk	10 seconds
Workload	Agent-based monitoring and alert	Applications running status /alerts	5 minutes

As shown in table 1, to capture visibility data for link quality, we collected specific measurement metrics with regular intervals termed as "active monitoring" [23]. For packet precise collection, we capture the network packets from each network interface of SmartX Micro-Box termed as "passive monitoring". To persistently monitor OF@TEIN+ infrastructure, reliable visibility data transfer is applied to sustain the data locally during network failure or application failure. Box-Agents are deployed to convey the status of running functionalities with the support of asynchronous messaging during Agents-based communication.

3.1.2 Integration and Storage Next Visibility Integration stage integrates collected data for generating consolidated reports over a period and identifies any anomalies. Formatted visibility data is sent to Visibility Storage and Staging stage where data is stored.

Table 2. Raw format Multi-View Visibility data for parsing metrics Multiple DB

Visibility Data for (CPU) in JSON Format at InfluxDB	Visibility Data (Load) in JSON Format at Elastic Search	Visibility Data (latency) in JSON Format at MongoDB
[”2020-02-01T02:55:01.54Z”, ”smartx-microbox-gist-1”, ”idle”, 99.5003109715353], [”2020-02-01T02:55:34.28Z”, ”smartx-microbox-gist-1”, ”system”, 5.60003485634267], [”2020-02-01T02:57:21.43Z”, ”smartx-microbox-gist-1”, ”softirq”, 0.0333330630191482]	{ ”_index”: ”pers_collectd_cpu”, ”_type”: ”mirror”, ”_id”: ”120”, ”_score”: 1, ”_source”: { ”@version”: ”1”,”@timestamp”: ”2019-06-27T10:46:33.738Z”, ”boxid”: ”smartx_microbox_um_2”, ”plugin”: ”cpu”,”type_instance”: ”softriq”,”values(%)”: 0.50001657} }}	{ ”_id” : ObjectId (”5ce19081497327a”), ”timestamp” : ”2019/05/20 02:21:05 KST ”microbox-SOURCE” : ”microbox-vnu-1 ”microbox-gist-1” : ”224 ”microbox-gist-2” : ”222 ”microbox-um-1” : ”76.5 ”microbox-um-2” : ”75.0 ”microbox-chula-1” : ”87.5 ”microbox-itc-1” : ”43.5 ”microbox-ucsm-1” : ”101 ”microbox-drukren-1” : ”128 ”microbox-itb-1” : ”78.7 ”microbox-ptit” : ”0.295”}

Table 2 shows the collected metrics for physical/underlay resource layer metrics. These metrics are parsed and integrated into multiple databases according to measurement type. As shown in Table 3 metrics for multiple layers of visibility collection after getting parsed and validated are stored to one of NoSQL Data Stores, which are deployed in our DataLake.

Depending on various monitoring requirements, we choose InfluxDB, MongoDB, and Elasticsearch. We consider three options i.e. MongoDB is suitable for configuration data and Elasticsearch is good for logs, and InfluxDB is suitable for time-series data. Currently, we extensively utilize MongoDB to store Playground configuration and various Playground entities status data such as real-time resource-layer data and aggregated data used for generating a daily visibility report based on aggregation of visibility. While InfluxDB and Elasticsearch are used to store near-real-time metrics data and flows data respectively. Since MongoDB (document-oriented), Elasticsearch (index-oriented), and InfluxDB (time series) are all special-purpose NoSQL Data Stores, they depend on separate store configuration and status data for different resources of OF@TEIN+ Playground. The configuration is handled via MongoDB collection. At the backend, java-based plugins are utilized to store and update Playground metrics in respective databased at regular intervals.

3.1.3 Aggregation and Summarization In this stage, we are concerned with multi-view visibility through time-series summarization and aggregation support. Summarization is performed first at the Micro-Box where packet tracing is summarized on a specific time window (5 minutes) to generate flows. These summarized flows are used to extract useful information and to reduce the size of visibility data.The implementation for the summarization is depicted in Algorithm 1.

Algorithm 1 Packet Tracing for Base Collection

```
Input: Basic IP Packet                                              ▷ p = packet buffer
Output: 6-tuples of header form IP-only packet as tracing criteria  ▷ 6-tuple packet
 1: procedure BASE TRACING($p$)
 2:     if Packet in the Interface does not match the given IP ($p.ethtype \neq IP$) then
 3:         Drop the packet p
 4:     end if
 5:     Extract source address ($srcaddr \leftarrow p.ip.srcaddr$)
 6:     Extract destination address $destaddr \leftarrow p.ip.destaddr$
 7:     Extract packet length$length \leftarrow p.ip.totallength$
 8:     if $p.ip.nextproto \neq TCP$ then
 9:         if $p.ip.nextproto \neq UDP$ then
10:             Drop the packet p
11:         end if
12:     end if
13:     Extract protocol ($protocol \leftarrow p.ip.nextproto$)
14:     Extract destination port ($dstport \leftarrow p.ip.tcp.dstport$)
15:     Extract source port ($srcport \leftarrow p.ip.tcp.srcport$)
16:     return Headers
17: end procedure
```

An appropriate spatio-temporal data-mining algorithm is selected to run on the pre-processed data, and produce output patterns. Common output pattern families include spatio-temporal summarization and change patterns. Spatio-temporal data mining algorithms often have statistical foundations and integrate scalable computational techniques. Output patterns are post-processed and then assist Playground operators to find novel insights and refine data mining algorithms when needed.

Next, a summarization tool is utilized to generate a time-specific summarized multi-view visibility data formatted as HTML format report. Figure 3 shows the equations used for aggregating visibility as Daily collection count, Daily collection percentage and Daily collections missing over a defined time. These compiled summarizations are dissipated animatedly at regular intervals to operators/administrators for troubleshooting and analysis purposes. These reports provide status of distributed Playground resources in one window by aggregating the data with average/percentage/collection for one-day duration. Summarization for multi-view visibility is performed as daily collection count, percentage, missing collection, etc.

3.1.4 Spatio-Temporal Visualization Visibility data in the databases is stored as multiview data in a time-series format. This data is processed and stored either as real-time

$Daily\ collection\ count\ DCC\ for\ space(S)\ and\ time(T)\ can\ be\ calculated\ by\ a\ simple\ method,$

$$DCC(S,T) = \textstyle\sum_{k=1}^{t} d(c_t) \qquad (1)$$

$$where\ d(c_t)\ can\ be\ defined\ as\ d(c_t) = d(s_{i_k}, t_{i_k}),$$
$$t\ as\ measurement\ interval,\ c\ as\ count,$$
$$s_{i_k}\ as\ value\ of\ variable,\ t_{i_k}\ as\ timestamp$$

$Similarly,\ Daily\ collection\ percentage\ can\ be\ calculated\ as,$

$$DCP(S,T) = \textstyle\sum_{k=1}^{t} d(c_t)/t * 100 \qquad (2)$$

$Finally\ Daily\ collections\ missing\ can\ be\ computed\ as,$

$$DCM = \textstyle\sum_{k=1}^{t} d(exp_{c_t}) - d(c_t) \qquad (3)$$

$$where\ d(exp_{c_t})\ is\ daily\ expected\ collection$$

Fig. 3. Equations for Aggregation

measurements or as summarised data. In Visualization stage, processed data is accessed through multiple visualization tools to auto-generates graphical views for further exploration of graphical visibility. Visualization support with the previous approach was limited and desktop-oriented while work presented in this paper provides a long-term visualization in varying perspectives.

1. First measurement metrics that belongs to same category are summarised over a particular time-period to show resource distribution among them.
2. Second we provide visualization of multiple metrics for single-site to identify mutuality between them. For example CPU utilization metrics are visualized on one page to identify similar patterns.
3. Third we multi-site visualization for distributed resources on a single view with the same time-line. This visualization assists in identifying the differences between resources/sites behavior. To further explore the multi-site visualization we employ analysis schemes such as trends patterns and seasonal patterns.

To visualize time-series based metrics we utilize open-source software named Grafana and Kibana, Matplot, abnd Seaborn library. In order to realize the multi-belt onion-ring visualization, we leverage open-source visualization library called psd3 [9]. That is, psd3 is primarily based on D3.js and supports multi-level pie charts, whereas, for deployment Node.js, JavaScript runtime is selected.

As shown in Figure 4, for utilization of Playground operations by end-user, we developed a cloud-native service through Kubernetes and distributed it through OF@TEIN+ Playground. One Kubernetes cluster was formed with Edge IoT-gateways distributed at Multi-site of OF@TEIN+ Playground. First, the Gwangju Institute of Science and Technology (GIST) in Korea uses two Micro-Boxes and one Raspberry Pi as Kubernetes Worker nodes and the P+O Center of SmartX Playground Tower as the master node.

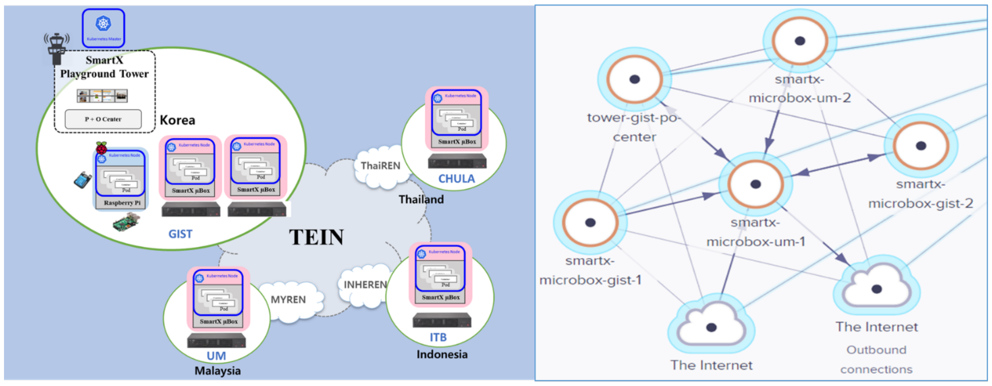


Fig. 4. Kubernetes clusters in OF-TEIN+ Playground.

We also set up a multi-site cloud-native environment with Micro-Box located at Chula-Thailand, ITB-Indonesia and UM-Malaysia. In a multi-site verification environment, Calico network add-on is used for functions connectivity.

3.2. Implementation for Spatio-Temporal MultiView Visibility

The smooth operations of visibility collection require first the identification of key monitoring metrics and corresponding collection tools. Secondly, tools are developed for later stages of visibility data Integration, storage, and summarization. Third, to manage visualization selective tools are employed.

Table 3. SmartX Micro-Box Software: Multiple Levels of Functionalities

Functionality	**Purpose**	**Responsible**
Core Functionalities	Base software functionality,to manage Micro-Box operation. OS, Kernel, Inter-connects, Kubernetes, Docker, etc.	Playground Operators
Basic Functionalities	Development at the Bare metal Monitoring of connection status/health with connected Boxes in the Playground,	Playground Operators
Application Functionalities	Applications in the form factor of containers, orchestrated through Kubernetes. Example:Smart Energy Service	Service Developers

The functionalities in SmartX Micro-Box and Visibility center are categorized as Core, Basic, and Application functionalities based on their purpose and user's role. As mentioned in table 3, *Core Functionalities* deals with base software such as OS, Kernel, etc. Whereas *Basic Functionalities* corresponds development at the bare-metal. The *Application Functionalities* are the services running in the containers.

Table 4. Raw format Multi-View Visibility data for parsing metrics Multiple DB

App Name	**App Role**	**Layer**	**Form factor**	**Source**
IOVisor	Passive Monitoring	Flow-layer	BareMetal	*Micro-Box
Collectd	Passive Monitoring	Flow-layer		
perfSONAR	Active Monitoring	Resource-layer		
Box Liveliness with Tower/App/Box	Active Monitoring		BareMetal	
Apache Kafka	Reporting	Resource-layer		
Box Agent	Reporting	Resource-layer		
ZeroMQ	Reporting	Resource-layer		
Docker Container	Accessibility	Resource-layer	VM/BareMetal	
Collector/Aggregator /Storage/Measurement Report	Preparation	Resource-layer		Visibility-Center
Automated Mail	Reporting	Resource-layer		
Integration	Summarization	Resource-layer		
Grafana/Kibana/Onion-Ring	Vizualization	Resource-layer		
Kubernetes	Orchestration	Resource-layer		P+O Center

Table 4 lists the selected functionalities as tools developed and deployed in Micro-Box and at the Visibility-Center along with their role, layer, and form factor. Currently, for resource layer visibility collection, we utilize Collectd to collect transfers and stores performance statistics of Micro-Box [24]. Besides, for flow-layer visibility collection, we use eBPF-based packet tracing tools [7]. While Apache-Spark with Scala is utilized to generate flows from packet tracing. Besides, for resource-layer visibility we employed PerfSONAR command-line tools for monitoring Playground resources, interconnects, and end-to-end network performance metrics. To ensure consistent running of monitoring applications we placed Box-agents written in python at each resource that periodically check the running status and actively start the application within a short interval. Besides sending status, Box-Agents communicate with Centre-Agent through zeroMQ-based communication to handle asynchronous communication. To reliably transfer the visibility data enduring temporary network loss and delays, we apply Apache Kafka for messaging. A customized python-based tool format the data in the required format with tags and field values. At the visibility Center together with Apache Kafka, to manage reliable and persistent transport of visibility data, we are using Apache Zookeeper, which provides automatic management of metadata and synchronization issues. A customized java-based tool parse and validate the visibility data before storing it in the appropriate databases (i.e., MongoDB, ElasticsSearch, and InfluxDB). At the end of the day, a java based tool aggregates values of multi-view visibility measurements which can be useful for operators and administrators for analysis purposes. For a single view unified visualization, we apply multi-belt onion-ring visualization. Besides, for visualization of measurements from flow-layer and resource-layer visibility, we utilize Kibana and Grafana tools. For orchestration of containerized application, we have employed Kubernetes.

4. Spatio-Temporal Summarized Visualization of SmartX Multi-View Visibility: Verification

For verifying the prototype implementation, we utilize the OF@TEIN+ Playground. We consider metrics from visibility tools and visualized visibility data at each step of visibility workflow.

4.1. Verification Setup

In the initial-stage, deployment of Visibility-center utilizes server-based hardware with the specs: Intel® Xeon CPU E5-2690 V2@3.00GHz, HDD 5.5TB, memory DDR3 12x8GB, and 4 network interfaces of 1 Gbits/s. SmartX Micro-Box consists of Supermicro Super-Server E300-8D server. This Mini-1U server consists of 4 CPU cores with 2.2GHz Intel processor, 240 GB of hard disk, and 32 GB memory. There is one dedicated physical interface for IPMI-based remote access management through CLI (command-line interface) and the web UI (user-Interface). In addition, there are 2 10G + 6 1 GB network interfaces. Visibility Center is configured with Ubuntu 16.04.4 LTS OS, while SmartX Micro-Box loads Ubuntu 18.04.2 LTS OS, with a minimum kernel version of 4.4.0. A dedicated tenant is provided to developer for executing different experiments. In OF@TEIN+ Playground eight SmartX Micro-Boxes are included under current with three types of functionalities i.e. Core, Basic, and Application. Each Box is configured with a scheduled application at synchronized time and intervals for reliable and consistent operations of the Playground.

4.2. SmartX Multi-View Visibility collection and storage

In the proposed solution, measurements are collected at defined regular intervals and sent through Kafka producer at the Visibility Center, where Kafka Consumer fetch the collected measurements. A customized Java-based application validates and parses and format these measurements to write the visibility data in the Database i.e. MongoDB, ElasticSearch, and InfluxDB. Each measurement is tagged with a timestamp and source identifier.

To emphasize the time-series collection several months of visibility data is collected. Figure 5 shows a daily collection of flows at the Visibility-Center from Micro-Box at GIST-1 site. Each collection is tagged with a timestamp and name of source site before storing in the database. In the distributed environment, visibility collections from multiple resources are collected and stored regularly at the center. Administrators are required to maintain an overview of incoming data in terms of volume generated and any missing collection for the observed time.

To facilitate the administrators for smooth operations, we build a customized tool that provides verification of collection through summarized visibility data. Multi-view visibility data is aggregated in-terms of collection values, count, and percentage. Figure 6 shows resource layers metric collection aggregated for a 24-hour duration.

In SmartX Multi-View visibility, the flow layer is key to integrate the multiple visibility layers. To integrate flow-layer with the resource layer, we inspect each network packet and correlate them with predefined virtual and physical resources from the Playground

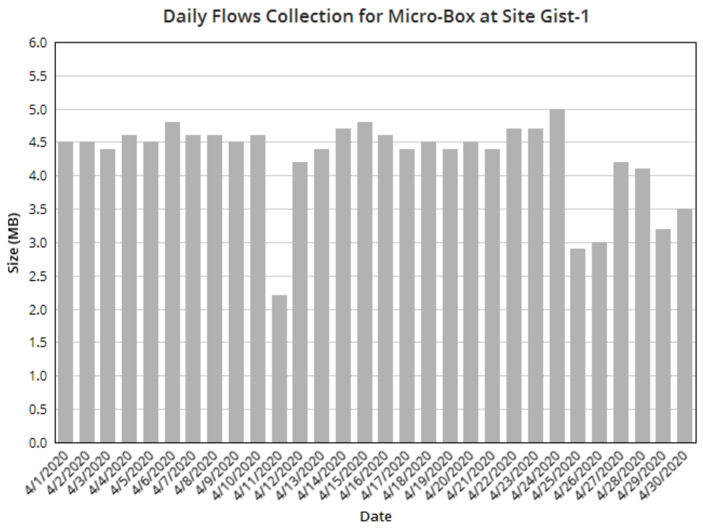
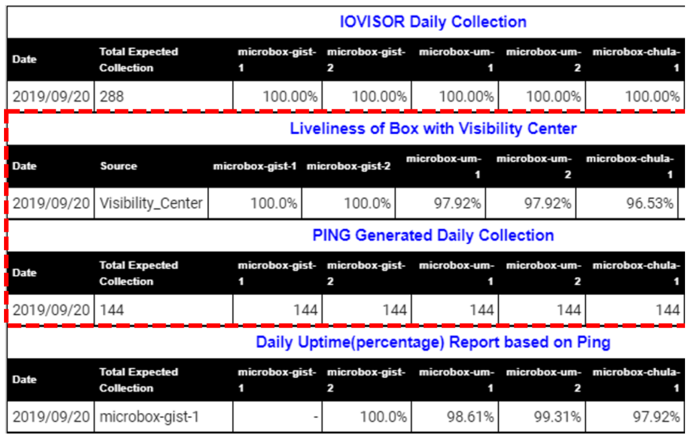


Fig. 5. Result of one-month flows collection for Micro-Box at Site Gist-1



IOVISOR Daily Collection

Date	Total Expected Collection	microbox-gist-1	microbox-gist-2	microbox-um-1	microbox-um-2	microbox-chula-1
2019/09/20	288	100.00%	100.00%	100.00%	100.00%	100.00%

Liveliness of Box with Visibility Center

Date	Source	microbox-gist-1	microbox-gist-2	microbox-um-1	microbox-um-2	microbox-chula-1
2019/09/20	Visibility_Center	100.0%	100.0%	97.92%	97.92%	96.53%

PING Generated Daily Collection

Date	Total Expected Collection	microbox-gist-1	microbox-gist-2	microbox-um-1	microbox-um-2	microbox-chula-1
2019/09/20	144	144	144	144	144	144

Daily Uptime(percentage) Report based on Ping

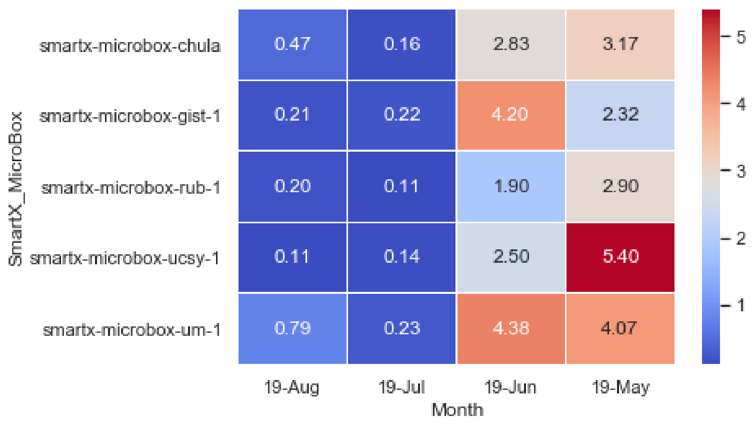
Date	Total Expected Collection	microbox-gist-1	microbox-gist-2	microbox-um-1	microbox-um-2	microbox-chula-1
2019/09/20	microbox-gist-1	-	100.0%	98.61%	99.31%	97.92%

Fig. 6. Daily Visibility Summarized Collection for analysis or multi-view visibility

database. The aggregated network traffic can be used to instantly highlight congested links and identify the source of the flow and associated applications. To minimize the induced size overhead, we have modified SmartX MVF to implement flow aggregation at the Micro-Box with the integration capabilities at the Visibility Center as shown in Figure 7 .

4.3. Spatio-Temporal Summarized Visualization

For verification, in this section we provided three types of summarized visualizations, i.e. (i) Single-site Visualization, (ii) Multi-Metrics on Single site Visualization, and (iii)



SmartX_MicroBox	19-Aug	19-Jul	19-Jun	19-May
smartx-microbox-chula	0.47	0.16	2.83	3.17
smartx-microbox-gist-1	0.21	0.22	4.20	2.32
smartx-microbox-rub-1	0.20	0.11	1.90	2.90
smartx-microbox-ucsy-1	0.11	0.14	2.50	5.40
smartx-microbox-um-1	0.79	0.23	4.38	4.07

Fig. 7. Results for storage volume of summarized flows and packet tracing without summarization over several months of collection

Multi-site Visualization. Each of these visualizations highlights an important visualization aspect for operators/administrators. We start with summarized time-based multi-view visibility data for analysis purposes. Next, we provide visualization for a single system metric for a single site with multiple parameters on the time-line. Followed by a comparison of multiple system metrics on a single site to understand their correlation. Next, we compared measurement from multiple sites to understand the metrics change together over time. Finally, we showed visualizations analysis, leveraging trends, and seasonal pattern schemes.

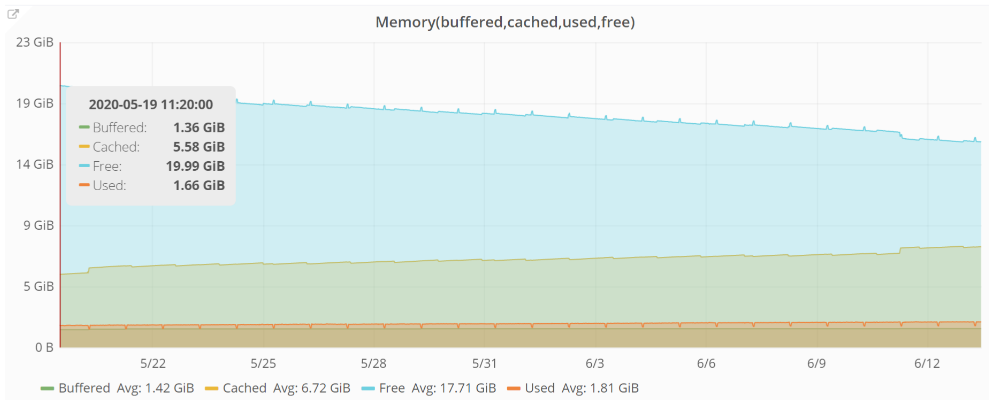


Fig. 8. Visualization results for system Utilization metrics (Memory)

Next we demonstrate time-series visualization for single metrics on a single site with multiple parameters. As shown in 8, Metrics for Memory plugin collects physical memory utilization with values for multiple parameters such as System, idle, nice.

After this we demonstrate summarized visualization for multiple system performance metrics (i.e. load, interface, disk, memory) on a single site in one layer. Unlike previous research, where these metrics were visualized separately. Here we have visualized on a single layer for comparison and analyze the variation over time-line as shown in figure 9 .

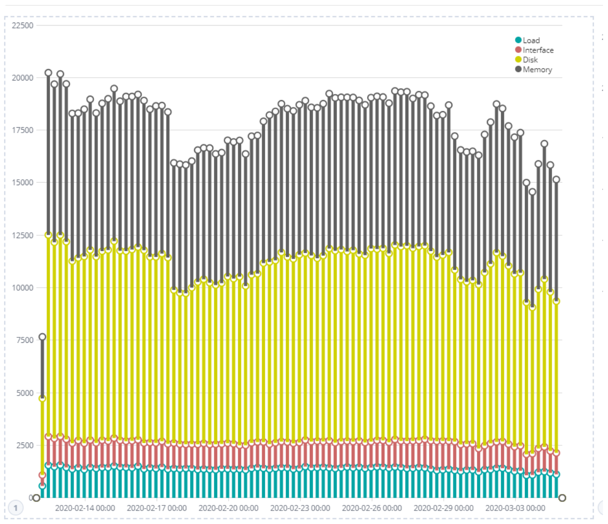
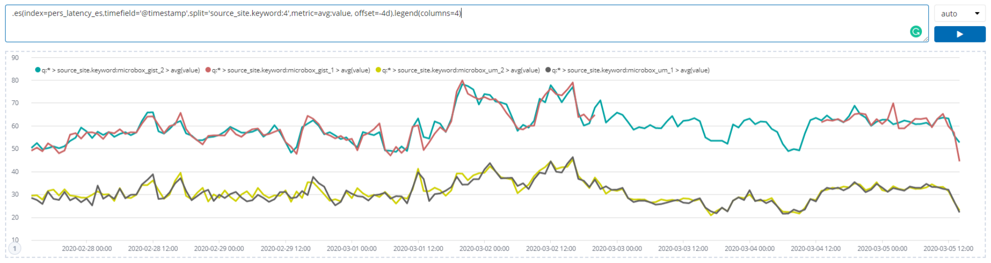


Fig. 9. Results of visualization for multiple metrics of physical resource layers from a single site over time-line



```
.es(index=pers_latency_es,timefield='@timestamp',split='source_site.keyword:4',metric=avg:value, offset=-4d).legend(columns=4)
```

Fig. 10. Result for comparison of visibility measurements from multiple sites over time-line

Figure 10 demonstrate summarized visualization from multiple sites (Micro-Boxes) for underlay resource-layer metrics i.e. latency . Such visualization is useful to identify the comparison of heterogeneous networks for identifying how these metrics change and differ from other sites. As shown in Figure, measurements from four sites are visualized over for a month. There is a clear similarity between measurements of sites (Gist-1 and Gist-2 at South-Korea), while the other two sites (UM-1 and UM-2 at South-Korea) have similar values. However, these two pairs of sites have different latency measurement between them.

In Spatio-temporal analysis, another useful function to use is trend. This function can be used to display a trend line using either a log or linear regression algorithm. There are several ways to think about identifying trends in time series. One popular way is by taking a rolling average, which means that, for each time point, average of the points is taken on either side of it. The number of points is specified by a window size, which needs to be chosen. Figure 11 shows trend analysis for resource layer metrics, i.e. Memory utilization for multiple resources on a single time-line.

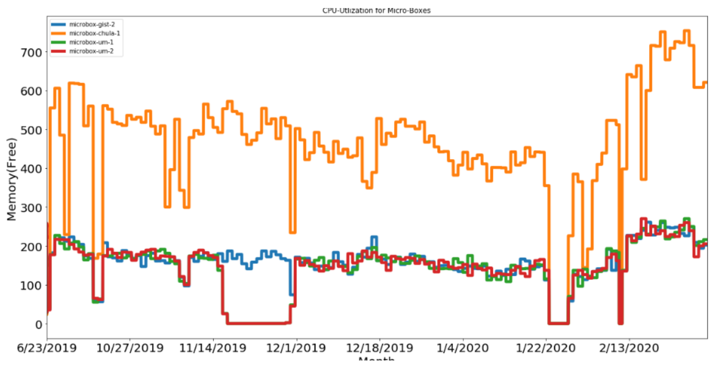


Fig. 11. Trends pattern based visualization of memory utilization for multi-sites

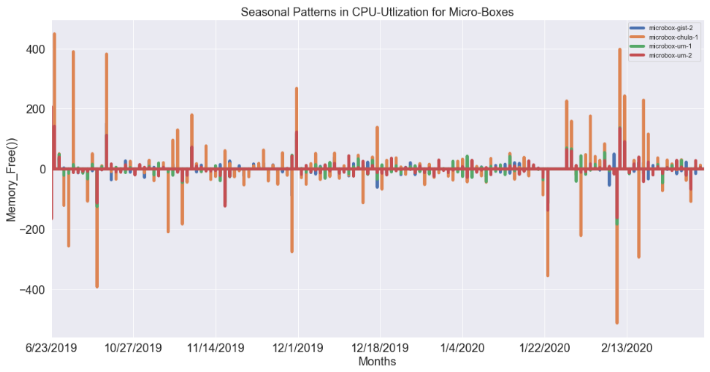


Fig. 12. Seasonal pattern based visualization of memory utilization for multi-sites

The seasonal component in spatio-temporal analysis describes the recurring variation of the time series. Seasonal patterns are utilized to detect seasonality in a time series i.e., Seasons are usually described in the context of mean values averaged per month or several months, and detailing relates mainly to spatial information[25]. For identifying seasonal patterns we subtract the trend computed above (rolling mean) from the original values. This, however, will be dependent on how many data points you averaged. Figure 12 shows seasonal Patterns as time Series Data for memory utilization in the physical resource-layer.

5. Conclusion

In this paper, we presented our initial effort to provide summarized spatio-temporal operations for OF@TEIN+ playground developers and operators to effectively operate and maintain the playground. We verified the work by visualizing multiple layers of visibility leveraging SmartX Multi-view visibility. As future step, we are working on incorporating visualization for containerized applications as temporal visibility collection.

Acknowledgments. (Security) Project: This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2017-0-00421, Cyber Security Defense Cycle Mechanism for New Security Threats). AI Graduate School Project: This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-01842, Artificial Intelligence Graduate School Program (GIST))

References

1. Shifen Cheng, Feng Lu, Peng Peng, and Sheng Wu. Multi-task and multi-view learning based on particle swarm optimization for short-term traffic forecasting. *Knowledge-Based Systems*, 180:116–132, 2019.
2. Polona Štefanič, Matej Cigale, Andrew C Jones, Louise Knight, Ian Taylor, Cristiana Istrate, George Suciu, Alexandre Ulisses, Vlado Stankovski, Salman Taherizadeh, et al. Switch workbench: A novel approach for the development and deployment of time-critical microservice-based cloud-native applications. *Future Generation Computer Systems*, 99:197–212, 2019.
3. J Kim, B Cha, Jongryool Kim, Namgon Lucas Kim, Gyeongsoo Noh, Youngwan Jang, Hyeong Geun An, Hongsik Park, J Hong, D Jang, et al. Of@ tein: An openflow-enabled sdn testbed over international smartx rack sites. *Proceedings of the Asia-Pacific Advanced Network*, 36:17–22, 2013.
4. Aris Cahyadi Risdianto, Junsik Shin, and JongWon Kim. Building and operating distributed sdn-cloud testbed with hyper-convergent smartx boxes. In *International Conference on Cloud Computing*, pages 224–233. Springer, 2015.
5. Aris Cahyadi Risdianto, Muhammad Usman, and JongWon Kim. Smartx box: Virtualized hyper-converged resources for building an affordable playground. *Electronics*, 8(11):1242, 2019.
6. Weisong Shi and Schahram Dustdar. The promise of edge computing. *Computer*, 49(5):78–81, 2016.
7. Muhammad Usman, Aris Cahyadi Risdianto, Jungsu Han, Moonjoong Kang, and JongWon Kim. Smartx multiview visibility framework leveraging open-source software for sdn-cloud playground. In *2017 IEEE Conference on Network Softwarization (NetSoft)*, pages 1–4. IEEE, 2017.

8. Lucas Lacasa, Bartolo Luque, Fernando Ballesteros, Jordi Luque, and Juan Carlos Nuno. From time series to complex networks: The visibility graph. *Proceedings of the National Academy of Sciences*, 105(13):4972–4975, 2008.
9. Bartolo Luque, Lucas Lacasa, Fernando Ballesteros, and Jordi Luque. Horizontal visibility graphs: Exact results for random time series. *Physical Review E*, 80(4):046103, 2009.
10. Jie Zhang and Michael Small. Complex network from pseudoperiodic time series: Topology versus dynamics. *Physical review letters*, 96(23):238701, 2006.
11. Yue Yang and Huijie Yang. Complex network-based time series analysis. *Physica A: Statistical Mechanics and its Applications*, 387(5-6):1381–1386, 2008.
12. Norbert Marwan, Jonathan F Donges, Yong Zou, Reik V Donner, and Jürgen Kurths. Complex network approach for recurrence analysis of time series. *Physics Letters A*, 373(46):4246–4254, 2009.
13. Xiaoke Xu, Jie Zhang, and Michael Small. Superfamily phenomena and motifs of networks induced from time series. *Proceedings of the National Academy of Sciences*, 105(50):19601–19605, 2008.
14. Grégoire Nicolis, A Garcia Cantu, and Catherine Nicolis. Dynamical aspects of interaction networks. *International Journal of Bifurcation and Chaos*, 15(11):3467–3480, 2005.
15. Andriana SLO Campanharo, M Irmak Sirer, R Dean Malmgren, Fernando M Ramos, and Luís A Nunes Amaral. Duality between time series and networks. *PloS one*, 6(8), 2011.
16. Zhe Jiang and Shashi Shekhar. Spatial and spatiotemporal big data science. In *Spatial Big Data Science*, pages 15–44. Springer, 2017.
17. Shashi Shekhar, Zhe Jiang, Reem Y Ali, Emre Eftelioglu, Xun Tang, Venkata Gunturi, and Xun Zhou. Spatiotemporal data mining: a computational perspective. *ISPRS International Journal of Geo-Information*, 4(4):2306–2338, 2015.
18. Yasushi Sakurai, Masatoshi Yoshikawa, and Christos Faloutsos. Ftw: fast similarity search under the time warping distance. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 326–337, 2005.
19. Saif Ahmad, Tugba Taskaya-Temizel, and Khurshid Ahmad. Summarizing time series: Learning patterns in 'volatile'series. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 523–532. Springer, 2004.
20. Shreya Ghosh, Anwesha Mukherjee, Soumya K Ghosh, and Rajkumar Buyya. Mobi-iost: mobility-aware cloud-fog-edge-iot collaborative framework for time-critical applications. *IEEE Transactions on Network Science and Engineering*, 2019.
21. Muhammad Usman, Nguyen Tien Manh, and JongWon Kim. Multi-belt onion-ring visualization of of@ tein testbed for smartx multi-view visibility.
22. Salman Taherizadeh, Andrew C Jones, Ian Taylor, Zhiming Zhao, and Vlado Stankovski. Monitoring self-adaptive applications within edge computing frameworks: A state-of-the-art review. *Journal of Systems and Software*, 136:19–38, 2018.
23. Sihyung Lee, Kyriaki Levanti, and Hyong S Kim. Network monitoring: Present and future. *Computer Networks*, 65:84–98, 2014.
24. Florian Forster and S Harl. collectd–the system statistics collection daemon, 2012.
25. Yury Kolokolov and Anna Monovskaya. Multidimentional analysis and visualization of changes in characteristic seasonal patterns of local temperature dynamics. In *2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, volume 1, pages 321–327. IEEE, 2017.

Muhammad Ahmad Rathore received his BS in Computer Science from Comsats Institute of Information Technology, Islamabad, Pakistan, and his MS in Information and Communication Systems Security from Royal Institute of Technology, Stockholm, Sweden. Currently, he is pursuing Ph.D. Degree from Networked Intelligence Lab, School of

Electrical Engineering and Computer Science, Gwangju Institute of Technology, GIST, South Korea. His main research interests are in cloud-based technologies, visualization, and data analysis.

JongWon Kim received Ph.D. degree in Control and Instrumentation Engineering from Seoul National University, Seoul, Korea, in 1994. In 1994-1999, he was with the Department of Electronics Engineering at the KongJu National University, KongJu, Korea, as an Assistant Professor. From 1997 to 2001, he was visiting the Signal & Image Processing Institute (SIPI) of Electrical EngineeringSystems Department at the University of Southern California, Los Angeles, USA, where he has served as a Research Assistant Professor since Dec. 1998. From Sept. 2001, he has joined Gwangju Institute of Science & Technology (GIST), Gwangju, Korea, where he is now working as the chair of GIST AI Graduate School, which was established late 2019 as one of 5 government-sponsored AI graduate schools in Korea.

Received: March 17, 2020; Accepted: September 15, 2020.