A Machine Learning-based Real-time Monitoring System for Classification of Elephant Flows on KOREN

Waleed Akbar[1], Javier J. D. Rivera[2], Khan T. Ahmed[1], Afaq Muhammad[1] and Wang-Cheol Song[1]*

[1] Department of Computer Engineering, Jeju National University
Jeju, South Korea
[E-mail : waleedwali786/talhajadun/afaq24@gmail.com, philo@jejunu.ac.kr]
[2] Department of Electrical Engineering, Jeju National University
Jeju, South Korea
[E-mail : shaifvier@gmail.com]
*Corresponding author : Wang-Cheol Song

Received March 19, 2022; accepted April 19, 2022; published August 31, 2022

Abstract

With the advent and realization of Software Defined Network (SDN) architecture, many organizations are now shifting towards this paradigm. SDN brings more control, higher scalability, and serene elasticity. The SDN spontaneously changes the network configuration according to the dynamic network requirements inside the constrained environments. Therefore, a monitoring system that can monitor the physical and virtual entities is needed to operate this type of network technology with high efficiency and proficiency. In this manuscript, we propose a real-time monitoring system for data collection and visualization that includes the Prometheus, node exporter, and Grafana. A node exporter is configured on the physical devices to collect the physical and virtual entities resources utilization logs. A real-time Prometheus database is configured to collect and store the data from all the exporters. Furthermore, the Grafana is affixed with Prometheus to visualize the current network status and device provisioning. A monitoring system is deployed on the physical infrastructure of the KOREN topology. Data collected by the monitoring system is further pre-processed and restructured into a dataset. A monitoring system is further enhanced by including machine learning techniques applied on the formatted datasets to identify the elephant flows. Additionally, a Random Forest is trained on our generated labeled datasets, and the classification models' performance are verified using accuracy metrics.

Keywords: Software Defined Network (SDN), Real-time monitoring, KOREN, NetFlow, Machine learning, Elephant flows.

A preliminary version of this paper appeared at the International Conference on Internet (ICONI 2021) on December 12-14, 2021, in Jeju Island, Korea. This version includes concrete information about real-time monitoring, such as deployment, configuration, and topology

http://doi.org/10.3837/tiis.2022.08.019 **ISSN : 1976-7277**

1. Introduction

The rapid growth of computer networks has a high impact on traffic velocity, volume, and variety. The SDN decouples the traditional network device into two segments, the data plane for forwarding and the control plane for management and control [1]. The data plane is distributed on each virtual switch, and the central control plane is placed on the controller. The SDN provides many advantages over the traditional network, such as scalability, elasticity, and policy handling [31]. In SDN networks, the virtual switches are directly instructed by the controller. The flow tables are dynamically updated according to the network requirements. A controller is a central entity that manages all the flows in the network. The ONOS (Open Network Operating System) is the most widely used controller [3]. Commonly, the OpenFlow protocol is used to deploy flow rules on the switches by the controller [2]. An OpenFlow is the most widely used southbound protocol for communication between the control plane and data plane.

Fig. 1 depicts the basic SDN architecture consisting of three layers: application layer, control layer, and infrastructure layer. An application layer provides the abstraction for services and applications from the underlying physical architecture. Northbound APIs are provided to communicate with the control layer. The intelligent central controller is the brain of SDN architecture, and it provides the services such as security policies, resources optimization policies, bandwidth management policies, and routing policies. The controller communicates with the underlying infrastructure through southbound APIs. Usually, OpenFlow is the major protocol used as southbound APIs. In SDN, Virtual Infrastructure Manager (VIM) manages and maintains a physical infrastructure. Logically, the controller extracts the information about physical resources via a VIM and provides this information to services on the application layer for further analysis.

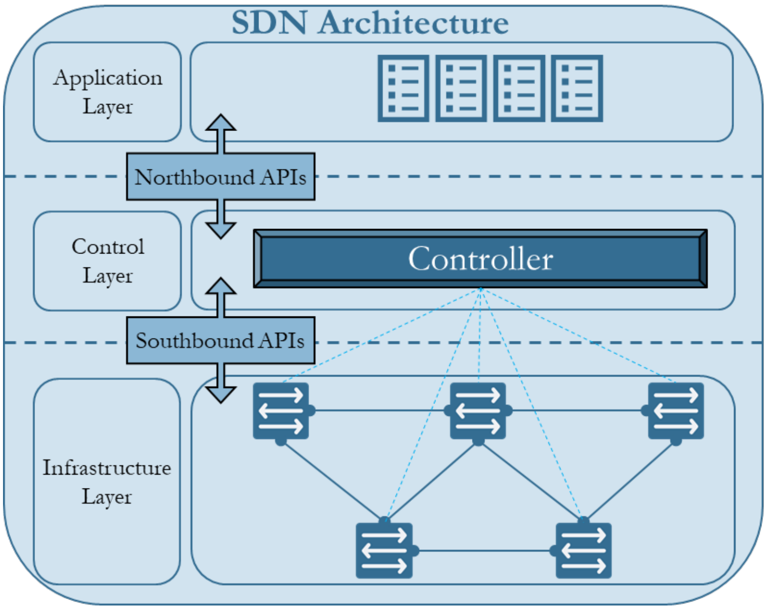

Fig. 1. Basic SDN Architecture consisting of application, control, and infrastructure layer. Whereas the southbound and northbound APIs is the way of communication between the layers.

One of the vital features in SDN is fine-grain monitoring. It assists the network to realize the behavior of underlying infrastructure and view the condition of network elements. With optimization of monitoring architecture, the network performance is highly improved. Therefore, to provide such lower-level monitoring detail about the virtual and physical entities, an efficient and scalable monitoring system is needed. A monitoring system should have capabilities to record network traffic and capture resource utilization. Network monitoring is classified into four steps. The first step is data collection from all the entities in the network. The second step is to do data processing in real-time to capture all the network dynamics. The popular tools for data processing and collection are the Prometheus server and Apache Kafka. The third step is data storage; a time-series database is used for data storage such as Influx DB, Prometheus DB, and Mongo DB. The fourth step is data visualization to view network behavior and resource utilization. The most popular tools for visualization are Grafana or Kibana.

One aspect of the monitoring system is to visualize the whole network, and the other aspect is to analyze the accumulated data. The network engineer utilizes the former method to keep an eye on the whole network and detect ambiguous activity such as high resource utilization, high request count on a single server, and low resource availability. A threshold-based notification is available to identify these ambiguous activities in the network. Mostly, the latter is use case constrained such as traffic classification, link utilization, path optimization, and many more. In this era, the more focus of researchers is on data analysis and data engineering. That means interpreting the data and predicting the future state of data. Many machine learning and deep learning models are implemented in this study. Such studies include traffic classification [34], encrypted traffic analysis [33], and malicious traffic detection [32].

Traffic classification has a significant impact on network performance. Unable to detect the malicious traffic on time will hamper the normal flow of the network, and in worse situations, it crashes the whole network. The large flows usually consume the high network resources, and their presence affects the minor flows on the network. In network terms, such flows are called elephant flows. The elephant flow has a high impact on the network performance because it tends to consume high bandwidth for a long duration [29]. There is a high probability of congestion in-network when an elephant flow occurs. These days, data generation is too fast, and network changes are happening at a higher frequency. If the elephant flow is not detected on time, it may crash or hamper the normal working of the network, and it will continue to impact the working of the entire system slowly. Therefore, the importance of flows classification and identification is increasing rapidly. Many researchers [23], [27] have already proposed a monitoring system in SDN. They are either deployed in test-bed environments [25], [26] or in simulation environments [21], [22], [24]. These implementations cannot completely reflect the real behavior of a network. Most of the assumptions mentioned in these implementations are not realistic, and real networks should not have such a type of constraint and conditions. Therefore, the deployment of monitoring systems in a real environment is foremost important.

To fulfill the SDN network requirements and provide the fine-grain monitoring capabilities, we proposed a real-time monitoring system to visualize and collect the real-time data of the KOREN infrastructure [4]. Later, we pre-process the collected data to construct a dataset. Further, we train the un-supervised classification models and evaluate the model performance using supervised learning models. Major modules of our proposed architecture are as follows.

The first module is a real-time network collection and visualization system. It consists of five major components Grafana [5], Prometheus [6], node exporter [7], push gateway [8], and a NetFlow collector [9]. The node exporter and NetFlow expose the API (Application programming interfaces) endpoints to connect and collect the metrics from physical and virtual entities. Prometheus is configured to pull the information from the node exporter endpoint. Similarly, Prometheus push-gateway is configured to push the metrics from the NetFlow collector. Then, the Grafana is used to visualize the information that helps the admin understand the network's state. After the data collection, we pre-process the data to understand the insights. Data cleaning, standardization, and feature selection are performed in the pre-processing step. Furthermore, we label the dataset using unsupervised machine learning models and verify models' classification accuracy using Random Forest. Fig. 2 shows the steps performed in our proposed methodology for data analysis.

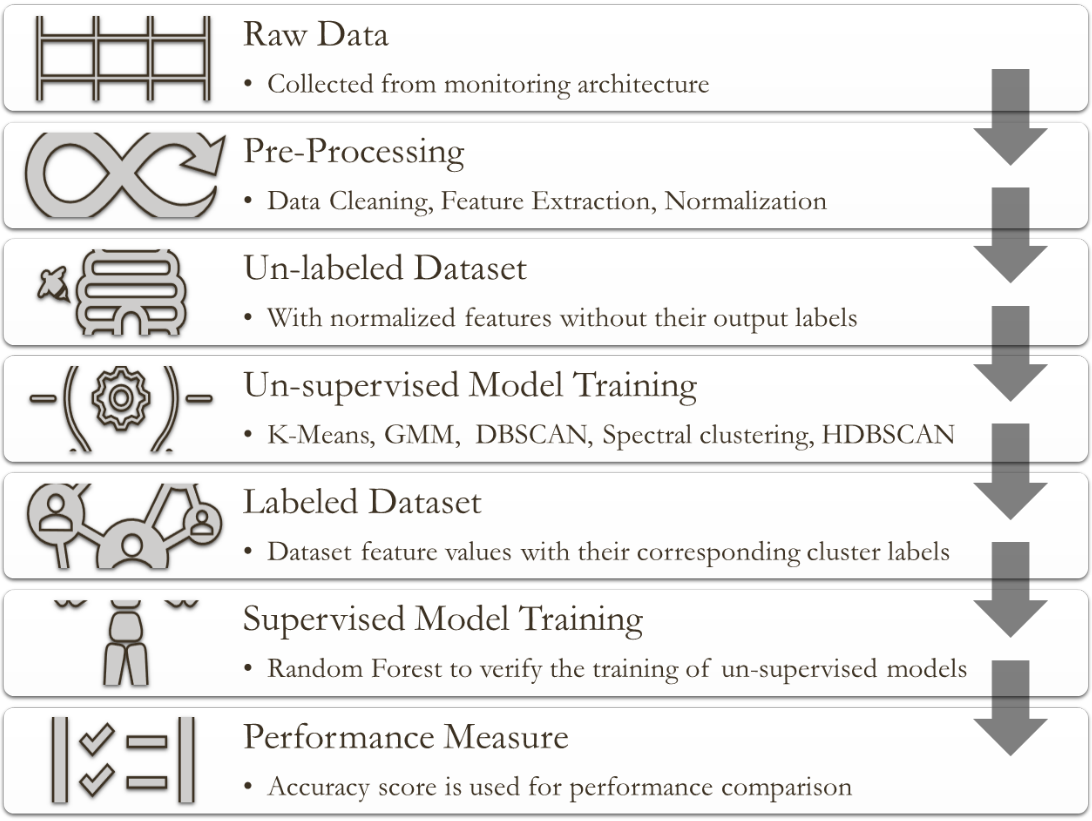
Raw Data
• Collected from monitoring architecture
Pre-Processing
• Data Cleaning, Feature Extraction, Normalization
Un-labeled Dataset
• With normalized features without their output labels
Un-supervised Model Training
• K-Means, GMM, DBSCAN, Spectral clustering, HDBSCAN
Labeled Dataset
• Dataset feature values with their corresponding cluster labels
Supervised Model Training
• Random Forest to verify the training of un-supervised models
Performance Measure
• Accuracy score is used for performance comparison

Fig. 2. The steps performed in our methodology for data analysis: How data is analyzed and pre-process to make an un-label dataset. Later, dataset labeled using un-supervised models and evaluate using supervised model

The manuscript division is as follows: the next section discusses the real-time monitoring system deployment and data collection module. Section three explains the data pre-processing and then describes the comprehensive detail of machine learning models, and this section ends with an explanation of the supervised learning model. Results and discussion are presented in section four, and the last section concludes the paper.

2. Real-time Monitoring System

We have developed a real-time monitoring system on KOREN (Korea Education Research Network). which is an SDN system consisting of multiple spines and leaf switches, the KOREN test-bed topology is shown in **Fig. 3**.

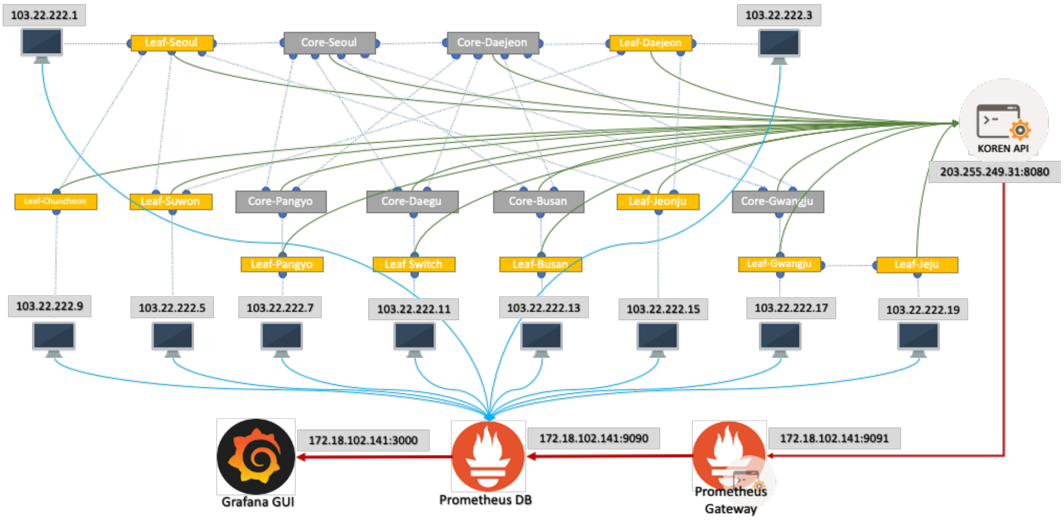

Fig. 3. Real-Time Monitoring Architecture, consisting of Agents, KOREN API, Prometheus Gateway, Prometheus DB and Grafana Dashboard

2.1 Proposed Monitoring System

Fig. 3 depicts our integrated monitoring system with the complete KOREN topology that consists of six spines and ten leaf switches. A blue dotted line presents the interconnection between spine-to-spine and spine-to-leaf switches. Multiple paths between the switches provide redundancy in case of path or device failure. The NetFlow agent is installed in switches to monitor the real-time traffic. To restrict the NetFlow agent's scope and secure public network traffic, the agent's output is provided through the KOREN API. API's output is directly pushed into the Prometheus push-gateway that is configured to collect the information from the short-term jobs running in the network. Later, push-gateway stores the data in Prometheus DB. Moreover, we deployed the node exporter agents on end hosts to monitor the physical and virtual entities inside the host. A node exporter extracts the metrics such as interface network traffic, CPU (Central Processing Unit) utilization, memory utilization, etc. [10]. Prometheus directly pulls the data from node exporter agents and stores the data in DB. Grafana queries the Prometheus data and visualizes the response according to the use case scenarios.

We design the Grafana dashboard to visualize network status and resource utilization, as shown in **Fig. 4**. The first block shows the lists of switches and hosts for selection. Host block represents the host's resource and network information. Host information includes processes utilization and memory utilization, and network information includes packets-in, packets-out, and packets per protocol. Moreover, it depicts virtual interface information. A switch block displays the selected switches link information such as packet-in, packet-out, in-drop, and out-drop.

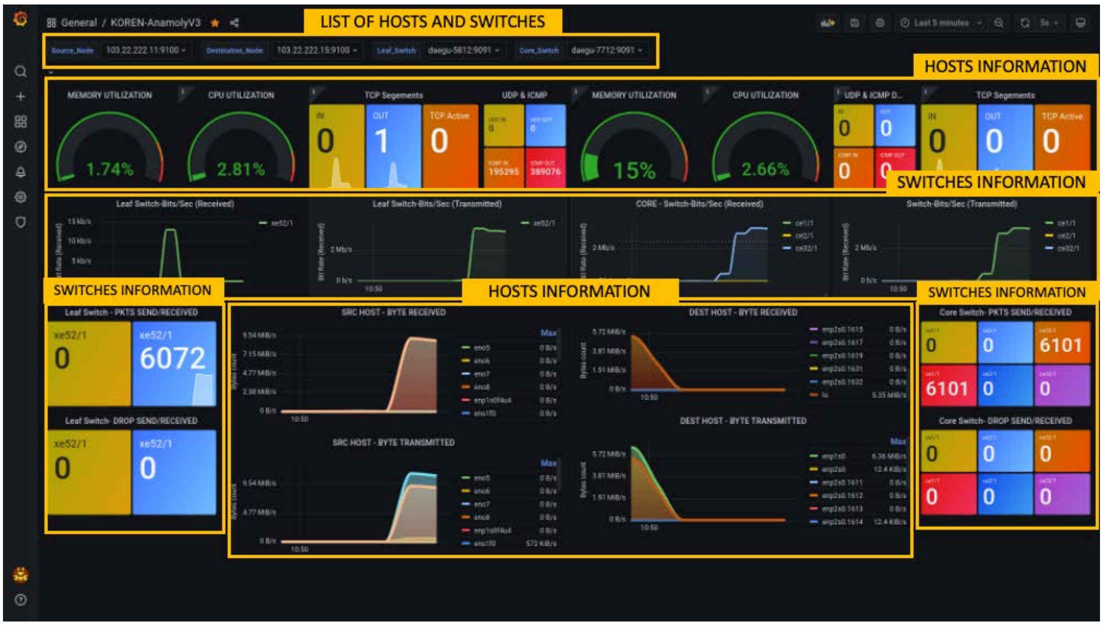
General / KOREN-AnamolyV3
LIST OF HOSTS AND SWITCHES
Source_Node 103.22.222.11:9100
Destination_Node 103.22.222.15:9100
Leaf_Switch daegu-5812:9091
Core_Switch daegu-7712:9091
Last 5 minutes
HOSTS INFORMATION
MEMORY UTILIZATION
1.74%
CPU UTILIZATION
2.81%
TCP Segements
IN 0
OUT 1
TCP Active 0
UDP & ICMP
195295
389076
MEMORY UTILIZATION
15%
CPU UTILIZATION
2.66%
UDP & ICMP D...
TCP Segements
IN 0
OUT 0
TCP Active 0
SWITCHES INFORMATION
Leaf Switch-Bits/Sec (Received)
Leaf Switch-Bits/Sec (Transmitted)
CORE - Switch-Bits/Sec (Received)
Switch-Bits/Sec (Transmitted)
SWITCHES INFORMATION
Leaf Switch - PKTS SEND/RECEIVED
xe52/1 0
xe52/1 6072
Leaf Switch- DROP SEND/RECEIVED
xe52/1 0
xe52/1 0
HOSTS INFORMATION
SRC HOST - BYTE RECEIVED
SRC HOST - BYTE TRANSMITTED
DEST HOST - BYTE RECEIVED
DEST HOST - BYTE TRANSMITTED
SWITCHES INFORMATION
Core Switch- PKTS SEND/RECEIVED
0 0 6101
6101 0 0
Core Switch- DROP SEND/RECEIVED
0 0 0
0 0 0

Fig. 4. Grafana Dashboard: with dropdown list of host and switch, the host resource utilization information such as memory and CPU, and switches traffic information such bytes and packets count.

2.1 System Configuration

The core of the monitoring system is Prometheus DB, and it stores entire information in real-time. First, Prometheus is installed, and the endpoint is configured to receive the metrics from the node exporter connected to the host device. The node exporter extracts the host machine's resource utilization and network-related metrics. NetFlow is configured on switches to monitor link network traffic. For security purposes of a public network, direct access is limited to admin only. Therefore, the KOREN provides HTTP (Hypertext Transfer Protocol) APIs to access the NetFlow output. A Grafana is configured to visualize the information available in DB. Table 1 shows the port of running services for Grafana, Prometheus, Prometheus push-gateway, node-exporter, and NetFlow collector.

Table 1. Port numbers of services

S. No.	Name	URLs
1	Grafana	localhost:3000
2	Prometheus	localhost:9090
3	Node Exporter	<node_ip>:9100
4	Push Gateway	localhost:9091
5	netFlow Collector	<server_ip>:6343

3. Machine Learning for Traffic Classification

Many research trends are moving towards the machine learning architecture model because of its high performance in other research fields [30]. Our proposed approach is also built on the machine learning architecture. In this manuscript, the following unsupervised learning techniques are analyzed to detect elephant flows. The K-Means [11], Gaussian distribution [12], DB-Scan [13], Spectral clustering [14], and HDB-Scan [15] are implemented. Before that, in the pre-processing step, feature selection methods are used to identify the key feature in the dataset.

Table 2. List of all the metrics collected from the monitoring system

S. No.	Metric Name	S. No.	Metric Name
1	Time stamp	8	Host
2	Protocol name	9	Protocol id
3	Bytes transferred	10	Packets transferred
4	Destination address	11	Source address
5	Destination port number	12	Source port number
6	Destination port name	13	Source port name
7	Flow records	14	Sequence number

3.1 Data Pre-Processing

The raw data is analyzed in the pre-processing step to extract the key features. **Table 2** shows the list of entire features. All columns with Nan, Null, None, or single value are removed in the data cleaning step. A “Kendell” [16] correlation matrix is applied to identify the essential features as a feature selection technique. Scores near one are highly co-related and negative co-relation on the other side. The features with values near -1 are dropped from the dataset since they negatively influence the model training process. Therefore, we calculated the “Kendell” correlation matrix again to show the strong correlation between the final selected features, as shown in **Fig. 5**. The remaining features are bytes transferred, a destination address, destination port, packet transferred, source port, and the protocol used. The impact of elephant

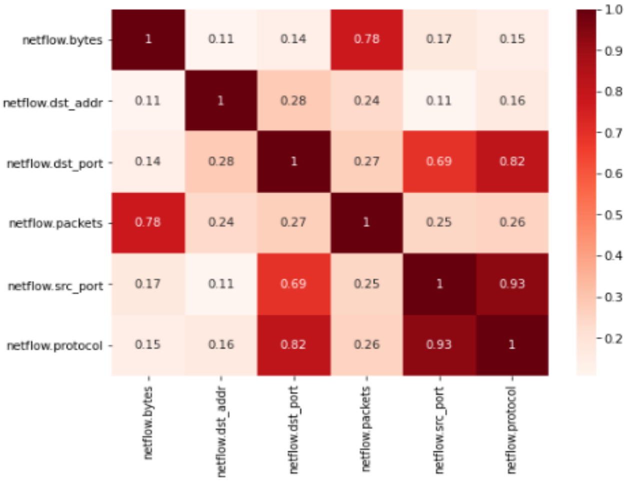

Fig. 5. Correlation matrix score of finalized features, close to one is best score and close to zero is worse score

flow is based on the number of bytes transferred on the network, and it is important to understand the relationship of bytes value with other features. Therefore, we applied a PCA (Principal Component Analysis) on the remaining features except for the bytes column [17]. After that, the dataset is re-scaled between 1 to 10 using the sci-kit learn Min-Max Scalar method [18]. **Fig. 6** visualizes the relationship of bytes with all other features.

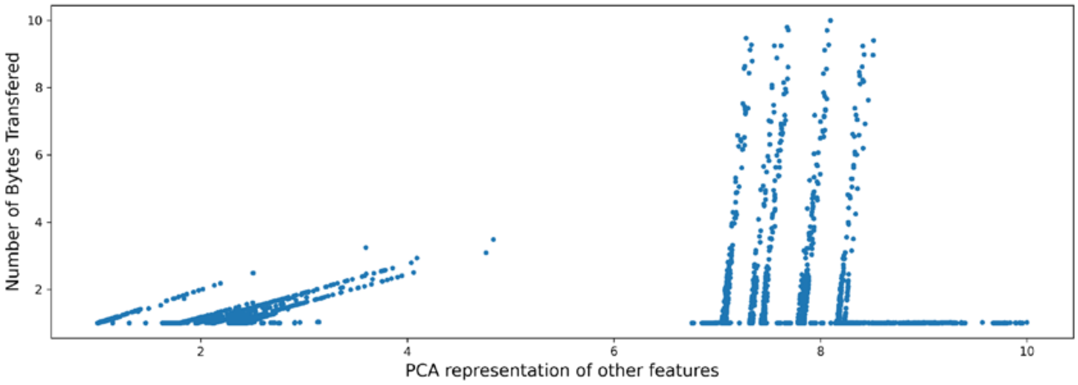

Fig. 6. Visualization of PCA score of all other features against the total number of bytes transferred

3.2 Unsupervised machine learning models

Once the pre-processing step is completed, all non-relevant features have been removed, and the dataset is normalized and standardized. At this time, the dataset is prepared for machine learning model training. Ensemble learning is a technique to combine the output of multiple models and select the best possible answer. An ensemble of unsupervised clustering algorithms is evaluated on the prepared dataset; this section describes the learning models.

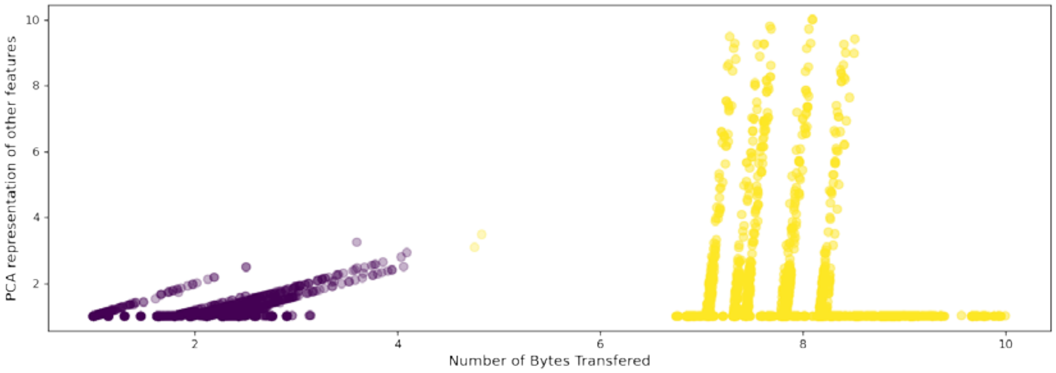

Fig. 7. K-Means clustering labels are visualized here, small purple circles show the normal data, and yellow are classified as elephant flows.

3.2.1 K-Means

K-means is a popular clustering algorithm that divides the dataset into k number of non-overlapping clusters. Overall, the algorithm's operating speed is fast, but learning usually falls in local minima. We select k=2 because our data has two visible sets. Initially, centroids are randomly placed and updated after each iteration. Every data point will belong to one of these clusters. The allocation of clusters is based on the lowest distance of a point from the centroid. The sum of squared distance between the data point and cluster centroid is calculated [35]. The mathematical representation for the calculation of centroid is shown in (1). **Fig. 7** shows the k-means clusters.

$$\sum_{i=0}^{n} \min_{\mu_j \in C}(\|x_i - \mu_j\|^2) \dots (1)$$

3.2.2 Gaussian Mixture Models (GMM)

GMM is a popular and powerful probabilistic clustering technique. It creates the n number of Gaussian distributions, and each distribution represents a single cluster. It performs the expectation step to calculate the expectations of likelihood based on the hyperparameter current estimates. Then, the maximization step computes the maximum probability found in the previous step. The point allocated to a cluster is based on the likelihood of data point mean and variance value. The assignment of mean and variance values to data points is based on the expectation-maximization (EM) technique [20]. The mathematical representation of the expectation and maximization steps is shown by (2) and (3), respectively.

$$Q\left(\theta \middle| \theta^{(t)}\right) = E_{\left(Z \middle| X, \theta^{(t)}\right)}[\log L(\theta; X, Z)] \dots (2)$$

$$\theta^{(t+1)} = arg \min_{\theta} Q\left(\theta \middle| \theta^{(t)}\right) \dots (3)$$

The GMM is applied to label the dataset and then plot to visualize the model predictions. GMM creates two clusters, but they overlap, as seen in Fig. 8.

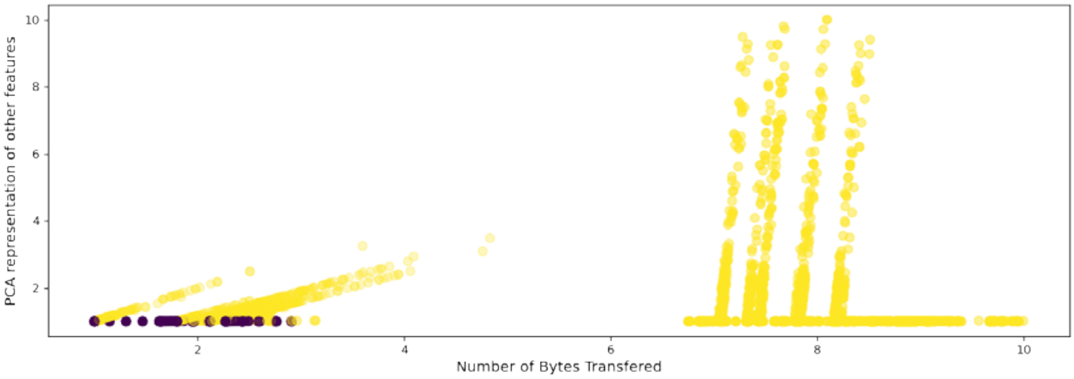

Fig. 8. GMM clustering labels are visualized here, small purple circles show the normal data, and yellow are classified as elephant flows.

3.2.3 Density-based Clustering Algorithm (DBSCAN)

DBSCAN (Density-based spatial clustering of applications with noise) is another data clustering algorithm for classifying data into two or more classes. In DBSCAN, there is no need to specify the number of clusters. It creates clusters based on the density of the points. It takes a minimum number of points in the region to be called a cluster and a minimum measured distance of neighborhood points. This algorithm divides all the data points into core, border, and noise points. The combination of core and data points makes the cluster, and noise points are the outliners. Fig. 9 depicts the DBSCAN clusters.

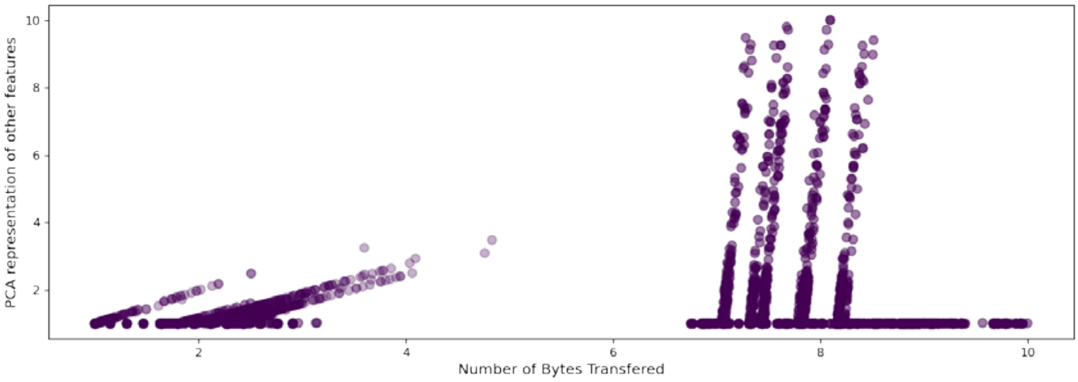

Fig. 9. DB-SCAN clustering labels are visualized and, it labels the data as single cluster

3.2.4 Spectral Clustering

It is a widely used clustering algorithm technique based on connected graph format used for non-convex data points. It inputs no information about the shape and size of the cluster. The data point cluster allocation is based on the eigenvalues and eigenvectors. It also performs the dimension reduction on eigenvalues and eigenvectors before clustering. **Fig. 10** portrays the spectral-based clusters.

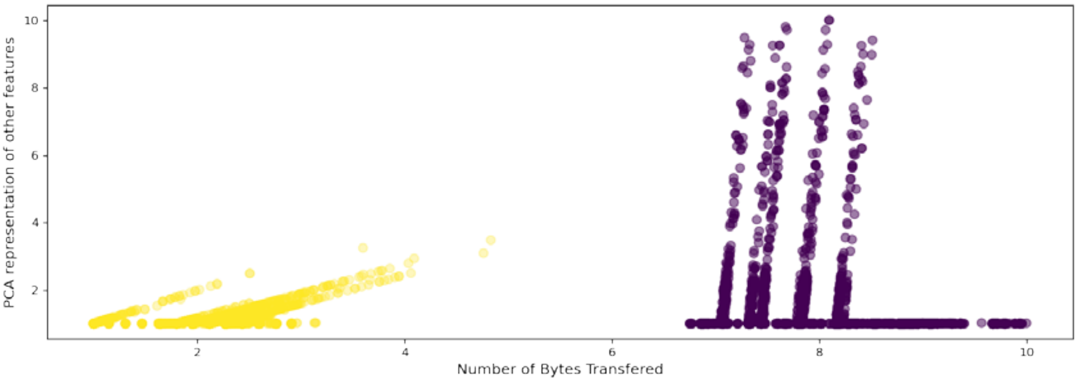

Fig. 10. Spectral clustering clustering labels are visualized here, small yellow circles show the normal data, and purple are classified as elephant flows.

3.2.5 Hierarchal DBSCAN

It is the enhanced version of DBSCAN that takes the core points and distance of data points from the core points. Then it calculates the readability distance between all the points. Further, it creates the spanning tree with minimum reachability distance and converts it into a connected points hierarchy. Later, it will sort the points and condense the complex tree into smaller trees. A cluster is chosen based on persistent and lifelong tree features, and stability is calculated. The cluster with the higher stability is selected. **Fig. 11** shows the HDBSCAN clusters.

Each of these models is trained on the pre-processed dataset. The model predicts the number of clusters and the number of data points in each cluster, and then those points are labeled against the cluster they lie in. After the prediction the datasets are labeled with those cluster numbers. Hence five datasets are generated with different cluster labels.

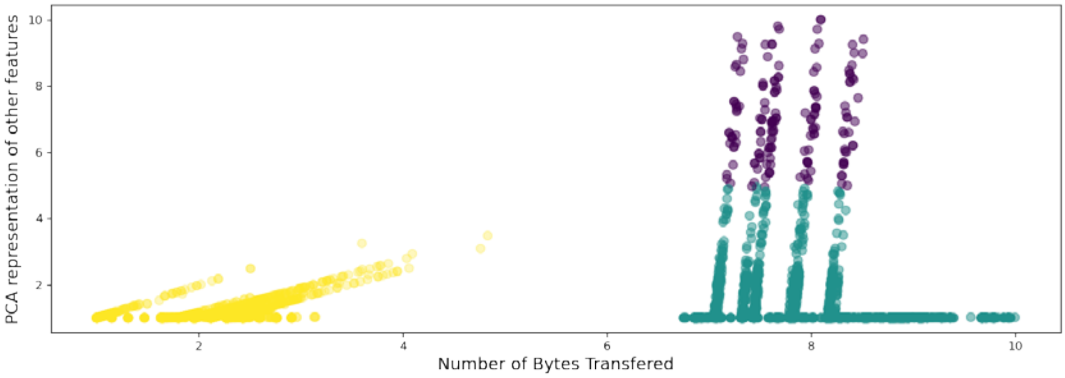

Fig. 11. Spectral clustering labels are visualized here, small yellow circles show the normal data, and purple and green are classified as elephant flows.

3.3 Model Configuration

The configuration setting of hyper-parameters for un-supervised models is mentioned in the **Table 3**. Only values that provided the best results are listed in table.

Table 3. Model configurations hyper-parameters with their selected values

S. No.	Model	Hyper-parameters	Values
1	K-Means	Number of clusters	2
		Maximum iterations	1000
		Algorithm	‘Elkan’
2	GMM	Number of components	2
3	DB-SCAN	‘eps’	5
		Minimum samples	10
4	Spectral Clustering	Number of clusters	2
		Assign labels	‘k-means’
5	HDB-SCAN	Cluster selection epsilon	5
		Minimum samples	1000
		Minimum cluster size	1000

3.4 Supervised Learning

A labeled dataset is created by assigning a cluster-id to each of the data points using multiple un-supervised learning models. All models predicted cluster labels based on provided input features. The labeled dataset consists of bytes transferred, PCA representation of other features and class label. There is no way to verify the accuracy of un-supervised machine learning models. Therefore, a supervised learning model is used to identify the efficiency of trained models. In our methodology, a random forest as a supervised learning model is selected. A detailed discussion of random forest is provided in the next section.

3.4.1 Random Forest

The random forest is a decision tree-based architecture [19]. It is an ensemble learning model that combines the outcomes of multiple classifiers to solve complex problems. Each classifier is a single tree decision tree, and they combine to make a forest. It is referred to as multiple forests consisting of multiple trees connected randomly. In such types of models, slight change in data will have a high impact on trees correlation. Less correlation means less chance of error during model training. The popular result aggregation techniques are bagging that is designed to increase the stability and accuracy of learning model. It also reduces the variance and overfitting of models. The decisive factors are entropy and information gain. We train the random forest model on each of our five labeled datasets. Random forest model inputs are features vectors with the y-labels and hyperparameters. In our case, y-labels are the cluster ids, and features are PCA representations of the other five features. The number of trees, minimum samples for split, minimum samples for the leaf node, maximum depth of tree are the hyperparameter we tune. We apply the gird search CV on random forest hyperparameters; the best values with tuning range are shown in **Table 4**[28].

Table 4. Random Forest hyperparameter name, range, and best values

S. No.	Metric Name	Best Value	Range
1	N_estimators	500	100, 200, … ,1000
2	Min_sample_split	20	10, 20, … ,100
3	Min_samples_leaf	10	10, 20, … ,100
4	Max_depth	25	5, 10, … ,50
5	Bootstramp	False	False, True

4. Results and Discussion

Our tested is built on KOREN virtualized infrastructure, there are ten virtual switches with NetFlow enabled services. NetFlow information is send and store to central location. Similarly, there are ten hosts directly connected with switches. Furthermore, there are hosts agents to collect the information from end devices. Both hosts information and network traffic are stored in Prometheus DB. Prometheus is configured on our lab-PC. After that we configure Grafana environment in another PC. Once both services and running, the Grafana can query Prometheus for metrics and visual them on dashboard.

We trained five instances of random forest algorithms on five labeled datasets. A dataset is evenly distributed, an 80% of data is utilized for training, and the remaining 20% is for testing, to assess the performance of individual models. We considered the k-means dataset as a reference dataset and evaluated all algorithms on the k-means test set. The accuracy is our primary evaluation metric, as shown in **Fig. 12**. The higher accuracy values mean our model accurately classifies the large flows. The DBSCAN would not achieve higher accuracy.

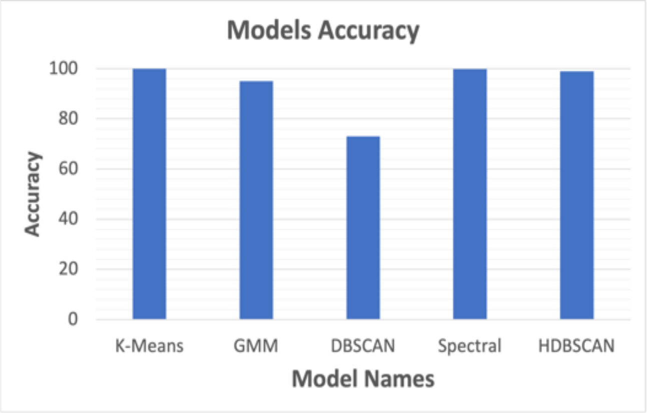

Fig. 12. Accuracy achieved by the different datasets

Comparatively, the GMM method performs well. However, HDBSCAN and Spectral are the best-performed models after the k-means. These results show that the models do not overfit.

5. Conclusion

In this paper, we have implemented a real-time monitoring system to monitor the status of network and collect the network utilization data. Furthermore, we deployed a Grafana based visualization system to understand the current network entities with their status and behavior. The intrinsic benefits of having a real-time monitoring system can be perceived by coupling it with data analysis and machine learning techniques. This manuscript proposed an ensemble learning model for elephant flows classification, and accuracy score approaches that were utilized to select the best classification models. The readiness of network data for computing the algorithms allowed multiple model applications to represent results that could be compared. The system is flexible enough, to allow further enhancements for network optimization with the use of more complex algorithms.

Acknowledgement

This research was supported by the 2020 scientific promotion program funded by Jeju National University.

References

[1] Bogineni, K., et al., "SDN-NFV reference architecture," Verizon, New York City, NY, USA, Verizon Network Infrastructure Planning Version 1, 2016.

[2] Afaq, Muhammad, Shafqat Rehman, and Wang-Cheol Song, "Large flows detection, marking, and mitigation based on netFlow standard in SDN," *Journal of Korea Multimedia Society*, 18(2), 189-198, 2015. Article (CrossRef Link)

[3] Berde, Pankaj, et al., "ONOS: towards an open, distributed SDN OS," in *Proc. of the third workshop on Hot topics in software defined networking*, pp. 1-6, 2014. Article (CrossRef Link)

[4] https://www.koren.kr/kor/index.asp

[5] https://grafana.com/

[6] https://prometheus.io/

[7] https://prometheus.io/docs/guides/node-exporter/

[8] https://github.com/prometheus/pushgateway

[9] Nugraha, Muhammad, et al., "Utilizing OpenFlow and netFlow to detect and mitigate SYN flooding attack," *Journal of Korea Multimedia Society*, 17(8), 988-994, 2014. Article (CrossRef Link)

[10] https://prometheus.io/docs/instrumenting/exporters/

[11] Manning, Christopher, Prabhakar Raghavan, and Hinrich Schütze, "Introduction to information retrieval," *Natural Language Engineering*, 16(1), 100-103, 2010. Article (CrossRef Link)

[12] Fu, Yinlin, et al., "Gaussian mixture model with feature selection: An embedded approach," *Computers & Industrial Engineering*, 152, 107000, 2021. Article (CrossRef Link)

[13] Lee, Changhun, and Chiehyeon Lim, "From technological development to social advance: A review of Industry 4.0 through machine learning," *Technological Forecasting and Social Change*, 167, 120653, 2021. Article (CrossRef Link)

[14] Chen, Yuxin, et al., "Spectral methods for data science: A statistical perspective," *Foundations and Trends® in Machine Learning*, 14(5), 566-806, 2021. Article (CrossRef Link)

[15] Cavalcante Araujo Neto, Antonio, "A Framework for Hierarchical Density-Based Clustering Exploration," 2021. Article (CrossRef Link)

[16] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.kendalltau.html

[17] https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

[18] David Paper, "Introduction to scikit-learn," *Hands-on Scikit-Learn for Machine Learning Applications: Data Science Fundamentals with Python*, pp. 1-35, 2020. Article (CrossRef Link)

[19] Shu, Jun Hua, Jiang Jiang, and Jing Xuan Sun, "Network traffic classification based on deep learning," *Journal of Physics: Conference Series*, Vol. 1087. No. 6, 2018. Article (CrossRef Link)

[20] Greff, Klaus, Sjoerd Van Steenkiste, and Jürgen Schmidhuber, "Neural expectation maximization," *arXiv preprint arXiv:1708.03498*, 2017.

[21] Isolani, Pedro Heleno, et al., "Interactive monitoring, visualization, and configuration of OpenFlow-based SDN," in *Proc. of 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, IEEE, 2015. Article (CrossRef Link)

[22] Van Tu, Nguyen, Jonghwan Hyun, and James Won-Ki Hong, "Towards onos-based sdn monitoring using in-band network telemetry," in *Proc. of 2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, IEEE, 2017. Article (CrossRef Link)

[23] Queiroz, Wander, Miriam AM Capretz, and Mario Dantas, "An approach for SDN traffic monitoring based on big data techniques," *Journal of Network and Computer Applications*, 131, 28-39, 2019. Article (CrossRef Link)

[24] Cheng, Tracy Yingying, and Xiaohua Jia, "Compressive traffic monitoring in hybrid SDN," *IEEE Journal on Selected Areas in Communications*, 36(12), 2731-2743, 2018. Article (CrossRef Link)

[25] Jang, RhongHo, et al., "Rflow+: An sdn-based wlan monitoring and management framework," in *Proc. of IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, IEEE, 2017. Article (CrossRef Link)

[26] Suárez-Varela, José, and Pere Barlet-Ros, "Sbar: Sdn flow-based monitoring and application recognition," in *Proc. of the Symposium on SDN Research*, pp. 1-2, 2018. Article (CrossRef Link)

[27] Ajaeiya, Georgi A., et al., "Flow-based intrusion detection system for SDN," in *Proc. of 2017 IEEE Symposium on Computers and Communications (ISCC)*, IEEE, 2017. Article (CrossRef Link)

[28] https://scikit-learn.org/stable/modules/grid_search.html

[29] Afaq, Muhammad, Shafqat Ur Rehman, and Wang-Cheol Song, "Visualization of elephant flows and qos provisioning in sdn-based networks," in *Proc. of 2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, IEEE, 2015. Article (CrossRef Link)

[30] Gomes, Heitor Murilo, et al., "A survey on ensemble learning for data stream classification," *ACM Computing Surveys (CSUR)*, 50(2), 1-36, 2018. Article (CrossRef Link)

[31] Amin, Rashid, Martin Reisslein, and Nadir Shah, "Hybrid SDN networks: A survey of existing approaches," *IEEE Communications Surveys & Tutorials*, 20(4), 3259-3306, 2018. Article (CrossRef Link)

[32] Kambar, Mina Esmail Zadeh Nojoo, et al., "A survey on mobile malware detection methods using machine learning," in *Proc. of 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, IEEE, 2022. Article (CrossRef Link)

[33] Oh, Chaeyeon, Joonseo Ha, and Heejun Roh, "A Survey on TLS-Encrypted Malware Network Traffic Analysis Applicable to Security Operations Centers," *Applied Sciences*, 12(1), 155, 2022. Article (CrossRef Link)

[34] Zhao, Jingjing, et al., "Network traffic classification for data fusion: A survey," *Information Fusion*, 72, 22-47, 2021. Article (CrossRef Link)

[35] Makarychev, Konstantin, and Liren Shan, "Near-optimal algorithms for explainable k-medians and k-means," in *Proc. of International Conference on Machine Learning*, PMLR, 2021. Article (CrossRef Link)

Waleed Akbar received MS (Computer Science) and BS (Telecommunication and Networking) degrees from COMSATS University Islamabad, Abbottabad campus in 2018 and 2014, respectively. He is currently pursuing a doctorate degree in computer engineering with Jeju National University, South Korea. He joined COMSATS as a research associate in 2014 and was later promoted to lecture after completing of MS degree in 2018. In 2020, he joins the Network Convergence Lab as a Ph.D. scholar. His research interests include SDN, NFV, Intent-Based Networking, Mobile Edge Computing, network configuration and management, server management, network monitoring, and network optimization.

Jose Javier Diaz Rivera in 2005 B.S. degree in Computer Systems Engineering – Instituto Tecnologico de Estudios Superiores de Monterrey, Monterrey, Mexico. 2017 ~ 2019, M.Sc degree in Computer Engineering – Jeju National University, South Korea. 2021 ~ Present, Ph.D. student in Electronic Engineering – Jeju National University, South Korea. Interest includes NFV, SDN, 5G, Anomaly Detection, ML, Blockchain

Talha Ahmed Khan received the BS(CS) degree from FAST- National University of Computer and Emerging Sciences Pakistan and received the MS(CE) degree from Jeju National University, South Korea, in 2019. He is currently pursuing the doctorate degree in computer engineering with Jeju National University, South Korea. His research interests include the SDN, NFV, 5G Mobile Networks, Intent Based Networking, Orchestration, and scaling of VNF(s), Mobile Edge Computing and VNF development.

Muhammad Afaq received a Ph.D. degree in Computer Engineering from Jeju National University, MS degree in Electrical Engineering with emphasis on Telecom from Blekinge Institute of Technology, Sweden, and BS degree in Electrical Engineering from the University of Eng. and Technology, Peshawar, Pakistan in 2017, 2010, and 2007 respectively. He is currently working as a Postdoctoral Researcher at Network Convergence Lab, Jeju National University. He also worked as an Assistant Professor in the Department of Computer Science and IT at the Sarhad University of Science and IT, Pakistan. Before starting his Ph.D., he worked as a Research Associate in the Faculty of Computer Science and Engineering at GIK Institute of Engineering Sciences and Technology, Pakistan, and as a Lecturer in the Department of Electrical Engineering at the City University of Science and IT. His research interests are cloud computing, software-defined networking, network function virtualization, wireless networks, and protocols, machine learning, and data science.

Wang-Cheol Song has worked at the Computer Engineering Department, Jeju National University, South Korea, Since 1996. He received B.S. degree in Food Engineering and Electronics from Yonsei University, Seoul, Korea, in 1986 and 1989, respectively. And He received M.S. and Ph.D. in Electronics studies from Yonsei University, Seoul, Korea, in 1991 and 1995, respectively. His research interests include VANETs and MANETs, SDN/NFV, Intent-based networking, and network management.