IBN@Cloud: An Intent-based Cloud and Overlay Network Orchestration System

Mir Muhammad Suleman Sarwar, Afaq Muhammad, and Wang-Cheol Song

***Abstract*—This paper presents an intent-based networking (IBN) system for the orchestration of OpenStack-based clouds and overlay networks between multiple clouds. Clouds need to communicate with other clouds for various reasons such as reducing latency and overcoming single points of failure. An overlay network provides connectivity between multiple Clouds for communication. Moreover, there can be several paths of communication between a source and a destination cloud in the overlay network. A machine learning model can be used to proactively select the best path for efficient network performance. Communication between the source and destination can then be established over the selected path. Communication in such type of a scenario requires complex networking configurations. IBN provides a closed-loop and Intelligent system for cloud to cloud communication. To this end, IBN abstracts complex networking and cloud configurations by receiving an intent from a user, translating the intent, generating complex configurations for the intent, and deploying the configurations, thereby assuring the intent. Therefore, the IBN that is presented here has three major features: (1) It can deploy an OpenStack cloud at a target machine, (2) it can deploy GENEVE tunnels between different clouds that form an overlay network, and (3) it can then leverage the advantages of machine learning to find the best path for communication between any two clouds. As machine learning is an essential component of the intelligent IBN system, two linear and three non-linear models were tested. RNN, LSTM, and GRU models were employed for non-linear modeling. Linear regression and SVR models were employed for linear modeling. Overall all the non-linear models outperformed the linear model with an 81% R^2 score, exhibiting similar performance. Linear models also showed similar performance but with lower accuracy. The testbed contains an overlay network of 11 GENEVE tunnels between 7 OpenStack-based clouds deployed in Malaysia, Korea, Pakistan, and Cambodia at TEIN.**

***Index Terms*—GENEVE, IBN, linear models, non-linear models, OpenStack, overlay network, TEIN.**

I. Introduction

INTERNET engineering task force (IETF) defines an intent-based network (IBN) as "a network that can be managed using intent" [1], [2]. An intent is "a set of operational goals (that a network should meet) and outcomes (that a network is supposed to deliver), defined in a declarative manner without specifying how to achieve or implement them" [1], [2].

IBN has the ability to continuously monitor the network and react to changing network conditions, such as path selection in an overlay network for communication between clouds. Cloud computing offers various services [3]. More and more individuals and organizations are turning to Multi-cloud solutions as a single cloud is often unable to meet their diverse needs [4]. Clouds may need to share their resources with other clouds for a variety of reasons, such as minimizing single points of failure, improving reliability, and ensuring high availability [5].

Tunneling [6] can provide the mechanism of communication between clouds as tunneling enables access to restricted networks and establishes an isolated and secure communication channel [7]. In networking, when one node does not have direct access to another node, tunnels can be established at a demilitarized zone (DMZ) between the two nodes to enable accessibility. In some cases, tunnels are set up solely to facilitate communication over an isolated and secure channel, even if the source and destination nodes have direct network accessibility [7]. To enable communication between OpenStack-based clouds, a GENEVE tunnel is implemented, and all GENEVE tunnels between the clouds collectively form an overlay network. The topology of this network can be displayed and managed through an SDN-Controller such as RYU.

Network modeling plays a critical role in the development of efficient solutions for network operation and optimization, especially with regard to the emerging field of self-driving networks [8], [9]. This model determines the most optimal path for network traffic between a given source and destination. For this purpose monitoring modules are deployed for collecting data of the overlay network topology. This data is used to train a machine learning model. Once the model has identified the best path, the SDN-Controller adds flow rules to route network traffic along that path.

To this end, in this paper, we present IBN that deploys OpenStack-based clouds at different locations and deploys GENEVE tunnels between the OpeStack-based clouds. Together all the tunnels form an overlay network. IBN receives an intent for a service request from a source to a destination cloud. IBN uses the ML model to proactively select a path that has minimum round-trip time (RTT). IBN then gives input to SDN-controller for communicating over the selected path for the given source and destination in the intent. IBN, therefore, acts as an intent-based closed-loop and Intelligent way of resource sharing through GENEVE tunneling between

Manuscript received July 7, 2023; revised October 13, 2023; approved for publication by Pack, Sangheon, Division 3 Editor, October 30, 2023.

This work was supported by the National Research Foundation(NRF), Korea, under project BK21 FOUR. This research was supported by Brain Pool program funded by the Ministry of Science and ICT through the National Research Foundation of Korea (2019H1D3A1A01102980).

The authors are with Department of Computer Engineering, Jeju National University, Jeju-si, Republic of Korea, email: {mirmohdsuleman, afaq24}@gmail.com, philo@jejunu.ac.kr.

W.-C. Song is the corresponding author.

Digital Object Identifier: 10.23919/JCN.2023.000051

1229-2370/23/$10.00 © 2023 KICS

multiple OpenStack-based clouds.

The testbed contains an IBN application, 7 OpenStack-based clouds, and an overlay network with 11 GENEVE tunnels deployed by IBN. The test-bed also contains a monitoring system for data collection that is used for machine learning model training and inference. Moreover, the test-bed also contains an RYU SDN-Controller that ensures communication over the path provided by IBN. The OpenStack-based clouds and overlay network are deployed in four partner countries, namely Pakistan, Cambodia, Malaysia, and South Korea at trans-Eurasia information network (TEIN) [10]. TEIN is a network between the Asia Pacific and Europe for research and education communities. TEIN is an EU co-funded Asia@Connect project and comprises 21 Asian partners.

Section II contains related work. Section III describes the architecture of the system. Section IV contains explanation of the test bed design and all its components. Section V presents machine learning. Section VI contains the results and discussion. Section VII concludes the paper.

II. Related Work

In an IBN, a user provides a networking requirement in an abstract manner that does not contain any technical configurations. An abstract networking requirement is called an intent. The IBN application itself decides the necessary configurations for satisfying an intent. The IBN application executes commands on the underlying network for deploying the decided configurations. This process is called intent assurance. An intent can be any abstract network requirement, e.g., "deploy tunnel from node A to node B." It is then the job of the IBN application to translate the intent into network configurations based on some policy [11].

A study proposed network slicing in 5G networks using generative adversarial neural network (GAN) and IBN [12]. Another study presented a generic intent-based networking platform for E2E network slice orchestration and lifecycle management using IBN and graph neural network (GNN) [13]. Zeydan, Engin, and Yekta Turk, in a study, provide a comprehensive survey on IBN [14].

IBN can facilitate cloud-to-cloud communication that is required for various reasons. One of the reasons is to avoid a single point of failure. In addition to the risk of a single point of failure, a cloud that is located far from a client can result in high latency when accessing the cloud. Cloud, therefore, needs to share its resources with another cloud near the client's location [15].

In a multi-cloud scenario, application developers face the challenge of managing a distributed application across multiple clouds. This includes ensuring that the application remains operational and performs optimally, while also considering the potential migration of services from one cloud to another. To address this challenge, developers need to have a comprehensive understanding of the cloud infrastructure, network topology, and application architecture. They also need to consider factors such as load balancing, data replication, and disaster recovery to ensure that the application remains available and performs well in a multi-cloud environment [16].

clouds, therefore, need an overlay network in the form of tunnels for communication and sharing of resources. [7].

Tunneling, therefore, provides the communication mechanism between two or more clouds. There can be different types of tunneling, such as HTTP tunneling [17], GRE tunneling [18], VXLAN [19] tunneling, and GENEVE tunneling [20]. If we examine the details of VXLAN we will see that it was introduced as a MAC-in-IP within an IP/UDP transport. While this feature is advantageous for bridging it is not important for routing purposes and could be utilized for better payload byte usage. Additionally, MAC to IP bindings requires signalling which calls for information exchange in the control plane or flood-based learning. The GENEVE [20] tunneling protocol has a flexible option header format that allows for the definition of the length, fields, and content depending on the instruction set provided by the tunnel endpoint (TEP) node that performs the encapsulation. Although some of the fields in GENEVE tunneling are simple and static, such as bridging or routing, other fields and formats used for telemetry or security are highly variable, for hop-by-hop independence . Since 2020 GENEVE has been therefore adopted as a standard tunnel protocol [21].

Once the overlay network is deployed between clouds using tunnels there can be several paths of communication between a source and destination. The best path can be selected proactively using Machine Learning. The goal of network traffic prediction is to forecast future network traffic based on historical data. This proactive approach is valuable for network management and planning purposes. Network prediction has become increasingly significant and is receiving greater attention, particularly for medium to large network providers [22]. Enhancing the accuracy of network prediction enables network providers to optimize resources more effectively and deliver higher service quality to their customers [23]. The family of recurrent neural network (RNN) approaches specializes in modeling time series data, with the objective of predicting future time series based on past information, even when there are long time lags of unknown durations. RNN encompasses various network architectures, including simple RNN, long short-term memory (LSTM), and gated recurrent unit (GRU) [24]. RNN is an advanced model derived from the traditional feed-forward neural network (FFN). It incorporates a self-recurrent loop that enables the flow of information from one time step to another, making it particularly effective for tasks involving time series and sequence modeling. LSTM, a type of RNN, was specifically designed to address the challenge of vanishing and exploding gradients encountered by traditional RNNs. GRU, on the other hand, was introduced as an alternative to LSTM with the goal of reducing computational complexity while achieving similar performance [24]. According to [25], it was argued that a nonlinear approach based on traffic prediction is the most suitable. Recurrent neural networks (RNNs) are recognized as highly effective models that exhibit strong accuracy in forecasting time series data [26], [27].

In summary, clouds offers various services [3]. This study presents IBN for deploying clouds to host services. These offered services have some essential requirements such as

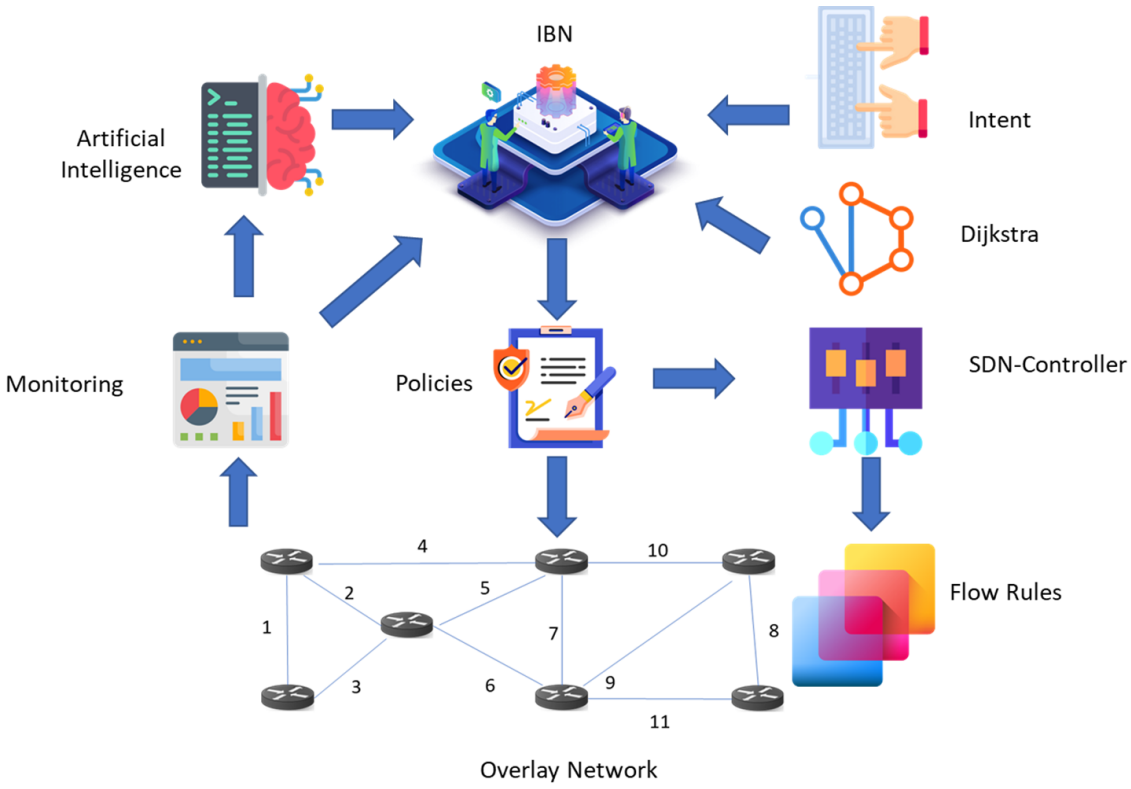


Fig. 1. System architecture.

high availability, reliability, performance, replication, and data consistency [4]. These essential requirements need communication between Multiple clouds [15], [16], [28]. Communication between Mutiple clouds can be provided by L2 tunnels [6], [7], [17]–[21].The IBN can then deploy tunnels between clouds. Collectively these tunnels form an overlay network. Communication on the overlay network can take several paths. IBN leverages the advantages of machine learning models to forecast RTT latency for each tunnel in the overlay network [8], [9], [22]–[27].

III. System Architecture

Fig. 1 shows the architecture of the system. IBN acts as the central component by receiving input from all other components and decides the output. IBN deploys clouds and overlay network links such as Layer 2 (L2) GENEVE tunnels between multiple clouds. Together all the tunnels form an overlay network. A total of 7 clouds and 11 GENEVE tunnels are deployed in our testbed using IBN. The overlay network is used by clouds for cloud-to-cloud communication. There can be several paths between a source cloud to a destination cloud. For optimal network performance, communication must be done proactively by selecting the best path. The best path has the minimum forecasted RTT latency in comparison to other paths between the same source and destination clouds. For this purpose monitoring tools collect network data such as the RTT latency of each link. The obtained data is then used by a machine learning model for training. Once the model is trained it can forecast the RTT latency of each link. For a given source and destination in intent, the IBN gets the forecasted data from the machine learning model. The IBN feeds the forecasted data to the Dijkstra algorithm as weights. The Dijkstra algorithm returns the shortest path based on these weights to the IBN. Thereby, the Dijkstra algorithm finds the best path. This path has the minimum RTT latency forecasted. IBN stores the path for a given intent. IBN then directs SDN-Controller to ensure communication over the best path. SDN-Controller then deploys flow rules for communication over the best path.

Once the best path is decided the IP addresses of the nodes and port numbers of tunnels in the decided path are used to formulate flow rules. A flow rule decides to forward a packet to a specific port based on the information of source and destination addresses in the packet in a virtual switch. A bridge of an Openstack OVS acts similarly to a virtual switch and contains ports. A GENEVE tunnel is a link and is connected to the bridge by a port. A bridge can be connected to multiple GENEVE tunnels by different ports. An SDN-Controller adds the decided flow rules on the bridge of each OVS to enable communication over the best path. Interaction between different components and obtaining the best path is done by an individually executing component of IBN called off-platform application (OPA). Section IV explains in detail the experimental setup and working of components.

IV. Testbed Design

IBN is deployed at a centralized location where it has a global view of all system components as well as the overlay

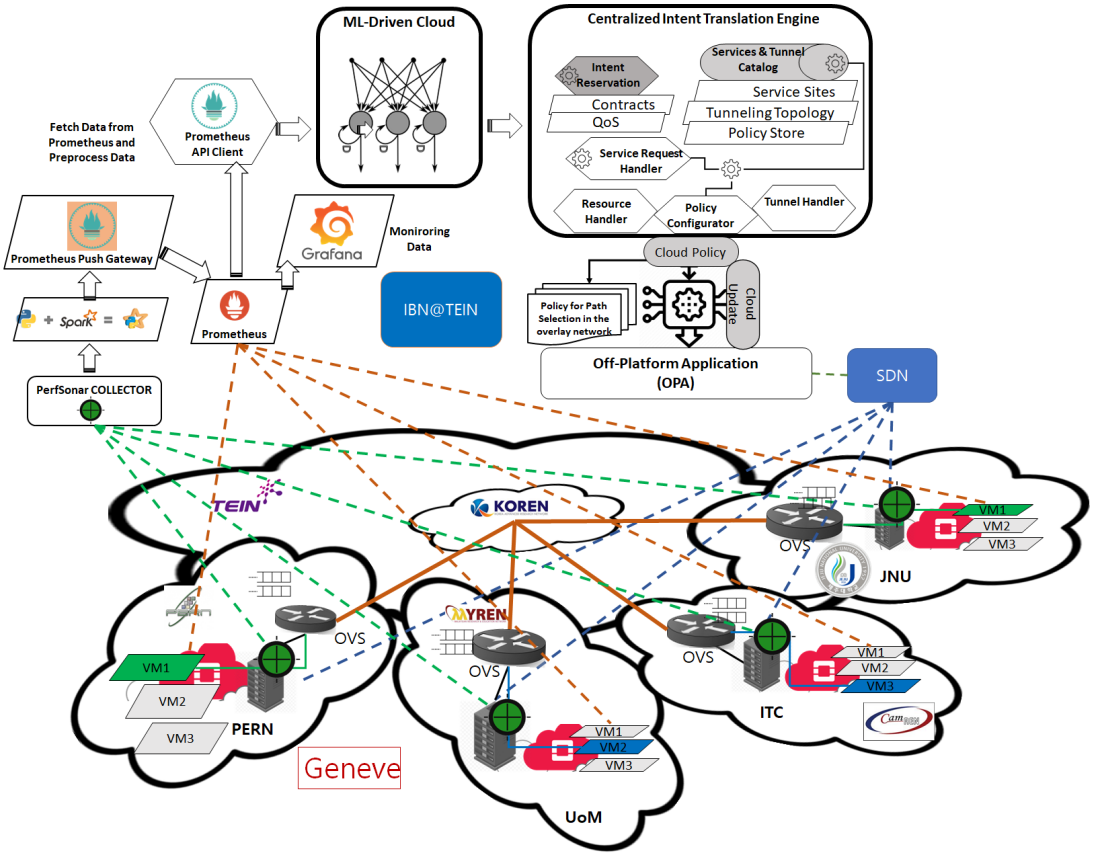


Fig. 2. Testbed design of IBN orchestrating clouds and overlay network between clouds at TEIN.

TABLE I
LIST OF CLOUD NODES DEPLOYED AT TEIN.

S. No.	Site	Site Name	Country
1	JNU	Jeju National University	Korea
2	UoM	University of Malaya	Malaysia
3	PERN	Pakistan Education & Research Network	Pakistan
4	KOREN-NOC	Korea Research & Education Network NOC	Korea
5	ITC	Institute of Technology Cambodia	Cambodia
6	SUIT	Sarhad University of Information Technology	Pakistan
7	NUST	National University of Sciences & Technology	Pakistan

network. IBN is used to deploy 7 OpenStack [29] clouds in four partner countries, namely Pakistan, Cambodia, Malaysia, and South Korea at TEIN, with 11 GENEVE tunnels, as shown in Fig. 2. In Pakistan we deployed OpenStack cloud nodes at three sites namely the Sarhad University of Science and Information Technology (SUIT), the National University of Science and Technology (NUST), and Pakistan Education and Research Network (PERN). In Korea, we deployed OpenStack cloud nodes at two sites namely Korea Research and Education Network (KOREN) – Network Operations Center (NOC), and Network Convergence Lab Jeju National University (NCL-JNU). In Cambodia and Malaysia, we deployed one cloud node at the Institute of Technology Cambodia (ITC) and the other at the University of Malaya (UoM). Table I shows a list of all sites and their Institute name in each country. Monitoring tools are also deployed at the centralized location so that it has a global view of the overlay network. Monitoring tools such as PerfSonar, PerfSonar-Collector, Prometheus-push gateway, Prometheus, and Grafana are deployed for network performance monitoring. Fig. 2 shows the GENEVE tunnel between all sites also deployed by IBN. PerfSonar is then deployed at each site to test the performance of the link. For this purpose, PerfSonar Collector is developed that executes any number of tests and exports the results to Prometheus a time series database (TSDB) through several intermediate modules. Monitoring Data is visualized by deploying Grafana. The obtained data is used for training of ML model for finding the best path. RYU SDN-Controller is also deployed at the same centralized location where IBN and monitoring tools are deployed. SDN-Controller can visualize the topology and deploy flows for a path based on input from IBN for given intent. Fig. 2 shows the experimental setup of IBN

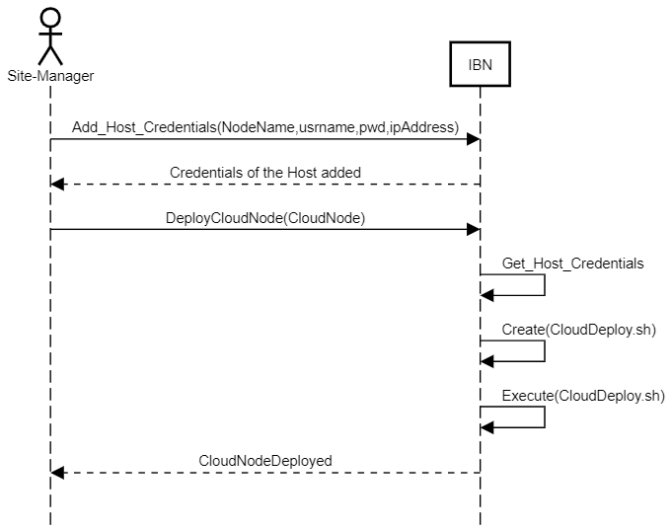


Fig. 3. Site-manager deploying cloud node at his site.

orchestrating clouds and overlay network between clouds. This section describes each of these components in detail.

A. Intent-based Networking (IBN)

The IBN application contains complete system information and has a global view of the current system status. The IBN application has a Catalog that contains information on nodes, Links between nodes, and services at each node. IBN is used for adding new services. The IBN also shows all tunnels and tunneling details, such as bridges, ports, and remote Ips. The IBN system has a simplified front-end web-based interface for users to request the automated orchestration of services. IBN uses the high-level service context/business goals provided by the user as intents and translates them into low-level system configurations. IBN is used to deploy cloud nodes at sites as shown in Fig. 3 and Algorithm 1. IBN is also used to deploy an overlay network such as GENEVE tunnels between cloud nodes as shown in Fig. 4 and Algorithm 2. In addition, the IBN application will ensure the provision of a requested service to a user. A user from any node from any partner country can request service at any other node at any other partner country. IBN will learn the best path for this request. Initially, the intent is pending but when the decided path is implemented and the service is acquired from source to destination nodes, the intent is "Assured" as shown in Fig. 5 and Algorithm 3. The IBN shows all pending and assured intents. In simple words, user intent is a requirement for which certain complex network configurations are needed. This section further elaborates on IBN users and the mechanism of intent translation.

1) IBN roles: IBN consists of three distinct user types: "IBN-Manager", "Site-Manager", and "Service-Provider". An IBN-Manager has a global view of the system. IBN-Manager can create users such as Site-Manager or Service-Provider. Users can then interact with IBN and carry out their activities.

2) Intent translation: IBN is used for network automation that can simplify the configuration process and reduce the risk of human error. By using intent-based commands, a user can specify the desired outcome without needing to know the specific details of the network configuration. The IBN can then automatically generate the necessary configuration commands to achieve the desired outcome. For example, the IBN-Manager can use IBN to deploy a tunnel, a Site-Manager can use IBN to deploy a cloud node at his site, or a Service-Provider can submit a request in the form of intent to acquire a service from another node. IBN generates and executes configurations for each of these intents.

A Site-Manager creates a site in IBN and deploys a cloud node at that site using IBN as shown in Fig. 3 and Algorithm 1. For this purpose, a Site-Manager first adds the credentials of the host on which the cloud node needs to be deployed. The Site-Manager then selects the cloud node to deploy it at the site. IBN extracts the credentials of the host on which the cloud node needs to be deployed. IBN creates a bash file with the name "CloudDeploy.sh". IBN then adds commands in the "CloudDeploy.sh" file that are related to accessing the host by SSH using host credentials. Further, IBN adds commands in the "CloudDeploy.sh" file for deploying cloud node and starting a Control node. IBN then executes the file "CloudDeploy.sh". Executing the file gets access to the host system by SSH and deploys cloud and starts a Control node. A Site-Manager can also manage his site using IBN such as

Algorithm 1 Deploy cloud node

Inputs:

- IPs: IP Address of the physical Host where the cloud needs to be deployed.
- Username: Username of the physical Host where the cloud needs to be deployed.
- Password: Password of the physical Host where the cloud needs to be deployed

Outputs:

- OpenStack cloud deployed at the physical host of IP address.

Step 1: Add host credentials

Use server-side script such as PHP for adding node name IP Address, username and password to the IBN database of the Host system where a cloud node needs to be deployed.

Step 2: Get host credentials

Use server-side script such as PHP to get node name, IP address, username and password from the IBN database of the host system where a cloud node needs to be deployed.

Step 3: Create file CloudDeploy.sh

Use file handling of a server-side script such as PHP to create a file "CloudDeploy.sh". Write Linux commands in the file that:

o Access the host by SSH using IP, username, and password.

o Deploy Microstack-based OpenStack cloud.

o Start a control node.

Step 4:Execute file CloudDeploy.sh

o Execute the "CloudDeploy.sh".

o Cloud is deployed and a control node of the cloud is started by accessing the host system.

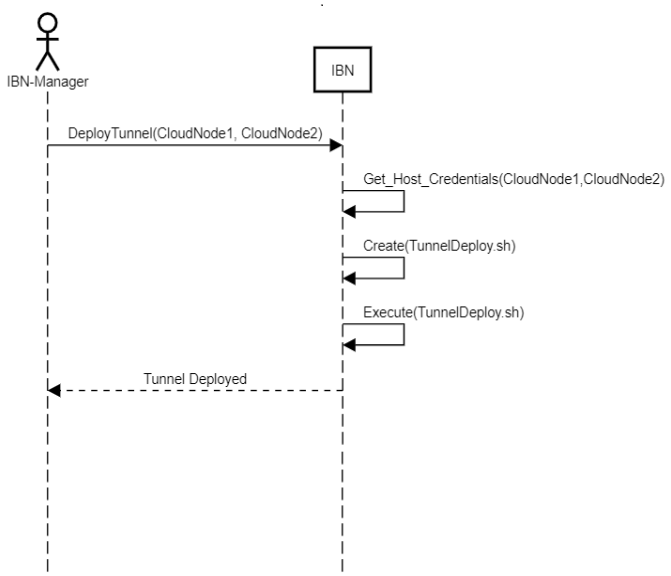


Fig. 4. IBN-Manager deploying a tunnel.

Algorithm 2 Deploy tunnel

Inputs:

- TEP-IPs: IP addresses of both tunneling endpoints.
- TEP-usernames: Usernames of the both tunneling end points.
- TEP-passwords: Passwords of the both tunneling end points.

Outputs:

- GENEVE tunnel deployed between two Openstack clouds.

Step 1: Get host credentials

Use server-side script such as PHP to get IP Addresses, usernames and passwords of the both TEPs from the IBN database where a tunnel node needs to be deployed.

Step 2: Create file TunnelDeploy.sh

Use file handling of a server-side script such as PHP to create a file "TunnelDeploy.sh". Write Linux commands in the file that:

o Access the host1 (TEP1) by SSH using IP, username, and password.

o Deploy GENEVE tunnel to the other TEP cloud and exit.

o Access the host2 (TEP2) by SSH using IP, username, and password.

o Deploy GENEVE tunnel to the first TEP cloud and exit.

Step 3: Execute file TunnelDeploy.sh

o Execute the TunnelDeploy.sh

o Deploy the tunnel by accessing both the TEPs.

launching Instances. There can be more than one site in a Country, such as Korea having two sites, i.e., one at NCLab-JNU and the other at KOREN-NOC-Seoul. There are a total of 7 cloud nodes in the project, and for each cloud node system, there is a corresponding Linux OS user account i.e., the host system account.

Algorithm 3 Auto path calculation and implementation

Inputs:

- Source-node-ID: The ID of the source node in intent.
- Destination-node-ID: The ID of the destination node in intent.
- Destination-instance-ID: The ID of the destination instance to which data or files will be transfered.
- Quality of service (QoS) parameter: The attribute of the requested service e.g., for 720p or HD for a video service
- Data: Current network and node performance data which is input to ML model.

Outputs:

- Optimal path: nodes and edges in a path.
- Flow rules for optimal path: SDN implementing flow rules for communication over optimal path.

Step 1:Submit service request

o Service-provider requests a service by providing a service name, QoS parameter, source-node-ID and destination-node-ID.

Step 2:Get inputs

o OPA gets current network performance data such as RTT of each link from monitoring tools.

o OPA gets source-node-ID and destination-node-ID from a pending intent

Step 3:Obtain link prediction

o The current RTT of each link is given as input to the ML model.

o Obtain predicted RTT values for each link.

Step 4:Obtain path

o The predicted RTT values are given as input to the Dijkstra along with a source node and a destination node.

o Obtain source to destination best path in the form of nodes and edges.

Step 5:Calculate E2E path for intent

o Get destination-instance-ID from intent.

o Decide source-instance-ID based on QoS parameter in intent.

o Calculate e2e path from source instance on source node to destination instance on destination node.

Step 6:SDN implementing optimal path

o OPA provides the E2E path to the SDN controller.

o The E2E Flow rules are calculated for the best path.

o The flow rules are deployed on OVSs of each node.

o Communication is established over the best Path for a given intent and resultant path is stored in IBN.

o Status of the intent is updated from pending to "assured"

IBN manager can then deploy tunnels between cloud nodes that were deployed by Site-Managers. As shown in Fig. 4 and Algorithm 2, two cloud nodes are selected by IBN-Manager to deploy a tunnel between them using IBN. Each cloud node is a TEP. The host credentials of each TEP are extracted from the IBN. IBN creates a bash file with the name "TunnelDeploy.sh". IBN adds commands in the "TunnelDeploy.sh" that are related

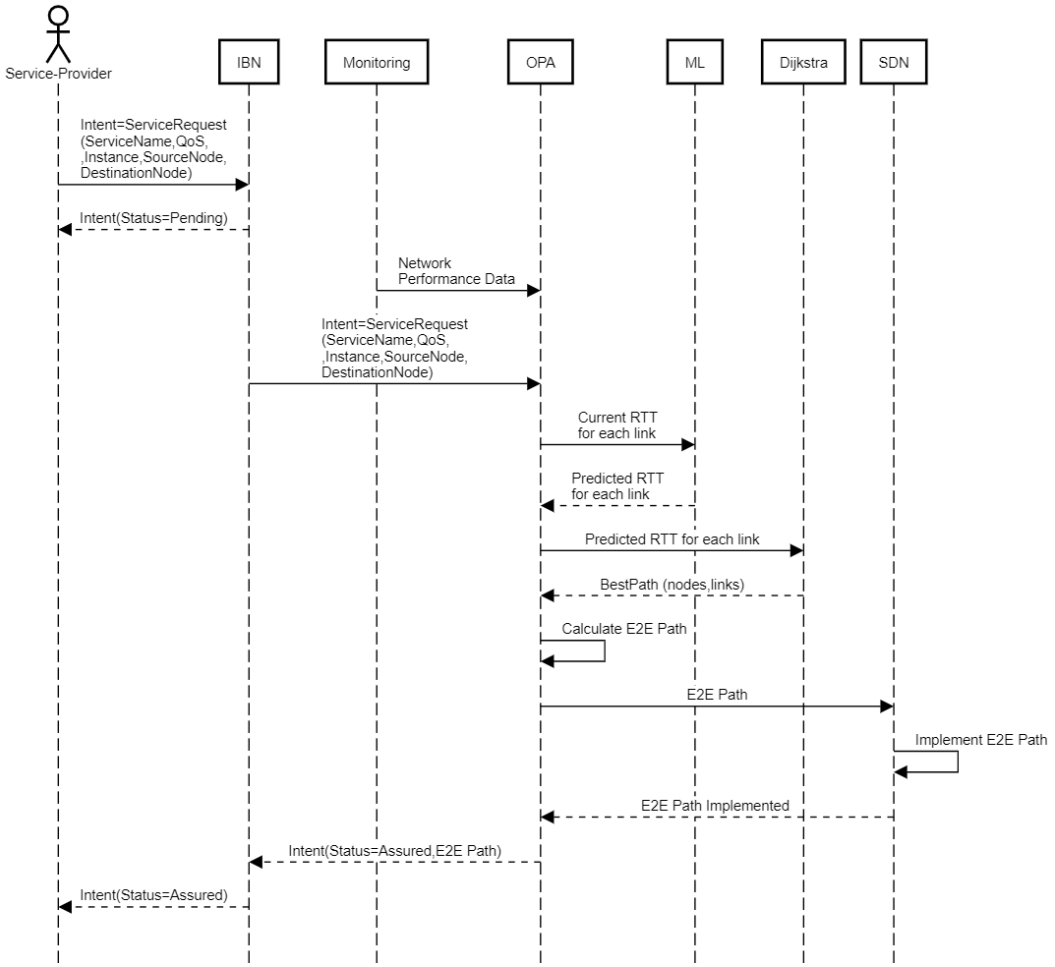


Fig. 5. Auto path calculation and implementation.

to accessing the first cloud node by SSH using host credentials. IBN then adds commands in the "TunnelDeploy.sh" that are related to deploying the tunnel from the first cloud node to the second cloud node and exit. Since tunnels are uni-directional and for bi-directional communication the tunnel needs to be deployed from both directions. Therefore, IBN adds commands in the "TunnelDeploy.sh" that are related to accessing the second cloud node by SSH using host credentials. Finally, IBN adds commands in the "TunnelDeploy.sh" that are related to deploying the tunnel from the second cloud node to the first cloud node and exit. IBN then executes the file. Executing the file gets access to both cloud nodes and deploys the tunnel.

A Service-Provider hosts services at Instances of a cloud site that is owned by a Site-Manager. For this purpose, a Service-Provider uses the IBN to request a Site-Manager for the provision of resources. Service-Provider can request a service from another cloud node to his cloud node, thereby submitting an intent. While posting an intent, the Service-Provider specifies the service-name, QoS parameter, source cloud node, target cloud node, and Instance that acquires a service. Initially, when a Service-Provider submits an intent, it is in a pending state.

Following this, an OPA gets the current network performance data such as RTT from the monitoring tools for each link. The OPA also gets the pending intent. The OPA obtains the predicted RTT values from the machine learning model for each link in the overlay network based on the current data. OPA then uses the predicted RTT values as weights with the Dijkstra algorithm. For the given source and destination in the intent, the Dijkstra algorithm returns the shortest path based on the weights. Since the weights given as input to the Dijkstra algorithm were the predicted RTT values, the shortest path returned by the Dijkstra algorithm represents the best path with a minimum predicted RTT. The path returned by Dijkstra is a node-to-node path, where each node represents a cloud node. The end-to-end (E2E) path from a source Instance on a cloud node to a destination Instance on another cloud node is decided by OPA from the intent. The destination Instance in which the Service-Provider wants to acquire a service is given in the intent. The OPA decides the source Instance based on the QoS parameter given in the intent. The Instance on the Source cloud node that hosts a service with the QoS desired by the Service-Provider is decided as the source Instance.

Once the E2E path is calculated the OPA provides the path to the SDN Controller. The E2E flow rules are calculated for the best path. A flow rule contains information about the path of communication that is based on the information of source and destination addresses in the packets. The flow rules are

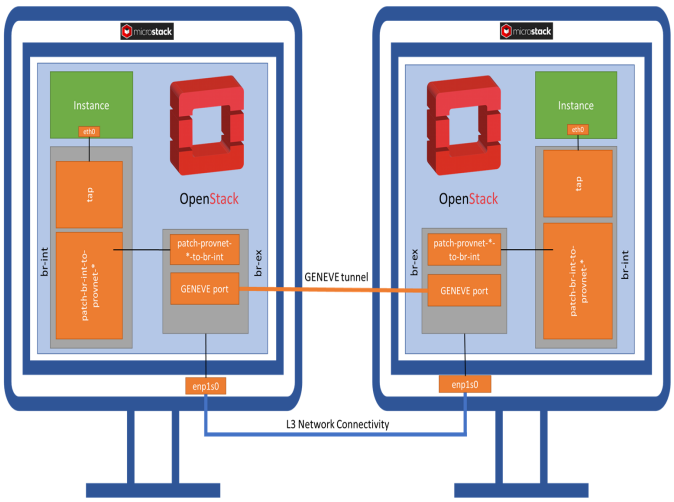


Fig. 6. GENEVE tunneling between OpenStack clouds [30].

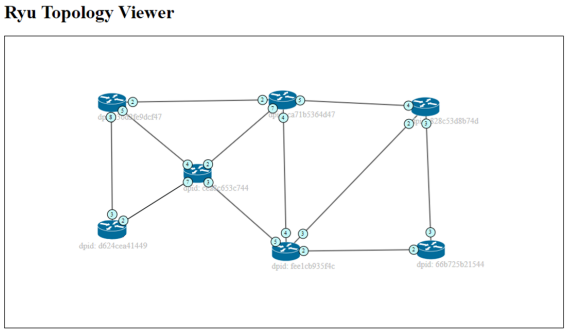


Fig. 7. Overlay network topology between OpenStack cloud OVSs at RYU topology viewer.

deployed on each OVS of each cloud node. Communication is established over the best path for a given intent. The resultant path is also stored in IBN and the status of the intent is updated to "Assured". The complete process of calculating and deploying the best path of communication for a given intent of a service request is given by Fig. 5 and Algorithm 3.

B. An Overlay Network between Openstack-based Clouds at TEIN

1) OpenStack-based clouds: OpenStack clouds are deployed using microstack [31]. Instances are launched using a custom Ubuntu image because the default image Cirros cannot install any package and is merely used for testing. Each of these clouds serves as a Control as well as a compute node, i.e., single node cloud deployment.

2) GENEVE tunnel at OpenStack OVS: As depicted in Fig. 6 [30], every Instance is equipped with an Ethernet interface (e.g., eth0), which facilitates connectivity to the Instance. An integration bridge, br-int, is present in the OpenStack Compute node to connect the Compute node with the Control node. To provide external connectivity via the physical Ethernet interface enps01 of the host machine, the OpenStack Control node utilizes an external bridge called br-ex [32]. As we have deployed a single-node cloud that acts as both the Control and Compute node, both the integration bridge, br-int, and external bridge, br-ex, are present on the same node. Each cloud Instance establishes a connection with br-int through tap-ports, which implies that br-int is equipped with a tap port for every Instance. In addition to tap-ports a patch-br-int-to-provnet-* port of bridge br-int connects to port patch-provnet-*-to-br-int of bridge br-ex, thereby connecting the two bridges. To enable communication between Instances of two separate clouds, a Geneve tunnel is deployed on bridge br-ex As depicted in Fig. 6, there are two single-node OpenStack clouds, each of which is equipped with bridges br-int and br-ex. From Fig. 6, it is evident that the tunnel is deployed on bridge br-ex. In a single-node cloud scenario, deploying the tunnel on either bridge br-ex or bridge br-int will not make any significant difference. This is because the single node serves as both Control and compute nodes, and both of the bridges are in the same node. However, in a multi-node cloud scenario, there is a significant difference between deploying the tunnel on bridge br-ex and bridge br-int. The Control node has bridge br-ex, whereas each compute node has its bridge br-int. Therefore, deploying the tunnel on bridge br-ex will enable communication between Instances from all compute nodes, while deploying the tunnel on bridge br-int will only facilitate communication between Instances within the specific compute node. L2 tunnels such as VXLAN or GENEVE are unidirectional. This means for establishing bidirectional communication; the tunnel must be deployed from both directions. The supported tunnel type can be found in the ML2 plugin document of a cloud. Kernel 3.18 or greater and OVS version 2.4 or greater of microstack-based OpenStack cloud support GENEVE tunnel but do not support VXLAN tunnel [33]. For a tunnel to be established between two clouds, both sides must support and use the same type of tunnel. If different types are used, the tunnel may be deployed without error, but Instances from one cloud will not be able to access instances from the other cloud.

3) SDN-Controller: Communication between a source and a destination cloud can take several paths. An SDN controller such as RYU [34] connected to each OVS of the cloud can control traffic over a selected path. The chosen path is provided by the ML model. Fig. 7 shows the topology viewer of RYU SDN-Controller between 7 nodes and 11 edges. This topology was deployed at TEIN.

C. Dataset Generation by Monitoring Network Performance

Network Performance Monitoring involves the gathering of data metrics such as speed, bitrate, congestion window (Cnwnds), jitter, re-transmissions (Retrans), delay, and RTT for each tunnel in the overlay network. PerfSonar [35] is a commonly used tool for network performance testing. To execute PerfSonar tests between a source and destination, a custom-coded PerfSonarCollector is employed. For testing, iperf3udp, nuttcp, and owping tests were conducted because these tests cover maximum network performance metrics. From the iperf3udp test, we collected metrics such as bitrate and jitter. From a nuttcp test speed, Cnwnds, Retrans, and RTT are collected. From an owping test, delay metric is collected. To perform the testing, three separate PerfSonarCollectors

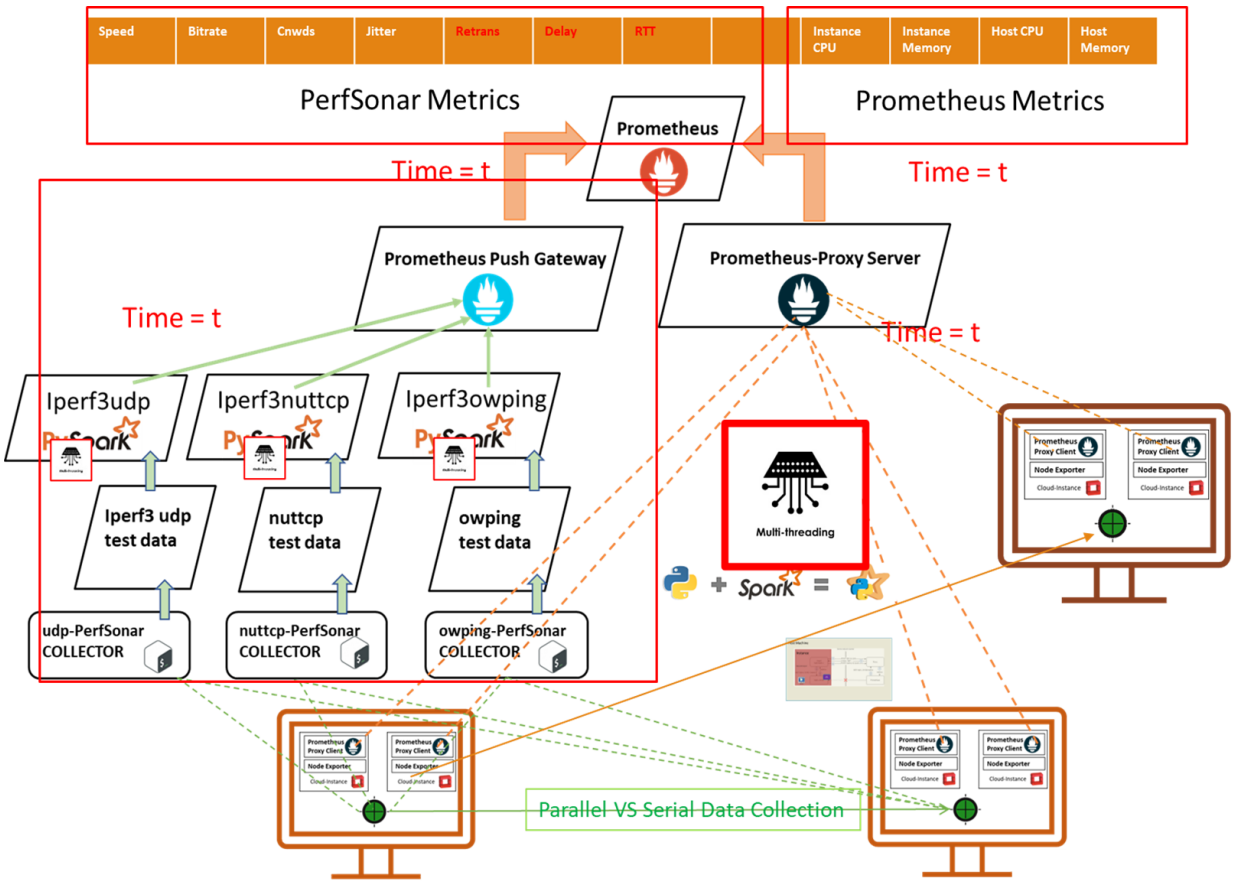


Fig. 8. Network and cloud performance monitoring.

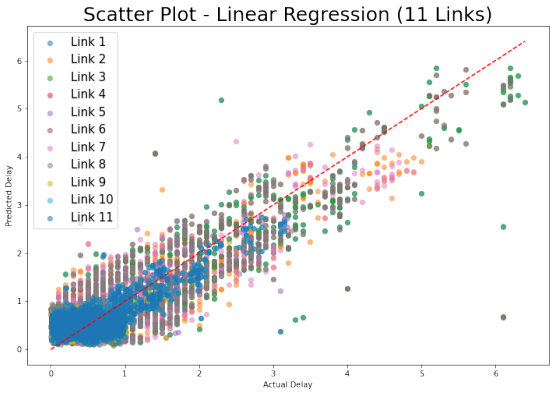
TABLE II
DATASET DESCRIPTION.

Sample	Timestamp	Link1-RTT (ms)	Link2-RTT (ms)	...	...	Link10-RTT (ms)	Link11-RTT (ms)
1	1634758561	93	182	...	...	98	98
2	1634758571	94	181	...	...	97	97
...		...	...	...	...	...	...
...		...	...	...	...	...	...
10893	163486749	92	180	...	...	98	98

were developed and named udp-PerfSonarCollector, nuttcp-PerfSonarCollector, and owping-PerfSonarCollector.

Each of these collectors was executed in parallel with its respective test. After executing the tests, each of the three PerfSonarCollectors saves its output to a separate CSV file. Specifically, the udp-PerfSonarCollector saves its output to iperf3udp.csv, the nuttcp-PerfSonarCollector saves its output to nuttcp.csv, and the owping-PerfSonarCollector saves its output to owping.csv. Whenever a new record is added to each of the CSV files, it is tailed with the IP address and a port number of the source of the test. This data is sent to Prometheus push gateway [36] by PySpark [37] from the IP and port number. It must be noted that three other PySpark components execute for the three different PerfSonar tests, iperf3udp, nuttcp, and owping. Running PySpark in multithreading is a good way to handle multiple tasks simultaneously. Each thread is responsible for collecting monitoring data of a specific source and destination pair of a link, for all three tests (iperf3udp, nuttcp, and owping) using their respective PerfSonarCollectors. This ensures that all data is collected at the same time stamp for comparison purposes. Once the data is collected, each thread sends the data to the Prometheus push gateway. The Prometheus push gateway then pushes the data to Prometheus. As a result, the network performance data of each source and destination of every link is received at the same time stamp. Fig. 8 illustrates the network performance monitoring data obtained at Prometheus at the same time stamp. The data obtained at Prometheus has metrics such as speed, bitrate, Cnwnds, jitter, Retrans, delay, and RTT. All the features are obtained at the same time stamp at an interval of 10 seconds. At this point, it is important to note that we need all features at the same timestamp to compare them. For example, at a particular time stamp t, we require traffic metrics of all 11 links for comparison. Data is fetched from Prometheus using

(a) Linear regression training MSE and MAE



(b) Linear regression scatter plot for RTT prediction

Fig. 9. Linear regression model performance.

Python API client [38]. Table II shows the dataset obtained by the Python API client from Prometheus for machine learning. Prometheus stores every record with a UNIX time stamp. Table II shows the number of samples, and UNIX timestamp of each record followed by 11 columns of RTT for each link. Machine learning models are trained using the RTT latency metric only because a metric is needed that can have a negative impact on the performance of communication. This data is then used to train machine learning models for time series forecasting of RTT latency for each link. The dataset obtained from monitoring tools has 10893 records. The dataset is then split into 70% for training an ML model and 30% for testing.

V. Machine Learning

When the predicted RTT is given input to the Dijkstra [39] algorithm as weights, the Dijkstra algorithm finds the shortest path with minimum weights. Dijkstra's algorithm guarantees to find the shortest path in a weighted graph with non-negative edge weights. It maintains a priority queue of nodes based on their tentative distances from the source node. It iteratively selects the node with the smallest tentative distance, relaxes its neighbors, and updates their distances if a shorter path is found. For machine learning non-linear models such as RNN, LSTM, and GRU were used as well as linear models such as SVM and LR were used. Overall non-linear models performed better than linear models. This section presents the performance of each model.

A. Linear Regression

Linear regression [40] is a statistical method used to model the relationship between a dependent variable and one independent variable. It is a linear approach represented by the main equation:

$$y = b_0 + \overrightarrow{b}_1 \overrightarrow{x}_1. \tag{1}$$

In linear regression, the dependent variable is represented by y, and the independent variable is represented by $\overrightarrow{x}_1$. The intercept of the regression line is denoted as b0, and the vector of slope coefficients is represented as $\overrightarrow{b}_1$. The objective of this algorithm is to determine the curve that provides the best fit to the given data, accurately describing the relationship between the dependent and independent variables. The algorithm achieves this by considering all possible trend lines through the data and calculating the squared differences between the observed values (y) and the predicted values ($\bar{y}$). It stores these squared differences $(y - \bar{y})^2$ for each trend line and selects the line that minimizes the sum of these squared differences, $\sum_i (y_i - \bar{y_i})^2$. In other words, it chooses the line of best fit that minimizes the overall distance between the actual data points and the points predicted by the line.

Overall linear regression showed an R^2 score of 0.56, a mean absolute error of 0.13, and a mean square error of 0.13. Fig. 9a shows the training MSE and training MAE of the linear regression model. Fig. 9b shows a scatter plot for the linear regression model predicting RTT for 11 links in the topology. Fig. 9b shows that most of the data in the scatter plot is scattered. This means that the values predicted are far from the actual data represented by a straight line.

B. Support Vector Machine

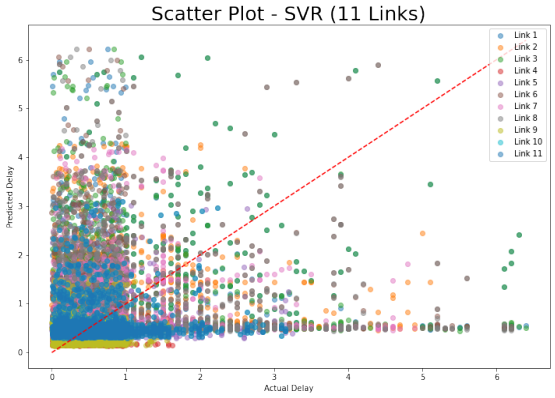
The support vector machines (SVM) method, initially introduced by Vapnik, was originally designed for solving pattern recognition problems. Subsequently, Vapnik extended the application of SVM to address function fitting problems in 1998, leading to the development of the support vector regression (SVR) method [41].

It is assumed that a set of data G = $\{(x_i, d_i)\}$, for all i $\in \{1, \cdots, N\}$ $x\_i \in R^n$ is a n dimension input vector, $d_i \in R^1$ is a corresponding target output, and N expresses the total number of pattern records. The linear regression estimating function can be expressed as [41]:

$$y = f(x)\ w\psi(x)\ +\ b. \tag{2}$$

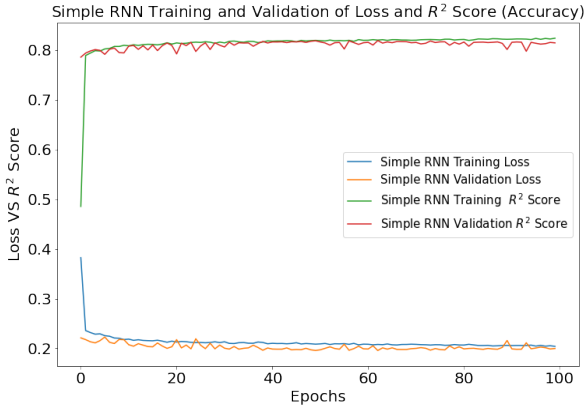
Overall SVR showed an R^2 score of 0.59, a mean absolute error of 0.28, and a mean square error of 0.13. Fig. 10a shows the training MSE and training MAE of the SVR model. Fig.

(a) SVR training MSE and training MAE

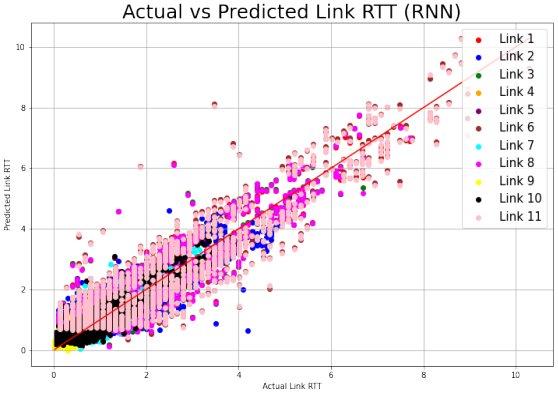


(b) SVR scatter plot for RTT prediction

Fig. 10. SVR model performance.



(a) RNN training and validation of loss and R^2score (accuracy)



(b) RNN scatter plot for RTT prediction

Fig. 11. RNN model performance.

10b shows a scatter plot for the SVR model predicting RTT for 11 links in the topology. Fig. 10b shows that most of the data in the scatter plot is scattered. This means that the values predicted are far from the actual data represented by a straight line.

C. Recurrent Neural Network

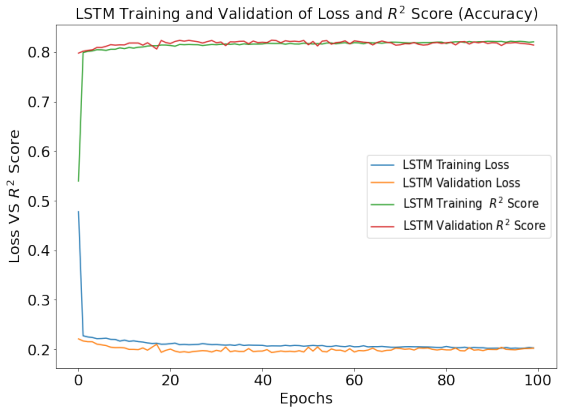
The Recurrent Neural Network (RNN) model shares similarities with the basic FFNN (Feedforward Neural Network) model, but there are key distinctions between them. The most fundamental difference is that RNN allows outputs from layers to cycle back into the network. This architecture grants the RNN the ability to incorporate dependencies among the data, a factor that traditional FFNN cannot capture. In the RNN model, given an input sequence $x = (x_1, \cdots, x_T)$, it iterates through the following equations from t = 1 to T, computing the hidden vector sequence $h = (h_1, \cdots, h_T)$ and the output vector sequence $y = (y_1, \cdots, y_T)$:

$$h_t = f\left(W_{xh}x_t + W_{hh}h_{t-1} + b_h\right), \quad (3)$$

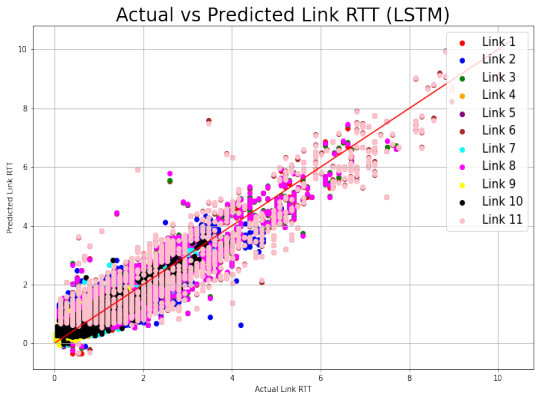
$$y_t = W_{hy}h_t + b_y. \quad (4)$$

RNN uses weight matrices W_{ij} between layers and employs the backpropagation through time (BPTT) algorithm for gradient computation during training [27].

Fig. 11a shows the training and validation loss versus the training and validation R^2 score of the RNN model. Overall RNN shows an R^2 score of 0.81. R^2 score shows the accuracy of the RNN model. A decreasing training loss and increasing training R^2 indicates that the model is improving its ability to fit the training data, while a decreasing validation loss and increasing validation R^2 score suggests that the model is generalizing well to unseen data. We can see in Fig. 11a that a decrease in training and validation loss directly causes an increase in training and validation of R^2 score i.e. the accuracy of the model. The convergence or stabilization of all four curves indicates that the model is learning effectively and not overfitting. Fig. 11b shows a scatter plot for the RNN model predicting RTT for 11 links in the topology. Fig. 11b shows that most of the values predicted are close to the actual data that is represented by the straight line.

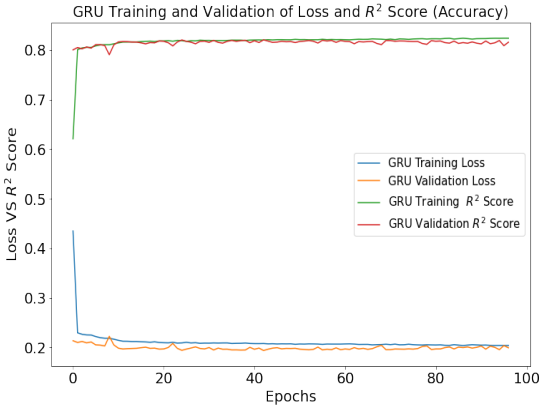


(a) LSTM Training and validation of loss and R^2score (Accuracy)

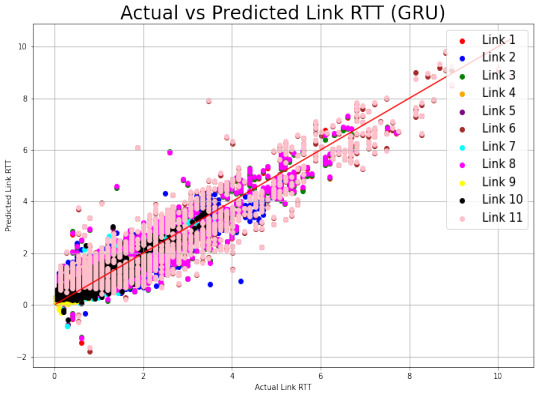


(b) LSTM Scatter plot for RTT prediction

Fig. 12. LSTM model performance.



(a) GRU Training and validation of loss and R^2score (Accuracy)



(b) GRU Scatter plot for RTT prediction

Fig. 13. GRU model performance.

D. Long Short-Term Memory

Training standard RNNs can encounter the vanishing gradient problem [27], where long-term dependencies become difficult to weight. To address this issue, two notable architectures, LSTM and GRU, have been proposed. LSTM incorporates a memory unit, consisting of a memory cell c_t and an output h_t, modifying the RNN model [27].

$$h_t = o_t \tanh(c_t) \tag{5}$$

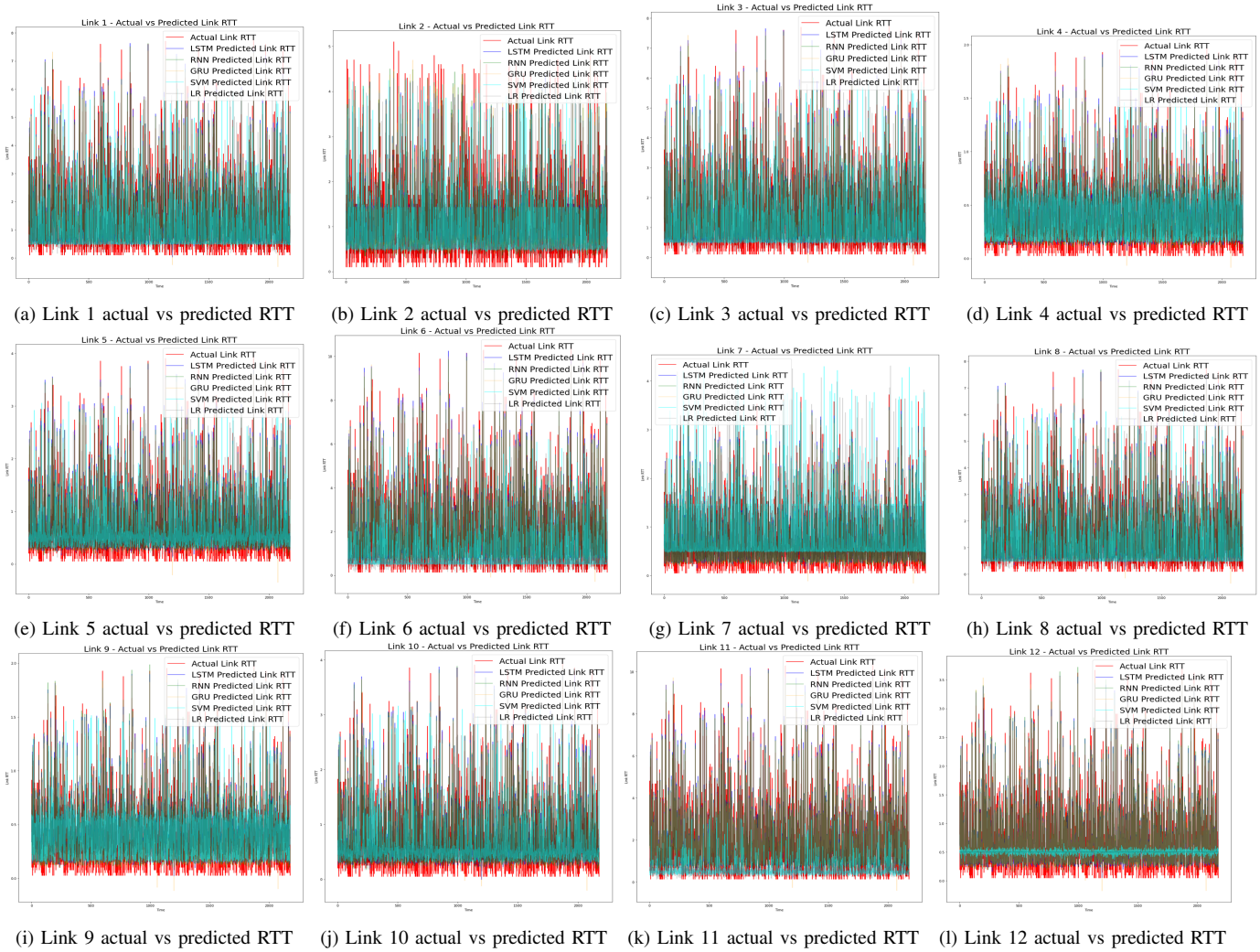
Fig. 12a shows the training and validation loss versus the training and validation R^2 score of the LSTM model. Overall LSTM shows an R^2 score of 0.81. R^2 score shows the accuracy of the LSTM model. A decreasing training loss and increasing training R^2 score indicates that the model is improving its ability to fit the training data, while a decreasing validation loss and increasing validation R^2 score suggests that the model is generalizing well to unseen data. We can see in Fig. 12a that a decrease in training and validation loss directly causes an increase in training and validation of R^2 score i.e. the accuracy of the model. The convergence or stabilization of all four curves indicates that the model is learning effectively and not overfitting. Fig. 12b shows a scatter plot for the LSTM model predicting RTT for 11 links in the topology. Fig. 12b shows that most of the values predicted are close to the actual data that is represented by the straight line.

E. Gated Recurrent Unit

Another variant of the RNN is the GRU [27], which has quite a similar architecture to an LSTM. The GRU also has gated units that control the flow of information inside the unit; however, unlike the LSTM, the GRU does not have separate memory cells. The activation h_t at time t of this unit is a linear combination of the activation at the previous time step h_{t-1} and the candidate activation $\widetilde{h}_t$ [27]:

$$h_t = (1 - z_t)\, h_{t-1} + z_t \widetilde{h}_t. \tag{6}$$

Fig. 13a shows the training and validation loss versus the training and validation R^2 score of the GRU model. Overall GRU shows an R^2 score of 0.81 R^2 score shows the accuracy of the GRU model. A decreasing training loss and increasing training R^2 score indicates that the model is improving its



(a) Link 1 actual vs predicted RTT (b) Link 2 actual vs predicted RTT (c) Link 3 actual vs predicted RTT (d) Link 4 actual vs predicted RTT

(e) Link 5 actual vs predicted RTT (f) Link 6 actual vs predicted RTT (g) Link 7 actual vs predicted RTT (h) Link 8 actual vs predicted RTT

(i) Link 9 actual vs predicted RTT (j) Link 10 actual vs predicted RTT (k) Link 11 actual vs predicted RTT (l) Link 12 actual vs predicted RTT

Fig. 14. Actual RTT vs LR, SVR, RNN, LSTM, and GRU predicted RTT for 11 links in the overlay network.

ability to fit the training data, while a decreasing validation loss and increasing validation R^2 score suggests that the model is generalizing well to unseen data. We can see in Fig. 13a that a decrease in training and validation loss directly causes an increase in training and validation of R^2 score i.e., the accuracy of the model. The convergence or stabilization of all four curves indicates that the model is learning effectively and not overfitting. Fig. 13b shows a scatter plot for the GRU model predicting RTT for 11 links in the topology. Fig. 13b shows that most of the values predicted are close to the actual data that is represented by the straight line.

VI. Results and Discussion

The performance of two linear models and three non-linear models was evaluated for predicting RTT in a network topology consisting of 11 links. For the linear models linear regression and SVR are used. For the non-linear models RNN, LSTM, and GRU models are used. Overall all non-linear models performed better than linear models. All non-linear models performed similarly but GRU showed even better mse. Linear models also performed similarly but showed poor performance in comparison to the non-linear models.

The scatter plot in Figs. 9b and 10b for LR and SVR reveals a larger spread of predicted values away from the actual data line, suggesting less accurate predictions. Both linear models showed 0.13 MSE. Linear regression showed 0.13 MAE which is less than 0.28 MAE of SVR, but on the other hand SVR shows a 0.59 R^2 score which is better than 0.56 R^2 score of the linear regression model. The graph of training MSE and MAE in Figs. 9a and 10a for the Linear regression and SVR models exhibits a zig-zag pattern, indicating an unstable and inconsistent learning process. This behavior suggests that the model's performance may be affected by model instability, or inappropriate learning rate, making it less reliable for making accurate predictions.

In contrast, the non-linear models demonstrated superior performance. RNN, LSTM, and GRU achieved an R^2 score of 0.81 and a MAE of 0.32. However, RNN, LSTM, and GRU showed 0.20, 0.19, and 0.13 MSE. This shows that GRU performed well in comparison to other non-linear models from the mse perspective.

Overall these results indicate that the non-linear models

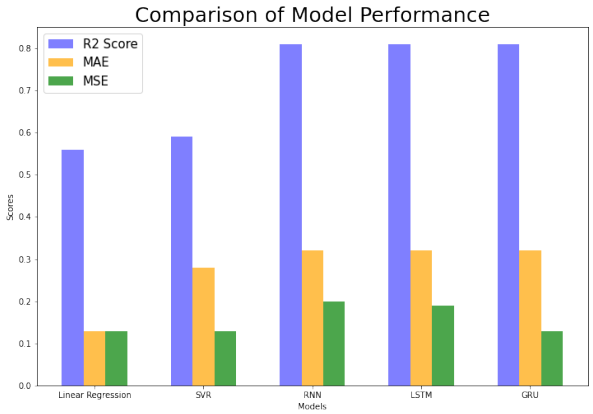


Fig. 15. Comparison of model performance.

TABLE III
MODEL PERFORMANCE COMPARISON

S. No.	Model	R^2 Score	MAE	MSE
1	Linear regression	0.56	0.13	0.13
2	SVR	0.59	0.28	0.13
3	RNN	0.81	0.32	0.20
4	LSTM	0.81	0.32	0.19
5	GRU	0.81	0.32	0.13

effectively capture the underlying patterns in the data, resulting in more accurate predictions. The plot for training and validation loss versus training and validation R^2 score in Figs. 11a, 12a, and 13a for non-linear models demonstrates successful convergence and model generalization. The scatter plot in Figs. 11b, 12b, and 13b for non-linear models reveals less spread of predicted values and is close to the actual data line, suggesting more accurate predictions. Fig. 15 and Table III shows model performance comparison. Fig. 15 and Table III summarize the R^2 score, MSE, and MAE for all models, further highlighting the superior performance of non-linear models. These findings emphasize the effectiveness of non-linear models in capturing the complex relationships and patterns within the RTT data. Additionally, Fig. 14 presents individual graphs comparing the actual RTT with RNN, LSTM, GRU, LR, and SVR predicted RTT for each of the 11 links. It is evident that the non-linear models predictions closely align with the actual RTT, indicating its ability to capture the nuances of link behavior. In contrast, linear models predictions exhibit greater deviations from the actual data, suggesting limitations in capturing the underlying dynamics accurately.

Overall, these results confirm that non-linear models outperforms linear models in predicting RTT in the network topology under consideration. The higher R^2 score and lower errors of the non-linear models demonstrate its superior capability to model complex relationships and make accurate predictions.

VII. CONCLUSION

A multi-cloud infrastructure offers benefits like increased resources, availability, and reliability but also brings some challenges, such as complex configurations for the deployment of clouds, an overlay network, and the best paths for communication between clouds. This paper presents IBN as an intent-based, intelligent, and closed-loop system to address these challenges. The IBN receives a networking requirement in the form of an intent from a user in an abstract and simple way. The IBN translates the intent into complex networking configurations. The IBN presents an intent translation mechanism for three types of intents i.e., cloud deployment, overlay network deployment in which each link is a GENEVE tunnel, and path selection in the overlay network for communication between clouds. For the path selection, a machine learning model predicts the RTT of each link in the overlay network. A monitoring system was developed to obtain data for the training and testing of machine learning models. In the context of a multi-cloud infrastructure, the node-to-node path, where each node represents a cloud node, can be determined using Dijkstra's algorithm with predicted RTT values as the weights. The E2E path is determined by selecting the source and destination instances based on the instance and QoS parameters specified in the intent. The E2E path is implemented by the SDN Controller in the form of flow rules. The process of path selection and implementation is handled by an individually executing component of IBN called an OPA.

For the validation of the proposed system, the presented IBN is used to deploy 7 OpenStack-based clouds at TEIN in Pakistan, Korea, Malaysia, and Cambodia. IBN is also used to deploy an overlay network of 11 Geneve tunnels between the clouds. IBN then utilizes machine learning to select the best path. For this purpose, 2 linear and 3 non-linear models are trained and tested, namely linear regression, SVR, RNN, LSTM, and GRU. The performance of all the linear models was similar and poor. Results show that non-linear models perform better than linear models. The non-linear models perform almost similarly, with a 0.81 R^2 score and 0.32 MAE. However, the RNN, LSTM, and GRU models show 0.20, 0.19, and 0.13 MSE. The IBN therefore utilizes a non-linear model such as GRU for predicting the RTT of all the 11 links in the overlay network. The predicted values are given as input to the Dijkstra algorithm as weights. The Dijkstra algorithm returns the shortest path based on the weights. The shortest path returned by the Dijkstra algorithm is the best path with minimum RTT.

In our future work, we aim to test other machine learning models such as GNN. We also aim to extend the functionality of IBN to cloud native functions (CNFs) and utilize machine learning for VNF and CNF relocation.

REFERENCES

[1] A. Clemm, L. Ciavaglia, L. Granville, and J. Tantsura, "Intent-based networking-concepts and overview," *Internet Engineering Task Force, Internet-Draft*, 2019.

[2] M. M. S. Sarwar, A. Muhammad and W.-C. Song, "IBN-based orchestration of overlay networks for multiple OpenStack clouds," in *Proc. ICONI*, 2022.

[3] R. Chayapathi, S. F. Hassan, and P. Shah, *Network Functions Virtualization (NFV) with a Touch of SDN: Netw Fun Vir (NFV ePub_1*. Addison-Wesley Professional, 2016.

[4] K. Subramanian and F. L. John, "Data security in single and multi cloud storage–an overview," *Int. J. Innovative Research Comput. Commun. Eng.*, vol. 4, no. 11, 2016.
[5] S. Gopinath and E. Sherly, "A comprehensive survey on data replication techniques in cloud storage systems," *Int. J. Appl. Eng. Research*, vol. 13, no. 22, pp. 15926–15932, 2018.
[6] T. Saad, B. Alawieh, H. T. Mouftah, and S. Gulder, "Tunneling techniques for end-to-end VPNs: Generic deployment in an optical testbed environment," *IEEE Commun. Mag.*, vol. 44, no. 5, pp. 124–132, 2006.
[7] M. M. S. Sarwar *et al.*, "VXLAN tunneling in openstack based multi-site cloud a secure mechanism for communication and data sharing," in *Proc. KNOM*, 2022.
[8] K. Rusek, J. Suárez-Varela, P. Almasan, P. Barlet-Ros, and A. Cabellos-Aparicio, "Routenet: Leveraging graph neural networks for network modeling and optimization in SDN," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2260–2270, 2020.
[9] M. Ferriol-Galmés *et al.*, "Routenet-erlang: A graph neural network for network performance evaluation," in *Proc. IEEE INFOCOM*, 2022.
[10] Asia@Connect, *Asia@Connect*, 2022 (accessed june 5, 2022), https://www.tein.asia/sub/?mc=2030.
[11] M. M. S. Sarwar, S. Alam, A. Muhammad, and W.-C. Song, "IBN@TEIN: An AI-driven intent-based networking platform for service deployment with qos assurance," *KNOM Review*, vol. 25, no. 02, pp. 1–10, 2022.
[12] K. Abbas, M. Afaq, T. A. Khan, A. Mehmood, and W.-C. Song, "IBNslicing: intent-based network slicing framework for 5G networks using deep learning," in *Proc. APNOMS*, 2020.
[13] T. A. Khan, K. Abbass, A. Rafique, A. Muhammad, and W.-C. Song, "Generic intent-based networking platform for E2E network slice orchestration & lifecycle management," in *Proc. APNOMS*, 2020.
[14] E. Zeydan and Y. Turk, "Recent advances in intent-based networking: A survey," in *Proc. VTC2020-Spring*, 2020.
[15] M. Kimmerlin *et al.*, "Network expansion in OpenStack cloud federations," in *Proc. EuCNC*, 2017.
[16] F. Paraiso, N. Haderer, P. Merle, R. Rouvoy, and L. Seinturier, "A federated multi-cloud PaaS infrastructure," in *Proc. IEEE Fifth International Conference on Cloud Computing*, 2012.
[17] R. Fielding *et al.*, "RFC2616: Hypertext transfer protocol–http/1.1," 1999.
[18] Z. Aqun, Y. Yuan, J. Yi, and G. Guanqun, "Research on tunneling techniques in virtual private networks," in *Proc. WCC 2000-ICCT 2000*, 2000.
[19] M. Mahalingam *et al.*, "Virtual extensible local area network (VXLAN): A framework for overlaying virtualized layer 2 networks over layer 3 networks," Tech. Rep., 2014.
[20] J. Gross, I. Ganga, and T. Sridhar, "RFC 8926: Geneve: Generic network virtualization encapsulation," 2020.
[21] CISCO, *All Tunnels Lead to GENEVE*, 2022 (accessed june 5, 2022), https://blogs.cisco.com/datacenter/all-tunnels-lead-to-geneve.
[22] S. Wang, Q. Zhuo, H. Yan, Q. Li, and Y. Qi, "A network traffic prediction method based on LSTM," *ZTE Commun.*, vol. 17, no. 2, pp. 19–25, 2019.
[23] V. Alarcon-Aquino and J. A. Barria, "Multiresolution fir neural-network-based learning algorithm applied to network traffic prediction," *IEEE Trans. Syst., Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 36, no. 2, pp. 208–220, 2006.
[24] R. Vinayakumar, K. Soman, and P. Poornachandran, "Applying deep learning approaches for network traffic prediction," in *Proc. ICACCI*, 2017.
[25] M. Barabas, G. Boanea, A. B. Rus, V. Dobrota, and J. Domingo-Pascual, "Evaluation of network traffic prediction based on neural networks with multi-task learning and multiresolution decomposition," in *Proc. IEEE ICCP*, 2011.
[26] S. C. Prasad and P. Prasad, "Deep recurrent neural networks for time series prediction," *arXiv preprint arXiv:1407.5949*, 2014.
[27] N. Ramakrishnan and T. Soni, "Network traffic prediction using recurrent neural networks," in *Proc. IEEE ICMLA*, 2018.
[28] A. Panarello, A. Celesti, M. Fazio, M. Villari, and A. Puliafito, "A requirements analysis for IaaS cloud federation." in *Proc. CLOSER*, 2014.
[29] OpenStack, *The Most Widely Deployed Open Source Cloud Software in the World*, 2022 (accessed june 5, 2022), https://www.openstack.org/.
[30] M. M. S. Sarwar, J. J. D. Rivera, A. Muhammad, and W.-C. Song, "GENEVE@ TEIN: A sophisticated tunneling technique for communication between OpenStack-based multiple clouds at TEIN," in *Proc. IEEE APNOMS*, 2022.
[31] V. Jyotinagar and B. B. Meshram, "Developing an OpenStack environment: An exploration of experimentation and the diagnostic research," in *Proc. ICECAA*, 2023.
[32] OpenStack, *Network Troubleshooting*, 2022 (accessed june 5, 2022), https://docs.openstack.org/operations-guide/ops-network-troubleshooting.html.
[33] OpenStack, *OVS L2 Agent*, 2022 (accessed june 5, 2022), https://docs.openstack.org/neutron/latest/contributor/internals/openvswitch_agent.html.
[34] M. T. Islam, N. Islam, and M. A. Refat, "Node to node performance evaluation through RYU SDN controller," *Wireless Personal Commun.*, vol. 112, pp. 555–570, 2020.
[35] B. Tierney *et al.*, "perfSONAR: Instantiating a global network measurement framework," *SOSP Wksp. Real Overlays Distrib. Syst.*, vol. 28, 2009.
[36] P. Pushgateway, *Prometheus Pushgateway*, 2022 (accessed june 5, 2022), https://github.com/prometheus/pushgateway.
[37] T. Drabas and D. Lee, *Learning PySpark*. Packt Publishing Ltd, 2017.
[38] M. M. S. Sarwar, A. Muhammad, and W.-C. Song, "LSTM-assisted traffic forecasting for path selection in an overlay network between multiple clouds at TEIN," *KNOM Review*, vol. 26, no. 1, pp. 69–78, 2023.
[39] Y. Deng, Y. Chen, Y. Zhang, and S. Mahadevan, "Fuzzy Dijkstra algorithm for shortest path problem under uncertain environment," *Appl. Soft Comput.*, vol. 12, no. 3, pp. 1231–1237, 2012.
[40] N. Uras, L. Marchesi, M. Marchesi, and R. Tonelli, "Forecasting Bitcoin closing price series using linear regression and neural networks models," *PeerJ Comput. Science*, vol. 6, p. e279, 2020.
[41] J. Wang, L. Li, D. Niu, and Z. Tan, "An annual load forecasting model based on support vector regression with differential evolution algorithm," *Appl. Energy*, vol. 94, pp. 65–70, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261912000165

Mir Muhammad Suleman Sarwar started his career in 2010 at the IT Excellence Center, Directorate of IT, Pakistan, as a programmer. In 2011 he started his research and teaching career at the Department of Computer Science, University of Peshawar, Pakistan. In 2014 he joined the Institute of Business and Management Sciences, Pakistan as a Lecturer. In 2014, he began working as a Research Associate and has been serving as a Lecturer since 2015 at the Department of Computer Science, COMSATS University Islamabad, Pakistan. He received his BS degree in IT from the Institute of Business and Management Sciences, Agricultural University Peshawar, Pakistan in 2010. He received his MS degree in Computer Sciences from the University of Peshawar, Pakistan in 2015. In 2021, he joined Network Convergence Lab as a PhD scholar in the Department of Computer Engineering, Jeju National University, South Korea. His research interests include overlay networks, Cloud, SDN, NFV, intent-based networking, and machine learning.

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.