Machine Learning-based VNF Deployment

- Master Thesis Defense -

Suhyun Park

Supervisor: Prof. James Won-Ki Hong

Dept. of CSE, DPNM Lab., POSTECH, Korea
sh.park11@postech.ac.kr

2020. 12. 22.

Table of Contents

Introduction

Introduction

▪ Network Function Virtualization (NFV)

- NFV technologies are to reduce cost and accelerate service deployment by decoupling functions(e.g. like a firewall, load balancer etc.) from dedicated hardware and moving them to virtual servers.

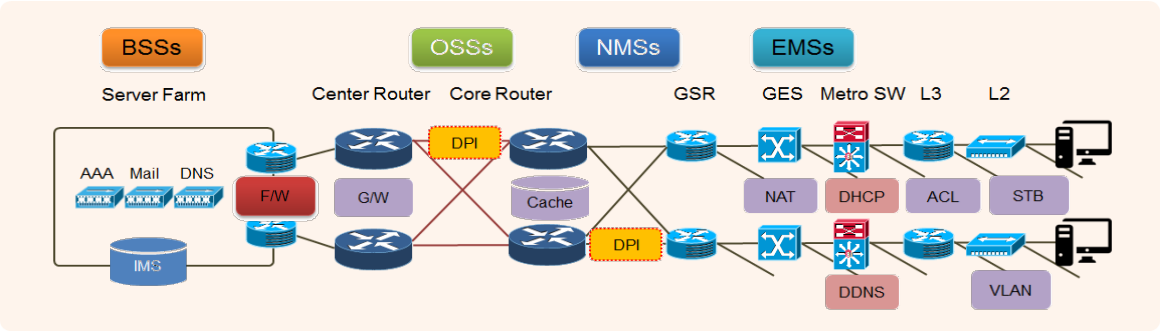


Fig. 1 : Traditional Network Environment

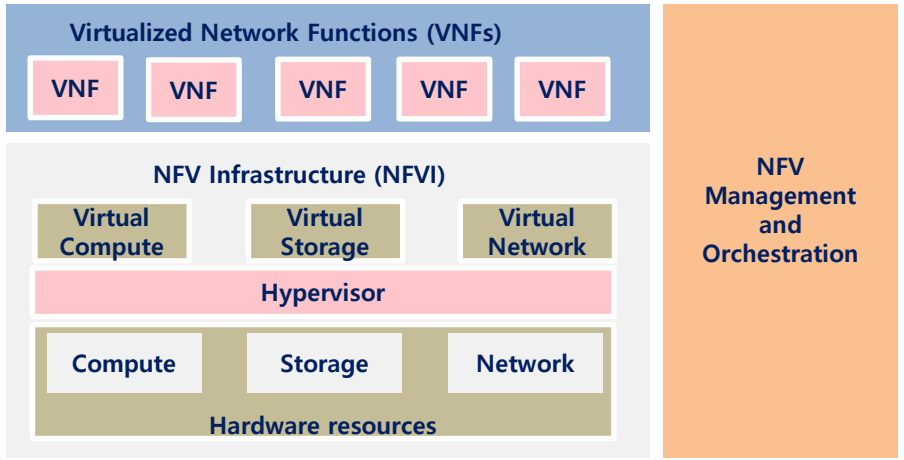


Fig. 2 : NFV Architecture

- Advantages
  - Reduced OPEX and CAPEX
  - Reduced Time to Market
  - Flexibility
  - Openness (Vendor neutrality)

- NFV Standardization

  On-going Standardization led by ETSI

  Phase-1 (2013~2014): Structure planning of NFV

  Release-2 (2015~2016): Interoperability among NFV components

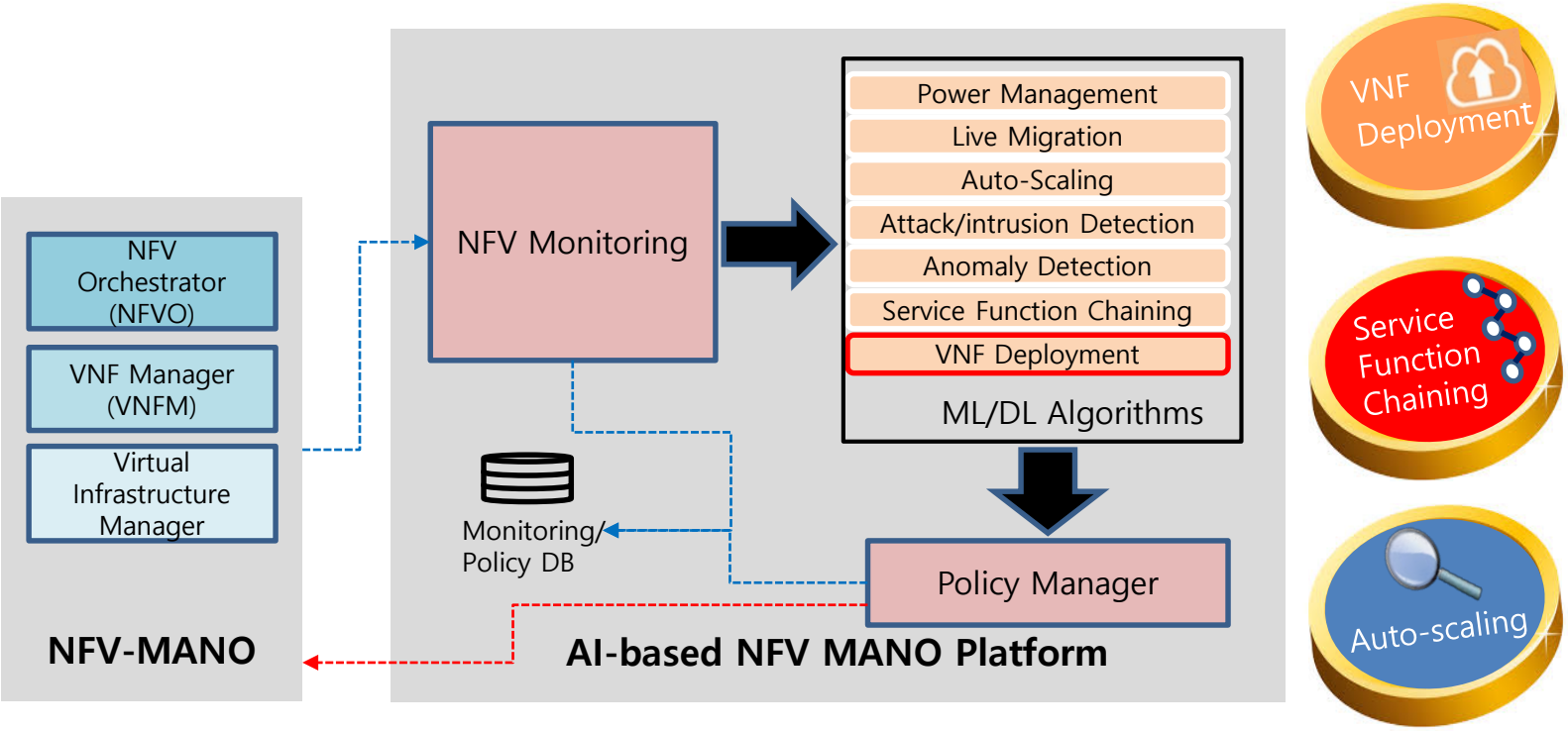
  Release-3 (2017~2018): Functional enhancement of NFV

  Release-4 (2019~Now): Automation, optimization of MANO (5G core)

Introduction

- NFV Research Issues
  - Increasing **complexity** of management
  - Especially for VNF Deployment :
    - Need to manage **tens of thousands** VNF instances
    - Need to allocate proper resources to **dynamically changing traffic requests**

⇒ **Development of AI-based MANO**



- Deciding the SFC routing path and VNF deployment decision is NP hard problem
- Running time optimization

→ **Machine Learning-based VNF deployment prediction**

Fig. 3 : NFV Lifecycle Management

Introduction

VNF Deployment Tasks

- Definition
  - Making VNF deployment decision to maximize the Quality of Service (QoS) with the minimum operating costs.
- Scope
  - Scaling : a decision to increase or decrease the capacity of each VNF
  - **Placement** : a decision including the exact location where each type of VNF will be installed on the network topology
- Goal
  - Making a **proactive deployment decision before the new requests coming**.

Related Work

Related Work

- **VNF Placement**
  - **VNF placement approach** not considering traffic dynamicity
  - **Readjustment approach** considering traffic dynamicity

- Cohen et al. access the VNF placement problem from the perspective of **minimizing deployment and connection costs** of VNFs
  - Use near optimal approximation algorithm

⇒ focusing on the relatively **static situation**,
while ignoring dynamic and changing aspects of the network status

- M. Ghaznavi et al. formulate **an elastic VNF placement problem of the dynamic situation** by considering elasticity overhead and resource consumption.

⇒ **responsively determining placement** upon the receipt of a new request,
resulting in high latencies and endangering the overall QoS

Related Work

- **ILP - Machine Learning based Approach**
  - M. F. Bari et al. address the problem that determines the VNF deployment and SFC routing.
    - The problem has been defined as an **optimization of the network operating costs** and utilization while **meeting the SLA standards.**
  - S. Lange et al. predict the demand for virtual servers for each VNF type at a future time by using randomly generated service request data and **the ILP solution**.

⇒ This study trains ML model to proactively predict the needed number of VNF instances for each type.

⇒ The output doesn't include the exact location of VNF instances.

Authors	Research Goal	Research scope	Dynamicity considered	Proactive approach	Methodology
Cohen et al.	Minimize deployment and connection costs of VNFs	Placement	NO	NO	Near optimal approximation algorithm
M. Ghaznavi et al.	Achieve elastic VNF placement	Placement	YES	NO	ILP/ Heuristic algorithm
M. F. Bari et al.	Decide VNF deployment and SFC routing	Placement	YES	NO	ILP/ Heuristic algorithm
S. Lange et al.	Predict the demand for virtual servers for each VNF type at a future time	Scaling	YES	YES	Machine learning

Table 1 : Comparison of Related Work

Methodology and Implementation

Methodology

▪ Overview of VNF Deployment ML Model Development

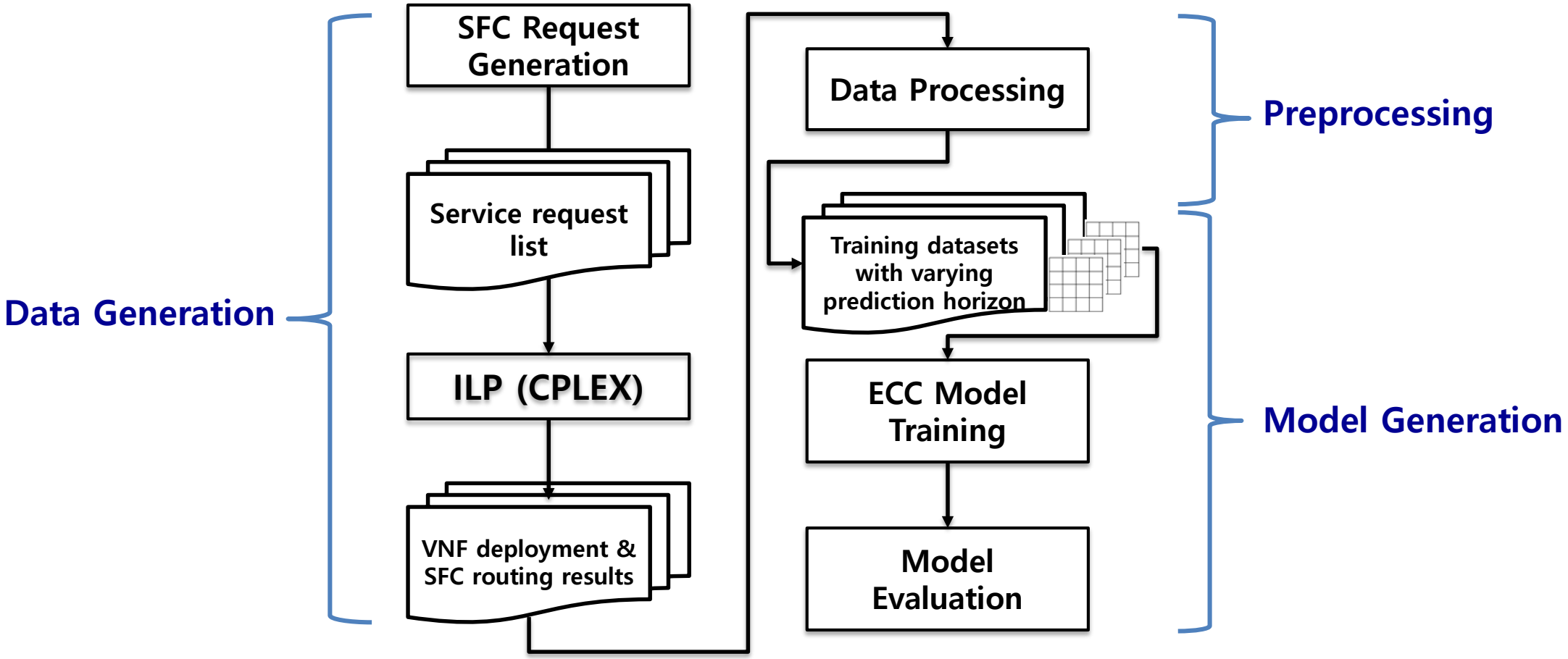


Fig. 4 : Overview of Machine Learning Model Development

SFC Request Generation

- **Network topology for SFC request generation**
  - Multi-access Edge Computing (MEC) as a simulation topology
  - In MEC Topology, VNFs can be deployed at the edge server instead of data center cloud to reduce the delay.
  - The topology data includes the network topology as a graph.

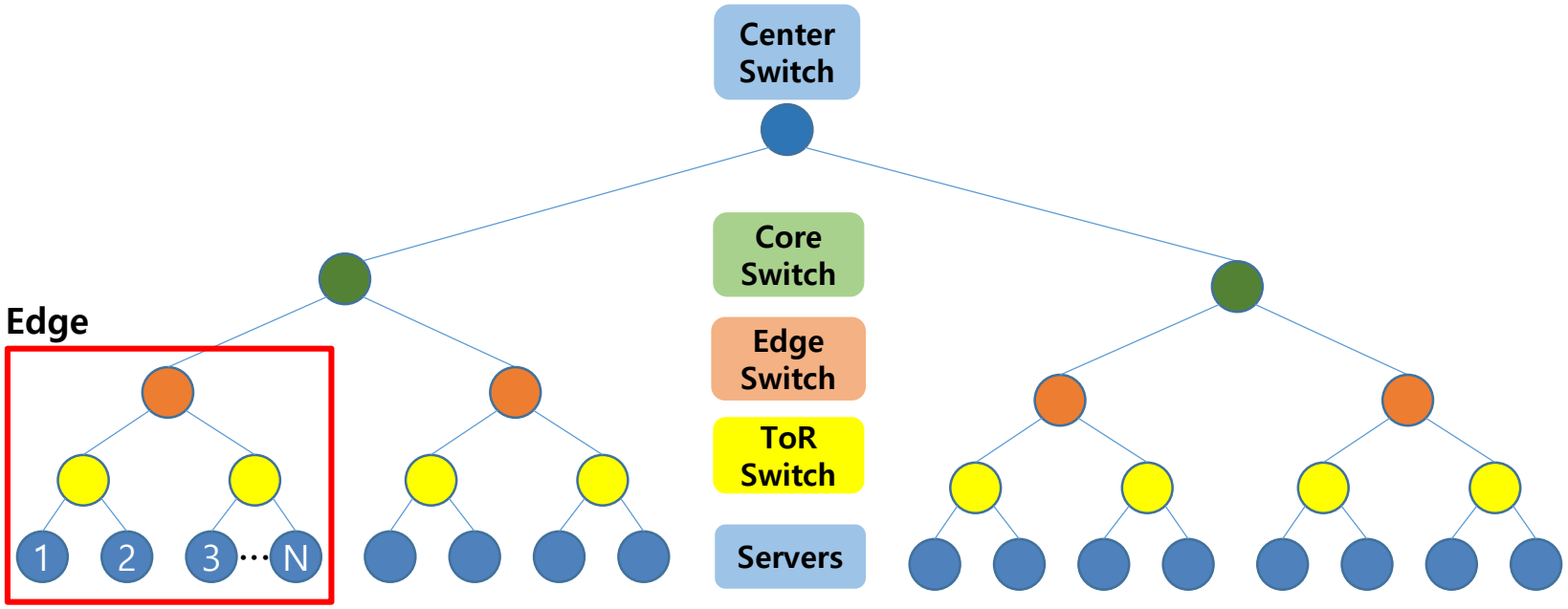


Fig. 5 : MEC Topology

SFC Request Generation

Data Generation

- When generating service requests, the overall traffic pattern follows **the simulation traffic pattern**.
- **The number of service requests** temporally vary according to the pattern.
- **The interarrival time of** service requests follows **the normal distribution** to generate the list as below.
- The proportions of **three SFC types** are predetermined as below.

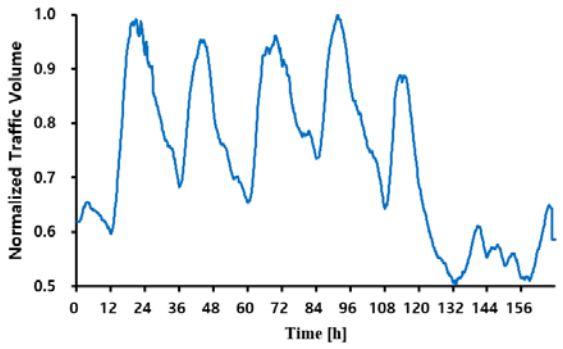


Fig. 6 : Simulation Traffic Pattern

Service id	Type	Proportion
1	NAT - Firewall - IDS	0.5
2	NAT - IDS	0.3
3	NAT - Firewall	0.2

Table 2 : SFC Catalog

arrival time	duration	src	dest	traffic	maxlat	penalty	vnf1	vnf2	vnf3	vnf4	sfcid
63.48384	833	4	7	25310	732	0.0000001	nat2	ids2	NA	NA	2
110.41171	710	5	6	25758	705	0.0000001	nat1	firewall1	ids1	NA	1
127.41385	1120	4	7	23752	737	0.0000001	nat2	ids2	NA	NA	2
185.9144	957	7	5	25249	704	0.0000001	nat3	firewall3	NA	NA	3
192.44491	1047	5	7	24508	701	0.0000001	nat2	ids2	NA	NA	2
238.69411	604	5	7	24704	714	0.0000001	nat1	firewall1	ids1	NA	1
327.88726	984	4	7	22719	715	0.0000001	nat1	firewall1	ids1	NA	1
372.20562	856	4	7	22730	701	0.0000001	nat1	firewall1	ids1	NA	1
401.19533	960	4	7	24441	717	0.0000001	nat2	ids2	NA	NA	2
409.91521	955	5	6	25999	728	0.0000001	nat1	firewall1	ids1	NA	1

Table 3 : An Example of Randomly Generated Service Requests

Data Generation (ILP Formulation)

▪ ILP Formulation for VNF Orchestration Problem (VNF-OP)

1. VNF deployment cost : the cost needed to transfer, boot, or attach the VM image

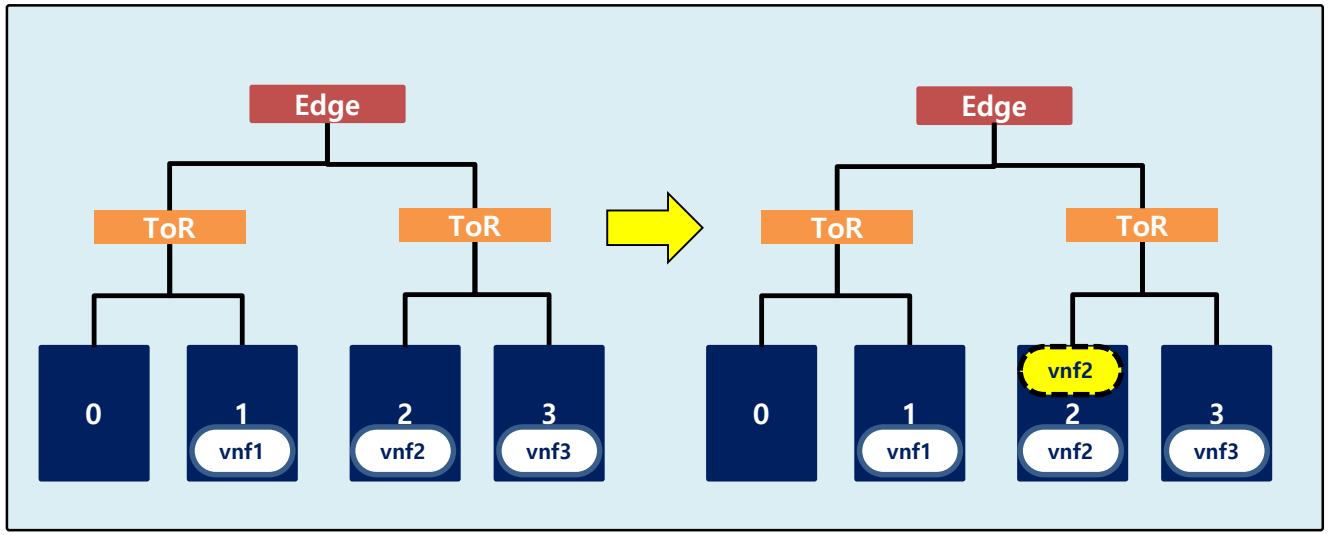


Fig. 7-1 : VNF deployment cost

$$D = \sum_{m \in M \mid y_m = 1} D_p^+ \times q_{mp} \times (y_m - \hat{y}_m)$$

D_p^+ : the deployment cost

q_{mp}: whether VNF m is of type p ∈ P or not

y_m: whether VNF m is active or not

$\hat{y}_m$: the previous status

2. Energy cost : the cost of energy consumption for the active servers

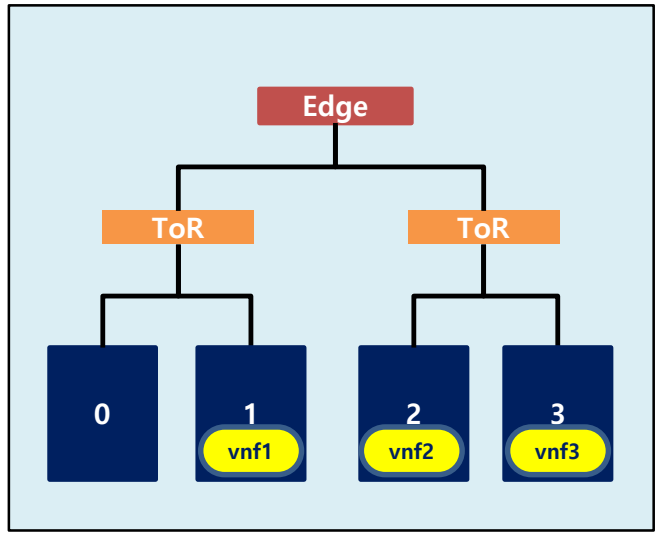


Fig. 7-2 : Energy cost

$$E_{\bar{n}} = \sum_{m \in \Omega_{\bar{n}}} y_m \times q_{mp} \times \left((e_{max}^r - e_{idle}^r) \times \frac{r_c}{r_t} + e_{idle}^r \right)$$

e_{idle}^r, e_{max}^r : energy cost in the idle and peak consumption state

r_t, r_c: total and consumed resource

Data Generation (ILP Formulation)

▪ ILP Formulation for VNF Orchestration Problem (VNF-OP)

3. Traffic forwarding cost : leasing cost of transit links and energy consumption of the network devices
4. Penalty for SLO violation : the penalty that must be paid to the customer for SLO violation

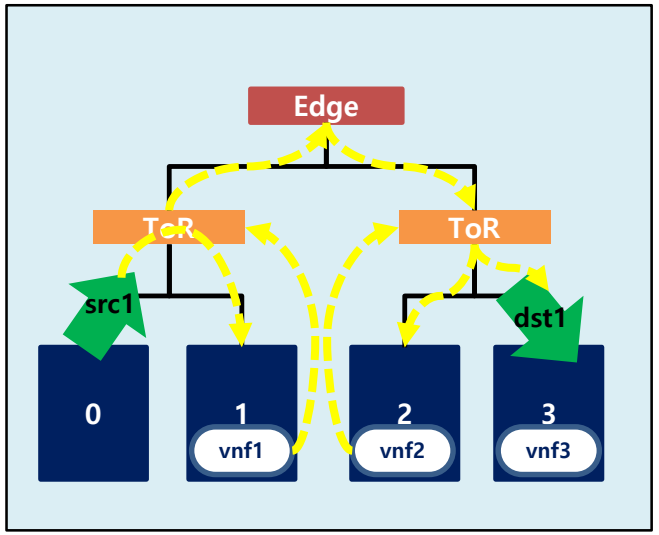


Fig. 7-3 : Traffic forwarding cost

$$F = \sum_{t \in T} \sum_{n_1 \in N^t} \sum_{n_2 \in \eta^t(n_1)} \sum_{\bar{u} \in \bar{S}} \sum_{\bar{v} \in \eta(\bar{u})} \left(\left(\omega_{\bar{u}\,\bar{v}}^{t n_1 n_2} - \hat{\omega}_{\bar{u}\,\bar{v}}^{t n_1 n_2} \right) \times \beta^t \times \sigma \right)$$

$\omega_{\bar{u}\,\bar{v}}^{t n_1 n_2}$: whether the link (n_1, n_2) uses physical link $(\bar{u}, \bar{v})$
β^t : the bandwidth demand of the traffic
σ : the cost of forwarding 1 Mbit data through one link in the network

$$P = \sum_{t \in T} \rho^t(\omega^t, \delta^t, \delta_a^t)$$

ω^t: the policy for determining penalty
δ^t: expected propagation delay
δ_a^t: actual propagation delay

Data Preprocessing

Monitoring Window and Prediction Horizon

- **Monitoring window (m)** : a certain amount of time we collect the monitoring data for the input of the machine learning model.
- **Prediction horizon (p)** : the time gap from the current time to the future time targeted for the optimal VNF deployment.

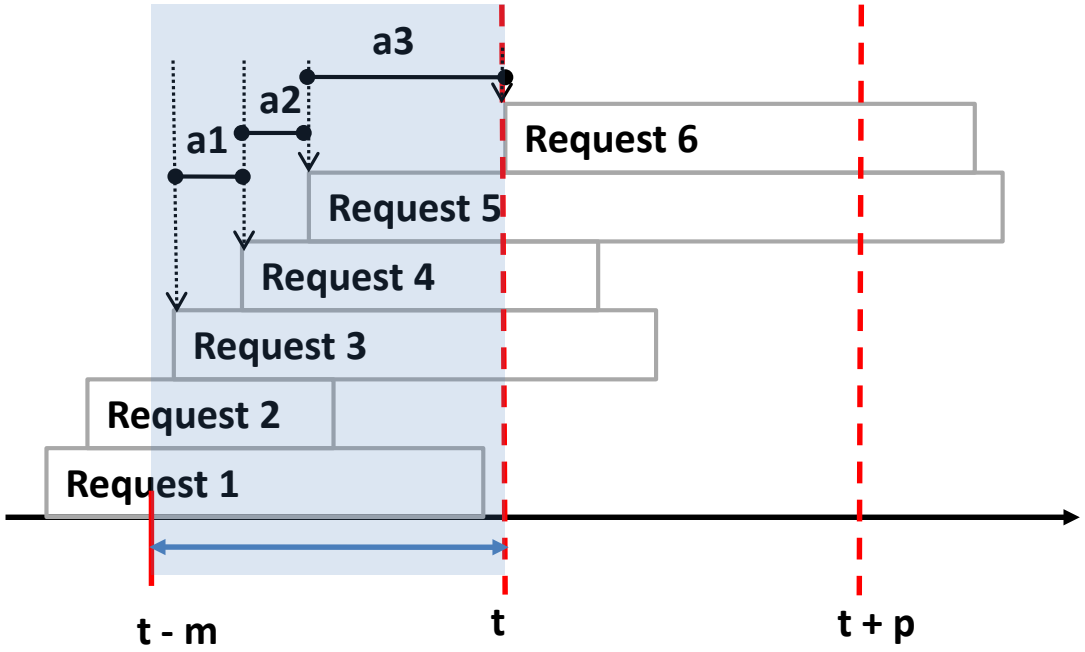


Fig. 9 : Monitoring Time Window (m) and Prediction Horizon (p)

arrival time	duration	src	dest	traffic	maxlat	penalty	vnf1	vnf2	vnf3	vnf4	sfcid
63.48384	833	4	7	25310	732	0.0000001	nat2	ids2	NA	NA	2
110.41171	710	5	6	25758	705	0.0000001	nat1	firewall1	ids1	NA	1
127.41385	1120	4	7	23752	737	0.0000001	nat2	ids2	NA	NA	2
185.9144	957	7	5	25249	704	0.0000001	nat3	firewall3	NA	NA	3
192.44491	1047	5	7	24508	701	0.0000001	nat2	ids2	NA	NA	2
238.69411	604	5	7	24704	714	0.0000001	nat1	firewall1	ids1	NA	1
327.88726	984	4	7	22719	715	0.0000001	nat1	firewall1	ids1	NA	1
372.20562	856	4	7	22730	701	0.0000001	nat1	firewall1	ids1	NA	1
401.19533	960	4	7	24441	717	0.0000001	nat2	ids2	NA	NA	2
409.91521	955	5	6	25999	728	0.0000001	nat1	firewall1	ids1	NA	1

Table 3 : An Example of Randomly Generated Service Requests

Data Preprocessing

▪ Graph Neural Network (GNN) input

- GNN algorithm uses the input data represented on the network topology in a graph form.
- The input data are defined as the network topology, the resource usage of servers, and the traffic usage of links of the network.

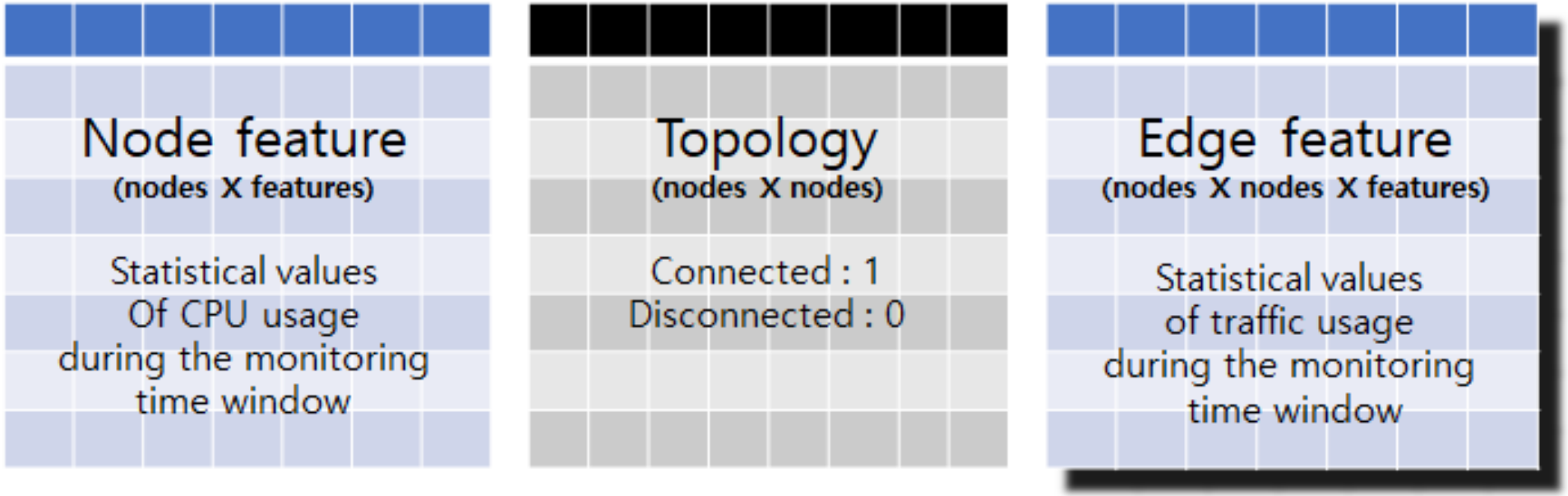
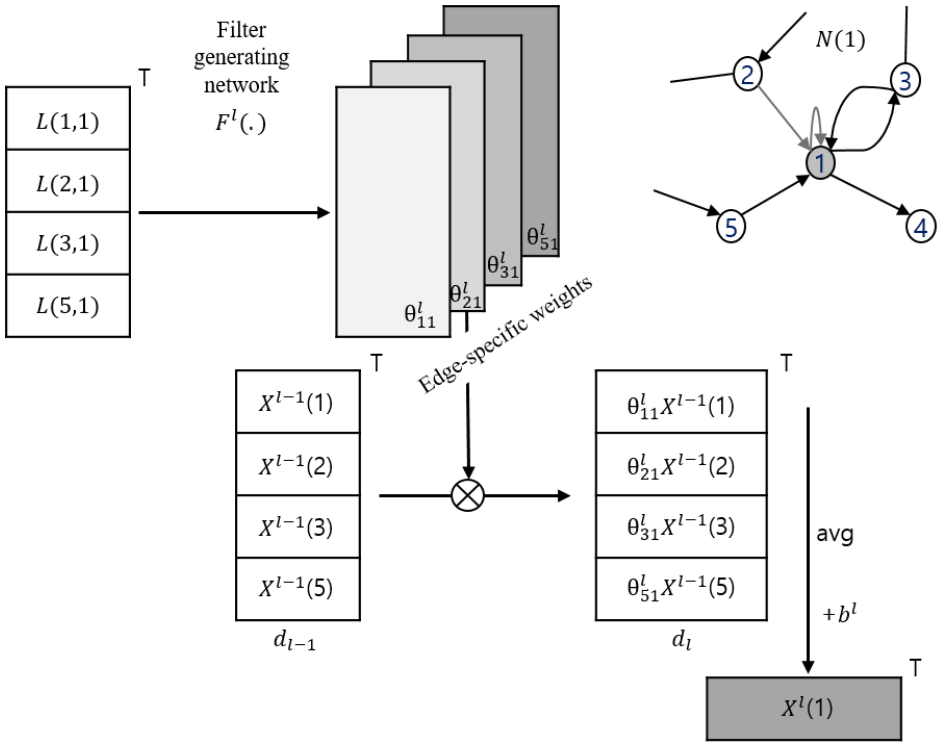


Fig. 8 : Input Data Format for GNN Encoding

Model Training

▪ ECC Algorithm

- The Edge-Conditioned Convolution (ECC) deals with our input data in a graph form in which each node and edge has a certain dimension of feature set respectively.
- ECC approach conditions each filtering weight on the edge features and computes the node as a weighted sum of its neighbor nodes.



$L(a,b)$: edge feature of the edge (a,b)
X^{l-1} (a) : node feature of the node (a) at layer, $l-1$
θ^l_{ab} : filtering weight for $L(a,b)$ at layer, l

Fig. 10 : Edge-conditioned filtered Graph Convolutional Neural Network

Model Training

Model Structure

- After the ECC layer, four feedforward layers are stacked.
- All layers are fully connected, and as activation functions, the Rectified Linear Unit (ReLu) for all hidden layers and Sigmoid for the output layer are used.
- Adam optimizer is used.

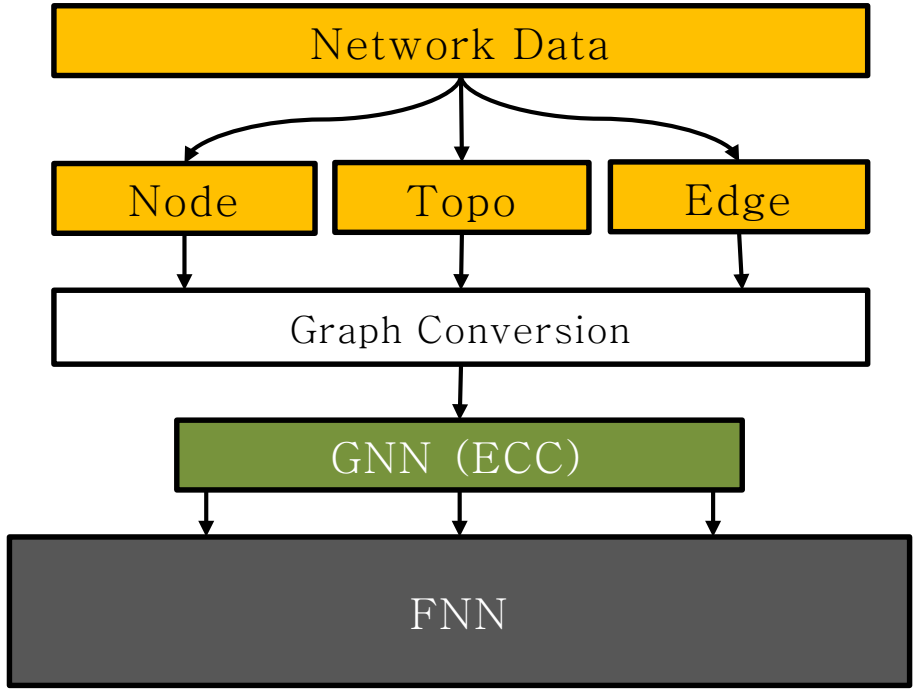


Fig. 11 : Model Structure

Module Implementation

- To apply the trained model to the OpenStack testbed, VNF deployment module loading the model has been implemented.
  - The module is implemented with RestAPI on Swagger 2.0.
  - The module consists of 7 functions.

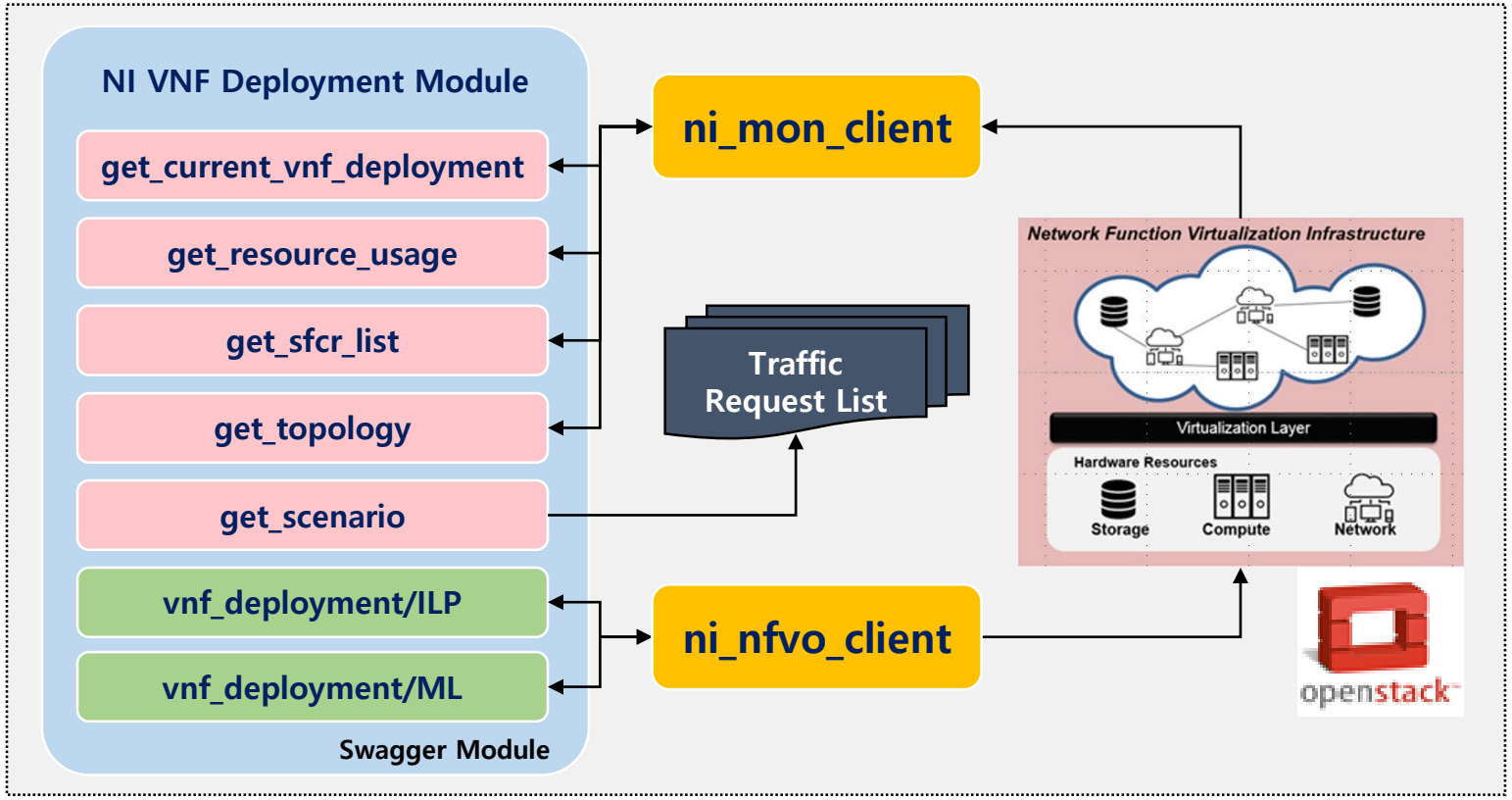
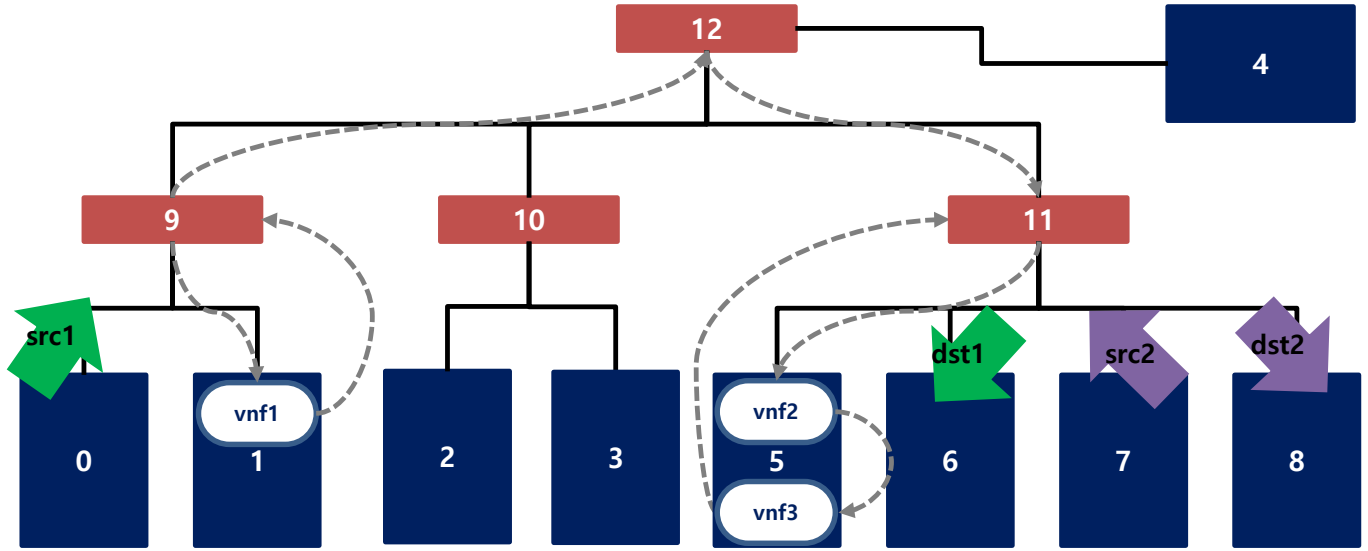


Fig. 12 : VNF Deployment Module with Testbed

Evaluation

Experiments

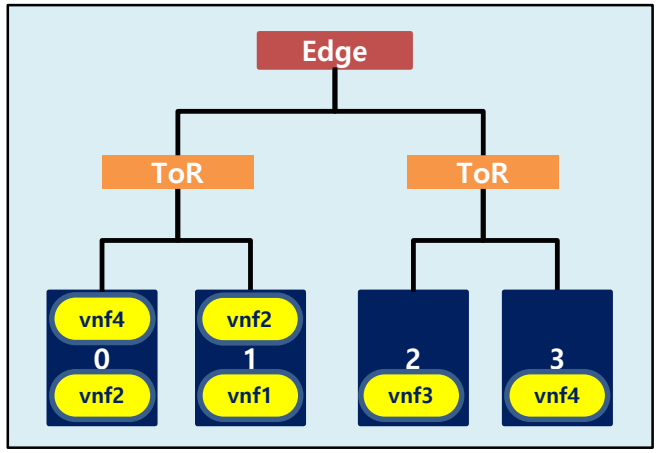
- **List of Experiments**
  1. With ILP simulation data, we trained three models for each dataset with different prediction time horizon and compared their mean accuracy.
  2. We compared the VNF deployment decision time of ILP and ML with the varying number of concurrent service requests.
  3. In the testbed environment, we deployed VNFs according to ILP and ML decisions and then generated HTTP traffic to measure the end-to-end latency of each service request and compare them between ILP and ML cases.



Evaluation

Evaluation Criteria

- An Example of VNF deployment output for 4 Servers and 4 VNF Types is as below: ILP solution (left), predicted result (right)
- On the ILP solution and the predicted result, ‘1’ means deployed and ‘0’ means not deployed.
- The accuracy is defined as the proportion of total matches for each position of the label matrix.



VNF types

Servers

0	1	0	1
1	1	0	0
0	0	1	0
0	0	0	1

ILP solution

VNF types

Servers

0	1	0	1
1	0	0	0
0	1	1	0
0	0	0	1

Predicted result

Fig. 13 : An Example of VNF deployment output for 4 Servers and 4 VNF Types is as below: ILP solution (left), predicted result (right)

Results

Accuracy of the Model

- Mean values of accuracy from three models for each prediction horizon time dataset.
  - Six datasets with the prediction horizon ranging from 50 seconds to 300 seconds with the interval of 50 seconds have been generated.

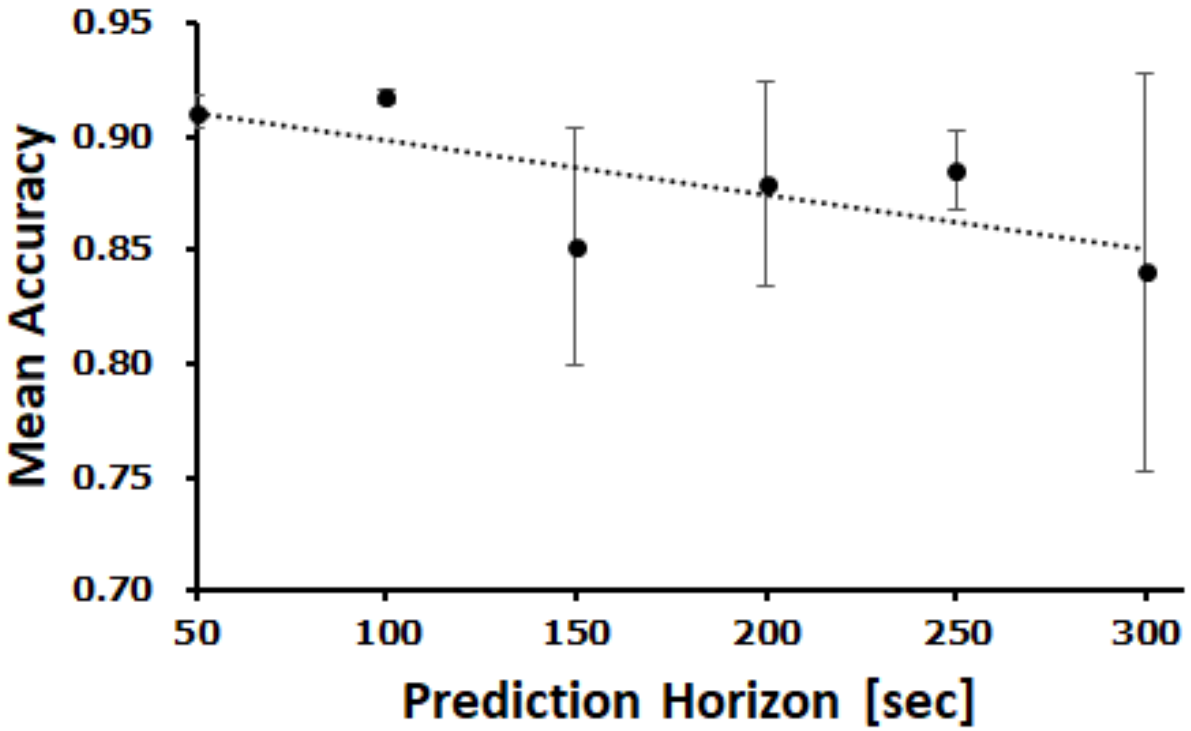


Fig. 14 : Impact of the Prediction Horizon on the Accuracy of Predicting Deployment

Results

- **Difference in VNF Deployment Decision Time between ILP and ML**
  - To compare the VNF deployment decision time of ILP and ML, the execution times are measured for each case with 1 to 5 concurrent service requests as the input.

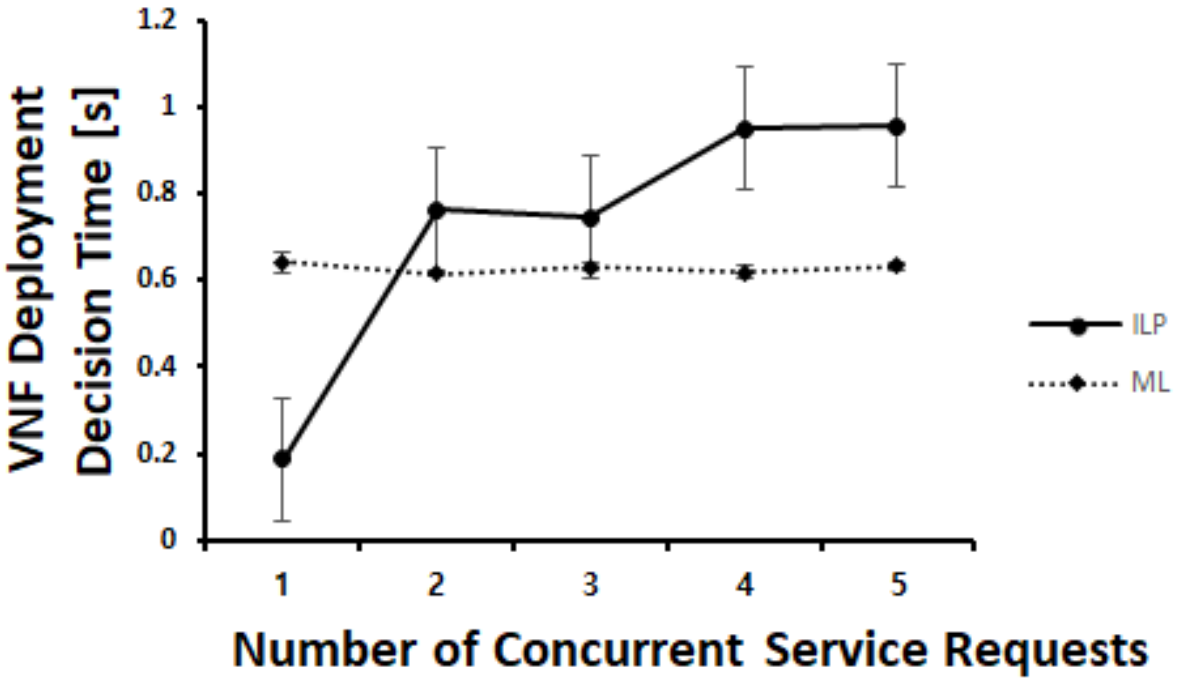


Fig. 15 : Difference in VNF Deployment Decision Time between ILP and ML according to the Number of Concurrent Service Requests

Results

- **Difference in VNF Processing Time between ILP and ML**
  - The deployment decisions with ILP or ML for arbitrarily generated service requests in the testbed are made.
  - HTTP traffics through the SFC path corresponding to the source, destination, and deployment are generated.

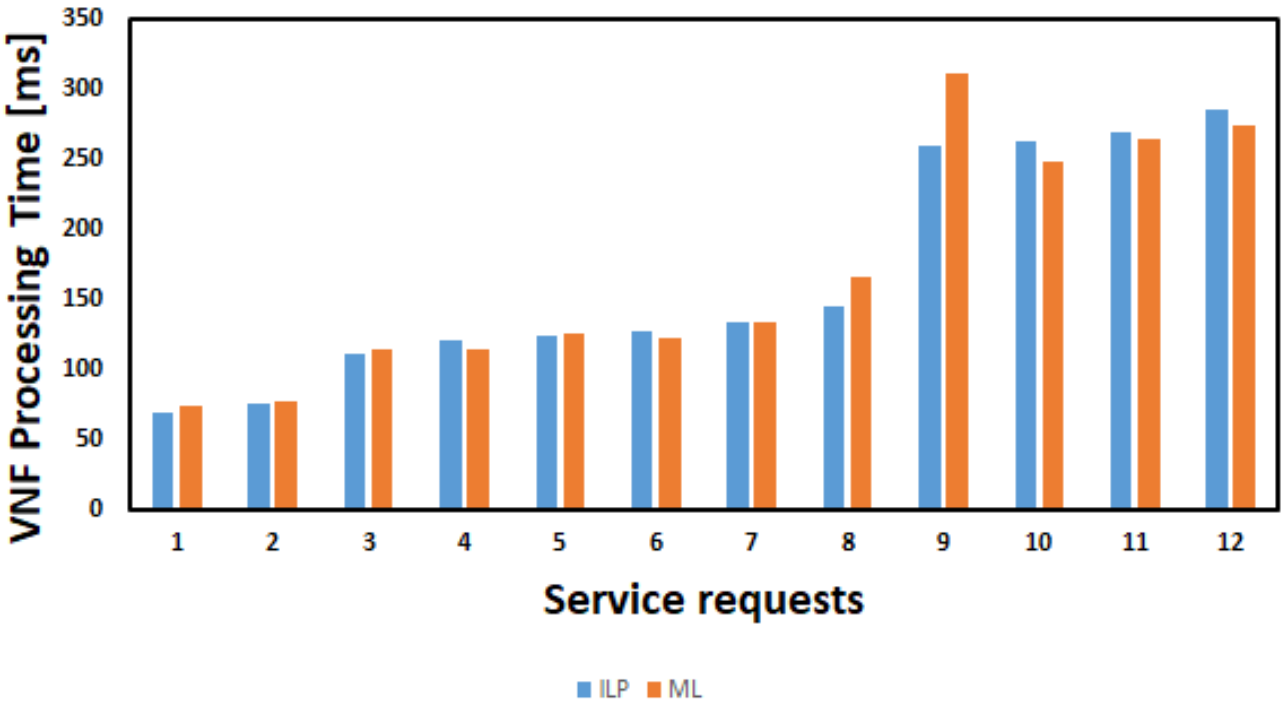


Fig. 16 : Difference in VNF Processing Time between ILP and ML for each Service Request

Discussion

- **Strength of this study**
  - We present a model that can determine the optimal VNF deployment at a time point after 5 minutes to the exact location of the server for each VNF type.
  - We apply this study to the real NFV environment testbed.

- **Weakness of this study**
  - The limitation of this study is that the model has been trained and verified only with simulation data.
  - It is difficult to judge the reliability of the model by the overall accuracy of the model as seen in the confusion matrix.

Summary and Future Work

Summary and Future Work

- **Summary**
  - We suggested the process of developing a machine learning model for VNF deployment.
  - The accuracy of the optimal VNF deployment model compared to the ILP solution achieved more than 84% for the 5-minute future time point.
  - We verified the performance of the model in the testbed by comparing the outcome VNF processing time when our model made the VNF deployment decision to that of ILP solution deployment.

- **Future Work**
  - Adding the source and destination node's information to the input data.
  - Applying Recurrent Neural Network (RNN) algorithm to the model for the prediction using time-series data.
  - Extending the prediction horizon over 5 minutes
  - Improving the accuracy of the model over 90% by fine-tuning the parameters

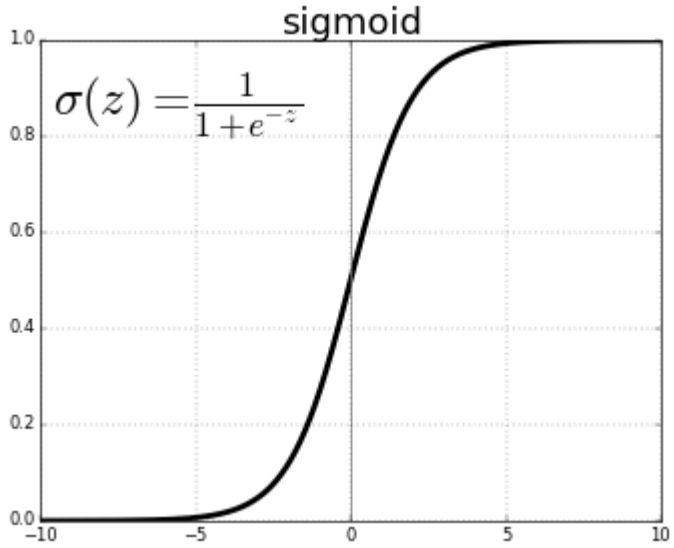
Q&A

감사합니다

Appendix 1. Activation Function

- Sigmoid

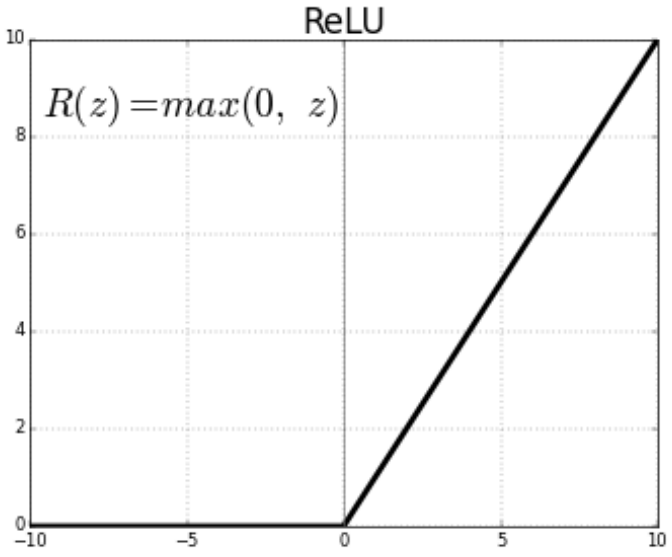
$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}.$$



⇒To make the output as a number between 0 and 1

- ReLU (Rectified Linear Unit)

$$f(x) = \max(0, x)$$

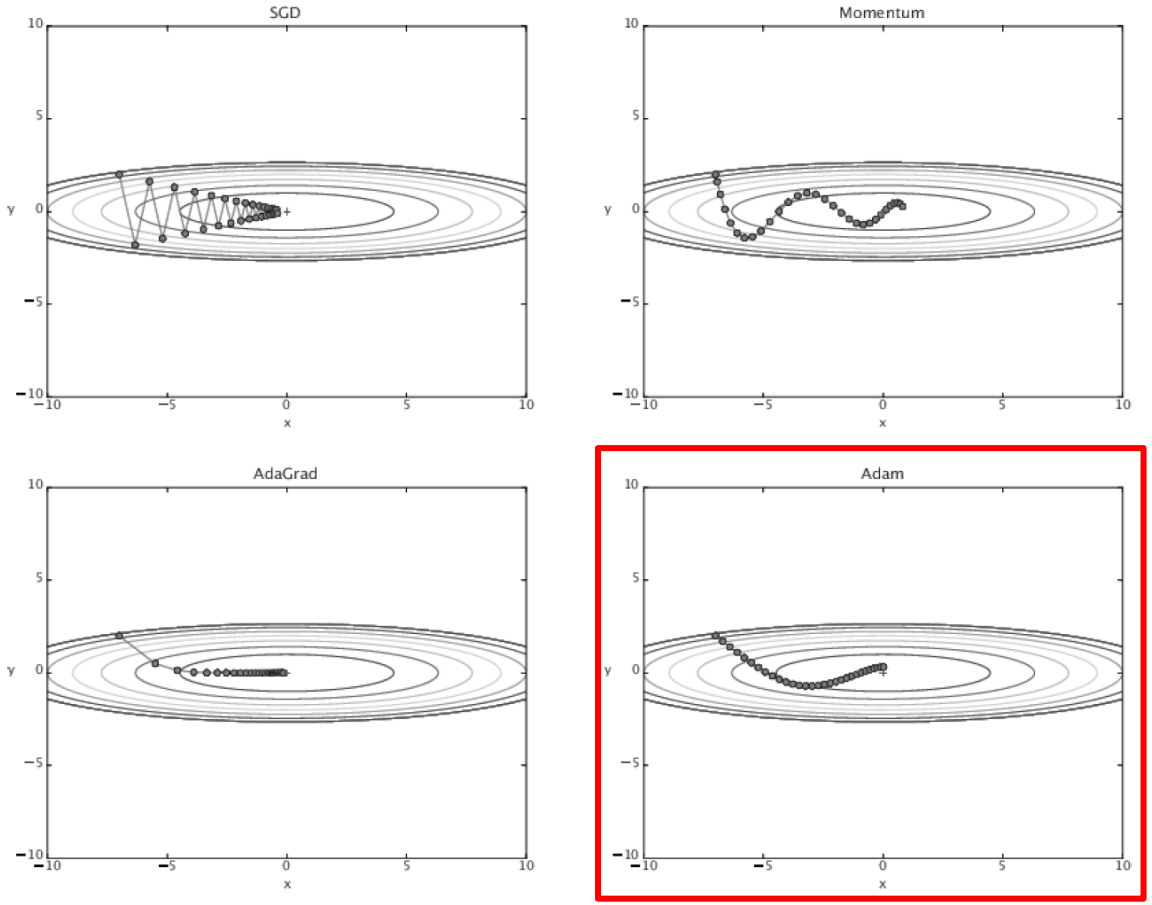


⇒To fix the Vanishing Gradients Problem

Appendix 2. Optimizer

- Adam

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial L}{\partial \mathbf{W}}$$



$$\mathbf{v} \leftarrow \alpha \mathbf{v} - \eta \frac{\partial L}{\partial \mathbf{W}}$$

$$\mathbf{W} \leftarrow \mathbf{W} + \mathbf{v}$$

$$\mathbf{h} \leftarrow \mathbf{h} + \frac{\partial L}{\partial \mathbf{W}} \odot \frac{\partial L}{\partial \mathbf{W}}$$

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{1}{\sqrt{\mathbf{h}}} \frac{\partial L}{\partial \mathbf{W}}$$

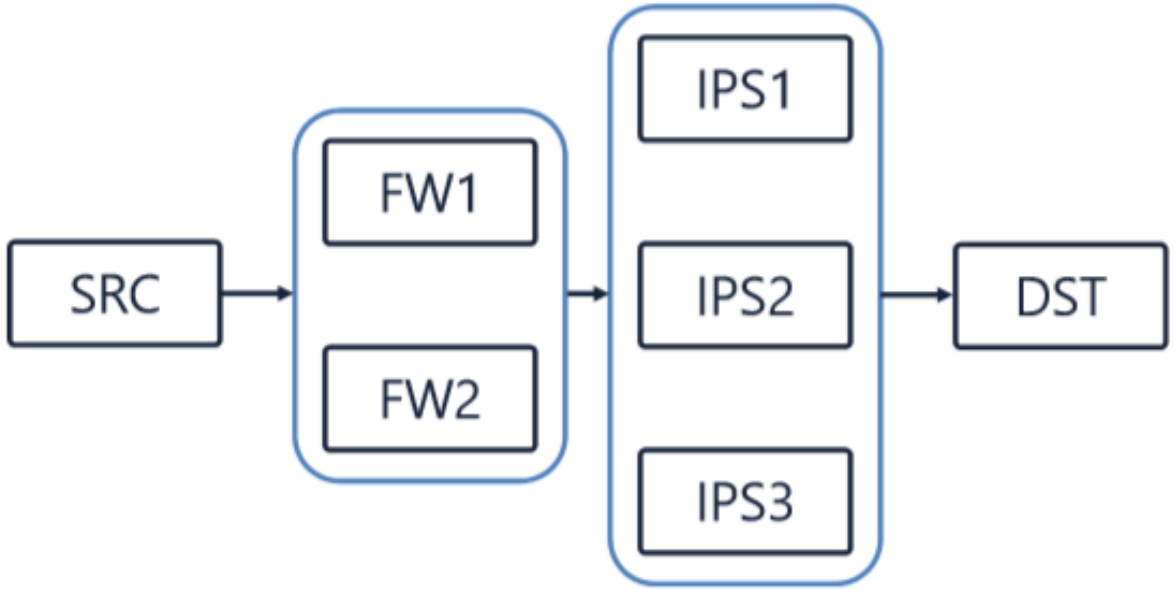
Momentum + AdaGrad ⇒Adam

⇒To efficiently converge to the optimal point

Appendix 3. SFC for VNF Deployment Module

- ML-based VNF Deployment module
  - ML model doesn't output the SFC result.

  ⇒ For the testbed experiment, VNF instances of the same type have been grouped so that the traffic can be load balanced.



Appendix 4. Experimental Environment

장비명	규격(ENV_SPEC)	수량	비고
OpenStack 노드(Node)	**Controller Node** (141.223.82.80) 모델: IBM System x3550 M4 ・CPU 1: Intel Xeon CPU E5-2690 2.90GHz (8Core) ・CPU 2: Intel Xeon CPU E5-2690 2.90GHz (8Core) ・Memory: 32GB (4x8GB) DDR3 ・Disk: HDD 1TB ・OS: Ubuntu 16.04.7 LTS / SW: OpenStack Rocky	1	-
	Compute Node #1~2 (141.223.82.81~82) 모델: Dell PowerEdge R430 ・CPU1: Intel Xeon CPU E5-2630v3 2.40GHz (8Core) ・CPU2: Intel Xeon CPU E5-2630v3 2.40GHz (8Core) ・Memory: 32GB (4x8GB) DDR4 ・Disk: HDD 600GB ・OS: Ubuntu 16.04.7 LTS	2	-
	Compute Node #3~4 (141.223.82.48~49) 모델: Dell PowerEdge R610 ・CPU1: Intel Xeon CPU X5650 2.67GHz (6Core) ・CPU2: Intel Xeon CPU X5650 2.67GHz (6Core) ・Memory: 24GB (6x4GB) DDR3 ・Disk: HDD 3TB ・OS: Ubuntu 16.04.6 LTS	2	-

Appendix 4. Experimental Environment

장비명	규격(ENV_SPEC)	수량	비고
OpenStack 노드(Node)	**Compute Node #5** (141.223.82.55) 모델: Dell PowerEdge R720 ・CPU1: Intel Xeon CPU E5-2670 2.60GHz (8Core) ・CPU2: Intel Xeon CPU E5-2670 2.60GHz (8Core) ・Memory: 96GB (12x8GB) DDR3 ・Disk: HDD 2TB ・OS: Ubuntu 16.04.7 LTS	1	-
	Compute Node #6~8 (141.223.82.91~93) 모델: Dell PowerEdge R610 ・CPU: Intel Xeon E5640 2.67GHz (4Core) ・Memory: 12GB (3x4GB) DDR3 ・Disk: HDD 1TB ・OS: Ubuntu 16.04.7 LTS	3	-