A summation of...

Machine Learning Driven Scaling and Placement of VNF at the Network Edges

PART II
Integer Linear Programming(ILP) + Formulation

Latency Optimal VNF Placement Problem in MEC-NFV Environment

1. System Modeling
2. Problem Formulation
3. ILP Model Evaluation

Recall in this paper...

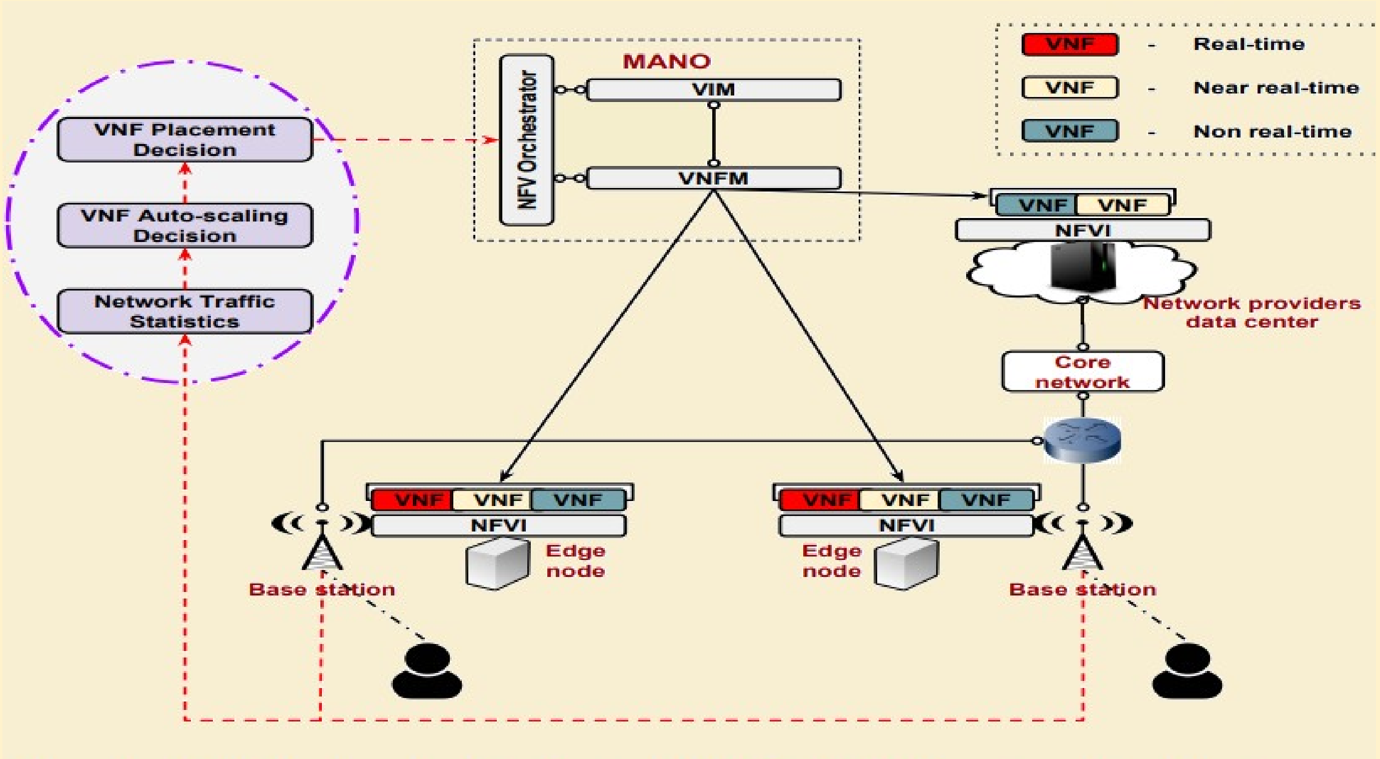
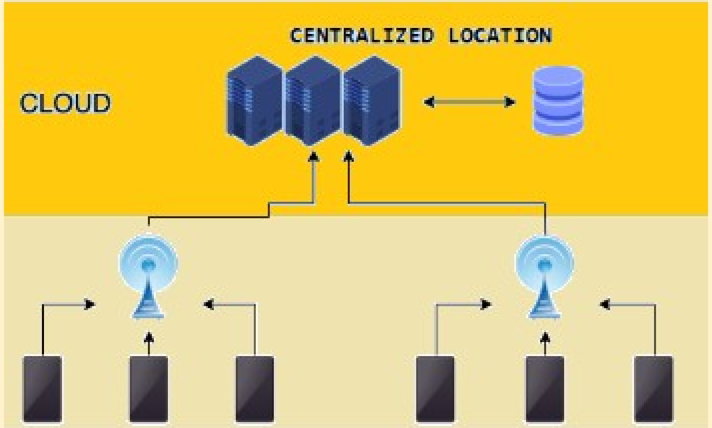


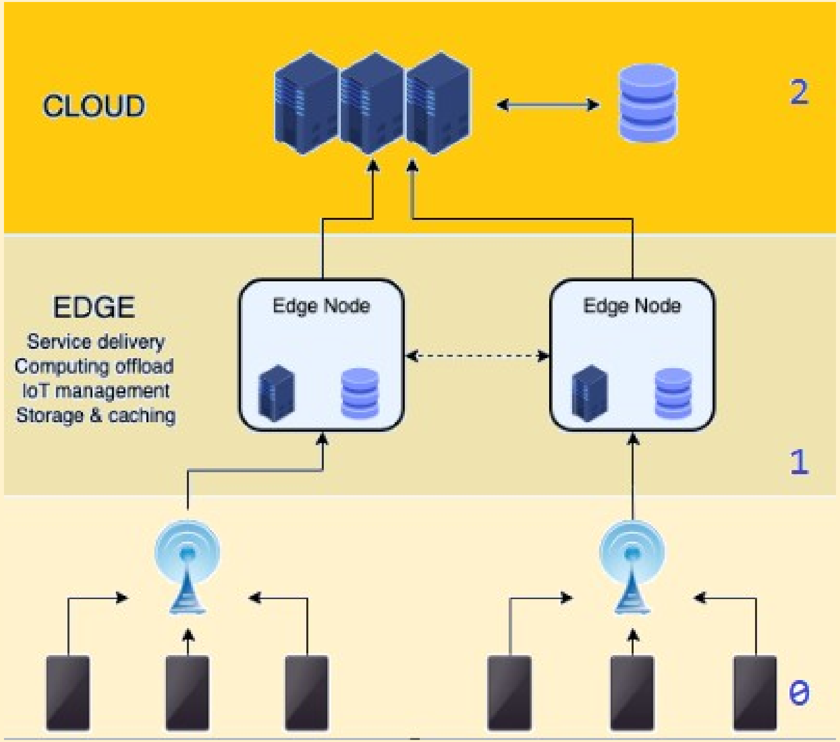
Fig. 1: A high-level distributed MEC-NFV System Architecture.

Brief Breakdown of MEC-NFV

BEFORE



AFTER



Integer Linear Programming

PROs

- Versatility in modeling ‘real life’ applications
- Improves modeling capabilities
- Provides logical thinking
- Ability to assist in making adjustments to changing conditions
- Great for optimization(maximize or minimize)

CONs

- Inability to process data that cannot be quantifiable
- Not easy to model or solve in certain cases

Fun Fact! One of many machine learning techniques!

General Process in Forming Integer Programs

1. **Select** set of decision variables
    a. Have to be integers .. hence ILP
    b. Unknowns of the selected mathematical model
2. **State** Objective
    a. Optimization(maximize or minimize) something
3. **List** out constraints
    a. Parameters you cannot alter the edge cases for
        i. Negative functions, unrealistically large numbers
    b. Slide 30 for complete list

System Modeling

Goal: **Minimize** end to end latency by:

- placing VNFs on edge devices closes to end users
- Once VNFs run out of capacity *then* fall back to VNFs in the providers cloud data center

Table: defines all parameters used in the formulation

Notation	Definition
$G = (N, E, Z)$	Graph of the NFVI.
$N = \{n_1, n_2, ..., n_i\}$	Set of physical nodes (edge and distant cloud) within the network.
$E = \{e_1, e_2, ..., e_l\}$	Set of physical links in the network.
$Z = \{z_1, z_2, ...z_q\}$	Set of users associated with VNFs.
θ^i	Hardware capacity (CPU, memory, network) of the physical node $n_i \in N$.
δ^l	Capacity of the physical link $e_l \in E$.
d^l	Latency on the physical link $e_l \in E$.
$V = \{v_1^1, v_2^2, ..., v_j^q\}$	VNFs associated to users (e.g. $v_j^q \in V$ is associated to user $z_q \in Z$).
$P = \{p_1, p_2, ..., p_k\}$	All paths in the network.
ψ^j	Required capacity (CPU, memory, network) of the physical node to host VNF $v_j \in V$.
d^j_{max}	Maximum end-to-end latency threshold VNF $v_j \in V$ tolerates from its user.
X_{ijk}	Binary variable denoting if VNF $v_j \in V$ is hosted by physical node $n_i \in N$ using path $p_k \in P$.
b_{ijk}	Required bandwidth between VNF $v_j \in V$ to the user, if the VNF is hosted by physical node $n_i \in N$ using path $p_k \in P$.
d_{ijk}	Required latency between VNF $v_j \in V$ to the user, if the VNF is hosted by physical node $n_i \in N$ using path $p_k \in P$.

TABLE V: Key notations in our model.

System Modeling – Important things to Note

- Each VNF has its own: CPU, Memory, and Network requirements
- VNF has an end to end delay threshold ($\mathbf{d^j}$) **AND** specifies a bandwidth requirement
- Latency from a user to a VNF($\mathbf{d_{ijk}}$)
- Decision variable($\mathbf{X_{ijk}}$) binary variable where 1 assign v_j to node n_i using path p_k

Problem Formulation

ILP model that takes in the following as **input**

-Set of users(U)

-Set VNFs (V)

-Set of VNFs hosts(N)

-Latency Array (d)

Then **outputs** optimal solution for VNF placement by minimizing the total end to end latency from all users

<u>Formulated Objective Function</u>

$$ILP : minimize \sum_{n_i \in N} \sum_{v_j \in V} \sum_{p_k \in P} X_{ijk}.d_{ijk}$$

Problem Formulation

Constraints of Optimization Objective: all help ensure the following:

$$\sum_{v_j^q \in V} \sum_{p_k \in P} X_{ijk}.\psi^j < \theta^i, \forall n_i \in N$$

$$\sum_{n_i \in N} \sum_{p_k \in P} X_{ijk}.d_{ijk} < d_{max}^j . \forall v_j^q \in V$$

$$\sum_{n_i \in N} X_{ijk} = 1, \forall v_j^q \in V, \forall p_k \in P$$

$$\sum_{n_i \in N} X_{ijk}.b_{ijk} < \delta^l, \forall e_l \in p_k, \forall p_k \in P$$

Constraint 9: ensures amount of hardware resources allocated to VNFs is within the available resources on the physical node

Constraint 10: end to end delay between user and VNF doesn’t exceed the max delay

Constraint 11: each VNF is hosted by exactly one physical node

Constraint 12: none of the physical links becomes overloaded

ILP Model Evaluation

Model was evaluated using simulation experiments

Simulation Environment: Based on backbone network by a private Mobile Network Operator

- Edge nodes at all base stations and capable of hosting finite number of VNFs
- 1 Cloud data center capable of hosting several VNFs

ILP Model Evaluation

VNFs were categorized into 3 categories depending on latency tolerance levels for service

1. Real Time
2. Near Time
3. Non-real Time

Used **equal** number of VNFs in all 3 categories

Generic applications	Expected latency
Real-time (e.g., Virtual Reality)	$< 5ms$
Near real-time (e.g., Video conference call)	$< 20ms$
Non real-time (e.g., Video streaming)	$< 100ms$

TABLE VI: Latency requirements for generic applications.

ILP Model Evaluation

Using **IBM ILOG CPLEX:**

- 1st scenario: all VNFs are assigned to cloud data center
- 2nd scenario: VNFs assigned to edge nodes first, then to cloud data center once capacity runs out

Had a **fixed** latency of 5ms from user to edge nodes

Number of VNF hosted on each node = 40

Total edge capacity of network = 240 VNFS

-Once over 240; automatically gets assigned to cloud data center

ILP Model Evaluation

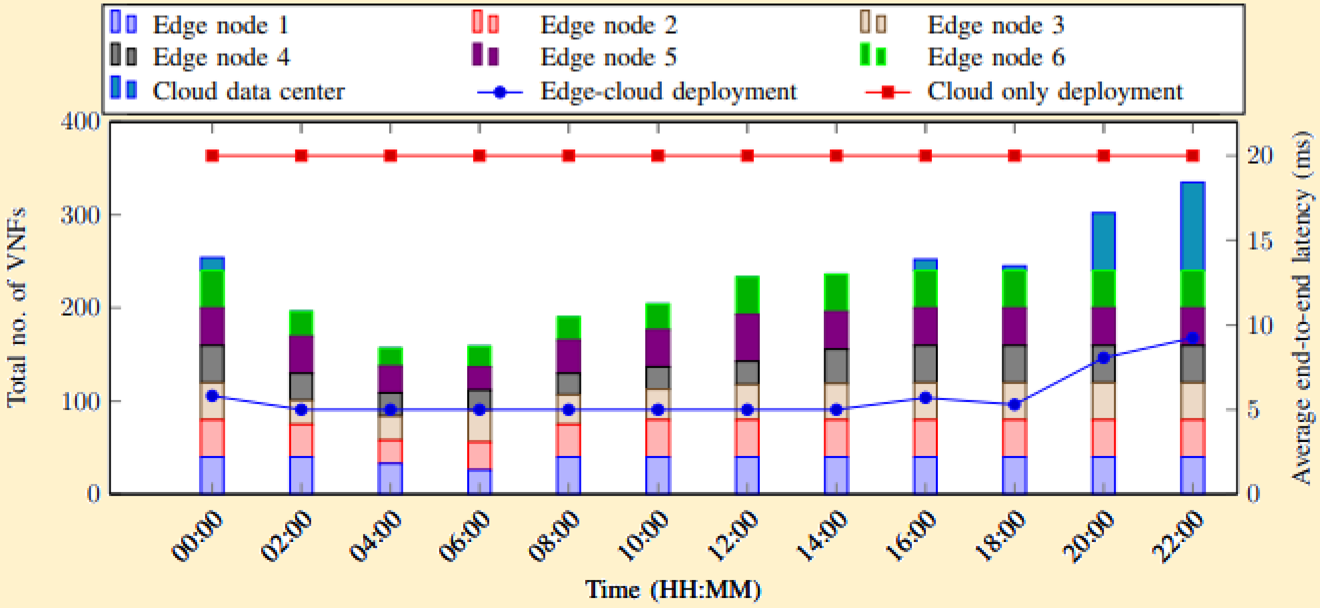
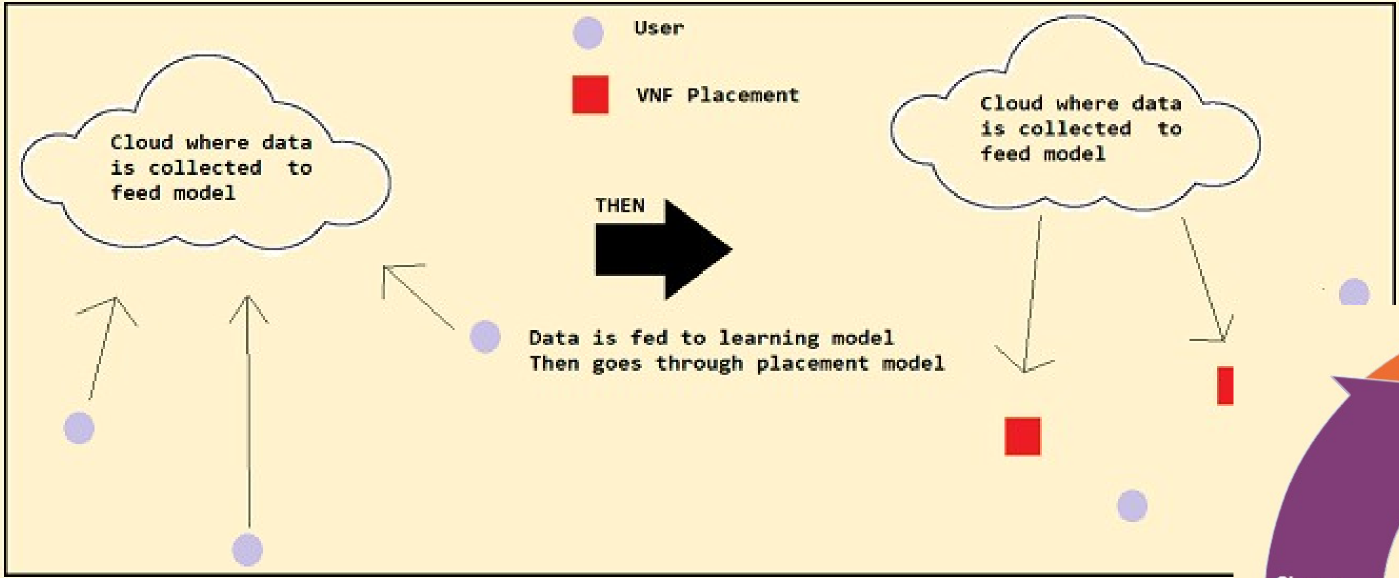


Fig. 5: Performance measure of the proposed system model.

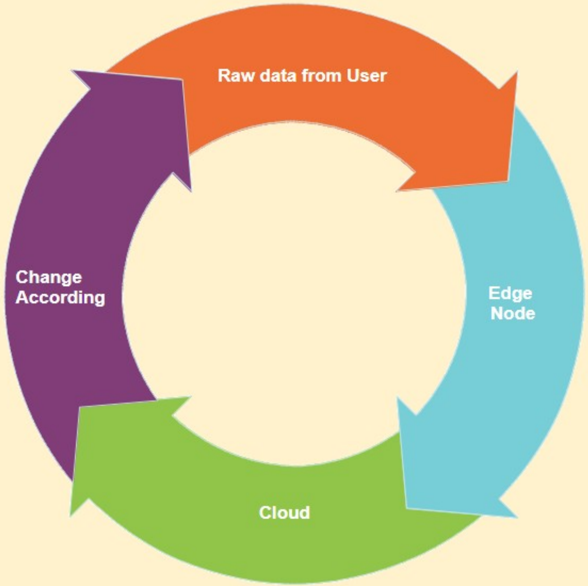
ILP model took 6.25 seconds to place 335 VNFs to help minimize aggregated user to VNF end to end latency

RED: Cloud only deployment; **fixed** latency of ***20ms***
BLUE: Edge + Cloud; **lower** latency times(avg: **5ms**) increased when the edge nodes were exhausted

A Room to Improve...



A proposition for federated learning...



Main weakness for their model is that it’s still dependent on the data being forwarded over to a centralized location to do their ‘learning’

IN CONCLUSION..

- MLP was the most effective model in predicting amount of VNFs to deploy
  - Beneficial in **proactive** auto scaling
  - Helped minimize downtime and reduce operational costs
- Proposed a optimal placement model that carefully selects where to place VNFs to reduce user -> VNF latency
  - Results averaged 75% reduction in end to end latency when *all* VNFs were placed at the network edges
- Future work potentially with federated learning

Citation

T. Subramanya and R. Riggio, "Machine Learning-Driven Scaling and Placement of Virtual Network Functions at the Network Edges," *2019 IEEE Conference on Network Softwarization (NetSoft)*, Paris, France, 2019, pp. 414-422.

doi: 10.1109/NETSOFT.2019.8806631

keywords: {integer programming;learning (artificial intelligence);linear programming;neural nets;telecommunication traffic;virtualisation;virtual network functions;network operators;migrate VNFs;network edges;efficient VNF placement;continuously changing network dynamics;neural-network model;network traffic;commercial mobile network;Network Function Virtualization;machine learning-driven scaling;Predictive models;Cloud computing;Real-time systems;Data centers;Load modeling;Measurement;Base stations;Network Function Virtualization;Machine learning;Proactive Auto-scaling;Virtual Network Function Placement;Multi-access Edge Computing},

URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8806631&isnumber=8806619