FIPAM: Fuzzy Inference based Placement and Migration Approach for NFV-based IoTs

Tariq, Muhammad Arslan; Farooq, Muhammad Umar; Zeeshan, Muhammad; Hassan, Ali; Akhunzada, Adnan

Published in:
IEEE Transactions on Network and Service Management

Link to article, DOI:
10.1109/TNSM.2022.3145658

Publication date:
2023

Document Version
Peer reviewed version

FIPAM: Fuzzy Inference based Placement and Migration Approach for NFV-based IoTs

Muhammad Arslan Tariq, Muhammad Umar Farooq, *Member, IEEE,* Muhammad Zeeshan, *Member, IEEE,* Ali Hassan, and Adnan Akhunzada, *Senior Member, IEEE*

***Abstract*—The advancement and spread of the internet-of-things (IoT) have massively been increased over a decade. With the widespread of IoT networks, it is becoming difficult to acquire and execute real-time data. Network function virtualization (NFV) provides a flexible and efficient solution for IoT-based applications and service management. NFV creates a virtualized environment that can run a large number of micro-services for different IoT applications by using the virtual network functions (VNFs) through placement and chaining. In this paper, we propose a novel fuzzy inference-based placement and migration (FIPAM) approach for placement and migration/chaining of VNFs to ensure that resource allocation is carefully carried out during VNF orchestration and embedding. Firstly, we formulate the VNF chaining and placement problem. Secondly, we propose a lightweight VNF placement solution that considers the underlying network conditions while making the placement decisions. A novel usage of fuzzy inference is proposed to optimize the chaining mechanism along with the dynamic instantiation of VNFs to meet specific service needs. Simulation results are shown to validate the superiority of the proposed algorithm over existing schemes.**

***Index Terms*—IoT, Network Function Virtualization, Fuzzy Inference Systems**

I. Introduction

THE internet of things (IoT) is becoming a reality with the exponential growth of information and communication technology. Computers, sensors, actuators, and software systems with communication capacity can be the "things" in the IoT. These connected things bring remarkable transformation by embedding data acquisition, contextual awareness, and actuation in every domain of life [1], [2], [3]. This unprecedented human-to-device and device-to-device interaction is continuously bringing holistic improvements in user experience. As a result, IoT-dependent services are multiplying every year. With the number of IoT devices expected to reach 25 billion by 2025, IoT applications are likely to claim a significant internet traffic share [4]. Besides, the rise in applications involving data analytics, artificial intelligence, and machine learning is encouraging a massive digitization roll-out. The need is so immense that many countries have already formulated their AI policy frameworks and have started massive digital transformation drives ranging from healthcare to industrial IoTs. With more and more IoT technologies emerging, it is becoming increasingly difficult for conventional network frameworks to match the diversity of IoT traffic specifications. These ubiquitous and autonomous networks of mainly heterogeneous devices require distinctiveness in many network services [5]. For instance, security and privacy needs are different for healthcare, precision farming, home appliances, etc. Similarly, the mobility of devices for intelligent transport systems and industrial IoTs require frequent adjustment of forwarding rules. Traditionally, different services like network address translation (NAT), authorization, and routing for IoT applications run over dedicated hardware appliances. As a result, it has become challenging to manage service policies and security parameters for IoT applications. The impact is not just increased capital/operational expenditures. These operational hindrances also limit innovation and increase time-to-market. The solution is to forfeit proprietary hardware. To avoid these challenges and offer separate hardware infrastructure for every IoT application to run, network function virtualization (NFV) [6] provides a new way to create, operate, and distribute networking services. It is a simple way to decouple different network functions from various hardware appliances. Those decoupled functions can run over commodity hardware or dedicated cloud infrastructure. Thus, the main aim of NFV is to replace conventional hardware-based network devices with the virtualized network functions (VNF). The primary motive for such transformation is to offer flexible services, and reduce costs and time-to-market.

The need for NFV is so immense that European Telecommunication Standards Institute (ETSI) has formed an industry specifications workgroup [7]. Several large telecom operators have already joined the workgroup to define general standards for orchestration, interfacing, and management of VNFs. The flexible interfacing and sequential ordering of VNFs can noticeably accelerate the offering of innovative and customized network services. Generally, it requires two stages: (1) Placement, and (2) Chaining [8]. The placement and chaining of VNFs involve interconnecting many system capacities (e.g., firewall, load balancer, cache servers) to guarantee that system streams are given the right treatment. These streams must experience a start-to-finish sequence of navigating a particular set of capacities. In placement, we decide how many network functions (and their instances) are to be orchestrated to complete the service requirements efficiently. It also involves finding the right spots for these functions. For instance, having a firewall service entertained at multiple points can help minimize latency and add to the network's performance. Usually, VNFs are best suited

M. A. Tariq, M. U. Farooq, M. Zeeshan, and A. Hassan are with the College of Electrical & Mechanical Engineering, National University of Sciences and Technology, Islamabad, 44000, Pakistan

A. Akhunzada is with DTU Compute, Technical University of Denmark, Kongens Lyngby, Denmark

Manuscript received April XX, 20XX; revised August XX, 20XX.

when placed at the network's points of presence (N-PoPs) [8] to efficiently manage the traffic flows and create the ideal environment for uninterrupted IoT services.

In the chaining phase, we create the process and service flows through interconnected VNFs throughout the system. Suppose we are running different IoT services simultaneously within the same network. Forwarding the traffic through the same paths (VNFs) may create latency and congestion issues over the network. However, with the help of chaining, we can make different network paths among multiple instances of the same set of VNFs according to their requirements to overcome such issues, i.e., one service requires user authentication. In contrast, the other one is fetching data from the server. Then we can choose different data paths according to their needs. Effective use of chaining involves flow-based traffic engineering empowered by software-defined networking (SDN) [8], that separates the control and data planes for better data engineering.

Modern IoT applications usually require micro-services. Since VNFs can also be orchestrated on commodity hardware by using various virtualization technologies, offering a large number of micro-services by chaining VNFs hosted at commodity servers is becoming easier. Such micro-services can be the core of many functional and non-functional requirements, including redundancy, and latency, etc. The rules for orchestration and chaining of such VNFs are similar. However, they significantly increase the complexity of the placement problem. Therefore, the need for placement and chaining of a large number of in-network VNFs in ultra-dense IoT networks needs to be investigated. Since the VNF placement and chaining problems are proved to be NP-hard [9], different heuristic/greedy approaches are usually applied to find closer approximations to the optimal solution. Furthermore, such approximations usually ignore parameters like traffic/computational load distribution and maximal bandwidth utilization. At the same time, points of interests for different IoT applications may change frequently, requiring re-adjustment (migration) of VNFs at appropriate location.

To meet these challenges, a fuzzy inference-based placement and migration (FIPAM) approach is proposed in this paper. The major contributions of this paper are as follow:

1) We first formulate the VNF placement as an optimization problem with an aim to maximize network utilization and minimize the number of congested links by maintaining balanced distribution of compute load, bandwidth, and latency.
2) To derive the solution for the placement problem, we first propose an algorithm to find the most appropriate locations to achieve near-optimal VNF placement. The proposed scheme summarizes latency, bandwidth, and compute capacity in cost matrix, and performance weight vector. Based on this information, the near-optimal placements are achieved by using the proposed shortest path algorithm.
3) Since, the proposed shortest path algorithm is computationally extensive, therefore, a fuzzy-inference based solution is proposed that achieves better distribution in terms of computational load, latency and bandwidth. This approach efficiently reduces the network convergence time, while showing superiority to the existing chaining solutions.
4) In contrast to the existing greedy/heuristics based placement and chaining solutions, the proposed lightweight FIPAM algorithm incorporates additional parameters like traffic/computational load distribution, and maximal bandwidth utilization etc, making it unsuitable for ultra-dense IoT networks.

II. Related Work

The VNF orchestration and chaining problem has received considerable attention during recent years. Most researchers have used several optimization techniques for solving this NP-Hard problem, mainly focusing on computation and communication cost [10], [9]. However, network dynamics and service function needs are highly variable in IoT-based large-scale networks. Therefore, latency and reconstruction costs are equally essential [11], [12].

Recently, several literature studies [13], [14], [15] have focused on various NFV challenges, including VNFs orchestration and management. These algorithms propose various models for the optimization and heuristic solutions for optimal or near-optimal placement of VNFs. Some other works have addressed the development of platforms for NFV management [16], [17]. In [11], authors employ admission control to satisfy resource management objective during SFC embedding and make use of MILP with relaxations and reformations to achieve their goal. In [18], VNFs placement and deployment problem in a mobile edge computing infrastructure is investigated to achieve the conflicting goal of latency, deployment cost, and performance.

In [19], a framework of dynamical service chaining in the software-defined NFV system is presented. In this framework, the role of SDN and NFV is to enable high efficiency and flexibility in the construction of service chaining. It includes the flows of steering through the required service chain. In this research, an optimization framework and a unified control are elaborated and explained for enabling the SDN-NFV framework that is utilized for the optimization of the service chaining according to the requirements of the user and the network environment. An idea of utilizing a net virtualization machine (NetVM) is proposed in [20]. NetVM carries virtualization to the network by empowering high transfer speed system capacities to work at close line speed while exploiting the ease item servers' adaptability and customization. NetVM operates the network function by this technique while gaining the flexibility and slow pace of the low-cost commodity servers. In [21], an enterprise security architecture is proposed. The authors placed firewall as virtualized network function at multiple cloud locations. This solved the locality problem according to changing network needs to protect enterprise networks from external and internal threats.

A group mapping scheme to satisfy network service requirements is proposed in [22]. The suggested scheme is based on dependency perception and adaptive mapping to save the network's computational and bandwidth resources. A

heuristic-based algorithm to solve a MILP problem is proposed in [23] to find a near-optimal solution. The algorithm first finds optimal first-hop scheduling and used a greedy approach to complete the SFC length. In [24], VNF mapping and scheduling scheme for space-air-ground integrated networks (SAGIN) is proposed. Authors have considered the dynamic nature of internet-of-vehicles (IoV) services. VNF mapping and scheduling are formulated as a MILP problem with specific delay and cost models. Two algorithms based on Tabu search are proposed, i.e., TS-based VNF remapping and rescheduling (TS-MAPSCH) algorithm, and TS-based pure VNF rescheduling (TS-PSCH) algorithm. A joint routing and placement (JORP) algorithm for IoT service chain is proposed in [12]. This algorithm is capable of dynamically scaling the number of VNF instances. To solve the formulated JORP, a deep learning approach is proposed that impersonates the branching, and mitigates the unlikely solutions.

An algorithm for VNF placement with the aim to maximize the number of accepted SFC requests is proposed in [25]. The proposed algorithm guarantees the network latency requirements. The optimization problem is solved by using delay guarantee heuristic approach. An SFC embedding problem is considered in [26] for multi-domain networks where there is no knowledge regarding the network topology and resources. A column generation method is proposed to solve the optimization problem. It is shown that the acceptance ratio is close to that of the optimal algorithm. A scalable SFC chaining algorithm is proposed in [27] for NFV-enabled networks through federated reinforcement learning technique. In [28], an intent-based networking (IBN) protocol is combined with NFV to achieve efficient VNF placement in a cloud-based infrastructure. The aim is to automatically configure network services according to the service quality and security requirements. With the aim to decrease the servers, a semi online VFN placement (SIVA) algorithm is proposed in [29]. This algorithm is based on bin-packing algorithm while taking care of migrations. Extensive simulation and experimental results show the superiority of the SIVA algorithm over existing state-of-the-art. In [30], the problem of SFC placement and chaining in NFV-enabled networks is modeled as capacitated shortest path tour problem (CSPTP) based on integer linear programming. The efficiency of the proposed scheme is analyzed in terms of the standard deviation in latency and link utilization.

For the placement of VNFs in the IoT network, an important challenge is to investigate the latency issue [31]. In [32], authors have proposed a resource orchestration for vertical industries of 5G transport networks. They have applied the placement model into the edge cloud model. In the content delivery network (CDN), an optimal solution in a multi-cloud scenario is derived in [10] after studying the VNF placement model. In contrast to the conventional cloud-based environment, one of the major concerns is that uncertain loads of IoT devices and propagation latency directly affect the VNF placement problem [33], [34].

After extensive literature review, one or more of the following limitations have been found, to the best of our knowledge; (1) large number of in-network VNF placement and chaining problem has not been explored, (2) existing greedy/heuristic approaches are computationally extensive and/or often ignore parameters like traffic/computational load distribution, and maximal bandwidth utilization etc, making them unsuitable for ultra-dense IoT networks, and (3) the VNFs are usually placed at nearest/high-capacity compute nodes without taking into account their corresponding link bandwidths that eventually results in congested links, and (4) lack of re-adjustment (migration) mechanisms for VNF instances to move network services at appropriate locations according to service needs. In this work, we address the VNF placement, migration, and chaining problem for large-scale IoT networks to overcome the above mentioned limitations.

III. System Overview

Application requirements and service provisioning policies define a specific order in which data flows traverse through the network, usually referred to as a service function chain (SFC). Before proceeding further, the four network functions usually used for enterprise network security are defined below.

- A firewall (FW) is considered a security tool that controls network traffic according to the security rules.
- A deep packet inspection (DPI) function is a packet processing application that digs deeper into IP data payload rather than only looking into protocol headers. Scanning payload helps identify worms like Slammer and Nimda that carry a specific signature (a string of bytes) in the payload.
- An intrusion prevention system (IPS) is a system security framework that checks and recognizes errors. This framework screens the system continuously, watches out for all possible malicious occurrences, and accumulates data accordingly.
- A deception system (DS) is a process that defends attacks from hackers by transmitting dummy or any random data to them. It works to identify the hacker's signature to block their access to the system.

To understand the VNF placement/migration and service chaining problem, we use a miniature IoT network presented in figure 1. We consider four VNFs namely firewall, DPI, IPS and DS to build a hypothetical security framework. Here, different security checks are possible for different applications/users. For instance, all traffic flows are passed through a firewall. Trusted traffic flows are immediately granted service access, whereas suspected traffic flows are passed through a DPI module to further investigate traffic patterns. Any malicious traffic flows are then passed to an IPS that may decide on the basis of certain parameters to log and discard the traffic or pass to a deception system for further traffic analysis (zero-day attack identification). The above mentioned mechanism divides the traffic flows in different service chains, i.e., $n_i \rightarrow FW \rightarrow n_e$, $n_i \rightarrow FW \rightarrow DPI \rightarrow n_e$, and $n_i \rightarrow FW \rightarrow DPI \rightarrow IPS \rightarrow DS$. It is worth mentioning that any VNF instance (e.g. firewall) may be shared between multiple SFCs or a separate instance may be generated for each SFC. Figure 1 shows the placement and chaining of multiple instance of firewall and a single instance of IPS, IDS, and DS, each to form different service paths, i.e., $n_i \rightarrow N1 \rightarrow N4 \rightarrow N8 \rightarrow n_e$,

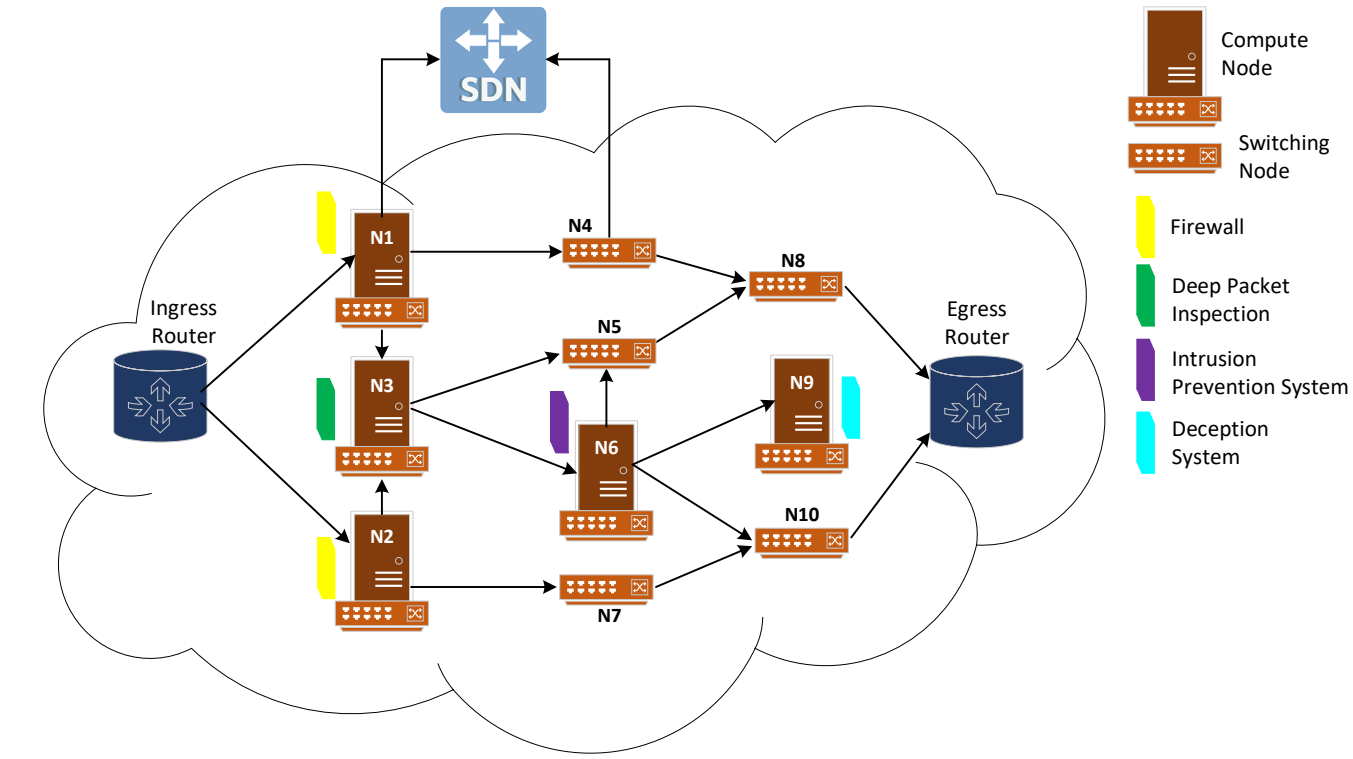

Fig. 1. An example enterprise network with multiple VNFs

$n_i \rightarrow N2 \rightarrow N3 \rightarrow N6 \rightarrow N9$ etc, to offer the network security service with different enforcement levels. Similarly, multiple replications of the same SFC may be triggered to minimize service latency/traffic load in different network segments, i.e. $n_i \rightarrow N1 \rightarrow n_e$, and $n_i \rightarrow N2 \rightarrow n_e$ represents replication of the same SFC in the enterprise network shown in figure 1.

As evident from the above discussion, interconnecting the VNFs fulfilling end-to-end service path is essential for NFV. When a traffic flow arrives, it must be steered through a specific set of VNFs defined by the policy and service provisioning agreements. Therefore, service policy and chaining problems are tied. This procedure makes ways that interconnect different sets of system capacities on the basis of start-to-finish latency, and system capacity etc.

IV. Problem Formulation

Let $G = \{N, E\}$ be the undirected graph, where N and E represent physical nodes and links, respectively. Here, $N = \{N_c \cup N_R\}$ gives the union of compute nodes (for hosting network functions) and switching nodes (for traffic forwarding). Considering the total number of distinct virtual functions be N_v, so that $F = \{f_1, f_2, ..., f_{N_v}\}$ represents the set of virtual functions. Any i^{th} virtual function f_i may comprise of several instances represented by $f_i = \{\lambda_i^1, \lambda_i^2, ..., \lambda_i^{N_{I,i}}\}$, where $N_{I,i}$ is the number of instances of i^{th} virtual function. Only one instance of an VNF can be implemented on a simple compute node. Let $\lambda_i^{j,k}$ represents j^{th} instance of i^{th} VNF running on k^{th} compute node.

TABLE I
Notations

Notation	Description
N	Number of physical nodes
E	Number of links in the network
N_c	Number of compute nodes
N_R	Number of switching nodes
N_v	Number of distinct VNFs
F	Set of virtual functions
f_i	i^{th} virtual function
$N_{I,i}$	Number of instances of the i^{th} VNF
$\lambda_i^{j,k}$	j^{th} instance of i^{th} VNF running on k^{th} compute node
S	Set of M service chains
S_l	l^{th} service chain
C_i	Unused compute capacity of the i^{th} compute node
B_i	Bandwidth of the i^{th} link
Δ_i	Latency of the i^{th} service chain
L_c	Cumulative latency
L_d	Demanded latency
C_d	Demanded compute load
L_0	Latency of a single VNF instance
Θ_0	Compute load of single VNF instance

Also let $S = \{S_1, S_2, ..., S_M\}$ represents M service chains, so that l^{th} service chain S_l is represented with a sequence $< n_i, f_{start}, ..., f_{end}, n_e >$, where n_i and n_e are ingress and egress routers, and f_{start} and f_{end} are the starting and ending virtual function in the service chain S_l. It is worth-mentioning that all virtual functions in between f_{start} and f_{end} are not necessarily part of the service chain. Let C_i, B_j, and Δ_k be the unused compute capacity, available bandwidth, and latency of

i^{th} compute node, j^{th} link, and k^{th} service chain, respectively. In order to efficiently deploy and execute multiple service chains, balanced utilization of computational load $\sum_{i=1}^{N_c} C_i$, network bandwidth $\sum_{j=1}^{E} B_j$, and network latency $\sum_{k=1}^{M} \Delta_k$ is desired. The compute load distribution, bandwidth utilization, and latency distribution are given by

$$D_C = \frac{\min_{i \in N_c} C_i}{\frac{1}{N_c} \sum_{i=1}^{N_c} C_i} \tag{1}$$

$$D_B = \frac{\min_{j \in E} B_j}{\frac{1}{E} \sum_{j=1}^{E} B_j} \tag{2}$$

$$D_L = \frac{\min_{k \in M} \Delta_k}{\frac{1}{M} \sum_{k=1}^{M} \Delta_k} \tag{3}$$

In order to efficiently deploy and execute multiple service chains, we define the cost function as the weighted sum of compute load distribution (D_C), bandwidth utilization (D_B), and latency distribution (D_L). The aim is to maximize network utilization and minimize the number of congested links by maintaining balanced distribution of compute load, bandwidth, and latency. Thus, the optimization problem can be stated as

$$\max J = \omega_1 D_C + \omega_2 D_B + \omega_3 D_L \tag{4}$$

subject to constraints,

$$PU_{i,req} < PU_i$$
$$L_d > \sum_{i=1}^{N_F} L_{0,i}$$
$$B_{i,req} < B_i$$

where ω_1, ω_2, and ω_3 are the weighting factors so that $\omega_1 + \omega_2 + \omega_3 = 1$. Here, the first constraint means that the demanded compute load of i^{th} VNF instance ($PU_{i,req}$) must be less than the available compute capacity of i^{th} VNF instance. The second constraint ensures that the cumulative latency of any service chain (having N_F functions) is less than the demanded latency (L_d). The last constraint ensures that the demanded bandwidth of i^{th} VNF instance ($B_{i,req}$) is less than the available bandwidth at the compute node hosting the i^{th} VNF instance.

V. Proposed FIPAM: Fuzzy Inference-based Placement and Migration

This section presents our proposed FIPAM approach for efficiently placing, assigning, and chaining network functions. This is done in two phases. In the first phase, deployment of VNF instances at different compute nodes is achieved. The deployment (placement) process assures low-cost (latency, bandwidth, etc.) reachability to all deployed network functions. In the second phase, service chains are formed by connecting already deployed VNFs. The chaining process is based on the novel usage of the fuzzy inference approach to solve the optimization problem presented in equation (4). The proposed mechanism efficiently selects VNF instances for service chains. In a large network, the network dynamics and service needs are constantly changing. Therefore, the proposed FIS-based chaining process is mandated to change VNF locations or instantiate new instances according to service needs.

Separating the placement and chaining problem has certain benefits. In placement, a deep insight in to a large number of parameters ranging from computational capacity to service policies is desired. Therefore, a comprehensive optimal placement algorithm is executed at network start-up and after certain events to identify the best locations for hosting the VNFs. On the other hand, the chaining process needs to be executed frequently, e.g. whenever a new service is created/replicated, or changes in network capacity/priorities are observed. Having a large number of VNF instances already placed at ideal locations allows the fuzzy inference system to efficiently chain the VNF instances to meet different service requirements with minimal computational needs that helps scaling the solution to much larger networks.

A. Proposed Placement Process

The proposed placement process deploys a small number of VNF instances at the most accessible locations in the network. Usually, such locations include the network's points of presence (PoPs), ingress routers, or egress routers. The aim is to minimize accumulative service latency and maximize network throughput by utilizing the most appropriate links. The process also ensures that any compute node hosting a certain VNF has sufficient computational capacity to host that VNF. Without loss of generality, performance units (PU) for each compute node are calculated using three different parameters, i.e., CPU, RAM, and GPU, as depicted by (5). More parameters can be added, if required, however we assume that the number of parameters for PU calculation never exceeds a small constant. Here, α, β and γ are aggregation factors. PU indicates a node's total computational capacity. After hosting a VNF instance on a compute node, we re-evaluate available PUs as

$$PU = \alpha \times CPU + \beta \times RAM + \gamma \times GPU \tag{5}$$

For the placement process, all edges are assigned weights based on their delay and throughput. The lower the edge-weight, the better is the service quality offered by that edge. A modified shortest path algorithm is used to populate a cost matrix ($\mathbf{\Lambda}$) having dimensions of $N_c \times N$, as depicted by algorithm 1. Each entry $\mathbf{\Lambda}_{ij}$ indicates the accumulative cost of the best possible path between nodes i and j, where i is necessarily a compute node. The general arrangement of matrix $\mathbf{\Lambda}$ is given by

$$\mathbf{\Lambda} = \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1N} \\ c_{21} & c_{22} & \dots & c_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ c_{N_c 1} & c_{N_c 2} & \dots & c_{N_c N} \end{bmatrix}$$

The accumulated path weight for i^{th} compute node is computed by adding i^{th} row of the cost matrix $\mathbf{\Lambda}$. These

Algorithm 1: Proposed algorithm for VNF placement

Notations:
N: Number of physical nodes
N_c: Number of compute nodes
E: Number of edges
$\mathbf{\Lambda}$: Cost matrix
$\bar{c}_v$: Cost vector
$\bar{p}_{av}$: Available performance unit vector
PU_i: Available performance units in i^{th} node
$\omega_{i,j}$: Edge weight for edge (i, j). If i and j do not form a link, then $\omega_{i,j} \rightarrow \infty$
Start Procedure:
for (Each edge (i, j) in E) **do**
  $\mathbf{\Lambda}_{ij} = \omega_{i,j}$;
end for
for $(i = 1 : N_c)$ **do**
  for $(j = 1 : N_c)$ **do**
    for $(k = 1 : N_c)$ **do**
      if $\mathbf{\Lambda}_{jk} > \mathbf{\Lambda}_{ji}\mathbf{\Lambda}_{ik}$ **then**
        $\mathbf{\Lambda}_{jk} = \mathbf{\Lambda}_{ji} + \mathbf{\Lambda}_{ik}$;
      end if
    end for
  end for
end for
for (Each $i \in N_c$) **do**
  $\bar{c}_{v,i}$ = sum of i^{th} row of $\mathbf{\Lambda}$;
  $\bar{p}_{av,i} = PU_i$;
end for
for (Each $i \in F$) **do**
  Select all $j \in N_c$ where $\bar{p}_{av,j}$ is greater than performance need of f_i ;
  for (Each $j \in N_{I,i}$) **do**
    Place $\lambda_i^{j,k}$ at lowest metric node k ;
    Update $\bar{p}_{av}$;
  end for
end for

Algorithm 2: Proposed algorithm for VNF migration/chaining

Notations:
N: Number of physical nodes
N_c: Number of compute nodes
PU_i: Available performance units in i^{th} node
$\Psi = \{\Psi_L \cup \Psi_M \cup \Psi_H\}$: Segments of NFs according to ascending latency
$\Pi = \{\Pi_L \cup \Pi_M \cup \Pi_H\}$: Segments of NFs according to ascending compute load
Start Procedure:
for (Each $S_l \in S$) **do**
  Input L_d ;
  for (Each $f_i \in S_l$) **do**
    Select C_d ;
    $L_c = L_c + L_{f_i}$;
    Select L_0 and Θ_0 using FIS ;
    if (No. of compute nodes in selected segment > no. of compute nodes in adjacent segment) **then**
      Keep selected segment ;
    else
      Select adjacent segment ;
    end if
    if $(\exists\ k \in N_c : \{n_k\} \in \{\Psi \cap \Pi\})$ **then**
      Place $\lambda_i^{j,k} \in f_i$ in n_k having lowest metric in $\bar{c}_v$;
    else

      if (PU_i > requirement) **then**
        Create new $\lambda_i^{j,k} \in f_i$ at $n_k \in \Psi$;
      else
        Deny service ;
      end if
    end if
  end for
end for

accumulated path weights are used to populate composite weight vector $\bar{c}_v$, as shown below.

$$\bar{c}_{v,i} = \sum_{j=1}^{N} c_{ij} \quad \forall i \in N_c \tag{6}$$

Similarly, an available performance unit vector ($\bar{p}_{av}$) is maintained to hold available number of PUs for each node. The proposed algorithm tries to place instance of VNFs at compute nodes having lowest composite weight values, only if their available PUs are higher that the performance needs of the current VNF.

B. Proposed FIS-based Chaining Mechanism

This section presents a novel usage of fuzzy inference system (FIS) to optimize the chaining mechanism of VNFs after placement using the scheme mentioned in the previous section. Another contribution of the proposed mechanism is the dynamic instantiation of VNFs to meet specific service needs. As the VNF chaining optimization problem is challenging to solve using traditional mathematical techniques, fuzzy logic seems appropriate to address this problem being simple and computationally less extensive. Fuzzy logic has recently been used in several applications, including channel allocation, buffer management, traffic scheduling, congestion control, simultaneous management of multiple radio resources [35], and resource management for multiple networks [36].

The first stage is the design of FIS to select appropriate VNF instance having suitable latency and compute load for inclusion in the service chain. In contrast to the standard optimization techniques, fuzzy logic is based on human intuition and is very close to natural language. A fuzzy inference system generates output(s) based on one or more inputs, similar to a human decision. In general, a FIS consists of three components.

1) Fuzzification
2) Fuzzy Rule Matrix
3) Defuzzification

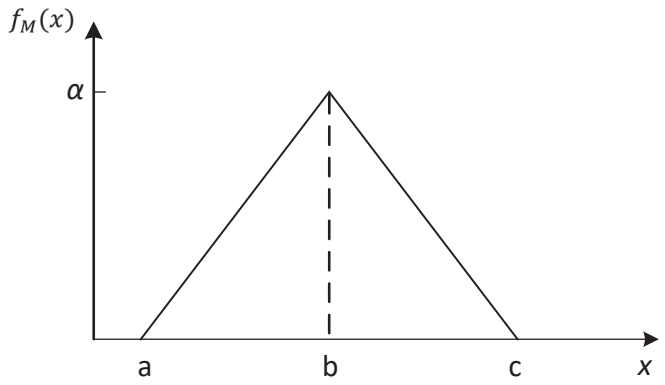

Fig. 2. Triangular Membership Function

The first component, fuzzification, is used to convert crisp input values into fuzzy numbers. The second component generates a fuzzy number corresponding to the compensated output by using the fuzzified input variables. Finally, these numbers are converted back to crisp values in the third component.

The main difference between traditional and fuzzy logic is that it considers the input and output parameters as vague, e.g., high throughput, low latency, moderate service quality, etc. In our proposed scheme, there are 3 inputs and 2 outputs of FIS, described below.

- *Input 1*: Cumulative latency (L_c) is the incremental latency of all VNF instances already included in the service chain
- *Input 2*: Demanded latency (L_d) is the end-to-end latency of all VNF instances included in the service chain
- *Input 3*: Demanded Compute load (C_d) is the quantified number of computational capacity required to execute a certain network function, measured in PU
- *Output 1*: Latency of VNF (L_0) is the latency of single VNF instance selected for participation in the service chain during the current iteration.
- *Output 2*: Compute load of VNF (Θ_0): It is the measure of performance units of the VNF instance selected for the service chain during the current iteration.

Once the inputs and outputs are defined, the next step is the selection of appropriate membership function (MF) for each input and output. We have used triangular MFs for all the inputs and outputs. A triangular MF $M(x)$ with endpoints $(a, 0)$ and $(c, 0)$ and the high point (b, α), as shown in Figure 2, is given as

$$M(x) = \begin{cases} \alpha\left(\frac{x-a}{b-a}\right), & a \le x \le b \\ \alpha\left(\frac{x-c}{b-c}\right), & b \le x \le c \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

The fuzzy sets for all the inputs and outputs are low, moderate, and high. The corresponding MFs for all the inputs and outputs are shown in figures 3 and 4, respectively. The universe of discourse (range of the parameter) for L_c, L_d, and

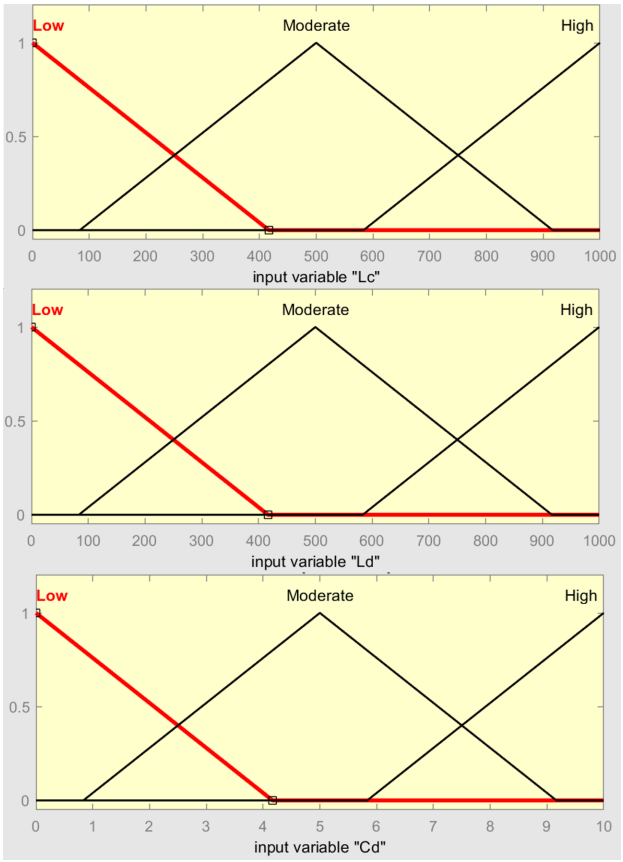

Fig. 3. MFs for inputs of the FIS

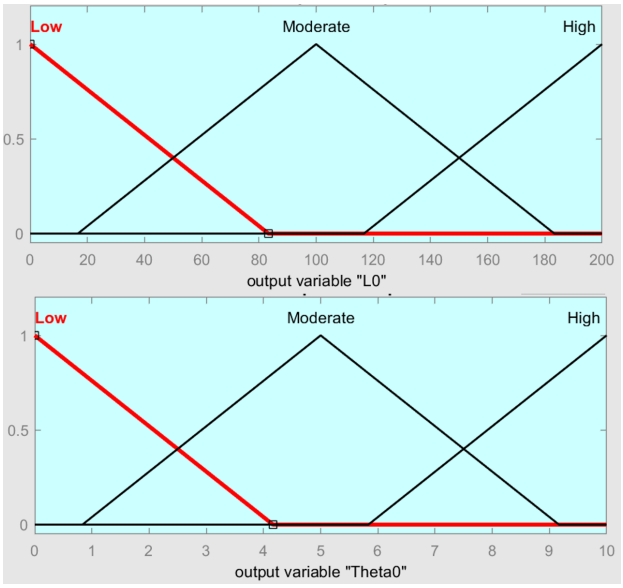

Fig. 4. MFs for outputs of the FIS

L_0 is taken in milliseconds, whereas, for C_d and Θ_0, it is taken in terms of PUs.

After the fuzzification stage, the next and the most crucial step is to formulate a fuzzy rule matrix (FRM). This table consists of IF-ELSE fuzzy rules based on human intuition for the proper selection of output variables. In our proposed

system, there are 3 inputs with 3 fuzzy sets each; therefore, the FRM consists of 27 rules shown in table II. An example of the IF-ELSE rule is given below:

IF (L_c is Low) & (L_d is High) & (C_d is High) **THEN** (L_0 is Moderate) & (Θ_0 is High)

In the above rule, moderate latency is selected because the proposed algorithm tries to offer better than demanded service while trying to achieve a fair distribution of service chains. After generating fuzzified output through FRM, the last stage of defuzzification converts these fuzzy outputs to crisp numbers through the centroid of area (COA) method.

To achieve load distribution, a dynamic scheme for selecting compute nodes, hosting the required VNF instance, is proposed. We first identify the number of nodes hosting the required VNF in the selected latency segment. This number is compared with that of the adjacent segments. The latency segment with the highest number of such nodes is selected. The node with minimum composite weight is included in the service chain from the nodes satisfying the selection criteria. If no such node is found, a new VNF instance is created at a node in the selected latency segment whose available PU is greater than the demanded. If no such node is found, the service is denied.

The two-phase placement and chaining approach helps in better capacity planning and traffic engineering, i.e., prioritization, and service-based traffic forwarding. Algorithm I results in appropriate placement of VNF instances and precise information of available PUs and latencies at all compute nodes. Algorithm II chains VNF instances while conforming to the required service needs. The service prioritization is an inherent attribute of the proposed FIS-based chaining algorithm controlled through FRM. If an SFC doesn't conform to the required service needs, it is discarded and re-constructed using the proposed FIS to meet the service needs. Algorithm I is executed at the network start-up and when the number of denied requests from FIS exceeds a certain threshold, thus minimizing the network compute overhead.

Now we discuss the computational complexity of the proposed FIPAM algorithm. The complexity of the first part i.e. VNF placement is $O(N_c^3)$, where N_c is the number of compute nodes. The complexity of the second part of the algorithm i.e. VNF migration/chaining is $O(M \times K)$, where M is the number of service chains and K is the complexity of the fuzzy inference system. The complexity of FIS depends on the number of fuzzy sets, number of inputs and outputs, number of rules, and discretization of inputs and outputs universe of discourse [37].

VI. Working Example

As a working example, we re-consider and further investigate the enterprise network security problem, already presented in figure 1. Such networks are usually protected by a perimeter gateway firewall (PGF). The security rules of PGF filter any incoming traffic. Placing PGF instances at clouds close to the network security boundaries can solve the PGF acquisition and management problems, reducing capital and operational costs. However, in the case of large IoT-based enterprise networks, many other considerations are essential. For instance, price reduction and development speed often result in compromised security fundamentals in IoT devices. Compromised IoT devices within the organization's security boundaries can generate serious threats to enterprise security. Different security policies may be needed according to the devices' make, location, communicating peers, inter-network communications, etc. Access to the same devices may be filtered with simple firewall rules. On the other hand, we may need to protect some of the devices with an additional intrusion detection system (IDS) and DPI modules. Computational requirements of these devices also vary greatly. For instance, only shallow packet inspection (SPI), focusing on mean-variance in packet sizes, inter-packet latency, and correlation to specific events, is enough. On the other hand, deep packet inspection, including packet headers, payload, and communicating peers' information, may be necessary. Similarly, VNF placement updates may trigger due to network dynamics or frequent location changes of specific IoT devices/services.

We introduce the model with different nodes, and starting/ending points for data packets' regulation and processing. As already discussed, the compute nodes' computational capacity is measured by three different parameters, i.e., CPU utilization, RAM, and GPU. The aggregation factors (α, β, and γ) are used to quantify the available computational capacity into a single variable. We use PU, defined in eq. 5, to evaluate the total capacity of each node, and after measuring usage of each node, we evaluate its utilization ratio. In this example, values of α, β, and γ are taken to be 0.5, 0.3, and 0.2, respectively.

Now we calculate the performance units of five compute nodes N1, N2, N3, N6, and N9, to determine the less utilized node. Assuming utilization ratio (UR) of five nodes to be 60%, 45%, 70%, 51%, and 40% respectively, the PU and APU of each node, computed using eq. (5), and (8), are shown in table III.

$$APU = PU\left(1 - \frac{UR}{100}\right) \tag{8}$$

Table III suggests that node N2 has more available computational resources than other compute nodes. Therefore, node N2 should be preferred for hosting a new VNF instance. However, latency and bandwidth of node N2 from other nodes hosting VNF instances corresponding to the same service chain should also be considered. Therefore, we determine each node's cost (latency, bandwidth, etc.) to other nodes to find the shortest path through the proposed placement mechanism, shown in algorithm 1. The weighted graph of the network, as mentioned earlier, is shown in figure 5. The weights are calculated based on link capacity and latency. The computed cost matrix by using the proposed algorithm is given by

$$\mathbf{\Lambda} = \begin{bmatrix} 3 & 0 & 6 & 2 & 4 & 8 & 7 \\ 5 & 6 & 0 & 4 & 6 & 10 & 9 \\ 5 & 2 & 4 & 0 & 2 & 6 & 7 \\ 7 & 4 & 6 & 2 & 0 & 4 & 6 \\ 11 & 8 & 10 & 6 & 4 & 0 & 8 \end{bmatrix}$$

TABLE II
FUZZY RULE MATRIX (FRM) FOR THE PROPOSED ALGORITHM

		C_d					
		Low		**Moderate**		**High**	
L_c	L_d	L_0	Θ_0	L_0	Θ_0	L_0	Θ_0
Low	**Low**	Low	Low	Low	Moderate	Low	High
	Moderate	Moderate	Low	Moderate	Moderate	Low	High
	High	High	Low	High	Moderate	Moderate	High
Moderate	**Low**	Low	Low	Low	Moderate	Low	High
	Moderate	Low	Low	Low	Moderate	Low	High
	High	Moderate	Low	Moderate	Moderate	Moderate	High
High	**Low**	Low	Low	Low	Moderate	Low	High
	Moderate	Low	Low	Low	Moderate	Low	High
	High	Moderate	Low	Moderate	Moderate	Moderate	High

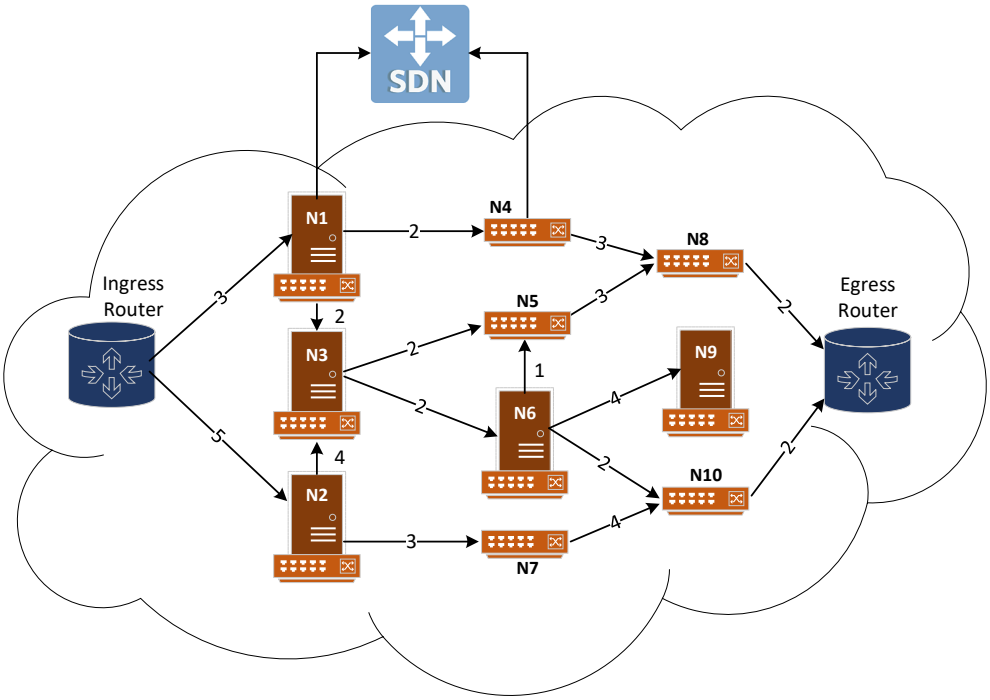

Fig. 5. Cost-weighted enterprise network

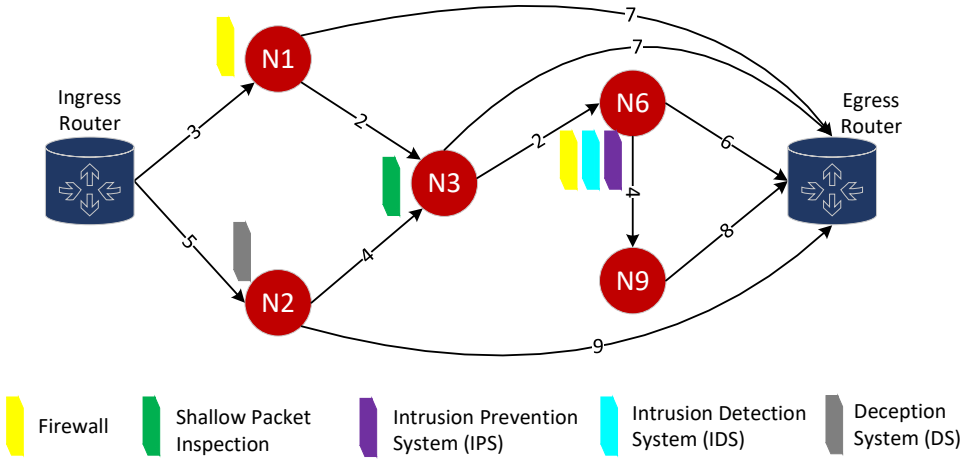

Fig. 6. Overlay network of compute nodes hosting VNFs

TABLE III
PERFORMANCE UNITS OF COMPUTE NODES

Parameters	CPU	RAM	GPU	PU	UR	APU
N1	4	8	4	5.2	60%	2.1
N2	6	16	4	8.6	45%	4.8
N3	4	4	8	4.8	70%	1.5
N6	6	16	4	8.6	51%	4.2
N9	4	8	4	5.2	40%	2.1

The sum of each row (except the first and last column) gives the cumulative cost of each compute node to all other compute nodes in the network, given by

$$\bar{c}_v = \begin{bmatrix} 20 \\ 26 \\ 14 \\ 16 \\ 28 \end{bmatrix}$$

The smaller cumulative cost value indicates better access of the compute node from all other nodes in the network. In this example, node N3 is the best candidate for hosting VNF instances. However, placement of VNF instances at any compute node also requires sufficient available computational capacity (i.e. APU). A subset of compute nodes having APU greater than the VNF compute demand is selected from the cost vector. In this example, five virtual functions form different SFCs offering micro-services to enhance the network's security. Initially, four virtual functions (FW, SPI, IPS, and IDS) are needed to form two SFCs i.e. $S_1 = FW \rightarrow IDS$, and $S_2 = FW \rightarrow SPI \rightarrow IPS$. For this purpose, one instance of each VNF is deployed on compute nodes N1, N3, and N6, as shown in the overlay network of compute nodes given in figure 6. These VNFs are embedded into S_1 and S_2 using the proposed fuzzy inference-based chaining mechanism described in algorithm II.

Now, let's suppose an internal network threat is suspected. Therefore, a new service chain $S_3 = FW \rightarrow DS$ is needed. Since the existing firewall instance cannot be over-scaled due to limited computing resources at N1, a new instance for the firewall needs to be orchestrated. To further evaluate the nature of malicious traffic, an instance of deception system is orchestrated to log all activity triggered by the malicious traffic. Using algorithm II, the firewall instance is placed on N6, and the deception system is placed on N2.

VII. SIMULATION RESULTS

For performance evaluations, we simulate an enterprise network security model having multiple SFCs comprising of 2-5 VNFs, and 50 compute nodes. The nodes' compute capacity is randomly set as 4, 10, and 20 performance units to indicate low, medium, and high compute capacity. The volume for one service traffic is 25, 50, and 100 MBs. The bandwidth for all links is set to 1 GB, and 50 edges are randomly placed in the network to connect all compute nodes. NFV chains are embedded using the FIPAM algorithm in MATLAB. The results are compared with stochastic hill climbing (SHC) based placement/chaining strategy and capacitated shortest path tour problem (CSPTP) [30]. The CSPTP scheme solves the placement problem as a shortest path tour problem, by applying greedy-based heuristic on sub-paths between source and destination. This makes CPSTP computationally expensive, on cost of achieving better service paths. Performance metrics include the acceptance ratio for the service requests and link load balancing indicators. For acceptance ratio, an increasing number of SFC requests are made with low, medium, and high service demands. Figure 7 depicts that the proposed FIPAM algorithm's intelligent placement and migration mechanism accommodated a larger number of service requests while conforming to the required service needs. For IoTs requiring a large number of short-lived micro-service requests, fuzzy rule-based chaining and VNF migration mechanism can yield better results. Furthermore, the proposed FIPAM algorithm also maintains a closer approximation to the optimal acceptance ratio.

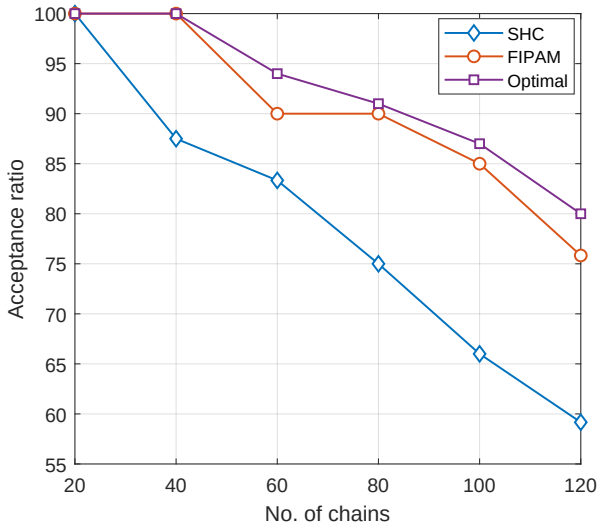

Fig. 7. Acceptance ratio against increasing service chains

For link-load balancing, standard deviations in terms of link utilization and end-to-end service latency are presented in figures 8 and 9, respectively. As one of the objectives of the proposed algorithm is to achieve better traffic load distribution that also impacts end-to-end delays by minimizing the number of congested links. The standard deviation of link utilization ratio and service latency indicate that the proposed FIPAM yields better load distribution as compared to both SHC and CSPTP schemes. The reason is that the proposed FIPAM algorithm looks into multiple parameters i.e. cumulative latency and demanded compute load to provide a diversified placement and chaining of VNFs. In addition, the proposed FIPAM algorithm maintains closer approximation to the optimal standard deviation of link utilization and latency.

The impact of increasing number of compute nodes on the computational time is depicted in figure 10. We measure the computational time consumed by the algorithms through MATLAB simulation. Although the computational complexity of the proposed VNF placement algorithm is $O(N_c^3)$, however,

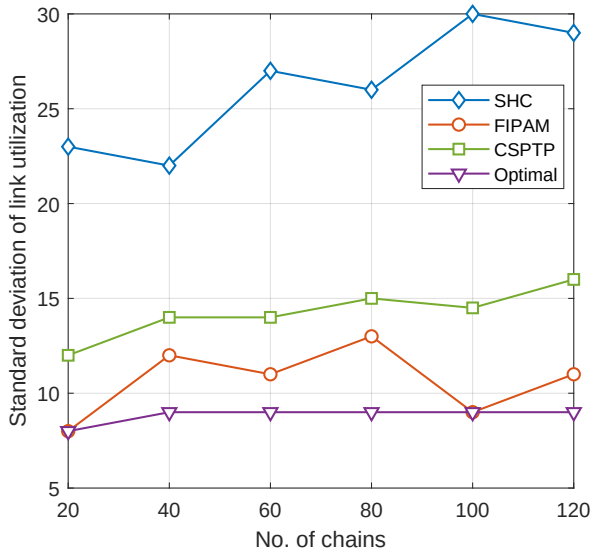

Fig. 8. Standard deviation of link utilization against increasing service chains

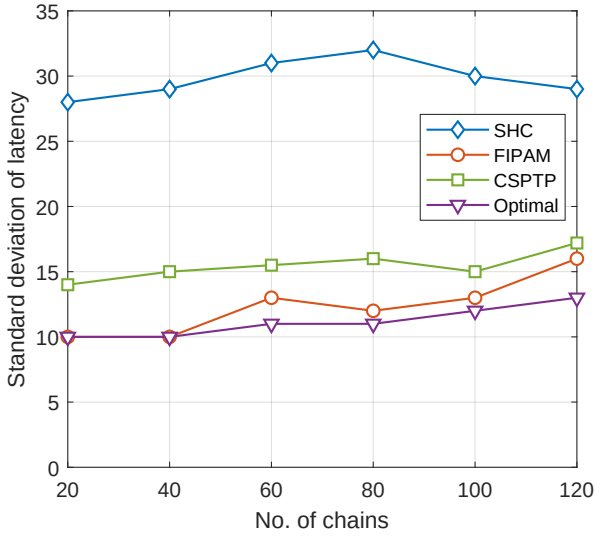

Fig. 9. Standard deviation of latency against increasing service chains

the number of compute nodes in any network is usually much smaller than the total number of nodes. It can be seen that the computational time for the proposed FIPAM algorithm is reasonably small as compared to both SHC and CSPTP schemes, even for large number of compute nodes (i.e. $N_c = 500$).

It is also worth mentioning that the proposed VNF placement algorithm is executed at a lower frequency, i.e. at network start-up and when the number of denied requests from fuzzy inference system exceeds a certain threshold. The threshold is usually adjustable as per the application requirements. A smaller threshold value will increase the frequency of readjustments that results in lesser number of VNF migrations, maintaining near-optimal placement of VNFS. However, it

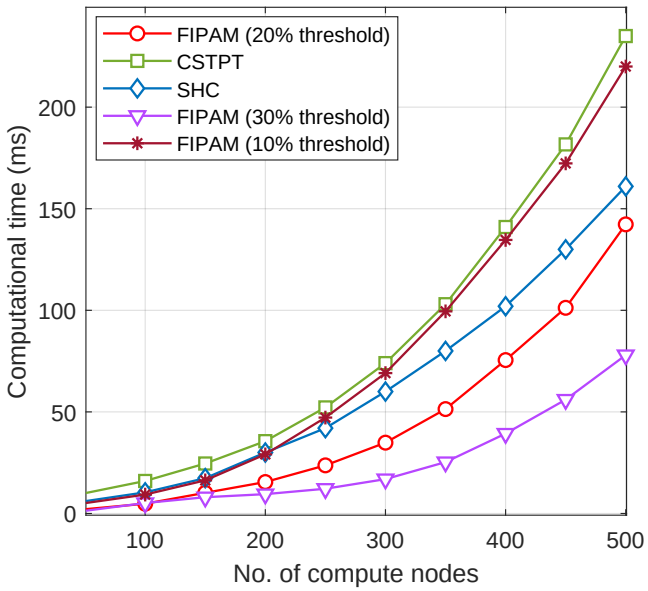

Fig. 10. Impact of increasing compute nodes on computational time

TABLE IV
SUMMARY IN TERMS OF PERCENTAGE IMPROVEMENT

	FIPAM vs SHC	FIPAM vs CSPTP
STD of latency	45–62%	7–33%
STD of link utilization	45–70%	14–38%
Computational time	12–66%	40–80%

will increase the computational time. Therefore, a trade-off is maintained between threshold value and computational time, and is adjusted on the basis of application scenario and computational resources in the network. The impact of threshold on computation time is depicted in figure 10. This is also worth mentioning that the points of interest for any service may continue to change from time-to-time in a large-scale IoT network. One contribution of the proposed algorithm is the readjustment of VNF instances to the most fitting locations. The percentage improvement of the proposed FIPAM algorithm in terms of the standard deviation (STD) of latency, link utilization and computational time is summarized in table IV.

VIII. Conclusion

Network function virtualization (NFV) has gained considerable attention during recent years to provide a flexible and efficient solution for IoT-based applications. This paper has proposed a novel fuzzy inference-based placement and migration (FIPAM) approach for VNF placement and chaining. The proposed approach carefully carries out the resource allocation process to properly utilize all the NFs. The complete FIPAM approach is divided into two stages; placement and chaining. After formulating the VNF optimization problem, we have proposed a lightweight placement solution considering underlying network parameters. For chaining, we have proposed a novel usage of a fuzzy inference system to optimize

the process that can meet specific service requirements. The superiority of the proposed FIPAM approach over existing techniques is validated through simulation results.

REFERENCES

[1] K. Ashton *et al.*, "That 'internet of things' thing," *RFID journal*, vol. 22, no. 7, pp. 97–114, 2009.

[2] D. Evans, "The internet of things: How the next evolution of the internet is changing everything," *CISCO white paper*, vol. 1, no. 2011, pp. 1–11, 2011.

[3] R. Minerva, A. Biru, and D. Rotondi, "Towards a definition of the internet of things (IoT)," *IEEE Internet Initiative*, vol. 1, no. 1, pp. 1–86, 2015.

[4] "IoT Middleware Market - Growth, Trends, and Forecasts (2020 - 2025)," Tech. Rep., 2020.

[5] X. Fu, F. R. Yu, J. Wang, Q. Qi, and J. Liao, "Dynamic service function chain embedding for NFV-enabled IoT: A deep reinforcement learning approach," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 507–519, 2019.

[6] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications surveys & tutorials*, vol. 18, no. 1, pp. 236–262, 2015.

[7] N. F. Virtualisation, "Architectural framework (RGS/NFV-002)," 2014.

[8] M. C. Luizelli, W. L. da Costa Cordeiro, L. S. Buriol, and L. P. Gaspary, "A fix-and-optimize approach for efficient and large scale virtual network function placement and chaining," *Computer Communications*, vol. 102, pp. 67–77, 2017.

[9] A. De Domenico, Y.-F. Liu, and W. Yu, "Optimal virtual network function deployment for 5G network slicing in a hybrid cloud infrastructure," *IEEE Transactions on Wireless Communications*, vol. 19, no. 12, pp. 7942–7956, 2020.

[10] I. Benkacem, T. Taleb, M. Bagaa, and H. Flinck, "Optimal VNFs placement in CDN slicing over multi-cloud environment," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 616–627, 2018.

[11] M. A. T. Nejad, S. Parsaeefard, M. A. Maddah-Ali, T. Mahmoodi, and B. H. Khalaj, "vSPACE: VNF simultaneous placement, admission control and embedding," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 542–557, 2018.

[12] C. Pham, D. T. Nguyen, N. H. Tran, K. K. Nguyen, and M. Cheriet, "Optimized IoT service chain implementation in edge cloud platform: A deep learning framework," *IEEE Transactions on Network and Service Management*, 2021.

[13] V. Eramo, E. Miucci, M. Ammar, and F. G. Lavacca, "An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures," *IEEE/ACM Transactions on Networking*, vol. 25, no. 4, pp. 2008–2025, 2017.

[14] A. M. Medhat, T. Taleb, A. Elmangoush, G. A. Carella, S. Covaci, and T. Magedanz, "Service function chaining in next generation networks: State of the art and research challenges," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 216–223, 2016.

[15] S. Dutta, T. Taleb, and A. Ksentini, "QoE-aware elasticity support in cloud-native 5G systems," in *2016 IEEE International Conference on Communications (ICC)*. IEEE, 2016, pp. 1–6.

[16] I. Cerrato, M. Annarumma, and F. Risso, "Supporting fine-grained network functions through intel DPDK," in *2014 Third European Workshop on Software Defined Networks*. IEEE, 2014, pp. 1–6.

[17] T. Taleb, A. Ksentini, and R. Jantti, ""anything as a service" for 5G mobile systems," *IEEE Network*, vol. 30, no. 6, pp. 84–91, 2016.

[18] A. Leivadeas, G. Kesidis, M. Ibnkahla, and I. Lambadaris, "VNF placement optimization at the edge and cloud," *Future Internet*, vol. 11, no. 3, p. 69, 2019.

[19] Y. Li, F. Zheng, M. Chen, and D. Jin, "A unified control and optimization framework for dynamical service chaining in software-defined NFV system," *IEEE Wireless Communications*, vol. 22, no. 6, pp. 15–23, 2015.

[20] J. Hwang, K. K. Ramakrishnan, and T. Wood, "NetVM: high performance and flexible networking using virtualization on commodity platforms," *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 34–47, 2015.

[21] C. Lorenz, D. Hock, J. Scherer, R. Durner, W. Kellerer, S. Gebert, N. Gray, T. Zinner, and P. Tran-Gia, "An SDN/NFV-enabled enterprise network architecture offering fine-grained security policy enforcement," *IEEE communications magazine*, vol. 55, no. 3, pp. 217–223, 2017.

[22] M. Jalalitabar, E. Guler, D. Zheng, G. Luo, L. Tian, and X. Cao, "Embedding dependence-aware service function chains," *Journal of Optical Communications and Networking*, vol. 10, no. 8, pp. C64–C74, 2018.

[23] L. Wang, Z. Lu, X. Wen, R. Knopp, and R. Gupta, "Joint optimization of service function chaining and resource allocation in network function virtualization," *IEEE Access*, vol. 4, pp. 8084–8094, 2016.

[24] J. Li, W. Shi, H. Wu, S. Zhang, and X. Shen, "Cost-aware dynamic SFC mapping and scheduling in SDN/NFV-enabled space-air-ground integrated networks for internet of vehicles," *IEEE Internet of Things Journal*, 2021.

[25] Y. Yue, B. Cheng, M. Wang, B. Li, X. Liu, and J. Chen, "Throughput optimization and delay guarantee VNF placement for mapping SFC requests in NFV-enabled networks," *IEEE Transactions on Network and Service Management*, 2021.

[26] R. Lin, S. Yu, S. Luo, X. Zhang, J. Wang, and M. Zukerman, "Column generation based service function chaining embedding in multi-domain networks," *IEEE Transactions on Cloud Computing*, 2021.

[27] H. Huang, C. Zeng, Y. Zhao, G. Min, Y. Y. Zhu, W. Miao, and J. Hu, "Scalable service function chain orchestration in NFV-enabled networks: A federated reinforcement learning approach," *IEEE Journal on Selected Areas in Communications*, 2021.

[28] A. Leivadeas and M. Falkner, "VNF placement problem: a multi-tenant intent-based networking approach," in *2021 24th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. IEEE, 2021, pp. 143–150.

[29] Y. Xie, S. Wang, and B. Wang, "Virtual network function placement with bounded migrations," *Cluster Computing*, pp. 1–12, 2021.

[30] M. Sasabe and T. Hara, "Capacitated shortest path tour problem-based integer linear programming for service chaining and function placement in nfv networks," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 104–117, 2020.

[31] A. Leivadeas, M. Falkner, I. Lambadaris, and G. Kesidis, "Resource management and orchestration for a dynamic service chain steering model," in *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2016, pp. 1–6.

[32] K. Antevski, J. Martín-Pérez, N. Molner, C.-F. Chiasserini, F. Malandrino, P. Frangoudis, A. Ksentini, X. Li, J. SalvatLozano, R. Martínez *et al.*, "Resource orchestration of 5G transport networks for vertical industries," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 2018, pp. 158–163.

[33] M. Otokura, K. Leibnitz, Y. Koizumi, D. Kominami, T. Shimokawa, and M. Murata, "Application of evolutionary mechanism to dynamic virtual network function placement," in *2016 IEEE 24th International Conference on Network Protocols (ICNP)*. IEEE, 2016, pp. 1–6.

[34] Y. Nam, S. Song, and J.-M. Chung, "Clustered NFV service chaining optimization in mobile edge clouds," *IEEE Communications Letters*, vol. 21, no. 2, pp. 350–353, 2016.

[35] M. W. Khan and M. Zeeshan, "QoS-based dynamic channel selection algorithm for cognitive radio based smart grid communication network," *Ad Hoc Networks*, vol. 87, pp. 61–75, 2019.

[36] L. Giupponi, R. Agusti, J. Perez-Romero, and O. S. Roig, "A novel approach for joint radio resource management based on fuzzy neural methodology," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 3, pp. 1789–1805, 2008.

[37] M. W. Khan and M. Zeeshan, "Fuzzy inference based adaptive channel allocation for ieee 802.22 compliant smart grid network," *Telecommunication Systems*, vol. 72, no. 3, pp. 339–353, 2019.

Muhammad Arslan Tariq received his BS degree in Electrical Engineering from Center for Advanced Studies in Engineering (CASE), Pakistan and Master's degree in Computer Engineering from National University of Science and Technology (NUST), Pakistan. His current research interests include network function virtualization, cloud computing, virtualization and software-defined networks.

Muhammad Umar Farooq received his Master's in computer science from Quaid-i-Azam University Pakistan, Master's in software engineering from National University of Sciences and Technology, Pakistan and Ph.D. in computer science from University Politehnica of Bucharest, Romania. He is currently an Assistant Professor of computer and software engineering with the College of Electrical and Mechanical Engineering, National University of Sciences and Technology. His research interests include routing and MAC protocols for wireless ad hoc networks, delay tolerant networks, and the internet of things. He is an inventor of one awarded US patent, reviewer of several international conferences and journals, and author of several international publications.

Muhammad Zeeshan (M'16) received Ph.D. degree in Electrical Engineering with specialization in wireless communications, from College of Electrical and Mechanical Engineering, National University of Sciences and Technology (NUST), Pakistan in 2015. From July 2010 to March 2016, he was with the Center for Advanced Research in Engineering (CARE), Pakistan, first as a Senior Design Engineer, and then as a Member Technical Staff starting July 1, 2014. In March 2016, he joined the faculty of the College of Electrical and Mechanical Engineering, National University of Sciences and Technology (NUST), Pakistan, where he is currently an Assistant Professor of wireless communications in the Department of Electrical Engineering. At this department, he is also heading the Cognitive Radio Wireless Communications (CoRWiC) research group. His research interests include SDR waveform development, physical layer design, synchronization techniques, and digital design of wireless communication systems. He has a number of international journal and conference publications. He is a reviewer of the IEEE Transactions on Communications. IEEE Access, and Wireless Personal Communications.

Ali Hassan received the BE and MS degrees in computer engineering from the College of Electrical and Mechanical Engineering (CEME), NUST, Islamabad, Pakistan, in 2004 and 2007, respectively, and the Ph.D. degree in electrical engineering from the University of Southampton, UK, in 2012. He is currently the Head of Department of the Department of Computer and Software Engineering, College of Electrical and Mechanical Engineering, NUST. His areas of specialization are machine learning and speech processing.

Adnan Akhunzada (SM) has more than 12 years of research and development experience both in ICT industry and academia. He has a proven track record of high impact published research (i.e., Patents, Journals, Transactions, Commercial Products, Book chapters, Reputable Magazines, Conferences, and Conference Proceedings). His experience as an Educator and a Researcher is diverse that includes work as a lecturer, a senior lecturer, a year tutor, and an occasional lecturer at other engineering departments, as an Assistant Professor at COMSATS University Islamabad (CUI), a Senior Researcher at RISE SICs Vasteras AB, Sweden, as a Research Fellow and a Scientific Lead at the DTU Compute, Technical University of Denmark (DTU), and a visiting professor having mentorship of graduate students, and a supervision of academic and research and development projects both at UG and PG level. His research interests include cyber security, machine learning, deep learning, reinforcement learning, artificial intelligence, large scale distributed systems (i.e., edge, fog, cloud, and SDNs), the IoT, Industry 4.0, and internet of everything (IoE). He is also a member of technical programme committee of varied reputable conferences, journals, and editorial boards.