Managing Chains of Application Functions over Multi-Technology Edge Networks

Nabeel Akhtar, Ibrahim Matta, *Senior Member*, *IEEE*, Ali Raza, *Student Member*, *IEEE*, Leonardo Goratti, Torsten Braun, *Senior Member*, *IEEE* and Flavio Esposito, *Member*, *IEEE*

***Abstract*—Next-generation networks are expected to provide higher data rates and ultra-low latency in support of demanding applications, such as virtual and augmented reality, robots and drones, *etc*. To meet these stringent requirements of applications, edge computing constitutes a central piece of the solution architecture wherein functional components of an application can be deployed over the edge network to reduce bandwidth demand over the core network while providing ultra-low latency communication to users. In this paper, we provide solutions to resource orchestration and management for applications over a virtualized *client-edge-server* infrastructure. We investigate the problem of optimal placement of pipelines of application functions (virtual service chains) and the steering of traffic through them, over a multi-technology edge network model consisting of both wired and wireless millimeter-wave (mmWave) links. This problem is $\mathcal{NP}$-hard. We provide a comprehensive "microscopic" binary integer program to model the system, along with a heuristic that is one order of magnitude faster than optimally solving the problem. Extensive evaluations demonstrate the benefits of orchestrating virtual service chains (by distributing them over the edge network) compared to a baseline "middlebox" approach in terms of overall admissible virtual capacity. Moreover, we observe significant gains when deploying a small number of *mmWave* links that complement the *Wire* physical infrastructure in high node density networks.**

***Index Terms*—Application Decomposition, Computer Network Management, Network Optimization, Softwarized Networks, Software-Defined Networking, Virtual Functions.**

I. Introduction

Next-generation mobile networks are expected to go beyond the delivery of static or streaming content, such as telephony, web browsing, and low-resolution video. They should be capable of serving many billions of users and smart devices at much higher data rates (over 500 Mbps) and ultra-low latencies (less than five milliseconds) [1]–[3]. Potential next-generation mobile network applications include robots and drones, virtual and augmented reality, healthcare, etc. Traditional network and application architectures can not support these stringent application requirements. Advances in the physical network infrastructure, *e.g.*, the integration of Gigabit Ethernet and millimeter wave (*mmWave*) technologies, and the virtualization of network and application functions are key to achieving these next-generation mobile network goals [1]–[3].

N. Akhtar, I. Matta and A. Raza are with Boston University, USA. N. Akhtar is currently at Akamai Technologies Inc. (e-mails: nabeel@bu.edu, matta@bu.edu, araza@bu.edu)

L. Goratti was with FBK CREATE-NET, Italy. (e-mail: lgoratti@fbk.eu)

T. Braun is with University of Bern, Switzerland. (e-mail: torsten.braun@inf.unibe.ch)

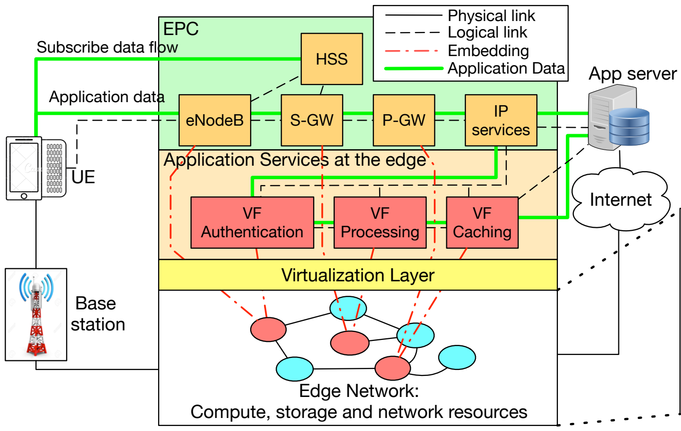
F. Esposito is with Saint Louis University, USA. (e-mail: espositof@slu.edu)

The virtualization of network functions, termed Network Function Virtualization (NFV), aims to decouple network software from proprietary, dedicated hardware appliances, termed "middleboxes" (*e.g.*, traffic shapers, Network Address Translation boxes). Similarly, application virtualization allows an application to work in an isolated virtualized environment. Moreover, in cloud-based or service-oriented application architectures, an application can be composed of many application components, where each component can run as a Virtual Function (VF). Thus, under application service virtualization, multiple VFs can run on any general-purpose computer within a virtual machine, in an operating system container, or as a serverless "Function as a Service" (FaaS). The flexibility with which VFs can be deployed and managed — *i.e.*, chained, allocated resources, migrated — allows their hosting "close" to the users, in an edge cloud/datacenter, thus meeting the application requirements of ultra-low latency and high throughput.

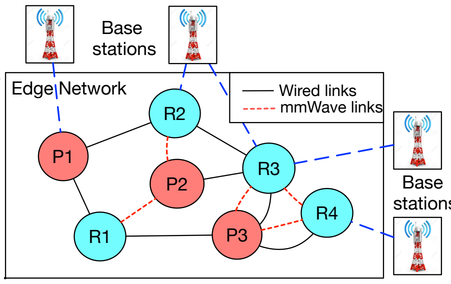
Figure 1a illustrates the evolution of cellular networks, where network services are moved from radio base stations and gateways into the edge cloud. In a traditional LTE architecture, user traffic traverses a series of devices on its way to the application server: the base station (eNodeB), a serving gateway (S-GW), and finally a packet data network gateway (P-GW) that connects to the outside world. On the other hand, in a virtualized environment, these network functions are envisioned to run virtualized, anywhere on the edge resources. They are chained together in a particular order based on processing requirements — in Figure 1a example, (eNode, S-GW, P-GW). To steer traffic across these VFs, Software Defined Networking (SDN) mechanisms are leveraged so that routes are established programmatically between components of the service chain.

Applications running on the edge network can also have different service chain requirements (*e.g.*, Authentication, Processing, Caching in Figure 1), and multiple application flows may need to use the same VF. Thus, understanding where to place VFs, or instances of the same VF, that are necessary to satisfy service chain requirements of different application flows, subject to physical resource (host and network) constraints, is a challenging problem. Furthermore, an edge network may consist of multiple link technologies, *e.g.*, Ethernet and *mmWave*, that may have different characteristics suitable for possibly different types of application flows.

Our Contribution: In this paper, leveraging optimization theory, we investigate the joint placement of virtual service chains consisting of virtual application functions (components) and the steering of traffic through them, over a multi-technology



a) Next Generation Infrastructure supporting virtualized services



b) Edge network with processing & routing nodes and heterogeneous transport network

Figure 1: Application Function virtualization for next generation mobile network

edge network model consisting of both wired and *mmWave* links. Our contributions in this work are:

- We propose a detailed "microscopic" binary integer program (BIP) to find the optimal placement of virtual functions.
- BIP is $\mathcal{NP}$-hard (*i.e.*, computationally expensive), so we provide a heuristic that is one order of magnitude faster than BIP.
- Our workload model captures virtual service chains that correspond to the needs of future applications described as "killer applications" (*i.e.*, virtual and augmented reality) over the edge network.
- Extensive evaluation results demonstrate the benefits of managing virtual service chains (by distributing them over the edge network) compared to a baseline "middlebox" approach (where all functions are run on one host).
- We observe significant gains when deploying *mmWave* links that complement the wired physical infrastructure. Moreover, most of the gains are attributed to only 30% of these *mmWave* links, which indicates that judicial placement of *mmWave* links is key for maximum gains.
- We show that the gains are highest in the high node-density networks, where *mmWave* links can be easily established between the nodes.
- To the best of our knowledge, this is the first work to study a multi-technology based edge infrastructure envisioned for next generation mobile networks. The developed model can be used by the "service" provider to allocate resources to service chains optimally, and by the "infrastructure" provider to understand the benefits of deploying *mmWave* links.

This paper extends our initial work published in [4]. In particular, we substantially revise and expand the evaluation results to include edge network topologies that are typical of edge city networks, namely those whose main structure is a bus or a ring. We also analyze the impact of the formation of mmWave links given the density of the network nodes.

Paper Organization: The paper is organized as follows: Section II & III provides a background and reviews related work. Section IV describes our system model. Section V explains our mathematical formulation. Section VI presents our evaluation model, parameters and proposed heuristic. Results are shown in Section VII. Section VIII concludes the paper.

II. Background

This section provides a review of the industry's direction to support high data rate and ultra-low latency applications on next-generation mobile networks. According to "IMT-2020", a program developed by the International Telecommunication Union's Radiocommunication Sector (ITU-R), the peak data rates are expected to be around 10 Gbits/s, while end-to-end latency is expected to be less than 5 ms [5]. To meet these strict requirements, there is a need for changes in the infrastructure (*e.g.*, using millimeter wave) and for having elasticity in hosting virtual functions (VFs) at the edge of the network. Users accessing application servers hosted in the public network experience average delays of 50-100ms, while such applications hosted in the operator's cloud experience delays ranging from 20-50ms. However, these delays are still significantly higher than those expected from a network that supports future applications. To meet the strict requirements of network applications for delays of 1-5 ms, the edge computing paradigm that places computation closer to end-users is necessary [1], [2]. As an example, *Telefonica*, one of the world's largest telecom operator, is using their central offices (COs) as datacenters (COdc). These COdc are closer to the end-users (at the network edge) and are capable of hosting user VFs [6].

Figure 1a shows the case where service-chain components are running as virtualized functions at the edge of the network. Here, all the traffic from users passes through Authentication, Processing, and Caching, which are running at the edge of the network, before arriving at the Application Server. Note that the operator's network services (*e.g.*, S-GW and P-GW), which are part of the Evolved Packet Core (EPC), can also be virtualized and hosted in the edge datacenter, as shown in Figure 1a. However, in this work, we are specifically studying virtual functions for applications running on the mobile network. *The internal functional split of the Radio Access Network, and virtual EPCs is beyond the scope of this work.*

Figure 1b shows an example of an edge network, consisting of processing nodes (P1-P3) and routing/switching nodes (R1-R4). This edge network covers a small geographical area, *e.g.*, a medium-size city. As the name suggests, processing nodes have the processing power and can host VFs, while routing nodes are responsible for routing traffic through the network. Note that a processing node can also act as a routing node. All the nodes are SDN enabled and can be programmed for traffic routing. The nodes are connected with two different link technologies, namely *Wire* and millimeter wave (*mmWave*) links. The *mmWave* technology is considered an important aspect of next-generation mobile networks. The enormous amount of spectrum available in the *mmWave* band, and the ease and flexibility of deploying a *mmWave* infrastructure, will greatly increase the network capacity, as well as decrease latency when *mmWave* links are used to create shortcuts between nodes [7].

Application service components are hosted at processing nodes. These components run as VFs and are dynamically instantiated, migrated, or removed from the network based on the system requirements. Applications have strict requirements for their traffic to traverse virtualized services in a certain order, *e.g.*, authentication followed by caching. This is known as "Service Function Chaining" (SFC). SFC is an important capability of virtualized networks as it provides both modularity and elasticity. A single function in a service chain can be dynamically changed/updated without having any impact on other functions. The efficient placement of virtualized functions and traffic steering through service chains are challenging problems.

III. Related Work

This section reviews some of the most prominent research work on function placement and traffic steering. The problem of placement of functions deals with the efficient instantiation of virtual function (VF) instances on processing nodes and routing traffic through them in a particular order, such that the overall cost is minimized and the demand of the system is satisfied. The problem of allocating resources in cloud providers is well-studied [8]–[15]. Goudarzi and Pedram [8] study the allocation of resources in multi-tier cloud providers based on SLA requirements. Gupta et al. [9] propose the P-ART framework for placing virtual network services in multi-cloud systems such that the service requirements are met. Su et al. [10] consider the affinity and conflict-aware placement of virtual machines in heterogeneous data centers. Pires and Barán [11] solve the virtual machine placement problem using a multi-objective formulation. Our work is different in that we look at the application VF placement, where applications have chains of functions that need to be placed on possibly multiple cloud nodes, and the order of flow traversal needs to be preserved. Unlike traditional function placement in clouds, these applications have strict service chaining and end-to-end performance requirements. Furthermore, different application flows have additional service chain requirements. Thus, a virtual service graph with resource requirements is created for each flow. This graph is embedded in a virtualized physical infrastructure, as shown in Figure 1a. The task of creating and deploying virtual service chains is similar to the Virtual Network Embedding (VNE) problem for Virtual Network Functions (VNF) [16], [17]. VNF orchestration is a widely studied topic, and different VNF placement schemes have been proposed [18], [19]. Similar to VNF placement, the application VF placement problem is $\mathcal{NP}$-hard. VNF placement approaches can be divided into the following three main categories.

1) Mathematical Optimizations: In this approach, the problem is formulated as a mathematical optimization problem and solved using different optimization algorithms [12]–[14], [18]–[22]. These formulations have an objective function, and the aim is to maximize or minimize the objective while satisfying the constraints. Qazi et al. [23], and Moens and Turck [24], aim to minimize the number of VNF instances being deployed. Mohammadkhan et al. [25] aim to minimize the total number of CPU cores used in cloud providers. Lin et al. [26] minimize the usage cost of cloud nodes. Gupta et al. [27], and Botero and Hesselbach [28], aim to minimize network resource costs, such as minimize energy consumption and total network bandwidth being used. Other works aim to maximize performance, such as minimize flow latencies [19] and minimize the maximum link utilization in the network [29]. Moreover, VNF placement algorithms model different aspects of the network. These include modeling the link capacities, link latencies, end-to-end flow delays, limits on the number of SDN routing rules in the switches, splitting/non-splitting of flows, and processing/storage capacities of nodes [23]–[26]. Tomassilli et al. [12] show that the VNF placement problem can be viewed as a set cover problem and proposes logarithmic factor approximation algorithms. Tajiki et al. [13] provide an Integer Linear Programming (ILP) formulation to solve the VNF placement and traffic steering problem. Other approaches use variations of ILP to solve this problem [30], [31]. Agarwal et al. [21] propose a queuing-based model. Since the problem of placement and routing is $\mathcal{NP}$-hard, these techniques are too slow to converge for most real-world networks. To solve the problem in a reasonable time, different techniques are used to obtain sub-optimal solutions quickly. Qazi et al. [23] use linear programming relaxation for routing while optimizing function placement. Gupta et al. [27] find K shortest paths through processing nodes and optimizes the VNF placement along these paths. The most widely used approach to solving this problem in a reasonable time is using heuristic algorithms, which we describe next.

2) Heuristics: To deal with the time inefficiency of *Mathematical Optimization* based solutions, heuristic approaches are used. However, heuristics provide no guarantee on the quality of the solution. Mohammadkhan et al. [25] use a multi-step greedy heuristic for function placement. Nguyen et al. [14] present an approximate algorithm for large scale problem instances that applies iterative rounding and variable fixing techniques to find the solution.

Khebbache et al. [32] propose an algorithm which makes use of multi-stage graph construction and max-flow to efficiently place VNFs. Leivadeas et al. [33] use a meta-heuristic algorithm based on Tabu search for VNF placement with the goal of minimizing the end-to-end communication delays, and deployment costs. Even though heuristics give sub-optimal solutions, it is the most popular approach for real-world applications and has been widely adopted.

3) Machine Learning (ML): Recently, several studies have suggested using various Machine Learning (ML) techniques to address the challenges of VNF placement and routing [15], [34]–[37]. DeepOpt [38] combines Deep Reinforcement Learning (DRL) and Graph Neural Network (GNN) for the VNF placement policy for different network topologies. Mijumbi et al. [39] suggest a distributed multiagent learning based approach for VNF-service chain allocation. Most of these approaches work similarly, where they explore different possible states of the network and optimize performance and cost. However, research in [40], [41] shows that these ML-based approaches can be ineffective for larger problem sizes, and they cannot be deployed for most real-world applications.

Although the VNF placement and routing problem is similar to application VF placement and routing, approaches designed for the former cannot be imported directly for the latter. Unlike the VNF case, which deals with network functions, our workload model captures virtual service chains that correspond to future applications' needs. We assume that next-generation networks will use multiple link technologies, and we provide a detailed model for modeling these new technologies and their impact on the network and applications. We provide a detailed link cost and performance modeling based on the link technology being used – the link cost function takes into account multiple cost metrics to accurately model the link technologies. Moreover, we provide optimal placement based on a detailed system model that captures many complexities that arise with virtualized services for a next-generation mobile network, including multi-technology wired and wireless links, detailed service demands, and realistic node/link constraints. For example, depending on the specific VF, the output rate from the VF may increase or decrease the input rate into the VF. We also provide a heuristic that solves the VF placement problem in a multi-technology edge network. The heuristic has polynomial runtime complexity and quickly solves the problem while sacrificing little on the quality of the solution.

IV. System Model

This section describes our envisioned system model for edge computing. We also describe our use cases (augmented and virtual reality applications) which have stringent processing and communication requirements that "thin" clients/mobile devices and traditional networks fail to support.

Our model of the infrastructure consists of a multi-technology edge network, where nodes are connected with wireless *mmWave* and wired links, as shown in Figure 1b. Nodes that are closer than a threshold distance are connected with *mmWave* links. There are two types of nodes in the network. Routing Nodes (RN) are OpenFlow enabled routers that forward packets to the next hop toward their destination. Processing Nodes (PN) are RNs with processing power, so a PN can also host Virtual Functions (VFs). A PN has multiple processing cores. For simplicity, we assume that a single core can only host a single VF instance.

There are costs associated with using the network. There is a fixed cost of running a VF instance on a PN. There are two different types of cost associated with using a communication link, namely, fixed cost and usage cost. A fixed cost is incurred if the link is being used, regardless of the amount of traffic flowing through the link. A usage cost is based on the cost per unit of traffic flowing through the link.

Each flow in the network has a source node, destination node, capacity demand, delay demand, and service chain. The capacity demand is the bit rate that a flow needs on each link as it goes from its source to destination. The delay demand is the maximum delay that packets of the flow can experience as they move from the source to destination. A service chain, as we discussed earlier, is an ordered list of VFs that the flow should pass through before reaching the destination node. This is shown in Figure 1a where an application flow passes through VFs running *Authentication*, *Processing*, and *Caching* before reaching the destination application server.

Online vs Offline: The resource allocation problem consists of placement of VFs and traffic steering, and it can be done either *online* or *offline*. In the *online* case, the resources are dynamically allocated for each flow as the flow arrives to the system. In the *offline* case, all the flow demands are known in advance and the resources are simultaneously allocated for all flows. Both the *online* and *offline* cases are $\mathcal{NP}$-hard [42]. The *offline* resource provisioning case is not always possible, especially when users' behavior cannot be accurately predicted. In this work, we only consider the *online* case. In the next section, we provide a detailed Binary Integer Programming (BIP) formulation for this problem, which can be used for both *online* and *offline* cases. Note that the *online* case is merely the *offline* case with a single flow.

To evaluate our system, we model the workload of service chains inspired by applications such as augmented and virtual reality applications. These applications have stringent requirements, and are described as "killer applications" for the next-generation mobile network [1], [2]. For example, VR applications requires high throughput and ultra-low latency. It is believed that VR applications, where users interact with other users, would need bandwidth up to 500Mbps and latency less than 5ms [1]. The challenge in advancing and deploying such applications is that traditional architectures (using remote clouds/datacenters) fail to satisfy such stringent requirements. To overcome this challenge, the VR application should be refactored as a chain of VFs that get deployed at the edge cloud. For instance, the 3D distributed game described in [43] may be decomposed into a chain of VFs as illustrated in Figure 2. The aim is to move most computation from Application Servers to the edge network, to reduce latency and increase throughput. As shown in Figure 2, when a user's request arrives, it first goes through the *Authentication and Access Control* VF to identify the user and check if the user is allowed to make the request. The request then moves to the *Processing and Storage* VF where the request is processed and actions are taken. These actions are also propagated to application servers over the Internet to update the global state of the game. This VF also has storage capability so it can provide caching and deliver data directly to the user. The delivered data finally

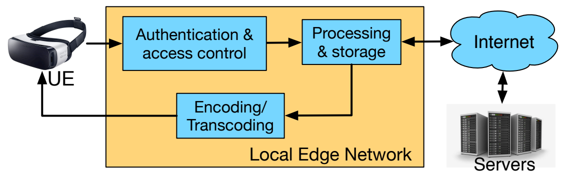


Figure 2: Virtual Reality use case

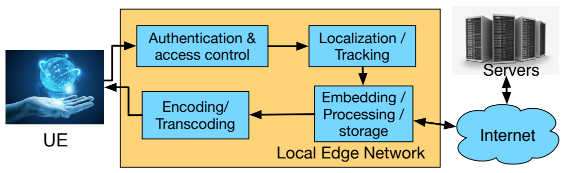


Figure 3: Augmented Reality use case

moves through the *Encoding/Transcoding* VF, where data is encoded/transcoded before being sent to the user. A similar case for AR is shown in Figure 3.

We choose augmented and virtual reality applications to test our proposed system since they have one of the highest throughput and one of the lowest latency requirements [1], [2]. Other applications with a different set of throughput and latency requirements can also use our proposed system.

V. Mathematical Model

In this section, we present the Binary Integer Programming (BIP) formulation for the joint placement of virtualized services (VFs) and traffic steering across the service chains. Although our formulation targets our envisioned next-generation mobile network, the system model described in Section IV can be applied to other scenarios by making appropriate changes to cost functions or constraints. Our model can be used by the "service" provider to optimally allocate resources to service chains, as we describe in this section. Specifically, we aim to minimize the operational (OPEX) cost by maximizing the resource utilization of the physical infrastructure. All network parameters are described in Table I. (Later in Section VI, we use this model, in conjunction with a network graph generation model, to also understand the benefits of deploying *mmWave* links from the point of view of the "infrastructure" provider.)

Notation	Description
$G(V, E)$	Network graph, V is the set of nodes: Routing Nodes (RNs) and Processing Nodes (PNs), and E is the set of all links (u, v).
$w_{(u,v)}$	binary $\{0,1\}$: 1 if there exists a physical link between nodes u and v, 0 otherwise.
$c(u, v)$	Capacity of link (u, v).
$l(u, v)$	Latency of link (u, v).
$k^c_{(u,v)}$	Fixed cost of using link (u, v). If any amount of traffic, greater than zero, passes through link (u, v), we incur this cost.
$k^d_{(u,v)}$	Usage cost of using link (u, v). It is the cost of unit flow that passes through link (u, v).
h^n_i	Fixed cost of instantiating a VF instance of type n on node $i \in V$.
O_v	Set of cores available at node $v \in V$. Each core can support one VF.
U_s	Load (in Mbps) that can be served by a single VF $s \in S$.
M^n_i	binary $\{0,1\}$: 1 if VF $n \in S$ can be supported at node i, 0 otherwise.
ϕ_s	Ratio of outgoing to incoming flow rate through VF $s \in S$.

Table I: Network Parameters.

In our model, a physical (or logical) network $G(V, E)$ is made up of nodes V, and links E between the nodes. Each link has capacity $c(u, v)$ and latency $l(u, v)$. The fixed cost of using a link is given by $k^c_{(u,v)}$, and it captures the cost incurred if the link is used by any flow. The usage cost of a link is given by $k^d_{(u,v)}$, which represents the cost per unit of flow that passes through the link. The cost of starting a new virtualized function instance on a Processing Node (PN) is given by h^n_i. Each PN has a set of cores available O_v, and each virtual function runs on a single core. Each VF has a load limit U_s (in Mbps) on the total bit rate that is served by the VF instance. Each PN can host certain types of VF n, as indicated by M^n_i. The volume of the incoming flow and outgoing flow through a VF changes based on the computation performed by the VF, *e.g.*, an encryption VF encrypts incoming traffic, so the amount of outgoing traffic leaving the VF is more than the incoming traffic. The ratio of the outgoing bit rate (in Mbps) over the incoming bit rate (in Mbps) for a VF is given by ϕ_s.

Notation	Description
F	Set of all flows in the network.
s^f	Start node of flow $f \in F$.
t^f	Destination node of flow $f \in F$.
d^f	Initial capacity demand of flow $f \in F$.
l^f	Latency demand of flow $f \in F$. Maximum delay that a flow $f \in F$ can tolerate on the path from source to destination.
K	Set of all different VFs that can be placed on nodes.
C^f	Service chain of flow $f \in F$. Set of VFs that flow $f \in F$ needs to traverse in a specific order, *i.e.* $n_1 \to n_2 \to ... \to n_l$, where $n_i \in K$.
C^f_{st}	$C^f_{st} = [n_{s^f} \to C^f \to n_{t^f}]$. The service chain of flow $f \in F$ which includes s^f and t^f nodes. To ensure that the flow starts at node s^f and ends at node t^f, two imaginary VFs n_{s^f} and n_{t^f} are introduced at s^f and t^f nodes, respectively. Since VFs n_{s^f} and n_{t^f} are only present at s^f and t^f nodes, these nodes are selected as the start and end nodes on the flow's path.
$d^{f(m\to n)}$	Capacity demand of flow $f \in F$ from VF m to n. $$d^{f(m\to n)} = d^f \prod_{i=s^f}^{m} \phi_i, \quad (\text{note}: \ \phi_{s^f} = 1)$$

Table II: Traffic Parameters.

Table II shows the traffic parameters. Each flow f in the network has a start node s^f, destination node t^f, initial capacity demand (in Mbps) d^f, latency demand (in milliseconds) l^f, and a service chain C^f. A flow is unsatisfied if any of its constraints are not met. As the flow traverses through the VFs in its service chain, its capacity demand changes based on the VF's ϕ_i. The capacity demand of a flow between two VFs is given by $d^{f(m\to n)}$.

A. Variables

Table III describes our model variables in detail. This includes decision variables, and derived variables (*i.e.*, variables dependent on decision variables).

B. BIP Formulation

1) Objective Function: Our objective is to find the optimal placement of VFs that minimizes the resource fragmentation in the system, *i.e.*, maximizes the utilization of resources. Since physical resources in the network are usually leased or rented from third parties, we aim to maximize the utilization

Variables	Description
$x^{f(m\to n)}_{(u,v)}$	binary {0,1}: 1 if link (u,v) is used to reach from VF m to n in the service chain C^f of flow $f \in F$, and 0 otherwise.
$x_{(u,v)}$	binary {0,1}: 1 if any flow uses link (u,v), and 0 otherwise. Note that it is not a decision variable, as it can be derived from $x^{f(m\to n)}_{(u,v)}$. $x_{(u,v)} = 1$ if $\sum_{f\in F}\sum_{(m\to n)\in C^f_{st}} x^{f(m\to n)}_{(u,v)} + \sum_{f\in F}\sum_{(m\to n)\in C^f_{st}} x^{f(m\to n)}_{(v,u)} > 0 \quad \forall (u,v)\in E \qquad (1)$ and 0 otherwise. Equation (1) above can also be written as a set of linear constraints as shown below. $x_{(u,v)} \le \sum_{f\in F}\sum_{(m\to n)\in C^f_{st}} x^{f(m\to n)}_{(u,v)} + \sum_{f\in F}\sum_{(m\to n)\in C^f_{st}} x^{f(m\to n)}_{(v,u)} \quad \forall (u,v)\in E$ $x_{(u,v)} \ge x^{f(m\to n)}_{(u,v)} \quad \forall f\in F,\ \forall (m\to n)\in C^f_{st},\ \forall (u,v)\in E$ $x_{(u,v)} \ge x^{f(m\to n)}_{(v,u)} \quad \forall f\in F,\ \forall (m\to n)\in C^f_{st},\ \forall (u,v)\in E$ Since $x_{(u,v)}$ is symmetrical, we also want to enforce that $x_{(u,v)} = x_{(v,u)}$
S^{fn}_{ia}	binary {0,1}: 1 if VF $n \in C^f_{st}$ is placed at core a of node i for flow $f \in F$, and 0 otherwise.
X^n_{ia}	binary {0,1}: 1 if any VF $n \in K$ is placed on core a of node i, 0 otherwise. Note that it is not a decision variable as it can be derived from S^{fn}_{ia}. $X^n_{ia} = 1$ if $\sum_{f\in F} S^{fn}_{ia} \ge 1 \quad \forall n\in C^f, \forall i\in V, \forall a\in O_i \qquad (2)$ and 0 otherwise. Equation (2) above can also be written as a set of linear constraints as shown below. $X^n_{ia} \le \sum_{f\in F} S^{fn}_{ia} \quad \forall n\in C^f, \forall i\in V, \forall a\in O_i$ $X^n_{ia} \ge S^{fn}_{ia} \quad \forall n\in C^f, \forall i\in V, \forall a\in O_i, \forall f\in F$

Table III: Variables.

of resources that are already in use as long as we can satisfy the flow demands. Following are the costs that we consider, and we aim to minimize.

VF Deployment Cost: To run a VF on a node, we assume a pricing/cost model that is similar to Amazon EC2 "dedicated host", in which a fixed cost is paid for leasing/renting the node on which the VF instance is run.

$$\mathbb{V}_c = \sum_{i\in V}\sum_{n\in K}\sum_{a\in O_i} h^n_i X^n_{ia} \tag{3}$$

Link Fixed Cost: If a link is used (in any direction) by any of the flows, regardless of the flow demand, we pay a fixed cost. Different link technologies (namely, *Wire* and *mmWave* links) can have different fixed costs, which we explain in detail later in Section VI.

$$\mathbb{E}_c = \sum_{(u,v)\in E} k^c_{(u,v)} x_{(u,v)} \quad \forall\ u > v \tag{4}$$

Link Usage Cost: This link usage cost is based on the amount of link resources used by flows. It represents the cost per unit of flow going through a link.

$$\mathbb{E}_d = \sum_{(u,v)\in E} k^d_{(u,v)} \sum_{f\in F}\sum_{(m\to n)\in C^f_{st}} x^{f(m\to n)}_{(u,v)} d^{f(m\to n)} \tag{5}$$

Our objective is to minimize the cost of the system and fragmentation of the resources in the system, while satisfying the flow demands. The objective function is given by:

$$minimize(\ \mathbb{V}_c + \mathbb{E}_c + \mathbb{E}_d\ )$$

This cost minimization is subject to the following constraints:

2) Link Capacity Constraint:

$$\sum_{f\in F}\sum_{(m\to n)\in C^f_{st}} d^{f(m\to n)} x^{f(m\to n)}_{(u,v)} \le c(u,v) \qquad \forall (u,v)\in E \tag{6}$$

Each link has a capacity limit. The aggregate input rate of all flows passing through a link should not exceed the capacity of the link.

3) Flow Latency Constraint:

$$\sum_{(m\to n)\in C^f_{st}}\sum_{(u,v)\in E} l(u,v) x^{f(m\to n)}_{(u,v)} \le l^f \qquad \forall f\in F \tag{7}$$

Each flow has a latency constraint. A flow, moving from source to destination, should not experience latency greater than its (end-to-end) latency requirement. Here we are only considering network delays, *i.e.,* propagation and transmission delays.

4) Physical Link Constraint:

$$x^{f(m\to n)}_{(u,v)} \le w_{(u,v)} \qquad (m\to n)\in C^f_{st} \tag{8}$$

A virtual link along the path of a flow should be using one of the existing physical links given by $w(u,v)$.

5) Flow Constraint:

$$\sum_{j\in V} x^{f(m\to n)}_{(i,j)} - \sum_{k\in V} x^{f(m\to n)}_{(k,i)} = \sum_{a\in O_i} S^{fm}_{ia} - \sum_{a\in O_i} S^{fn}_{ia} \tag{9}$$

$\forall i \in V, \quad (m \to n) \in C^f_{st}$, where VF n is after VF m in the service chain C^f_{st}.
This constraint ensures that there is a single continuous path between the pair of nodes on which VFs m and n are placed, where m and n are nodes of service chain C^f_{st}, and traversed in the order $(m \to n)$.

6) VF Placement Constraint:

$$S^{fn}_{ia} \le M^n_i \qquad \forall f\in F, \forall n\in C^f_{st}, \forall i\in V, \forall a\in O_i \tag{10}$$

VF $n \in C^f_{st}$ can only be hosted on nodes that can host VF n.

7) Single VF Node Selection Constraint:

$$\sum_{i\in V}\sum_{a\in O_i} S^{fn}_{ia} = 1 \qquad \forall n\in C^f_{st} \tag{11}$$

Only a single node is selected to host a VF in the service chain C^f_{st} of flow $f \in F$.

8) Node Capacity Constraint:

$$\sum_{n\in K}\sum_{a\in O_i} X^n_{ia} \le |O_i| \quad \forall i\in V \tag{12}$$

Each free core at a node can host a single VF. The number of VFs hosted at a node is limited by the number of cores available at that node.

9) VF Capacity Constraint:

$$\sum_{f\in F} d^{f(m\to n)} S^{fn}_{ia} \le U_n \quad \forall i\in V,\ \forall a\in O_i,\ \forall n\in C^f \tag{13}$$

Each VF at a node has a capacity limit and can only serve flow demands (in Mbps) within that limit.

10) Single VF per core:

$$\sum_{n \in C^f} S_{ia}^{fn} <= 1 \qquad \forall f \in F \quad \forall i \in V \quad \forall a \in O_i \qquad (14)$$

Each core at a node can host at most one VF.

VI. Evaluation Model, Parameters and Proposed Heuristic

In this section, we present our evaluation model and parameters for both the edge network and the workload of VR and AR service chains. We then provide a description of our proposed heuristic.

A. Edge Network Graphs

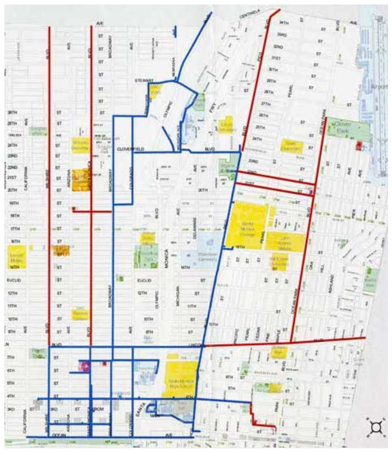
We used different types of edge network topologies for the simulation. We generated synthetic graphs using BRITE [44], a widely used network graph generator. Moreover, we used real edge network topologies for two cities, Santa Monica (CA, USA) and Palo Alto (CA, USA). Next, we explain the topologies generated using each technique.

1) Synthetic Edge Network Topology using BRITE: BRITE is a widely used network graph generator [44]. BRITE supports multiple graph generation models, including models for flat and hierarchical graphs. BRITE separates the placement of the nodes from the process of growing the graph and interconnecting the nodes. We use BRITE's *random node placement* model for placing nodes in a plane, and BRITE's *Waxman* model for interconnecting the nodes probabilistically [45]. BRITE network has 25 nodes and node density of 6.25 $nodes/km^2$.

2) Real Edge Network Topology: We used real city network topologies to generate edge networks. We used the optical fiber network topology of two cities in the USA. The original wired network topologies for Santa Monica and Palo Alto are shown in Figures 4a and 5a, respectively. We mapped the original topologies onto the Google Maps [46], as shown in Figures 4b and 5b. Mapping a topology on Google Maps helped us in finding the coordinates of different points on the maps. The original maps do not show the router nodes. We assumed there exists a router node at the intersection of the wires and the end of each wire. Moreover, we introduced some additional router nodes within long-distance links for the Santa Monica topology.

Santa Monica:
We used the wired network topology for Santa Monica, CA, USA. As shown in Figure 4, Santa Monica's wired network topology is a distributed bus topology [47], with few connections between backbone bus lines. Distributed bus topology is a widely used edge network topology for city networks. Santa Monica wired network has 43 nodes and a node density of 3.61 $nodes/km^2$.

Palo Alto:
We used the wired network topology for Palo Alto, CA, USA. As shown in Figure 5, Palo Alto's wired network topology is a ring topology with few connections across the diameter of the ring network. Ring topology is a widely used edge network topology for city networks. Palo Alto's wired network has 36 nodes and a node density of 0.91 $nodes/km^2$.


(a) Original Topology

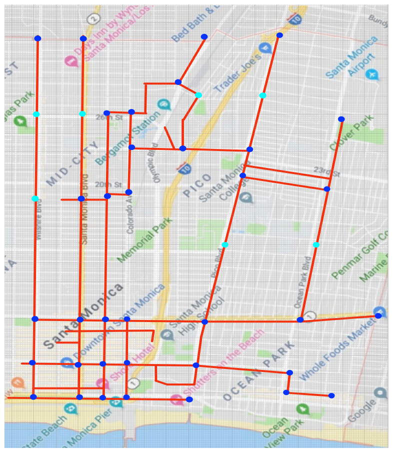

(b) Generated Topology

Figure 4: Santa Monica network topology.

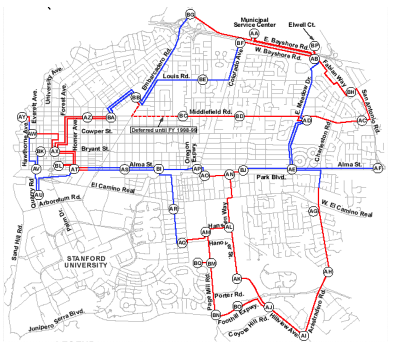
3) Adding mmWave links: The initial topologies that we generate represents a base edge network that consists of only wired links. We then augment this base graph with *mmWave* links to obtain three different types of graph, which are described next.

Wire: This is the initial graph generated with only wired links. An example of such graph is shown in Figure 6a.

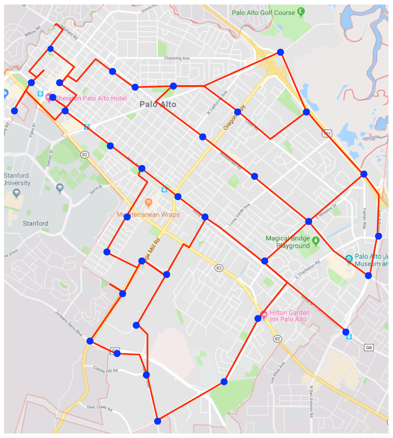
Single: *mmWave* links are added to the *Wire* graph if the distance between any two nodes in the graph is less than a given distance/*mmWave*-range (elaborated on below). However, if there is already a wired link between the two nodes, a *mmWave* link is not added. So we have only a *single* type of link technology (*mmWave* or *Wire*) between any two nodes, as shown in Figure 6b.

Dual: *mmWave* links are added to the *Wire* graph if the distance between any two nodes in the graph is less than a given distance/*mmWave*-range (elaborated on below). In this scenario, two nodes may have *dual* technology links, *i.e.*, both *mmWave* and *Wire* links, as shown in Figure 6c. *Dual* has the maximum number of possible *mmWave* links between nodes in the network.

Characteristics of different graphs for our evaluation, in terms of nodes, area and links, are summarized in Table IV. Graphs generated using BRITE covers a small area of 4.0



(a) Original Topology



(b) Generated Topology

Figure 5: Palo Alto network topology.

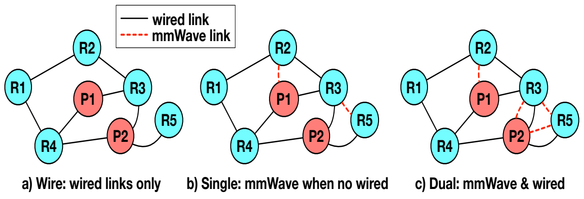


Figure 6: Multi-technology edge network consisting of processing and routing nodes.

km^2 and has the highest node density of 6.25 $nodes/km^2$. Santa Monica has an area of 11.90 km^2 and a node density of 3.61 $nodes/km^2$ and Palo Alto has an area of 39.65 km^2 and the lowest node density of 0.91 $nodes/km^2$. Because of the differences in the node density, BRITE has the highest percentage of *mmWave* links. Santa Monica has the second highest percentage of *mmWave* links and Palo Alto has the lowest percentage of *mmWave* links.

Table V shows the various parameters used in our evaluation campaign. The range of a *mmWave* link is defined by variable $range_{mm}$. Two nodes in the network cannot have a *mmWave* link if their distance is beyond $range_{mm}$. $range_{mm}$ is chosen to be 500m, which can be achieved in urban environments with Line Of Sight (LOS) connections [48]. The capacity of *mmWave* links can vary in the 1 Gbps–10 Gbps range, based on channel conditions [49]. We have taken the link capacity $c(u,v)_{mm}$ to be 2 Gbps for *mmWave* links [49], and $c(u,v)_w$ to be 10 Gbps for *Wire* links.

The fixed cost for using a *mmWave* link, $k^{c_{mm}}_{(u,v)}$, is kept low by setting it to 1, since it is less costly to establish *mmWave* links between two sites if they are within the range $range_{mm}$. On the other hand, the fixed cost for *Wire* links is higher, and so we set it to 50, since *Wire* links are usually leased / rented from an infrastructure provider.

Type	# Nodes	Area	Technology	avg. # of links	%age of links
BRITE					
Dual	25	4.0 km^2	*mmWave*	47.4	65.5
			Wire	25	34.5
Single	25	4.0 km^2	*mmWave*	35.6	58.8
			Wire	25	41.2
Wire	25	4.0 km^2	*mmWave*	0	0.0
			Wire	25	100
Santa Monica					
Dual	43	11.90 km^2	*mmWave*	90	62.50
			Wire	54	37.50
Single	43	11.90 km^2	*mmWave*	65	54.62
			Wire	54	45.38
Wire	43	11.90 km^2	*mmWave*	0	0.0
			Wire	54	100
Palo Alto					
Dual	36	39.65 km^2	*mmWave*	44	50.57
			Wire	43	49.43
Single	36	39.65 km^2	*mmWave*	18	29.51
			Wire	43	70.49
Wire	36	39.65 km^2	*mmWave*	0	0.0
			Wire	43	100

Table IV: Graph Parameters and Characteristics

Parameter	Description	Value
$range_{mm}$	*mmWave* range	500 m
$c(u,v)_{mm}$	Capacity of *mmWave* links	2 Gbps
$c(u,v)_w$	Capacity of *Wire* links	10 Gbps
$k^{c_{mm}}_{(u,v)}$	Fixed cost for using *mmWave* link	1
$k^{c_w}_{(u,v)}$	Fixed cost for using *Wire* link	50
$k^{d_{mm}}_{(u,v)}$	Cost per unit flow for using *mmWave* link	$1/PS$
$k^{d_w}_{(u,v)}$	Cost per unit flow for using *Wire* link	1
$l_{(u,v)}$	Latency of link (u,v) is the sum of propagation and transmission delays	-
h^n_i	Fixed cost of instantiating a VF instance of type n on node i	200
$\|O_v\|$	Number of cores available at processing node v	10
U_s	Capacity of the VF s	15 Gbps
$ratio_{PN}$	Ratio of processing nodes to routing nodes	0.3

Table V: Evaluation Parameters

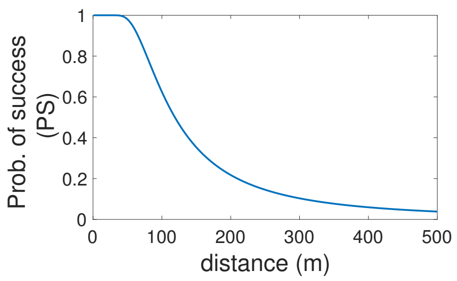


Figure 7: Probability of successful bit delivery over a *mmWave* link

The usage cost for *mmWave* links, $k^{d_{mm}}_{(u,v)}$, depends on link performance and is set to $1/PS$, where PS is the probability that a bit sent over the link successfully reaches the other side. PS is obtained using the empirical studies on *mmWave* technology described in [50], [51]. Figure 7 shows PS as a function of distance. Note that the usage cost $k_{(u,v)}^{d_{mm}}$ becomes significantly higher as the distance between the two nodes connected via a *mmWave* link increases. Hence, shorter *mmWave* links are favored over longer *mmWave* links. For *Wire* links, the usage cost $k_{(u,v)}^{d_w} = 1$, since the cost (delivery performance penalty) associated with using *Wire* is relatively much lower. The latency of a link is given by $l(u, v)$, and is equal to the sum of propagation and transmission delays. Note that there will be zero or negligible queuing delays when demands match allocated capacities.

We select a fraction of the nodes in the network graph to be processing nodes (PNs). This ratio, denoted by $ratio_{PN}$, is set to 0.3, *i.e.* only 30% of the nodes are PNs. Each PN node has $|O_v|$ cores available, and we set $|O_v| = 4$. This means that each PN can host at most 4 VFs. The capacity of a single VF U_s is set to 15 Gbps. The cost associated with instantiating a VF h_i^n is set to 200. It represents the cost of leasing a virtual machine or container from the edge datacenter. A high value has the effect of packing as many flows as possible on a VF as long as the flow demands can still be fulfilled. Next, we explain the process of selecting processing nodes.

4) Processing Node Selection: We assume that any node in the network can be chosen as a processing node. $ratio_{PN}$ fraction of nodes are selected as processing nodes. Processing nodes are selected such that the sum of the distances from each node to the closest processing node is minimized.

We formulate this problem as Integer Linear Program (ILP). The distance from the node i to processing node (PN) p is denoted by c_{pi}. X is the total number of processing nodes. Note that the nodes and the processing nodes share the same set of points. We define the following variables:

$z_{pi} = 1$ if node i is satisfied by PN p, 0 otherwise

$x_p = 1$ if PN p is being used, 0 otherwise

We formulate the problem as integer-optimization model.

$$minimize \sum_{p \in N} \sum_{i \in N} c_{pi} z_{pi}$$

Subject to:

1. The total number of processing nodes are equal to X

$$\sum_{p \in N} x_p = X$$

2. A single processing node p is selected for each node i

$$\sum_{p \in N} z_{pi} = 1 \quad \forall i \in N$$

3. $x_p = 1$ if node p is selected as PN

$$z_{pi} \leq x_p \quad \forall p \in N \quad \forall i \in N$$

$$\sum_{p \in N} z_{pi} \geq x_p \quad \forall x \in N$$

We used CPLEX solver[1] to solve the ILP above for the selection of processing nodes for Santa Monica and Palo Alto network graphs. Since the graphs generated using BRITE are synthetic, we generated multiple graph topologies and randomly selected processing nodes.

[1]IBM ILOG CPLEX Optimizer,
http://www-01.ibm.com/software/integration/optimization/cplex-optimizer

VR Flow		
Parameter	**Description**	**Value**
d_{VR}^f	Initial flow demand	μ = 10 Mbps σ = 2 Mbps
l_{VR}^f	Latency demand	μ = 5 ms σ = 1 ms
$\phi_{A\&AC}$	Ratio of outgoing to incoming flow rate through the Authentication & access control VF	0.9
$\phi_{P\&S}$	Ratio of outgoing to incoming flow rate through the Processing & storage VF	20
$\phi_{E\&T}$	Ratio of outgoing to incoming flow rate through the Encoding / Transcoding VF	0.8
AR Flow		
Parameter	**Description**	**Value**
d_{AR}^f	Initial flow demand	μ = 150 Mbps σ = 20 Mbps
l_{AR}^f	Latency demand	μ = 4 ms σ = 1 ms
$\phi_{A\&AC}$	Ratio of outgoing to incoming flow rate through the Authentication & access control VF	0.9
$\phi_{L\&Tk}$	Ratio of outgoing to incoming flow rate through the Localization / Tracking VF	0.9
$\phi_{E/P/S}$	Ratio of outgoing to incoming flow rate through the Embedding / Processing / Storage VF	1
$\phi_{E\&T}$	Ratio of outgoing to incoming flow rate through the Encoding / Transcoding VF	0.8

Table VI: Flow Parameters

B. Input Flow Parameters

There are two different types of flow in the network, each type has different service chain requirements representing either Virtual Reality (VR) or Augmented Reality (AR). For each of the generated network graphs, we generate five sets of flows, where each incoming flow is either VR or AR flow with probability 0.5. Each flow starts and ends at the same node (representing the user/client), which is randomly selected. We only consider the allocation of the service chains on the edge network. Flow parameters for VR and AR flows are described in Table VI.

C. Proposed Heuristic

We used the CPLEX solver to solve the BIP that we described in Section V-B. The running time for obtaining the optimal solution for each of our evaluations is very high because of the $\mathcal{NP}$-hardness of the problem. The problem has exponential worst-case complexity. To reduce the running time, Algorithm 1 shows a fast heuristic whose solution we compare against the CPLEX solution in terms of performance and running time.

This heuristic takes a flow and returns a least cost path, while fulfilling the flow requirements. It takes the current state of the network graph (G with nodes, edges, residual link capacities, fixed and dynamic costs, processing nodes) as input, along with the input flow requirements, *i.e.*, source and destination nodes, service chain, flow latency and ϕ_i (bandwidth ratio after the use of each VF along the chain).

Initially (line 1), we use the function *getFeasibleGraph* to get a subgraph (G') from the original graph G that includes only those links that have enough capacity to satisfy the flow end-to-end rate demand. The function *getNearbyPN* (line 2) finds the q nearest (in number of hops) processing nodes

Algorithm 1 Service Chain Placement Heuristic

```
Input:
$f$: incoming flow
$G(V, E)$: Network graph, $V$ is set of nodes and $E$ is set of links
$PN$: set of processing nodes, where $PN \subseteq V$
$q$: number of nearest processing nodes used for virtual function placement
Output: $minPath$

1: $G' = getFeasibleGraph(G, f)$; // subgraph G'(V,E'), E' can carry flow demand
2: $PN_q^{s^f} = getNearbyPN(G', s^f, PN, q)$;// get set of $q$ nearby processing nodes
3: $P^f = getShortestPaths(PN_q^{s^f}, G, f)$;//all possible paths through processing nodes
4: minPath = $null$
5: minPathCost = $\infty$
6: for path $p$ in $P^f$ do
7:     if $pathFeasible(p, l^f, d^f)$ then
8:         $c_p^f = getCost(p, f)$ // cost = fixed cost + usage cost + VF placement cost
9:         if $c_p^f$ < minPathCost then
10:            minPathCost = $c_p^f$
11:            minPath = $p$
12:        end if
13:    end if
14: end for
```

($PN_q^{s^f}$) from the source (s^f) on subgraph G' using Dijkstra's shortest path algorithm. After getting $PN_q^{s^f}$ processing nodes, *getShortestPaths* is invoked (line 3), which calculates all possible least-cost paths through every permutation of the processing nodes in $PN_q^{s^f}$. While calculating paths, *getShortestPaths* makes sure that for each path, the segment from the source to the first processing node has available capacity that is at least equal to the rate outgoing from the source (d^f). Also, from the first processing node to the last processing node, it has the maximum possible capacity required by the flow, and from the last node to the destination, it has at least a capacity of $d^f \prod_{i=s^f}^{n_l} \phi_i$.

Next, we evaluate each path individually. We perform additional feasibility checks using *pathFeasible* in line 7. *pathFeasible* checks if the path's latency is less than the flow's end-to-end latency requirement and the path can provide/deploy the function chain. If the path is feasible, we calculate the cost of allocating the flow f on the path p using the function *getCost* (line 8); the cost includes link usage and VF deployment cost along the path p. Here, we take a greedy approach where we try to use VFs that are already deployed along the path, otherwise collocate other missing VFs on the same processing node(s) if feasible. After evaluating all paths in P^f, we pick the path with the lowest cost for the flow.

The worst-case time complexity of the heuristic to satisfy a single flow is $O(V^3)$. The function *getFeasibleGraph* (line 1) has a time complexity of $O(E)$ (or $O(V^2)$ for a complete graph). *getNearbyPN* (line 2) and *getShortestPaths* (line 3) use the all-pairs shortest path algorithm, which has a time complexity of $O(V^3)$. *pathFeasible* (line 7) has a time complexity of $O(V)$ and *getCost* (line 7) has a time complexity of $O(V^2)$. Hence, the overall time complexity of the heuristic is $O(V^3)$.

VII. Evaluation Results

In this section, we discuss the results of our study where we evaluate the performance and cost of allocating service chains as flows arrive to the edge network. We consider the following performance metrics: (1) *Flow Acceptance Ratio*: is the ratio of flows accepted (*i.e.*, resources are available to allocate to these flows) to the total number of flow arrivals, (2) *Virtual Capacity Allocated*: is the total virtual capacity of all links along the service chains of accepted flows, and (3) *Average Link Utilization*: is the ratio of link usage over link capacity averaged over all links, or over each of the two types of link (*Wire* and *mmWave*). Results are shown for BRITE, Santa Monica and Palo Alto topologies. For each type of typology, results with 90% confidence intervals are shown for *Wire*, *Single*, and *Dual* networks for both BIP and Heuristic.

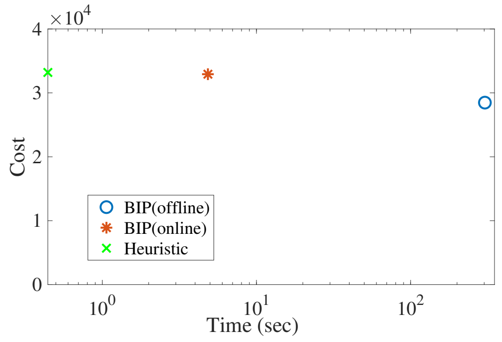
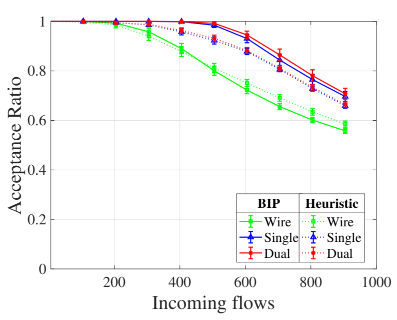


Figure 8: Cost vs running time comparison of BIP vs Heuristic

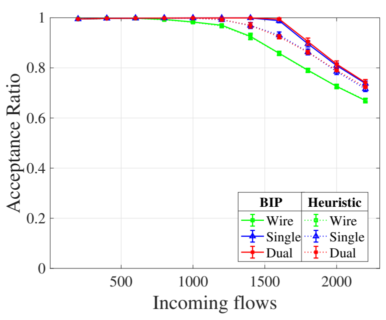
Observations: Before presenting the details of our results, we summarize our main observations as follows: (1) Augmenting the physical (*Wire*) infrastructure with *mmWave* links yields significantly higher flow acceptance ratio and virtual capacity allocated (up to 20% higher); (2) Most significant gains using *mmWave* links are in high node density networks, where nodes are closer to each other, and reliable *mmWave* links can be established between the nodes. (3) These *mmWave* links should complement the connectivity provided by *Wire* links and only a small number of *mmWave* links needs to be deployed to achieve most performance gains; (4) The flexibility in resource allocation afforded by decomposing applications into service chains that can be deployed *anywhere* on the edge infrastructure yields significant gains (up to three times higher accepted virtual capacity) over a traditional "middlebox" static deployment; and (5) The proposed heuristic decreases the running time by up to one order of magnitude when compared with BIP while giving performance results close to BIP.

The cost versus running time for BIP and our heuristic is shown in Figure 8 for the *Dual* scenario. As explained in Section IV, in the BIP *online* case, the resources are dynamically allocated for each flow as it arrives, while in the *offline* case, all flow demands are known in advance and resources are simultaneously allocated for all flows. Since *offline* has advance knowledge of all incoming flow demands, it can efficiently allocate the flows on the network and the cost is lowest. However, the running time for *offline* is orders of magnitude larger than the online case. The proposed heuristic yields a cost comparable to the BIP *online* case, with running time that is one order of magnitude lower. The *offline* resource provisioning is not always possible since we cannot accurately predict incoming flows. For this reason, in the remainder of the paper, results are shown for the BIP online case.

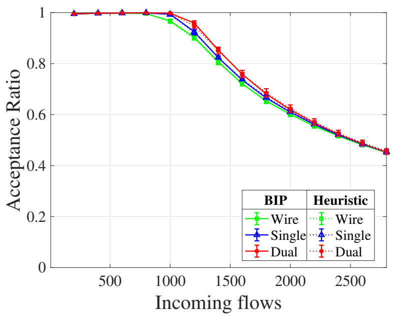
Figure 9 shows the flow acceptance ratio of BRITE, Santa Monica, and Palo Alto as a function of incoming flows for different types of network. The high node-density networks



(a) Acceptance Ratio for BRITE

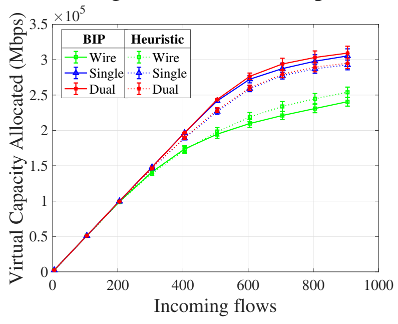


(b) Acceptance Ratio for Santa Monica

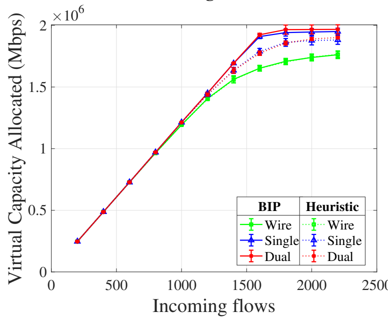


(c) Acceptance Ratio for Palo Alto

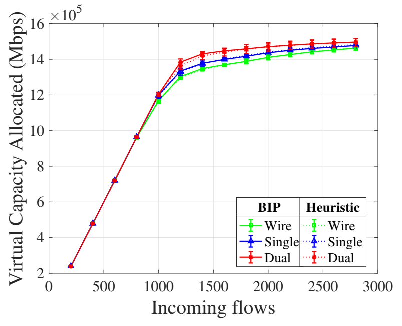
Figure 9: Flow Acceptance Ratio for *Wire*, *Single* and *Dual* networks with BIP and Heuristic



(a) Virtual Capacity Allocated for BRITE



(b) Virtual Capacity Allocated for Santa Monica



(c) Virtual Capacity Allocated for Palo Alto

Figure 10: Virtual Capacity Allocated for *Wire*, *Single* and *Dual* networks with BIP and Heuristic

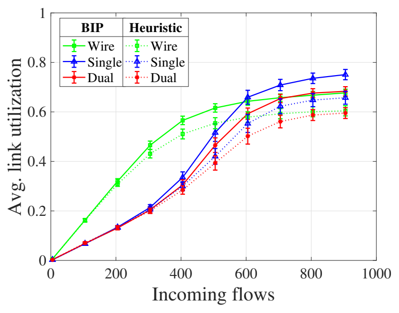
of BRITE and Santa Monica with *mmWave* links (*Single* and *Dual*) accept more flows than the same network with only wired links (*Wire*). The low-density network of Palo Alto yields little/no gain with *mmWave* links. As the density of nodes decreases, the probability of having a *mmWave* link between two nodes (within range) is small. This yields a network with sparse *mmWave* links, thus a small increase in the connectivity and capacity of the network, and a little/no increase in the accepted flows. The flow acceptance difference between *mmWave* and wire networks is more significant in high-density networks. The percentage of *mmWave* links are higher in high-density networks as nodes are closer to each other, and the probability of having *mmWave* links between two nodes (within range) is higher. This yields a network that is better connected and with increased capacity. Thus, we observe a higher number of accepted flows.

Since each flow can have different capacity requirements along its virtual service chain, the number of flows accepted does not necessarily mean that the network capacity is efficiently allocated. Figure 10 shows the virtual capacity allocated for BRITE, Santa Monica and Palo Alto. Again, we see that *Single* and *Dual* have higher virtual capacity allocated than *Wire* for high node-density networks. There is little/no gain for Palo Alto, which has low node density. For both Figure 9 and Figure 10, results obtained by the proposed heuristic are very close to BIP.

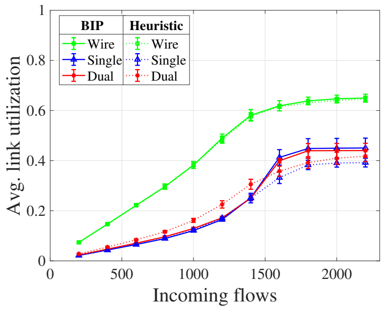
Figures 11 to 13 show the average link utilization for both *mmWave* links and *Wire* links. We observe that the *Wire* network has higher link utilization because the network has lower capacity and links get congested quickly. Figures 12 and 13 show the link utilization for *Wire* links, and for *mmWave* links, respectively. We see in Figure 12 that *Wire* links are better utilized (up to 20%) when there are *mmWave* links in the high node-density networks. The existence of *mmWave* links makes the network better connected, which leads to better utilization of the resources and higher number of flows accepted. Figure 13 shows that *mmWave* links are better utilized (up to 10% in BRITE) in *Single* networks compared to *Dual* networks, although the acceptance ratio and virtual capacity allocated for both networks are the same. However, *mmWave* links have higher usage cost. Thus, initially, when the network is not yet congested, only a few *mmWave* links are used. So initially, the average utilization for *mmWave* links is low, as shown in Figure 13. On the other hand, as more flows enter the system and the network becomes congested, more and more *mmWave* links are used to satisfy the flow demands. This leads to higher utilization of *mmWave* links, but at a higher cost. In all the graphs, the gain is significant in the high node-density networks. We also provide a comparison with the proposed heuristic. We observe that the heuristic performance is close to the performance given by BIP.

Figure 14 shows the CDF of utilization of the *mmWave* links for *Single* scenarios for BRITE, Santa Monica and Palo Alto. We observe that in the *Single* scenario, 60% of the links have utilization of less than 6%, and around 20% of the links are completely saturated with utilization close to 100%. This shows that significant performance gains can be achieved by judiciously deploying a small number of *mmWave* links.

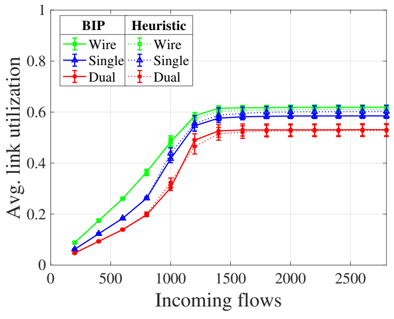
Middlebox Scenario: To highlight the benefit of using (optimal) distributed virtual NF placement, we compare it with a traditional middlebox scenario. In the middlebox scenario, a powerful hardware appliance, with all the required services, is placed at the edge of the network. For each network (*i.e.*, *Wire*, *Single* and *Dual*), we chose a single Processing Node (PN) to host the middlebox, *i.e.*. We set this middlebox to be 10 times more powerful (*i.e.*, it can serve 10 times more flows) than a



(a) Average Link Utilization for BRITE

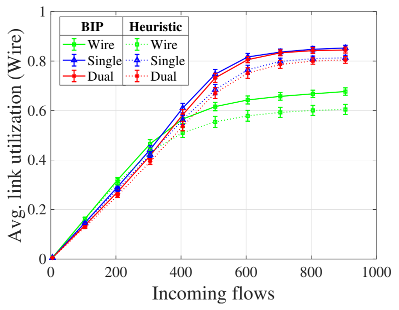


(b) Average Link Utilization for Santa Monica

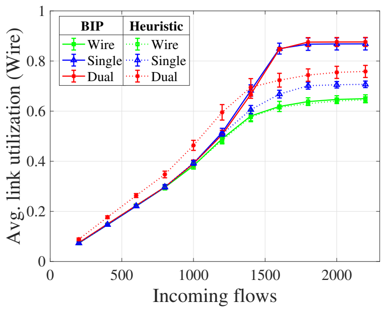


(c) Average Link Utilization for Palo Alto

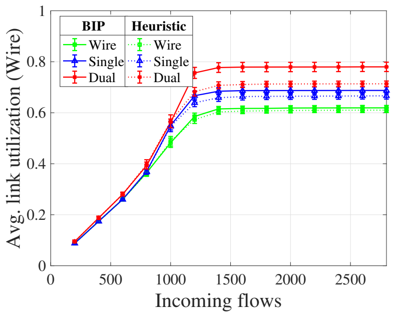
Figure 11: Average Link Utilization for all links as a function of incoming flows for *Wire*, *Single* and *Dual* networks



(a) Average Link Utilization for BRITE

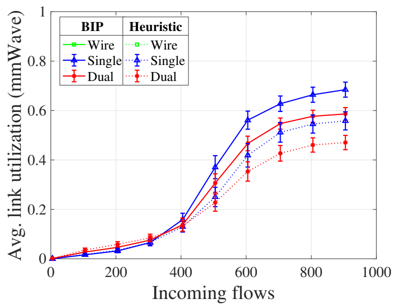


(b) Average Link Utilization for Santa Monica

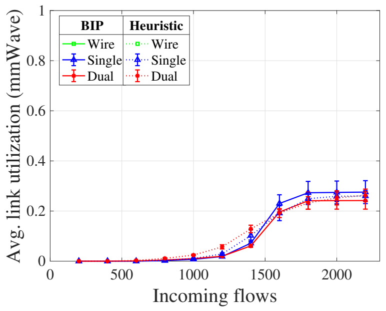


(c) Average Link Utilization for Palo Alto

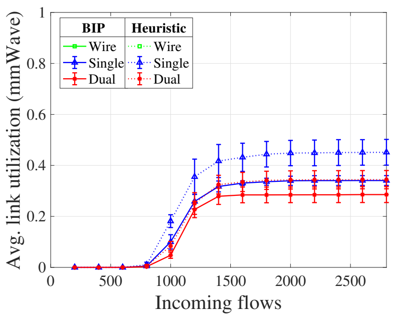
Figure 12: Average Link Utilization for *Wire* links as a function of incoming flows for *Wire*, *Single* and *Dual* networks



(a) Average Link Utilization for BRITE

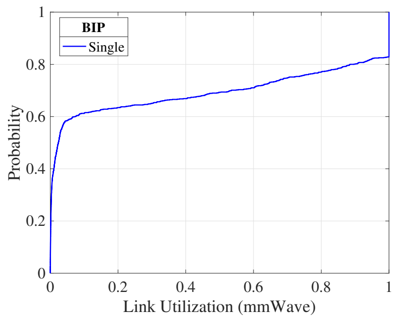


(b) Average Link Utilization for Santa Monica

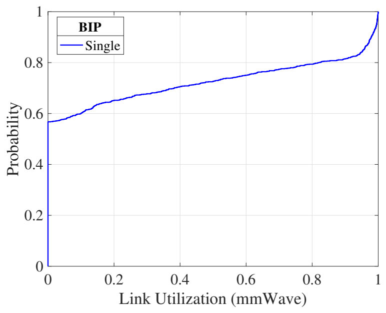


(c) Average Link Utilization for Palo Alto

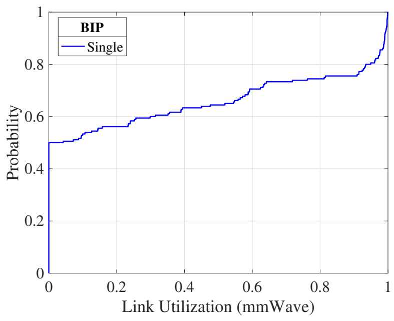
Figure 13: Average Link Utilization for *mmWave* links as a function of incoming flows for *Single* and *Dual* networks



(a) CDF of *mmWave* link utilization for BRITE

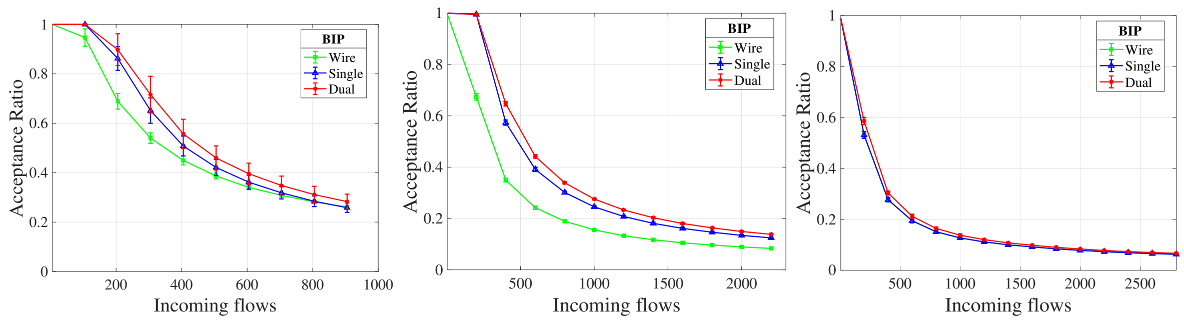


(b) CDF of *mmWave* link utilization for Santa Monica



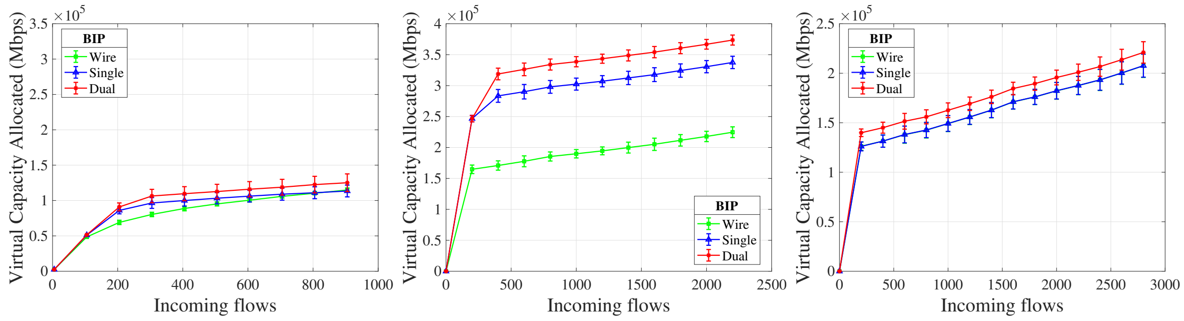
(c) CDF of *mmWave* link utilization for Palo Alto

Figure 14: CDF of *mmWave* link utilization for *Single* networks



(a) Acceptance Ratio for BRITE (b) Acceptance Ratio for Santa Monica (c) Acceptance Ratio for Palo Alto

Figure 15: Flow Acceptance Ratio under middlebox schenario for *Wire*, *Single* and *Dual* networks



(a) Virtual Capacity Allocated for BRITE (b) Virtual Capacity Allocated for Santa Monica (c) Virtual Capacity Allocated for Palo Alto

Figure 16: Virtual Capacity Allocated under middlebox schenario for *Wire*, *Single* and *Dual* networks

virtualized service placed on a PN, and it runs all the needed services. Figures 15 and 16 show the flow acceptance ratio and virtual capacity allocated, respectively, for the middlebox scenario. The number of flows accepted in the middlebox case (Figure 15) are far lower than that accepted in the distributed VF placement scenario (Figure 9). As shown in Figures 10 and 16, the virtual capacity allocated for the distributed VF placement scenario is three times higher than the traditional middlebox scenario for higher density networks.

Discussion: The results clearly show the benefits of introducing *mmWave* links in the network with higher node density. However, it is important to wisely deploy these *mmWave* links. As shown in Table IV, the *Dual* network has a larger number of *mmWave* links compared to the *Single* network. However, if we look at the marginal utility of using *Dual* over *Single*, the gains are negligible. The flow acceptance ratio and the virtual capacity allocated (Figures 9 and 10) for both cases are within the 90% confidence interval. Furthermore, the average utilization of links is higher in *Single* compared to *Dual* (Figure 11) for high density network, which means links are better utilized in the former. Figure 14 also shows that only a small number of *mmWave* links are needed to achieve most performance gains. This leads us to conclude that a small number of *mmWave* links should be introduced such that the overall connectivity between the nodes is increased, rather than to just increase the capacity of the network.

We also note that the middlebox scenario fails to take advantage of introducing *mmWave* links, as the number of flows accepted for the *Wire* network is similar to that for networks with additional *mmWave* links (Figures 10 and 16).

VIII. Conclusion

In this paper, we studied the problem of allocating resources at the edge in support of envisioned next-generation applications, *e.g.*, virtual and augmented reality. We presented a model of an edge network with multiple link technologies, namely, *Wire* and *mmWave*. We also developed a workload model that consists of the service chains with varying capacity requirements as the traffic flow traverses its chain. We formulated a binary integer optimization problem whose objective is to minimize the cost of deploying these service chains over the edge network, while satisfying their high throughput and ultra-low latency requirements. We also introduced a fast heuristic to solve the problem. Our extensive evaluations demonstrate the benefits of managing virtual service chains (by distributing them over the edge network) compared to a baseline "middlebox" approach (where all services are run on one host) in terms of overall admissible virtual capacity.

Moreover, we observe significant gains when deploying a small number of *mmWave* links that complement the *Wire* physical infrastructure. We show that a network topology with a high density of nodes has the highest performance gains since a large number of reliable *mmWave* links are formed between the nodes, thus increasing both the capacity and connectivity of the network.

Acknowledgements

This work has been partially supported by Comcast and by NSF awards CNS-1647084, CNS-1836906, CNS-1908574, and CNS-1908677.

References

[1] AT&T, "Enabling Mobile Augmented and Virtual Reality with 5G Networks," January, 2017. [Online]. Available: goo.gl/DKNbHx

[2] Qualcomm, "Augmented and Virtual Reality: the First Wave of 5G Killer Apps," February 1, 2017. [Online]. Available: https://goo.gl/kdgvCg

[3] Y. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing:A key technology towards 5G," *ETSI W. Paper*, vol. 11, 2015.

[4] N. Akhtar, I. Matta, A. Raza, L. Goratti, T. Braun, and F. Esposito, "Virtual Function Placement and Traffic Steering over 5G Multi-Technology Networks," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, June 2018, pp. 114–122.

[5] "Report ITU-R M.[IMT-2020.TECH PERF REQ] - Minimum requirements related to technical performance for IMT-2020 radio interface(s)," 2017-02-23. [Online]. Available: https://www.itu.int/md/R15-SG05-C-0040/en

[6] R. S. Montero, E. Rojas, A. A. Carrillo, and I. M. Llorente, "Extending the Cloud to the Network Edge," *Computer*, vol. 50, no. 4, April 2017.

[7] Y. Niu, Y. Li, D. Jin, L. Su, and A. V. Vasilakos, "A survey of millimeter wave communications (mmwave) for 5g: opportunities and challenges," *Wireless Networks*, vol. 21, no. 8, pp. 2657–2676, Nov 2015.

[8] H. Goudarzi and M. Pedram, "Multi-dimensional SLA-Based Resource Allocation for Multi-tier Cloud Computing Systems," in *2011 IEEE 4th International Conference on Cloud Computing*, 2011, pp. 324–331.

[9] L. Gupta, R. Jain, A. Erbad, and D. Bhamare, "The p-art framework for placement of virtual network services in a multi-cloud environment," *Computer Communications*, vol. 139, pp. 103 – 122, 2019.

[10] K. Su, L. Xu, C. Chen, W. Chen, and Z. Wang, "Affinity and Conflict-Aware Placement of Virtual Machines in Heterogeneous Data Centers," in *2015 IEEE Twelfth International Symposium on Autonomous Decentralized Systems*, 2015, pp. 289–294.

[11] F. L. Pires and B. Barán, "Multi-objective Virtual Machine Placement with Service Level Agreement: A Memetic Algorithm Approach," in *2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, 2013, pp. 203–210.

[12] A. Tomassilli, F. Giroire, N. Huin, and S. Pérennes, "Provably Efficient Algorithms for Placement of Service Function Chains with Ordering Constraints," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 774–782.

[13] M. M. Tajiki, S. Salsano, L. Chiaraviglio, M. Shojafar, and B. Akbari, "Joint Energy Efficient and QoS-Aware Path Allocation and VNF Placement for Service Function Chaining," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 374–388, 2019.

[14] M. Nguyen, M. Dolati, and M. Ghaderi, "Proactive service orchestration with deadline," in *2019 IEEE Conference on Network Softwarization (NetSoft)*, 2019, pp. 369–377.

[15] J. Pei, P. Hong, M. Pan, J. Liu, and J. Zhou, "Optimal VNF Placement via Deep Reinforcement Learning in SDN/NFV-Enabled Networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 263–278, 2020.

[16] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual Network Embedding: A Survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 1888–1906, Fourth 2013.

[17] M. G. Rabbani, R. P. Esteves, M. Podlesny, G. Simon, L. Z. Granville, and R. Boutaba, "On tackling virtual data center embedding problem," in *IFIP/IEEE IM 2013)*, pp. 177–184.

[18] L. Guo, J. Pang, and A. Walid, "Dynamic Service Function Chaining in SDN-enabled networks with middleboxes," in *IEEE ICNP*, 2016.

[19] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *IEEE CloudNet*, 2014.

[20] J. Pei, P. Hong, K. Xue, and D. Li, "Efficiently Embedding Service Function Chains with Dynamic Virtual Network Function Placement in Geo-Distributed Cloud System," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 10, pp. 2179–2192, 2019.

[21] S. Agarwal, F. Malandrino, C. F. Chiasserini, and S. De, "VNF Placement and Resource Allocation for the Support of Vertical Services in 5G Networks," *IEEE/ACM Transactions on Networking*, vol. 27, no. 1, pp. 433–446, 2019.

[22] Y. Yue, B. Cheng, X. Liu, M. Wang, and B. Li, "Resource Optimization and Traffic-Aware VNF Placement in NFV-Enabled Networks," in *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, 2019, pp. 153–158.

[23] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying Middlebox Policy Enforcement Using SDN," in *ACM SIGCOMM 2013*.

[24] H. Moens and F. Turck, "VNF-P: A model for efficient placement of virtualized network functions," *10th International Conference on Network and Service Management (CNSM) and Workshop*, pp. 418–423, 2014.

[25] A. Mohammadkhan, S. Ghapani, G. Liu, W. Zhang, K. K. Ramakrishnan, and T. Wood, "Virtual function placement and traffic steering in flexible and dynamic software defined networks," in *2015 IEEE International Workshop LANMAN*, April 2015.

[26] Tachun Lin and Zhili Zhou and M. Tornatore and B. Mukherjee, "Demand-Aware Network Function Placement," *Journal of Lightwave Technology*, vol. 34, pp. 2590–2600, 2016.

[27] A. Gupta, M. F. Habib, P. Chowdhury, M. Tornatore, and B. Mukherjee, "On service chaining using Virtual Network Functions in Network-enabled Cloud systems," in *2015 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS)*, 2015, pp. 1–3.

[28] J. F. Botero and X. Hesselbach, "Greener networking in a network virtualization environment," *Computer Networks*, vol. 57, no. 9, pp. 2021 – 2039, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128613001151

[29] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," in *IEEE CloudNet*, 2015.

[30] R. Behravesh, E. Coronado, D. Harutyunyan, and R. Riggio, "Joint User Association and VNF Placement for Latency Sensitive Applications in 5G Networks," in *2019 IEEE 8th International Conference on Cloud Networking (CloudNet)*, 2019, pp. 1–7.

[31] A. Hmaity, M. Savi, F. Musumeci, M. Tornatore, and A. Pattavina, "Virtual Network Function placement for resilient Service Chain provisioning," *2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, pp. 245–252, 2016.

[32] S. Khebbache, M. Hadji, and D. Zeghlache, "Scalable and cost-efficient algorithms for vnf chaining and placement problem," in *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, 2017, pp. 92–99.

[33] A. Leivadeas, G. Kesidis, M. Ibnkahla, and I. Lambadaris, "VNF placement optimization at the edge and cloud," *Future Internet*, vol. 11, no. 3, 2019, publisher Copyright: © 2019 by the authors. Copyright: Copyright 2019 Elsevier B.V., All rights reserved.

[34] J. Pei, P. Hong, K. Xue, D. Li, D. S. L. Wei, and F. Wu, "Two-Phase Virtual Network Function Selection and Chaining Algorithm Based on Deep Learning in SDN/NFV-Enabled Networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1102–1117, 2020.

[35] S. Yang, F. Li, S. Trajanovski, R. Yahyapour, and X. Fu, "Recent Advances of Resource Allocation in Network Function Virtualization," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 2, pp. 295–314, 2021.

[36] H. R. Khezri, P. A. Moghadam, M. K. Farshbafan, V. Shah-Mansouri, H. Kebriaei, and D. Niyato, "Deep Reinforcement Learning for Dynamic Reliability Aware NFV-Based Service Provisioning," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[37] S. Lange, H. Kim, S.-Y. Jeong, H. Choi, J.-H. Yoo, and J. Hong, "Machine Learning-based Prediction of VNF Deployment Decisions in Dynamic Networks," *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 1–6, 2019.

[38] P. Sun, J. Lan, J. Li, Z. Guo, and Y. Hu, "Combining Deep Reinforcement Learning With Graph Neural Networks for Optimal VNF Placement," *IEEE Communications Letters*, pp. 1–1, 2020.

[39] R. Mijumbi, J. Gorricho, J. Serrat, M. Claeys, F. De Turck, and S. Latré, "Design and evaluation of learning algorithms for dynamic resource management in virtual networks," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, 2014, pp. 1–9.

[40] P. T. A. Quang, Y. Hadjadj-Aoul, and A. Outtagarts, "A Deep Reinforcement Learning Approach for VNF Forwarding Graph Embedding," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1318–1331, 2019.

[41] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, vol. abs/1509.02971, 2016.

[42] A. Schrijver, *Theory of Linear and Integer Programming*. New York, NY, USA: John Wiley & Sons, Inc., 1986.

[43] A. Jurgelionis, P. Fechteler, P. Eisert, F. Bellotti, H. David, J. P. Laulajainen, R. Carmichael, V. Poulopoulos, A. Laikari, P. Perälä, A. De Gloria, and C. Bouras, "Platform for Distributed 3D Gaming," *Int. J. Comput. Games Technol.*, 2009.

[44] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: An Approach to Universal Topology Generation," in *MASCOTS '01*. IEEE Computer Society, 2001.

[45] B. M. Waxman, "Routing of multipoint connections," *IEEE Journal on Selected Areas in Communications*, vol. 6, pp. 1617–1622, Dec 1988.

[46] Google, "Google Maps," June 2019. [Online]. Available: https://maps.google.com

[47] E. L. Hahne, A. K. Choudhury, and N. F. Maxemchuk, “Improving the fairness of distributed-queue-dual-bus networks,” in *Proceedings. IEEE INFOCOM 90: Ninth Annual Joint Conference of the IEEE Computer and Communications Societies The Multiple Facets of Integration*, June 1990, pp. 175–184 vol.1.

[48] Y. Azar, G. N. Wong, K. Wang, R. Mayzus, J. K. Schulz, H. Zhao, F. Gutierrez, D. Hwang, and T. S. Rappaport, “28 GHz propagation measurements for outdoor cellular communications using steerable beam antennas in New York city,” in *IEEE ICC)*, June 2013.

[49] M. Weib, M. Huchard, A. Stohr, B. Charbonnier, S. Fedderwitz, and D. S. Jager, “60-GHz Photonic Millimeter-Wave Link for Short- to Medium-Range Wireless Transmission Up to 12.5 Gb/s,” *Journal of Lightwave Technology*, vol. 26, no. 15, pp. 2424–2429, Aug 2008.

[50] S. Rangan, T. S. Rappaport, and E. Erkip, “Millimeter-Wave Cellular Wireless Networks: Potentials and Challenges,” *Proceedings of the IEEE*, vol. 102, no. 3, pp. 366–385, March 2014.

[51] S. Singh, M. N. Kulkarni, A. Ghosh, and J. G. Andrews, “Tractable Model for Rate in Self-Backhauled Millimeter Wave Cellular Networks,” *IEEE JSAC*, vol. 33, Oct 2015.

Nabeel Akhtar (nabeel@bu.edu) received his Ph.D. in Computer Science from Boston University in 2019. He is part of the Network Control group at Akamai Technologies Inc. His research involves Network Optimization and Modeling, Architecture Design, and Resource Optimization. He has served as a TPC member for multiple conferences. He has served as a reviewer in over 30 conferences and journals. He received “Neal Shepherd Memorial Best Propagation Paper Award 2020” from IEEE Vehicular Technology Society. He received the Best Project Awards at ICMV 2010 and IEEE INFOCOM-CNERT 2018.

Ibrahim Matta (matta@bu.edu) (M’93-SM’06) received his Ph.D. in computer science from the University of Maryland at College Park in 1995. He is a professor and chair of computer science at Boston University. His research involves network protocols, architectures, and performance evaluation. He has published over 100 peer-reviewed articles. He received the National Science Foundation CAREER award in 1997 for his research on QoS routing. He has served as the chair or co-chair of many technical committees, including IEEE 2011 CCW and 2005 ICNP. He is a senior member of ACM and IEEE.

Ali Raza (araza@bu.edu) is a Ph.D. student at the Computer Science department of the Boston University (BU). Currently, his work deals with the orchestration of cloud-resources for applications with strict performance requirements. Before joining BU, he was a researcher at the NYU Abu Dhabi, where he worked on improving the web connectivity in developing regions through advanced web-caching techniques.

Leonardo Goratti (leoedasgoratti@yahoo.it) received his Ph.D. degree from the University of Oulu, Finland, and his MSc from the University of Firenze, Italy. He is currently a senior systems engineer at Safran Passenger Innovations GmbH, Germany, where he is involved in the 5G innovation strategy. His interests span across cloud technology, SDN and NFV for 5G, LTE-Advanced, millimeter-wave communications, and Visible Light technology. He has authored more than 75 research papers and he has served as a TPC member and reviewer in several international conferences.

Torsten Braun (torsten.braun@inf.unibe.ch) received the Ph.D. degree from the University of Karlsruhe, Germany, in 1993. Since 1998, he has been a Full Professor of Computer Science at the University of Bern. He has been a Director of the Institute of Computer Science and Applied Mathematics, University of Bern, from 2007 to 2011 and since 2019. He received Best Paper Awards from LCN 2001, WWIC 2007, EE-LSDS 2013, WMNC 2014, and the ARMS-CC-2014 Workshop as well as the GI-KuVS Communications Software Award in 2009.

Flavio Esposito (flavio.esposito@slu.edu) is an Assistant Professor with the Department of Computer Science at Saint Louis University (SLU). He received the Ph.D. degree in Computer Science from Boston University in 2013. His research interests include network management, network virtualization, artificial intelligence, and distributed systems. Dr. Esposito is currently a principal investigator of four National Science Foundation grants, and one grant from the International Center for Responsible Gaming. He was awarded with the Comcast Innovation Fund Award 2020, and is the recipient of three best paper awards from IEEE.