IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 2, 2020, accepted August 14, 2020, date of publication September 7, 2020, date of current version September 24, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3021354

Delay-Aware SFC Provisioning in Hybrid Fog-Cloud Computing Architectures

NAZLI SIASI[1], (Member, IEEE), MOHAMMED JASIM[2], (Member, IEEE), ADEL ALDALBAHI[3], (Member, IEEE), AND NASIR GHANI[4], (Senior Member, IEEE)

[1]Department of Physics, Computer Science and Engineering, Christopher Newport University, Newport News, VA 23606, USA
[2]School of Engineering, University of Mount Union, Alliance, OH 44601, USA
[3]Department of Electrical Engineering, King Faisal University, Al-Ahsa 31982, Saudi Arabia
[4]Department of Electrical Engineering, University of South Florida, Tampa, FL 33620, USA

Corresponding author: Mohammed Jasim (jaesimad@mountunion.edu)

ABSTRACT Network function virtualization (NFV) technology enables service providers to implement software-based network processing functionalities on standard computing servers (nodes). As such, this approach mandates the need for mapping virtual network functions (VNFs) in service function chains (SFCs) on these nodes for incoming service requests. Now traditional VNF mapping schemes use cloud nodes for its abundant available resources, at the detriment of prolonged network delays. Alternative schemes use fog nodes that return reduced delays, at the detriment of limited resources. Hence in this work, a novel SFC provisioning scheme is proposed for a hybrid fog-cloud architecture of various resources. The architecture is composed of a single fog and single cloud layer, in order to accommodate both delay-sensitive and delay-tolerant requirements for large number of incoming requests, respectively. This scheme yields a tradeoff between standalone cloud and fog solutions when implemented on the proposed hybrid architecture, in terms of the number of satisfied requests, network delay, resources consumption, energy consumption, and realization cost at large traffic volumes. The proposed scheme achieves 15-40% higher traffic capacity than fog solutions, 21-43% reduced delay, 45-52% less energy consumption levels and 28-30% less cost as compared to cloud solutions.

INDEX TERMS Cloud computing, fog computing, hybrid fog-cloud architecture, network function virtualization, service function chain.

I. INTRODUCTION

Cloud computing technology has emerged as a distributed paradigm that provides large-scale and dynamic pool of host infrastructure [1]. This technology brings multiple advantages, such as flexibility, scalability, and efficient resources utilization. Here the computing resources are shared among multiple networks of different applications and traffic volumes. This allows service providers to provision applications at reduced cost by eliminating on-site hardware and maintenance. However, cloud computing suffers from increased network delays for time-sensitive applications, e.g., real-time online gaming. This in turn results in services outages, network congestion at high traffic scenarios, limited bandwidth, as well as security and privacy challenges. For instance, cloud computing has recently been considered as a potential solution for underlying *internet of things* (IoT) infrastructure to benefit from the abundant storage capabilities [2]. However, it can be inefficient for time-sensitive applications due to the introduced delays, which can degrade *quality-of-service* (QoS). Thus yielding reduced network performance. Therefore, alternative computing solutions are necessary for time-sensitive applications.

Fog computing architectures [3] have been proposed to overcome the aforementioned cloud limitations. This is possible by utilizing edge devices to locally perform substantial amount of the network computing and storage (memory) requirements. Hence, this approach acts as a service-oriented intermediate interface between terminals (end-users) and cloud nodes. A major advantage here is reducing propagation delays associated with links connecting distant-based cloud nodes. It also alleviates extended bandwidth usage in these links. In spite of that, fog nodes feature low computing and storage capabilities at high traffic volume, i.e., limited capacity. This limits the support of high bandwidth applications.

Hence this paper proposes a *hybrid fog-cloud* (HFC) architecture that merges fog and cloud technologies, as depicted

The associate editor coordinating the review of this manuscript and approving it for publication was Aniello Castiglione.

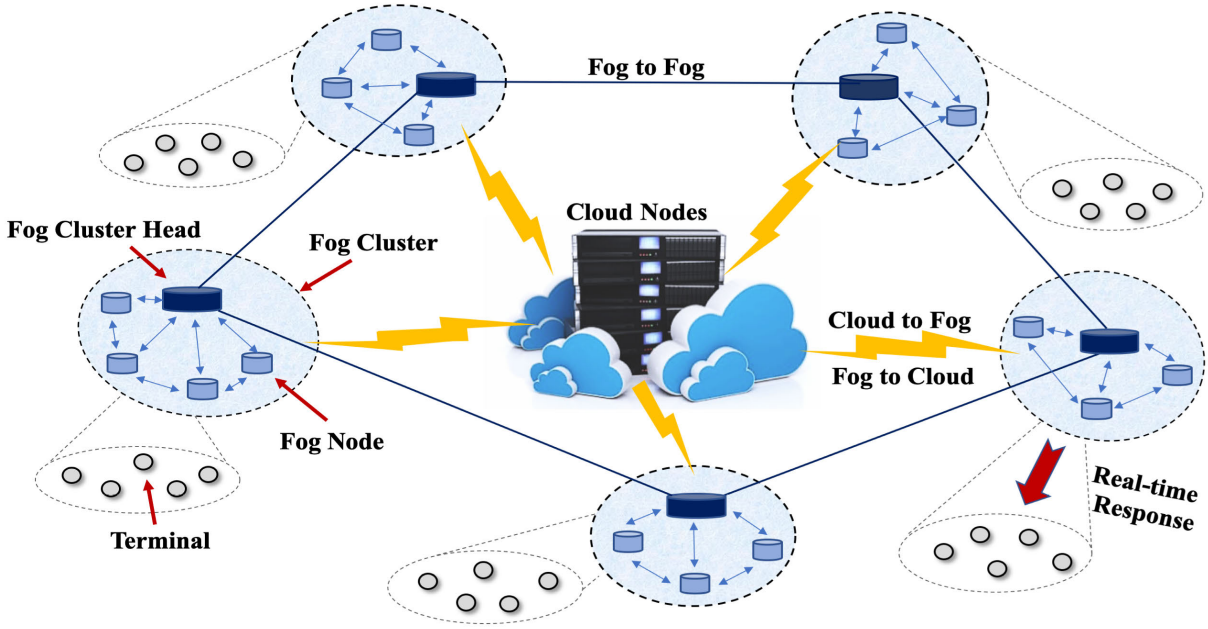

FIGURE 1. Hybrid fog-cloud architecture.

in Figure 1. Hence it combines the advantages of both technologies in one architecture at reduced limitations. This architecture can then support different types of applications and services. Namely, it aims to process specific application components at the fog nodes at reduced network delays and low bandwidth usage. Meanwhile, it provides high bandwidth capabilities at the cloud nodes at higher delays. Specifically, if the request demands a delay-sensitive service, then it will be mapped on the fog nodes, thereby achieving reduced delays. This reduces mapping on the cloud nodes, which minimizes traffic bottlenecks in the fronthaul of the network. Moreover, if the request demands a delay-tolerant service, then it is mapped on the *fog cluster heads* (FCHs) or the cloud nodes, since it has no stringent requirements on the delay. Here mapping delay-tolerant requests on the FCHs or cloud nodes keeps the fog nodes resources better utilized for future incoming delay-sensitive requests.

Furthermore, NFV is another key technology that decouples *network functions* (NFs) from proprietary hardware (nodes) [4]. Namely, it enables VNFs to run as software instances over dispersed cloud or fog nodes at various locations, in order to execute specific NFs such as firewall, DNS, caching, and evolved packets. Now a major design task here is VNF mapping onto the underlying cloud or fog nodes, in order to realize the network virtualization paradigm. Moreover, the VNFs here are often chained together in a certain sequence specified by terminal requests, thereby forming a SFC. Now performing SFC provisioning on NFV-based fog or cloud nodes at efficient resources utilization is a challenging task. This has been investigated by various efforts as presented in Section II.

This paper is organized as follows. Section II reviews the recent efforts on SFC provisioning and VNF mapping, along with existing fog/cloud architectures, and proposed contributions. Section III presents the novel multi-layer HFC architecture. This is followed by the system model in Section IV, composed of a proposed terminal request model, design metrics and assumptions. The SFC provisioning scheme is then proposed in Section V. Then network setup, performance evaluation, and simulation results are presented in Section VI. Finally, Section VII presents the conclusion remarks and future directions.

II. BACKGROUND SURVEY

A. SURVEY OF SFC PROVISIONING SCHEMES IN CLOUD ARCHITECTURES

Now a wide range of studies have been proposed to achieve SFC provisioning objectives in cloud architectures, see [5], [6] for a detailed review for some recent survey articles. First, global optimization techniques have been widely proposed to solve the problem of SFC provisioning in cloud computing. Foremost, the authors in [7] propose a *mixed-integer linear programming* (MILP) and a heuristic-based algorithm to jointly optimize NFV resource allocation in the SFC composition, VNFs forwarding graph embedding, and VNFs scheduling. Furthermore, the work in [8] formulates the provisioning problem as an optimization model and proposes a *mixed integer quadratic constrained program* (MIQCP) solution. Also, multiple dependent directional acyclic graph schemes are presented in [9]–[11]. These schemes consider the priority dependence between nodes at reduced usage of links bandwidth. An *integer linear program* (ILP) scheme is proposed in [9] and [10] that maps requests of multiple instances onto a single node to minimize resources consumption. However, the aforementioned schemes in general require traffic to traverse at longer paths to reach nodes hosting the VNFs. Thus yielding increased network delays and bandwidth consumption. Moreover, the work in [11]

proposes a graph-based heuristic that combines graph centrality and multi-stage graphs for the placement of service functions chains. Authors in [12] implement a scalable SFC placement by proposing an eigendecomposition-based approach for VNFs mapping on the cloud nodes. The proposed method here reduces the complexity and convergence times that essentially depend only on the physical graph sizes. Finally, the work in [13] formulates the physical network and SFC request as two weighted graphs and formulates the SFC placement problem in the NFV environment consisting of graph matching and VNF mapping. Then authors propose an LP-based approach and a Hungarian based algorithm to solve the graph matching and SFC mapping problem.

B. SURVEY OF SFC PROVISIONING IN HYBRID FOG-CLOUD ARCHITECTURES

The work in [14] integrates NFV with fog computing to leverage the advantages of both technologies to support a handover scheme for 5G systems. This integration achieves reduced overhead and network flexibility for fog nodes of fixed catches. Also, a VNF mapping approach is proposed in [15] that accounts for stringent delay constraints. The mapping problem is formulated as a graph-clustering optimization model and a genetic algorithm is applied to reduce cost. However, the aforementioned schemes lack SFC provisioning.

Few studies have looked into HCF architectures, along with VNF mapping. Foremost, authors in [16] propose an architecture for *platform as-a-service* (PaaS) that splits VNF mapping onto cloud and fog nodes to measure network delay. However this study is limited to a single VNF and it lacks SFC requirements. The authors in [17] also propose to span VNF components between cloud and fog infrastructure for IoT healthcare applications. This work focuses only on mechanisms for providing control, signaling, and data interfaces between the cloud and fog nodes. Also work in [18] proposes to migrate and monitor applications components between cloud and fog nodes, while considering the tradeoff between power consumption and delay. Here the mapping problem is again formulated as an optimization model. Then an approximate solution is proposed to decompose the primary problem into three subproblems (solved independently). This work achieves reduced response times and enhanced throughputs at the expense of high computation resources. However, it lacks VNF mapping onto any of the nodes.

Moreover, metaheuristic schemes have been leveraged in fog architectures to solve the SFC provisioning problem. For instance, the work in [19] proposes Tabu search VNF mapping scheme, where the feasible region consisting of all nodes is divided into equally-adjacent smaller sub-region (of fewer nodes). Then the scheme conducts an inner pattern search to select a host node based upon the shortest and the least-load path to the source.

Additionally, an application component placement in NFV-based HFC architecture is presented in [20] and [21]. Placement decisions are determined from an ILP solver to achieve cost minimization. This work is limited to small-scale scenarios and considers a single VNF type in a single fog layer. Authors in [22] present a multi-layer fog and cloud architecture for video-streaming applications, as opposed to single layer fog solutions. Three layers are classified here in terms of their coverage, computing and storage capacities. Nonetheless, this work only demonstrates suitable services for multi-layer architectures. It lacks SFC provisioning and it is limited to video-streaming applications. Also, the three layers model here can increase realization costs.

Furthermore, some studies leverage *software defined networks* (SDN) technology in fog and cloud architectures. Foremost, authors in [23] integrate cloud and fog computing in conjunction with SDN and NFV for 5G systems based on a SFC model. The work here only considers the type of hypervisors, virtualization, and security issues. It lacks consideration of the constrained-resources in the single fog layer. Moreover, it does not account for delay-sensitive applications. Additionally, work in [24] studies the benefits associated with service migration from cloud to multi-layer fog nodes based on SDN for video distribution applications. Namely, it measures the required time for migration between the different layers in efforts to reduce traffic at the core network. However, this study does not consider resources constraints in highly congested multi-layer fog nodes. It also lacks consideration of delay-tolerant applications of high capacity demands, as well as absence of VNF mapping. Moreover, authors in [25] use a heuristic scheme based on ILP to provide SFC in SDN-based networks with a failure recovery scheme. This work considers computational complexity, average failure probability in the selected paths, in addition to link and server utilization. Finally, a combined architecture based on SDN and fog computing is presented in [26] for *vehicular ad-hoc networks* (VANETs) of delay-sensitive and location-awareness services. Also, the work in [27] proposes an edge architecture for vehicular service composition that aims to replace conventional vehicular cloud solutions in next generation networks. The architecture is based on *vehicular service clouds* (VSC) that are available on-the-fly, which are accessed as per the needs of vehicular users. This provides an intra-vehicle resource sharing model that delivers a wide range of cloud services, such as on-demand entertainment and speech recognition for driver assistance at reduced latencies. The overall objective here is to achieve low latency based on reduced end-to-end task completion times.

Moreover, authors in [28] proposes a cooperative communication scheme for fog-to-fog and fog-to-cloud nodes. The scheme provides a range of terminal-defined services that involve a one-time requester/provider interaction. Hence short-term *service level agreements* (SLA) are developed based upon Tabu search heuristic method that uses previous solutions when selecting new optimal choices. However, this work focuses on establishing workflows for new service requests. It lacks the definition of the VNFs for the terminal-defined multimedia services and it does not present

SFC model on the fog or cloud nodes. Finally, the work in [29] proposes an SLA-aware fine-grained QoS provisioning (SFQP) scheme for *multi-tenant software-defined networks* (MTSDN), i.e., a service platform that manages and shares VNFs between tenants and terminals. The proposed scheme automatically extracts the eigen characteristics of each packet by an application-aware methodology. It also leverages *K-nearest neighbor* (KNN) algorithm to predict the SLA popularity.

C. MOTIVATIONS AND CONTRIBUTIONS

The aforementioned SFC provisioning schemes on cloud nodes map the VNFs without accounting for delay requirements of incoming requests. Note that applying these schemes to fog or *hybrid fog-cloud* (HFC) architectures can yield in various limitations, due to the case of heterogeneous nodes of different available resources. These schemes also lack network delay and cost models for SFC deployment. Moreover, the deployment and communication costs here are highly dependent on the node location, at which the VNF is hosted. Thus making the provisioning task more challenging. Now the lack of SFC provisioning in HFC architectures presents a major motivation for this work. Specifically, existing efforts often focus on dependency-unaware VNF mapping schemes with application components splitting between fog and cloud nodes. Hence, this paper proposes a novel SFC scheme for application components in synergetic HFC paradigm, termed as *delay-aware fog-cloud* (DAFC) provisioning. Namely, the proposed scheme accounts for the ultra-low latency requirements for real-time applications, where requests are categorized as either delay-sensitive and delay-tolerant based upon the delay requirements.

The goal here is to achieve SFC provisioning for HFC architectures that yields in practical advantages to network operators and regulators, as specified in the the fog computing standard *OpenFog Consortium* in [30]. This includes reduced network delays, shorter latencies, minimized energy consumption and realization costs. This work also enables efficient and dynamic resources utilization, in terms of processing usage (e.g. CPU usage), memory, and link bandwidth. Hence achieving high number of satisfied requests (reducing access attempts). Thus increasing network capacity and scalability.

III. PROPOSED HYBRID FOG-CLOUD ARCHITECTURE

The proposed hierarchical HFC architecture is composed of two layers. Terminals in Layer I, fog nodes and fog cluster heads in Layer II, and cloud nodes in Layer III, as detailed next, see Figure 1 [31].

A. LAYER I (TERMINALS)

This layer is composed of a set of terminals that connect to the substrate network via radio links. Such as mobile devices and vehicles, as well as stationary devices. Terminals here initiate SFC requests.

B. LAYER II (FOG NODES)

This layer consists of resources-constrained fog nodes that cooperatively process, compute, and temporarily store received requests from nearby terminals within the fog node footprint (coverage area). The adjacent fog nodes in a specific geographical area form fog clusters. The nodes in each cluster communicate with each other for traffic and available resources information exchange. The fog nodes in each cluster are interconnected with *fog cluster heads* (FCH) of higher computing and storage resources. The FCH allows traffic offloading to adjacent nodes, that possess abundant resources within the same cluster or with the neighbor cluster members. The FCH can also be used to host medium-delay or delay-tolerant services for requests relayed from the fog nodes (which only host delay-sensitive requests). Consequently, this saves resources in the fog nodes to accommodate additional delay-sensitive requests. This layer also includes multiple devices, such as edge routers, gateways, switches, access points, and set-top boxes.

C. LAYER III (CLOUD NODES)

This is the highest layer composed of high-end cloud nodes connected to the fog nodes in the lower layers via a high bandwidth network backbone. They feature very high computing resources to process and store enormous amounts of data, at the detriment of high network delays. Hence this layer is utilized for delay-tolerant requests demanding high resources consumption.

IV. SYSTEM MODEL

Consider the following assumptions for the proposed DAFC provisioning scheme, implemented on the aforementioned HFC architecture. I) The incoming service requests from terminals in Layer 1 are first processed at the fog nodes in Layer II through one-hop or multi-hop connection. II) Requests are classified into two different types based upon their delay requirements. Namely, delay-sensitive and delay-tolerant requests. III) Different VNFs in the requests demand various amount of resources, such as processing and storage resources and delay. IV) The total capacity at each fog or cloud node is represented by the total amount of available resources at the node. The various design models are presented next.

A. MULTI-LAYER SUBSTRATE NETWORK TOPOLOGY

The multi-layer substrate network for the proposed HFC architecture (Figure 1) is modeled as an undirected graph, $G = (N, E)$. It consists of vertices, N, representing the sets of terminals n_t^i, fog n_f^j, fog cluster heads n_h^k, and cloud nodes n_c^l, i.e., $N = \{n : n_t^i, n_f^j, n_h^k, n_c^l\}$. Also, E represents the sets of edges/links in the network, i.e., $E = \{e : e_{t,f}, e_{t,h}, e_{t,c}, e_{f,h}, e_{t,c}, e_{h,c}, e_{f,f}, e_{h,h}, e_{c,c}\}$. Here an edge is denoted by e, and it can be either a link between a terminal and a fog node, fog cluster, cloud node, denoted by $e_{t,f}$, $e_{t,h}$ and $e_{t,c}$, respectively. Also, it can be a link between a fog node and FCH or cloud node, denoted by $e_{f,h}$ and

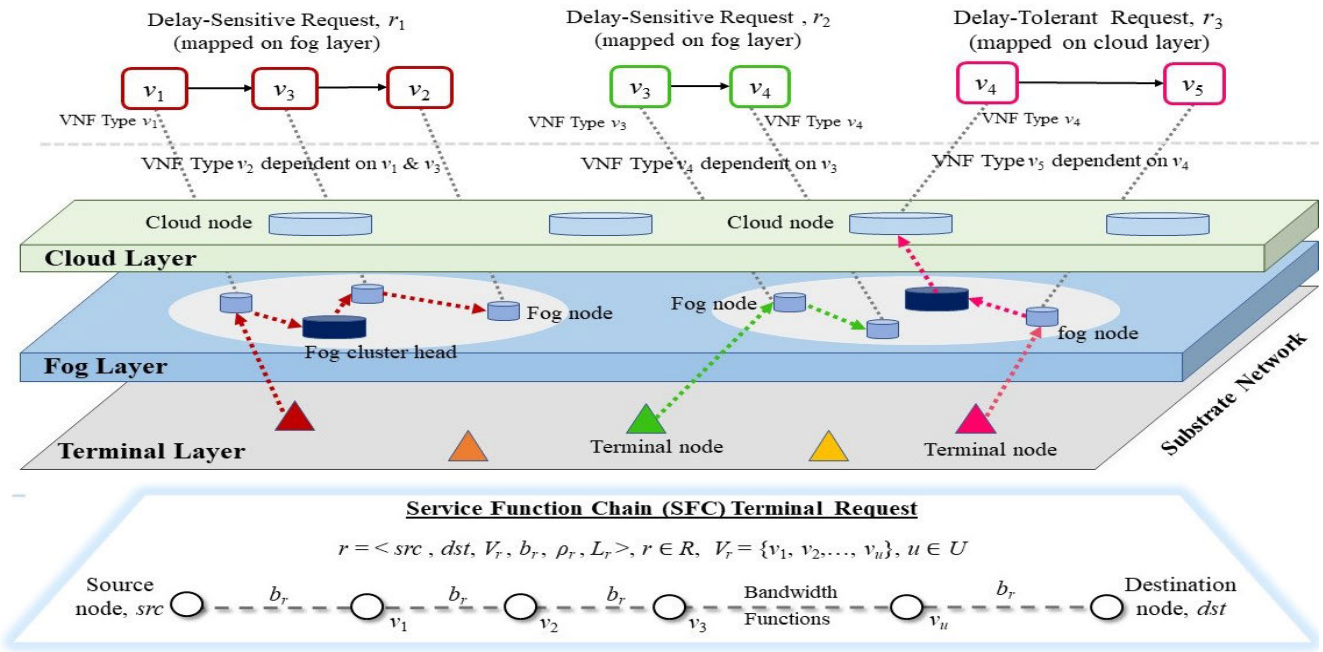

FIGURE 2. SFC provisioning examples on the proposed HFC architecture.

$e_{t,c}$, respectively. Moreover, it can be a link between cluster head and cloud node, $e_{h,c}$. Finally, it can be a link between fog nodes $e_{f,f}$, cluster heads $e_{h,h}$, and cloud nodes $e_{c,c}$. Furthermore, each node can host one or more VNFs, $v_u \in V_r$, i.e., $u = 1, 2, \ldots, U$, where U denotes the total number of VNF types. Also, V_r is the set of all required VNFs in the request r.

Moreover, the available substrate resources are bounded by a finite set of constraints. This is necessary for practical and realistic operational settings. Namely, the overall resources capacities at ascending layer numbers possess higher available resources. For example, cloud nodes have much higher resources than fog clusters heads. Now each node has a specific computing capability, which constitutes to the amount of available resources in its layer. These resources are presented as a set of three main attributes, i.e., processing (CPU), memory and bandwidth. Here, the total memory at any node is bounded by $Q_{me}(n)$, and the total processing capacity is bounded by $Q_{proc}(n)$. Also, the available link bandwidth on a substrate link between two nodes is bounded by $B(e)$. Note that a key saliency for the proposed scheme is scalability for larger number of node or link resources at high traffic volumes.

B. TERMINAL REQUEST MODEL

A novel model is developed here for request $r \in R$ of specific resources and delay requirements, where R is the set of total requests from terminals. Each request r is expressed by 6-tuple $r = <src, dst, V_r, b_r, \delta_r, L_r>$, where $src \in N$, denotes the source node (terminals in Layer 1), $dst \in N$, denotes the destination node hosting last VNF in the SFC. Also, V_r denotes the set of desired VNF types, v_u, ordered in the SFC in request r, i.e., $V_r = [v_1, v_2, \ldots, v_u], u \in U$, where U is the set of defined VNF types in the network. See Figure 2 for requests of various delay and resources requirements, mapped on nodes in Layers II & III based on the destination, e.g., r_1, r_2 and r_3. Furthermore, each hosted VNF in the network nodes requires specific processing $Q_{pr}(v_u)$ and memory $Q_{me}(v_u)$ resources, i.e., $Q_{pr}(v_u) \in Z^+$ and $Q_{me}(v_u) \in Z^+$. The variable b_r is the required link bandwidth to interconnect the VNFs in SFC, δ_r denotes the network delay requirement for the request (either delay-sensitive or latency-tolerant), and L_r represents the service lifetime. Note here that different number of instances of the same VNF can be generated from requests R, hence the same VNF can be shared on the same node by more than one request.

Figure 2 shows various SFC provisioning examples on the proposed HFC architecture for requests demanding various delay and resources levels, mapped on nodes in Layers II & III. First, a delay-sensitive request r_1 is received at a nearby fog node from the terminal layer with a dependency requirement of $v_1 \rightarrow v_3 \rightarrow v_2$. Hence it is required here to map v_1 first, then v_3, followed by v_2. These VNFs are all mapped on the fog layer in order to achieve the delay requirements here. The dotted arrows in red shows the SFC initiated from the terminal, passing through the nodes hosting the VNFs to the destination. The red dotted lines in Figure 2 show the SFC route for r_1 from terminal through the hosting fog nodes to the final destination. Similarly, the delay-sensitive request r_2 has a dependency requirement of $v_3 \rightarrow v_4$, i.e., mapping v_3 followed by v_4. Lastly, the received delay-tolerant request r_3 features a dependency requirement of $v_4 \rightarrow v_5$. Hence VNF v_4 is mapped first on a fog node in Layer II, followed by the next VNF in the SFC, v_5, which is also mapped on another fog node in the same layer. This VNF is mapped on a nearby node that satisfies least delay and cost, while achieving shortest

path. Here the green dotted lines depict the SFC route from terminal through the hosting nodes (FCH and cloud) to the destination.

Prior to presenting the SFC provisioning scheme, key design measures are taken into account that considers the resources requirements and availability. In particular, node and edge capacity requirements, as detailed next.

C. NODES CAPACITY REQUIREMENTS

Each VNF, v_u, in the request list is mapped to either a fog node n_f^i, fog cluster head n_h^k, or cloud node n_c^l, on substrate network $n \in N - \{n_t^i\}$, that has enough available computing $Q_{pr}(n)$ and memory resources $Q_{me}(n)$. A key condition here is that the sum of processing and memory capacities required by VNF instances mapped to the nodes cannot exceed the amount of available physical resources. This in turn avoids node overloading and requests drops. In notations,

$$\sum_{r\in R}\sum_{v_u\in V_r}\sum_{s\in S_{v_u}} \lambda^r_{v_u}.Q_{pr}(v_u) \leqslant Q_{pr}(n), \quad \forall n \in N - \{n_t^i\}, \quad (1)$$

$$\sum_{r\in R}\sum_{v_u\in V_r}\sum_{s\in S_{v_u}} \lambda^r_{v_u}.Q_{me}(v_u) \leqslant Q_{me}(n), \quad \forall n \in N - \{n_t^i\}, \quad (2)$$

where $\lambda^r_{v_u}$ denotes the number of VNFs of a specific type in request r mapped to a node, and S_{v_u} is the set of instances for the VNF of type v_u.

D. EDGE CAPACITY REQUIREMENTS

Here each link between two consecutive VNF nodes in the SFC request must be mapped to a substrate link, e, $e \in E$ of enough available bandwidth, $B(e)$, i.e., larger than the request bandwidth, b_r. This is necessary in order to transverse sufficient amount of flow (required by SFC requests) at reduced overloading. This is modeled as,

$$\sum_{r\in R}\sum_{e\in E} \Gamma^r_e.b_r \leqslant B(e), \quad (3)$$

where Γ^r_e denotes the number of virtual links in request r mapped to the network.

E. DELAY MODELS

For each request, as data packets travel from the source node to the subsequent nodes along the path, unit reaching the destination node, these packets encounter multiple types of network delay along their paths. In this work, the network delay is composed from the processing, transmission and propagation delays as presented next (note that the queuing is not considered here). Given a delay bound, δ_r, the network delay along the designated path, $D(p)$, for request, r, must satisfy the delay constraint, where p, $p \in P$, is the valid path for P total number of paths. This path delay is modeled as,

$$D(p) = \sum_{n\in p} D_{pr}(n) + \sum_{n\in p} D_{qe}(n) + \sum_{e\in p} D_{tran}(e) + \sum_{e\in p} D_{prop}(n) \leqslant \delta_r, \quad \forall r \in R, \quad (4)$$

where $D_{pr}(n)$, $D_{qe}(n)$, $D_{tran}(e)$ and $D_{prop}(e)$ represent the node processing, node queuing, link transmission and propagation delays, respectively.

1) PROCESSING DELAY

This value measures the total time required by the node to process a mapped VNF, v_u, in the SFC of the request, r. The total processing time for R requests is formulated as,

$$D_{pr}(R) = \sum_{r\in R}\sum_{v_u\in V_r}\sum_{s\in S_{v_u}} \frac{A_r}{\delta_{v_u}(n)}, \quad \forall n \in N - \{n_t^i\}, \quad (5)$$

where A_r is the traffic load per request, $\delta_{v_u}(n)$ denotes the processing rate of VNF type u in the request on a cloud or fog node, which is the request traffic unit per time (in ms).

2) QUEUING DELAY

The average queuing delay for each request, $D_{qe}(n)$, depends on the instantaneous requests arrival rate to the queue at the node, $\Upsilon(n)$ (requests/sec), the processing capacity requirements of the request, $Q_{pr}(r)$, service processing rate at node n, $\delta_{pr}(n)$, as well as transmission rate at the link, $\delta(e)$. Moreover, the queuing delay depends on the nature of the traffic, e.g., exponential during peak-hours or Poisson arrival rate. The above parameters form the traffic intensity, Ω, the key factor in determining the queuing delay. It is modeled as,

$$\Omega = \frac{\Upsilon(n)Q_{pr}(r)}{\delta(e)}, \quad (6)$$

where the request processing capacity requirements is the total VNF requirements, $Q_{pr}(v_u)$, $\forall v_u \in V_r$, expressed as,

$$Q_{pr}(r) = \sum_{v_u\in V_r} Q_{pr}(v_u). \quad (7)$$

The traffic intensity starts at very low rate when the number of requests is small (low requests arrival rate). Hence when a request arrives, it is unlikely that it will find another request in the queue. As a result, the average queuing delay will be very small (in order of microseconds). Meanwhile, as the number of requests increase, i.e., higher arrival rate. Then the traffic intensity increases until a point where the arrival rate exceeds the transmission rate and processing capacity of the link. Therefore, the queuing delay is increased. Note that if $\Omega > 1$, then the average arrival rate at the queue at node n exceeds the transmission rate at link e. However, the work here assumes bounded traffic intensity, in order to avoid an infinite increase in the queuing delay. This in turn avoids network failure for all incoming requests, when exceeding the requests delay bounds. Hence this is achieved by having the arrival rate to be less than or equal to transmission capacity of the link. This makes the service time the major factor that determines the queuing delay. Along these lines, the requests arrival rate at the node is modeled as Poisson distribution, and the queuing delay follows exponential increase. This model is characterised by the $M/M/n$ queuing model and Erlang $\mathbb{C}$ formula, thereby achieving a bounded traffic intensity.

Overall, the queuing delay at node n for each request r is formulated as,

$$D_{qe}(n) = \frac{\mathbb{E}(M(n), \zeta(n))}{M(n)\delta_{pr}(n) - \psi(n)}, \tag{8}$$

The parameter $\mathbb{E}(M(n), \zeta(n))$ is the Erlang $\mathbb{C}$ formula, i.e., $\mathbb{E}(\mathbb{C})$ expressed as [32],

$$\mathbb{E}(\mathbb{C}) = \frac{(\frac{M(n)\zeta(n)}{M(n)!})(\frac{1}{1-\zeta(n)})}{\sum_{m=1}^{M(n)}(\frac{M(n)\zeta(n)}{m!})^m + (\frac{M(n)\zeta(n)}{M(n)!})(\frac{1}{1-\zeta(n)})}, \tag{9}$$

where $\zeta(n)$ is the utilization rate of the node, expressed as,

$$\zeta(n) = \frac{\hat{\delta}_{pr}(n)}{M(n)\delta_{pr}(n)}. \tag{10}$$

The variable $\hat{\delta}_{pr}(n)$ is the instantaneous processing rate provided by node n, defined by,

$$\hat{\delta}_{pr}(n) = \psi_r(n)\Upsilon_{tot}(n). \tag{11}$$

Here the parameter $\psi_r(n)$ represents the percentage of requests that the node n can process utmost, $\Upsilon_{max}(n)$, from the total arriving requests at the same node, $\Upsilon_{tot}(n)$. This is formulated as,

$$\psi_r(n) = \begin{cases} 1, & \Upsilon_{max}(n) \leq \Upsilon_{tot}(n), \\ \dfrac{\Upsilon_{max}(n)}{\Upsilon_{tot}(n)}, & \Upsilon_{max}(n) < \Upsilon_{tot}(n). \end{cases} \tag{12}$$

The maximum workload at each node is represented by the maximum request arrival rate that a node can receive, $\Upsilon_{max}(n)$, as a fraction of the total arrival rate, $\Upsilon_{tot}(n)$. Here the parameter $\Upsilon_{max}(n)$ represents the maximum workload at the node, i.e., the maximum request arrival rate that a node can receive. It avoids excessive queuing delay when the fog node is heavily loaded.

3) TRANSMISSION DELAY

It relates to the transmission rate of the link, i.e., amount of traffic units that are forwarded/transmitted from one node to another. Note that work here adopts first-come-first-serve transmission scheme. Therefore, the transmission delay for all requests, $D_{tran}(R)$, is gauged as,

$$D_{tran}(R) = \sum_{r\in R}\sum_{e\in E}\frac{A_r}{\delta(e)}, \tag{13}$$

where $\delta(e)$ is the link transmission rate, which defines the elapsed forwarding time per traffic unit for the request.

4) PROPAGATION DELAY

It represents the time encountered for data to propagate on link e between any two nodes. It is gauged by the separation distance between the nodes divided by the propagation speed of the medium (e.g., wireless, fiber). The overall propagation delay for request r accounts for all interconnecting links between the source node, nodes hosting the VNFs to the destination node (propagation delay over all links joining the nodes at which VNFs are mapped). This delay is modeled as,

$$D_{prop}(R) = \sum_{r\in R}\sum_{e\in E}\frac{A_r}{\delta(r)}, \tag{14}$$

TABLE 1. Edge bandwidth cost.

Edge cost	($/GB)
$\rho(e_{t,f})$	1
$\rho(e_{f,f})$	2
$\rho(e_{f,h})$	2.5
$\rho(e_{h,h})$	3
$\rho(e_{h,c})$	3.5
$\rho(e_{t,c})$	4
$\rho(e_{c,c})$	0.1

F. COST MODELS

The cost model includes deployment, processing, and communication costs, where their summation formulates the overall realization cost of SFC provisioning in different network architectures, fog, cloud or HFC architectures.

1) DEPLOYMENT COST

The total license cost of deploying VNF software instances, modeled as,

$$C_{dep}(R) = \sum_{r\in R}\sum_{v_u\in V_r}\sum_{s\in S_{v_u}}\lambda^r_{v_u}.\rho(v_u), \quad \forall n \in N - \{n^i_t\}, \tag{15}$$

where $\rho(v_u)$ is the license cost of VNF type u.

2) PROCESSING COST

The cost of resources assigned and reserved for the overall number of mapped VNFs in the SFC requests, expressed as,

$$C_{pr}(R) = \sum_{r\in R}\sum_{v_u\in V_r}\sum_{s\in S_{v_u}}\rho_{pr}(n)Q_{pr}(v_u) + \rho_{me}Q_{me}(v_u), \tag{16}$$

$\forall n \in N - \{n^i_t\}$, where $\rho_{pr}(n)$ and $\rho_{me}(n)$ are the node processing and memory costs per resource unit, respectively.

3) COMMUNICATION COST

The total cost of edges assigned and used for all the mapped VNF edges in the requests. This includes communication cost between terminals and their affiliated VNFs on fog and/or cloud nodes. This is modeled as,

$$C_{com}(R) = \sum_{r\in R}\sum_{e\in E'}v^r_e.\rho(e).b_r, \tag{17}$$

where $\rho(e)$ accounts for the transmission cost per traffic unit between links in different layers. See Table 1 for different edge costs [24].

V. DELAY-AWARE FOG-CLOUD MAPPING (DAFC)

A novel scheme is now presented for delay-aware SFC provisioning implemented on the the proposed multi-layer HFC architecture, termed as *delay-aware fog-cloud mapping* (DAFC). In particular, the scheme uses a heuristic search method in efforts to achieve a tradeoff between delay and cost. The detailed pseudocode for this scheme is also presented in Algorithm 1, as detailed next. Consider a group of terminals in Layer I within a specific a footprint that generate service requests $\boldsymbol{R}$ transversed via radio interfaces to fog nodes n^i_f in Layer II. The neighboring fog nodes here form a cluster to process delay-sensitive requests. Moreover, the fog nodes in each cluster are connected to cluster heads in Layer III,

Algorithm 1 Delay-Aware SFC Provisioning Scheme

Input: Network $G = (N, E)$, set of incoming online requests $r \in R$, $r = <src, dst, V_r, b_r, \delta_r, L_r>$
Output: Fog/Cloud SFC provisioning
/ Loop and process all requests in R*/*
for (each $r \in R$) **do**
  / Initialize tracking variables */*
  $D(r) = 0$ */* Overall network delay */*
  $C(r) = 0$ */* Total cost (node, link) */*
  $Prev \leftarrow src$ */* Assign source to previous node */*
  Subs / Node with subsequent mapped VNF */*
  / Set route vector */*
  if (delay-sensitive), $\delta_r < \delta_{th}$ **then**
    / Map on fog nodes in Layer II*/*
    / Process all functions in function list */*
    for (each $v_u \in V_r$) **do**
      / Prune fog layer to build feasible graph, $G' = (N', E')$*/*
      if $n_f^j \in N$ w. $Q_{me}(n_f^j) > Q_{me}(v_u)$ & $Q_{pr}(n_f^j) > Q_{me}(v_u)$ & $e_{f,f} \in E$ w. $B(e_{f,f}) > b_r$ **then**
        $N' \leftarrow$ Add n_f^j and $E' \leftarrow$ Add $e_{f,f}$
        / Check if valid routes */*
        Compute delay-bound candidate routes set P from $Prev$ to all $n_f^j \in N'$, p s.t. $min(D(r) + D(p)) < \delta_r$
        for each $p \in P$ s.t. $P \neq \{0\}$ **do**
          / Evaluate feasible nodes in N' */*
          Choose n_f^j with $min(D(r)+D(p))$, $\forall n_f^j \in N'$
          / Update tracking variables */*
          $D(r) = D(r) + D_{pr}(n_f^j) + D_{qe}(n_f^j) + D_{tran}(e_{f,f}) + D_{prop}(e_{f,f})$
          $C(r) = C(r) + C_{pr}(n_f^j) + C_{com}(e_{f,f})$
          $Prev \leftarrow n_f^j$ */* update prev */*
          / Reserve resources along selected path */*
          $Q_{me}(n_f^j) = Q_{me}(n_f^j) - Q_{me}(v_u)$,
          $Q_{pr}(n_f^j) = Q_{pr}(n_f^j) - Q_{pr}(v_u)$
          $B(e_{f,f}) = B(e_{f,f}) - b_r$
        end for
      elsif proper node not found **then**
        / Map on cluster heads in Layer II */*
      elsif proper node not found **then**
        / Map on cloud nodes in Layer III */*
      end if
    end for
  else if (delay-tolerant) **then**
    */*Map on fog cluster heads in Layer II*/*
    FCH n_h^k with $min(D(r) + D(p))$, $\forall n_h^k \in \{N'\}$
    / Check resources availability, if not found, check Layer III */*
    Cloud node n_c^l with $min(D(r) + D(p))$, $\forall n_c^l \in \{N'\}$
  else if (not found) **then**
    Request r failed, exit
  end if
end for

n_h^k, over which delay-tolerant requests are processed. Overall, the communication and management protocols among nodes in different layers follow a hierarchical clustering approach, as depicted in Section III. Thus nodes can communicate horizontally within nodes in the same layer and vertically to nodes in higher layers for relaying purposes.

Consider the set of incoming online requests, R, generated from the terminals, i.e., received at the closest fog node within proximity (over a wireless link). Now the closest fog node examines the delay δ_r and lifetime L_r requirements when mapping VNFs, v_u, $[v_u \in V_r]$, in the SFC of each request r. The scheme initiates variables $D(r)$ and $C(r)$, to account for the network delays and costs, respectively. Also, a route vector is required that establishes routes between previous hosting node with mapped VNF, *Prev*, to the candidate node that maps the subsequent VNF, *Sub*. Thus forming route vectors.

The VNFs are mapped at different layers based on delay requirements, where the node selection mapping policy over which fog nodes and cluster heads is to achieve lowest latency and maximum available capacity. For instance, a close neighboring fog node that provides minimum total delay and maximum remaining resources at the current state is selected for an incoming request. Namely, if request is delay sensitive, then it is mapped on the fog nodes in the Layer II. For all VNFs in the SFC, the network is then pruned to build a feasible graph G' composed of candidate nodes $n_f^j \in N'$ with $Q_{me}(n_f^j) > Q_{me}(v_u)$ or $Q_{pr}(n_f^j) > Q_{pr}(v_u)$, and links $e_{f,f} \in E'$ with $B(e_{f,f}) > b_r$.

Then the scheme computes delay-bound candidate routes set, P, from *Prev* to all $n_f^j \in N'$. Moreover, each route p here satisfies minimum network delay for incoming request, δ_r, i.e., $min\{D(r) + D(e)\} < \delta_r$. From these routes, a candidate node is selected based on minimum delay requirements, hence achieving minimum delay with previous node that host last mapped VNF in the same layer.

Overall, nodes are selected based upon delay requirements. If the request is delay-sensitive, specified by δ_r less than a threshold value, δ_{th}, then VNFs are mapped to Layer II. Once a VNF is mapped, then resources are reserved in $G' = (N', E')$ by updating node resources and links bandwidth. This is achieved by subtracting the resources and bandwidth capacity requirements of all mapped VNFs on the node, i.e., $Q_{me}(n_f^j) = Q_{me}(n_f^j) - Q_{me}(v_u)$, $Q_{pr}(n_f^j) = Q_{pr}(n_f^j) - Q_{pr}(v_u)$ and $B(e_{f,f}) = B(e_{f,f}) - b_r$. Note that mapping here is contingent upon available resources in nodes at this layer. Namely, if a fog node in the cluster does not possess sufficient resources, then it handovers the VNF to its adjacent neighbors within the same layer. Moreover, if no available resources at all nodes in the cluster n_f^j, then the VNF is relayed to the cluster heads in Layer II n_h^k. Similarly, if Layer II does not possess enough resources, then the VNF mapping is performed at Layer III, i.e., cloud nodes n_c^l, $[n_c^l \in N']$.

Finally, when all the VNFs in r are mapped (see pseudocode in Algorithm 2), then r is flagged as successful. The VNFs are now mapped using a hypersivor or container on the node [33].

Algorithm 2 Successful SFC Provisioning for Request r

1: **if** All $v_u \in V_r$ successfully mapped, reserve resources **then**
2: /* *Update* $G = (N, E)$, *link and node capacity along route for* r */
3: /* *Start Docker containers or VMs*/
4: /* *Start data transmission phase for mapped request* r */
5: Counter $= 0$
6: **while** *Counter* $< L_r$ **do**
/* *Start service time counter*/
7: *Counter*++
8: **end while**
9: /* *Stop service for request* r *when Counter* $= L_r$ */
10: /* *Free reserved nodes and links resources along the path for request* r*/
$Q_{me}(n) = Q_{me}(n) + Q_{me}(v_u)$,
$Q_{pr}(n) = Q_{pr}(n) + Q_{pr}(v_u)$,
$B(e) = B(e) + b_r$
11: **else if** All $v_u \in V_r$ could not mapped successfully **then**
12: /* *Fail the request* */
13: **end if**

Then data transmission phase is initiated (service starts) for terminals, and resources are reserved for the entire lifetime period, L_r, of the request. Namely, a counter is initiated here to record the elapsed time over which a request is in transmission phase. Once this time has reached the request lifetime, $Counter = L_r$, then data transmission phase is terminated and resources are released by updating node and link capacity in $G' = (N', E')$. In notations, $Q_{me}(n_f^j) = Q_{me}(n_f^j) + Q_{me}(v_u)$, $Q_{pr}(n_f^j) = Q_{pr}(n_f^j) + Q_{pr}(v_u)$ and $B(e_{f,f}) = B(e_{f,f}) + b_r$. As a result, this setting relieves the limited resources in the nodes for use by other requests. In turn, network capacity and traffic volume is increased, i.e., accommodating more requests.

Meanwhile if the request is delay-tolerant, $\delta_r > \delta_{th}$, then the VNF is directly transversed to the fog cluster heads (FCHs) for mapping. This saves resources in the fog nodes for other delay-sensitive requests. Similarly, the network is pruned to select the hosting nodes for all the VNFs in the SFC for the delay-tolerant request. Here a feasible graph G' is built that is composed of candidate nodes $n_h^k \in N'$ with $Q_{me}(n_h^k) > Q_{me}(v_u)$ or $Q_{pr}(n_h^k) > Q_{pr}(v_u)$, and links $e_{h,h} \in E'$ with $B(e_{h,h}) > b_r$. Then the scheme also computes P routes from $Prev$ to all $n_h^k \in N'$, where each route p here satisfies the minimum network delay for request r, δ_r, i.e., $min\{D(r)+D(e)\}$. Following the route and node selection process, service is provided for the request lifetime L_r, during which resources are reserved. Note that if no resources are available at the FCH in Layer II, then the VNFs are mapped to cloud nodes in Layer III. Furthermore, if no resources are available at the cloud nodes, then the request is dropped. Consequently increasing the dropping rates.

TABLE 2. Simulation parameters.

Parameter	Value
Number of nodes: n_t^i, n_f^j, n_h^k, n_c^l	25, 5, 3, 2
Number of nodes per cluster	5
Node processing capacity (GHz): $Q_{pr}(n_f^j)$, $Q_{pr}(n_h^k)$, $Q_{pr}(n_c^l)$	5, 15, 100
Node memory (GB): $Q_{me}(n_f^j)$, $Q_{me}(n_h^k)$, $Q_{me}(n_c^l)$	32, 64, 128
Link bandwidth capacity (Gbps): $B(e_{t,f})$, $B(e_{f,f})$, $B(e_{f,h})$, $B(e_{h,h})$, $B(e_{h,c})$, $B(e_{t,c})$, $B(e_{c,c})$	0.054, 0.1, 0.2, 0.3, 10, 10, 20
Number of VNF per request	Rand[1-5]
Requested VNF processing: $Q_{pr}(v_u)$	1-3 (uniform)
Requested VNF memory: $Q_{me}(v_u)$	Rand [1-5]
Request bandwidth (Mbps): b_r	1-10 (uniform)
Number of requests per batch: R	100:100:1000
Traffic load: A_r (KB)	70
Edge transmission rate (rand[] ms)/traffic unit (GB): $\delta(e_{t,f})$, $\delta(e_{f,f})$, $\delta(e_{f,h})$, $\delta(e_{h,h})$, $\delta(e_{h,c})$, $\delta(e_{c,c})$	[0.5, 1.5], [2, 3], [2.5, 3.5], [15, 35], [15, 35], [20, 30]
VNF processing rate at nodes: $\delta(n_f^j)$, $\delta(n_c^k)$, $\delta(n_c^l)$	0.03, 1.03, 3.12
VNF license cost ($): $\rho(v_u)$	100
Node cost per resource unit ($/processor): $\rho_{pr}(n_f^j)$, $\rho_{pr}(n_h^k)$, $\rho_{pr}(n_c^l)$	5, 3, 2
Node cost per resource unit ($/memory): $\rho_{me}(n_f^j)$, $\rho_{me}(n_h^k)$, $\rho_{me}(n_c^l)$	6, 4, 3
Power parameters: $\beta(n)$, w_x, $w(n_f^j)$, $w(n_h^k)$, $w(n_c^l)$	0.7, 15W, 65W, 130W, 300W

VI. PERFORMANCE EVALUATION

The proposed DAFC provisioning scheme is now evaluated for the multi-layer HFC architecture versus traditional cloud and fog architectures using key performance metrics. This includes the number of successful (satisfied) requests, network delay, energy consumption, and realization cost. See Table 2 for the network parameters and their assigned values [23], [24], [37]–[39]. Now consider the followings in the simulation settings. The generated requests from the terminals are sent to a request pool. Then they are managed by the provisioning scheme based upon their arrival time, where the scheme aims to find the best combination among fog and cloud nodes to serve incoming requests. Thereafter, some statistical data are recorded when all requests are processed, such as the average number of satisfied requests, and network delay and cost, as presented next.

A. NUMBER OF SUCCESSFUL REQUESTS

High traffic volume from terminals in Layer I results in aggregated number of incoming requests of various requirements to the higher layers. These requests impose high demand on the available resources in the substrate network and impose challenges on the SFC provisioning process. Consequently, some requests can be dropped when resources and links become more congested. Hence SFC provisioning scheme is performed here on the three network architectures to conclude an optimum placement with the highest number of satisfied requests. This value is gauged in Figure 3 and it

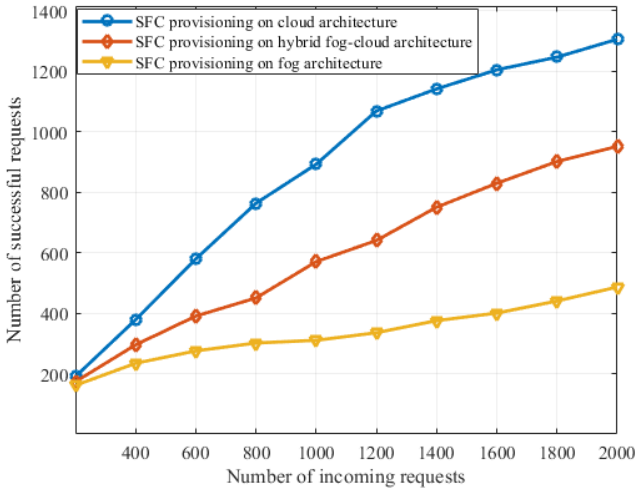

FIGURE 3. Number of successful requests for different architectures.

shows that cloud architectures return the highest capacity in terms of the number of satisfied incoming requests. Cloud solutions approximately accommodate 90-95% for the first 1000 incoming requests and 65-75% for 1600-2000 requests. This is attributed to the abundant available resources at the cloud nodes, at the detriment of increased propagation delays. Meanwhile, fog architectures suffer from significant requests drop rate, due to the limited resources on the fog nodes, i.e., satisfying 70%, 37%, and 25% for 400, 800 and 1600-2000 incoming requests. Note that fog nodes become highly saturated after the first 800 incoming requests, which results in dropping many incoming traffic from Layer 1 (60-70% dropping rates). Meanwhile, the proposed HFC architecture yields a tradeoff between cloud and fog solutions. Despite the close success rates in the hybrid and fog architectures for the first 400 requests, however the proposed HFC architecture is capable of satisfying more requests when traffic volume increases further. For instance, the HFC architecture yields in 75%, 62% and 50% success rates for 400, 800 and 1600-2000 incoming requests, respectively. Hence it achieves 15-40% higher capacity as compared to fog solutions for the same traffic volume. Moreover, the architecture here utilizes the *fog cluster head* (FCH) to accommodate the increased demand on resources at reduced processing times and latencies.

Note that the number of satisfied requests is an important key metric for delay-sensitive requests. When mapping the incoming delay-sensitive requests on the fog nodes, then these requests are deemed successful if $D(r) < \delta(r)$. Here its important to avoid mapping delay-sensitive on the cloud nodes, as this can yield in prolonged delay on the request delay, $D(r)$, until it exceeds $\delta(r)$. Consequently, the request is dropped from the network and denied access. Therefore, the terminal repeatedly attempts to access the network again. These attempts result in additional propagation, queuing and processing times in the network, hence yielding increased latencies in the control-plane for delay-sensitive applications.

TABLE 3. Processing times for various VNFs.

Network Function	Processing Time for VNFs (ms)
Load balancer	0.6475
Firewall	7.0771
VPN Function	1.6385

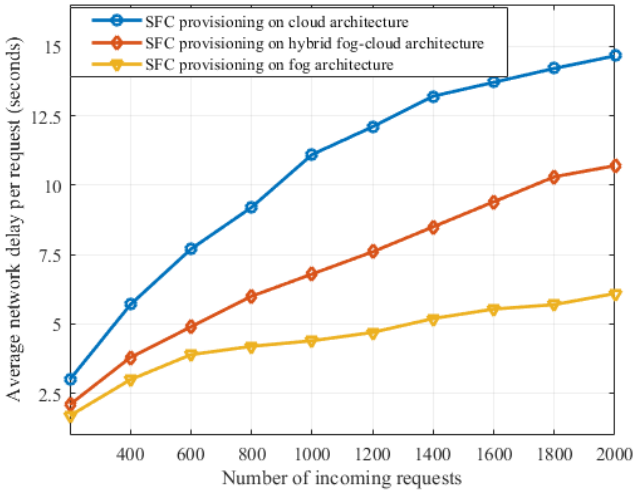

FIGURE 4. Overall network delay for different architectures.

B. OVERALL NETWORK DELAY

The overall network delay is composed of the processing, queuing, transmission and propagation delays. It is defined as the time required by the VNFs to process incoming packets of various applications, e.g., firewall, load balancer, and VPN function. This parameter is gauged by the processing time of the overall number of software-implemented VNFs at each fog or cloud node, as per Table 3 [34].

Figure 4 shows the network delay for the proposed DAFC provisioning scheme implemented on the cloud, fog, and HFC architectures. It is noticed here that cloud architectures yield excessive network delays, e.g., 6.4 and 9 seconds for 800 and 1600 incoming requests, respectively. This is attributed to the long propagation delays for packets traversed over nodes separated by large geographical areas. Finally, fog architectures yield in very short delays due to the low processing times at fog nodes and proximity to the terminals (short propagation times), e.g., 4 and 6 seconds for 800 and 1600 requests, respectively (at the expense of low capacity).

Meanwhile, the proposed HFC architecture achieves reduced delays at various number of requests. Namely, it returns significant reduction as compared to the cloud architecture, and slight increment over fog solutions at larger number of requests, e.g., 9.2 and 13.6 seconds for 800 and 1600 requests, respectively. Note that the small privilege for fog architectures here is contingent upon resources availability at the fog nodes, in order to support such a high number of satisfied requests. Also, standalone fog architectures can suffer from increased delays at high traffic volumes. This is in contrast to the proposed scheme that can accommodate high traffic volumes.

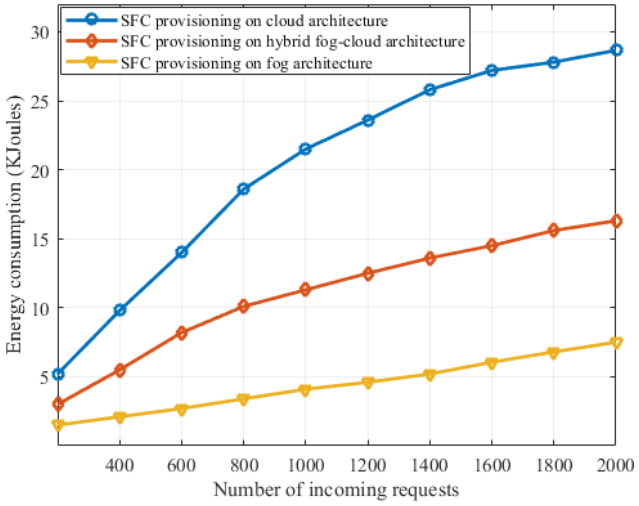

FIGURE 5. Energy consumption levels for different architectures.

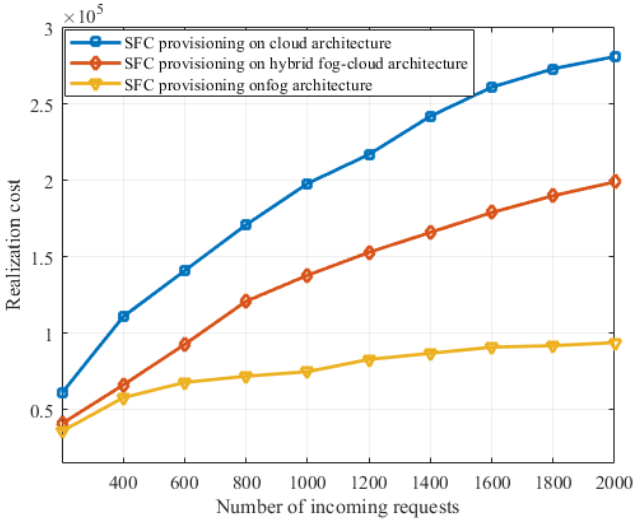

FIGURE 6. Realization cost for different architectures.

A key saliency for the propsoed HFC architecture is the relatively low processing times at increased number of requests. For instance, fog architectures can suffer from aggregated processing times when resources are limited. Consequently, high number of incoming requests can be dropped. This is in contrast to the abundant resources at the HFC solution. Moreover, this multi-layer HFC solution achieves significant reduction in processing times versus cloud architectures, i.e., approximately 25% and 48% less times at 1000 and 2000 requests, respectively. Note Table 2 for various delay ranges between different layers.

C. ENERGY CONSUMPTION

The energy consumption for the SFC provisioning scheme is plotted in Figure 5 for the various architectures. Specifically, this value is gauged by measuring the overall power consumption, W, during the entire network delay $D(r)$. It accounts for the power consumption levels $w(n)$ at the fog and cloud nodes in Layers II & III, as well as w_x power consumption in X number of switches between these nodes. See Table 2 for parameters settings [35], [36]. In notations,

$$\epsilon = D(r).W = D(r).(\sum_{n \in N} n.w(n)).Xw_x, \tag{18}$$

where the power consumption at each node, $w(n)$, i.e.,

$$w(n) = \beta(n).w(n)|_{max} + (1 - \beta(n)) + \zeta(n)|_{max}\,, \tag{19}$$

where $\beta(n)$ denotes the power consumption rate in idle mode, $w(n)_{max}$ is the maximum power consumption for nodes in different layers, and $\zeta(n)$ represents the utilization (saturation) factor at any node used for SFC provisioning.

Figure 5 shows that the proposed HFC architecture consumes reduced energy levels at high number of incoming requests. For example, the HFC architecture requires in order of 11 and 16 KJoules for 1000 and 2000 requests, versus 21-27 KJoules for cloud, and 4-8 KJoules for fog architectures for the same number of requests. Note here that fog nodes in Layer II approaches saturation earlier (100% utilization) after 300 requests, $\zeta(n_f^j) = 1$. However, this architecture still accommodates requests, as previously occupied resources are released when earlier hosted requests reach their lifetime counter. As a result, resources become available to host new requests (e.g., at 1600 incoming requests). This is compared to 900 requests for nodes in Layer III, i.e., $\zeta(n_h^k) = 1$. This demonstrates the benefit of the fog cluster heads (FCHs) in the proposed architecture. Finally, cloud architectures require excessive amounts of energy to accommodate the large number of requests, e.g., 22 KJoules for mapping 1000 requests, at abundant amount of resources, $\zeta(n_c^l) = 0.7$ at 1000 requests. Overall, the proposed HFC architecture again yields a tradeoff between cloud and fog counterpart.

D. REALIZATION COST

This cost is calculated for the different architectures at various numbers of incoming requests, as per Figure 6. Here the proposed HFC architecture yields an average cost tradeoff between cloud and fog architectures. Namely, it leverages the reduced processing times in fog architectures, and the abundant resources available in cloud architectures. Also, the proposed DAFC scheme yields efficient fog and cloud nodes utilization to satisfy each type of request separately according to the service requirements, while keeping costs and delay at minimum.

Overall, the proposed provisioning scheme shows that the performance for the HFC architecture yields a tradeoff between cloud and fog solutions. It approaches the performance of cloud architectures in terms available resources and successful requests. Meanwhile, it approaches the performance of fog architectures in terms of reduced network delays, realization costs, and energy consumption. Thus making the proposed architecture a suitable solution for delay-sensitive and delay-tolerant requests of various requirements. Such as, real-time video applications and data storage.

TABLE 4. Computational complexity of provisioning schemes.

Provisioning Scheme	Computational Complexity
Graph-based heuristics, [11]-[13], [18]	Polynomial time, $O(N^k)$
Metaheuristics [19] [21]	Exponential time,$O(2^n)$
Global optimization [23],[27]	Exponential time, $O(2^N)$
Proposed DAFC provisioning scheme	Linear time, $O(\|N\| log \|E\|)$

E. COMPUTATIONAL COMPLEXITY

The computational complexity of the different SFC provisioning schemes in cloud and fog computing architectures are classified into the following. First, graph-based heuristics such as the work in [11]–[13] and [18] implement graph centrality, graph-clustering and multi-stage graphs to find the best path for request r in the SFC placement. Generally, the proposed methods here reduce the complexity and convergence times, since the computational complexity here essentially depends only on the physical graph sizes. These schemes feature polynomial time computational complexity, modeled as $O(N^k)$. See Table 4 for the run-time computational complexity of the different provisioning schemes.

Meanwhile, global optimization techniques such as the MILP and ILP in [7] and [21] suffer from high computational complexity, i.e., exponential run-time, modeled as, $O(2^N)$. This high complexity is attributed to the evaluation of the objective function on the entire search space (nodes) in the network, in order to find the best solution to host the VNFs. Furthermore, metaheuristic schemes, such as Tabu search [19] performs greedy process to select the host node from each sub-region, where it maintains locally optimal selection for each VNF. The computational complexity here is characterised by an exponential process $O(2^n)$, $n \ll N$.

Meanwhile, the computational complexity of the proposed SFC provisioning scheme implemented on the HFC architecture along with the fog, cloud solutions is now analyzed. The proposed provisioning scheme determines E number of routes from the source to the first candidate node in the network. This route has multiple nodes and links. Therefore, the algorithm iterates over E routes between the previous node (e.g., source or *prev*) to the candidate node, in order to select a single node from the route that yields the least delay and load. Thereafter, the scheme examines the route possesses a node with sufficient resource with the least delay, i.e., first-fit (first suitable) node in the shortest route. Hence a single route is selected for each request from the network graph $G(N, E)$. Assuming the worst-case scenario (utilizing the whole network nodes and routes), then the run-time computational complexity is bounded by $O(|N| \, log \, |E|)$.

In light of the above, the proposed provisioning scheme on the HFC architecture features in reduced computational complexity as compared to the cloud and fog counterpart for the same number of successful requests. This is because it consumes reduced network resources, i.e., reduced nodes (links) usage compared to standalone fog (cloud) solutions. This is in contrast to the proposed scheme the computational run-time complexity for the proposed hybrid model is based on a first-fit approach, where it selects the first best node in the selected shortest path p that yield the least delay or load.

Overall, the aforementioned performance results of the proposed DAFC provisioning scheme benefit network operators in achieving higher user capacities, accommodating larger traffic volume, at improved *quality-of-service* (QoS) in terms of latency and bandwidth. It also yields in reduced operating and capital expenses (OPEX and CAPEX), as less number of nodes and links are used to accommodate incoming requests, at reduced power and energy consumption levels at nodes and switches.

It is also interesting to consider the proposed SFC provisioning scheme on multi-tier fog architectures, composed of heterogeneous nodes. Namely, implementing fog nodes of different available resources, at the edge of the network, over which the network capacity, delay, cost and energy efficiency can be investigated. Moreover, this work can be extended to include failure probabilities associated with nodes and links in the network. Hence it is also important to consider SFC provisioning schemes that account for network failures. Here developing various restoration methods becomes important for network recovery, while taking into account single- and multi-failure, and recovery times.

VII. CONCLUSION

There is a growing need to implement service function chaining support in emerging fog-cloud infrastructures. Hence this paper presents a novel delay-aware provisioning scheme for that maps the virtual network functions in fog-cloud architecture, composed of fog nodes, fog cluster heads and cloud nodes. Moreover, the scheme accounts for several key attributes for incoming requests. Foremost, delay, resources and bandwidth requirements, and lifetime. Overall findings confirm that the proposed hybrid architecture outperforms fog and cloud counterparts at high number of incoming requests. Future efforts will investigate failure awareness service provisioning on the proposed hybrid architecture. Moreover, these efforts will study the impact of the cluster heads as protection nodes in network survivability.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, Jun. 2015.

[3] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 1826–1857, 3rd Quart., 2018.

[4] D. Oliveira, J. Crichigno, N. Siasi, E. Bou-Harb, and N. Ghani, "Joint mapping and routing of virtual network functions for improved disaster recovery support," in *Proc. SoutheastCon*, Apr. 2018, pp. 1–8.

[5] J. G. Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 3, pp. 518–532, Sep. 2016.

[6] Z. Ye, X. Cao, J. Wang, H. Yu, and C. Qiao, "Joint topology design and mapping of service function chains for efficient, scalable, and reliable network functions virtualization," *IEEE Netw.*, vol. 30, no. 3, pp. 81–87, May 2016.

[7] 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.

[8] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Proc. IEEE 3rd Int. Conf. Cloud Netw. (CloudNet)*, Luxembourg City, Luxembourg, Oct. 2014, pp. 7–13.

[9] A. Gupta, B. Jaumard, M. Tornatore, and B. Mukherjee, "A scalable approach for service chain mapping with multiple SC instances in a wide-area network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 529–541, Mar. 2018.

[10] N. Siasi, N. I. Sulieman, and R. D. Gitlin, "Ultra-reliable NFV-based 5G networks using diversity and network coding," in *Proc. IEEE 19th Wireless Microw. Technol. Conf. (WAMICON)*, Clearwater, FL, USA, Apr. 2018, pp. 1–4.

[11] N. Tastevin, M. Obadia, and M. Bouet, "A graph approach to placement of service functions chains," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, May 2017, pp. 134–141.

[12] M. Mechtri, C. Ghribi, and D. Zeghlache, "A scalable algorithm for the placement of service function chains," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 3, pp. 533–546, Sep. 2016.

[13] M. Wang, B. Cheng, W. Feng, and J. Chen, "An efficient service function chain placement algorithm in a MEC-NFV environment," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[14] Y. Qiu, H. Zhang, K. Long, H. Sun, X. Li, and V. C. M. Leung, "Improving handover of 5G networks by network function virtualization and fog computing," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, Qingdao, China, Oct. 2017, pp. 1–5.

[15] J. Liu, S. Zhou, J. Gong, Z. Niu, and S. Xu, "Graph-based framework for flexible baseband function splitting and placement in C-RAN," in *Proc. IEEE Int. Conf. Commun. (ICC)*, London, U.K., Jun. 2015, pp. 1958–1963.

[16] S. Yangui, P. Ravindran, O. Bibani, R. H. Glitho, N. B. Hadj-Alouane, M. J. Morrow, and P. A. Polakos, "A platform as-a-service for hybrid cloud/fog environments," in *Proc. IEEE Int. Symp. Local Metrop. Area Netw. (LANMAN)*, Rome, Italy, Jun. 2016, pp. 1–7.

[17] O. Bibani, C. Mouradian, S. Yangui, R. H. Glitho, W. Gaaloul, N. B. Hadj-Alouane, M. Morrow, and P. Polakos, "A demo of IoT healthcare application provisioning in hybrid cloud/fog environment," in *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Luxembourg City, Luxembourg, Dec. 2016, pp. 472–475.

[18] R. Deng, R. Lu, C. Lai, and T. H. Luan, "Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, London, U.K., Jun. 2015, pp. 3909–3914.

[19] N. Siasi, A. Jaesim, and N. Ghani, "Tabu search for efficient service function chain provisioning in fog networks," in *Proc. IEEE 5th Int. Conf. Collaboration Internet Comput. (CIC)*, Dec. 2019, pp. 145–150.

[20] C. Mouradian, S. Kianpisheh, and R. H. Glitho, "Application component placement in NFV-based hybrid cloud/fog systems," in *Proc. IEEE Int. Symp. Local Metrop. Area Netw. (LANMAN)*, Washington, DC, USA, Jun. 2018, pp. 25–30.

[21] C. Mouradian, S. Kianpisheh, M. Abu-Lebdeh, F. Ebrahimnezhad, N. T. Jahromi, and R. H. Glitho, "Application component placement in NFV-based hybrid cloud/fog systems with mobile fog nodes," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1130–1143, May 2019.

[22] E. S. Gama, R. Immich, and L. F. Bittencourt, "Towards a multi-tier fog/cloud architecture for video streaming," in *Proc. IEEE/ACM Int. Conf. Utility Cloud Comput. Companion (UCC Companion)*, Zurich, Switzerland, Dec. 2018, pp. 13–14.

[23] R. Chaudhary, N. Kumar, and S. Zeadally, "Network service chaining in fog and cloud computing for the 5G environment: Data management and security challenges," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 114–122, Nov. 2017.

[24] D. Rosário, M. Schimuneck, J. Camargo, J. Nobre, C. Both, J. Rochol, and M. Gerla, "Service migration from cloud to multi-tier fog nodes for multimedia dissemination with QoE support," *Sensors*, vol. 18, no. 2, p. 329, Jan. 2018.

[25] M. Tajiki, M. Shojafar, B. Akbari, S. Salsano, and M. Conti, "Software defined service function chaining with failure consideration for fog computing," *Concurrency Comput., Pract. Exper.*, vol. 31, pp. 1–4, Sep. 2018.

[26] N. B. Truong, G. M. Lee, and Y. Ghamri-Doudane, "Software defined networking-based vehicular adhoc network with fog computing," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, Ottawa, ON, Canada, May 2015, pp. 1202–1207.

[27] V. Balasubramanian, S. Otoum, M. Aloqaily, I. Al Ridhawi, and Y. Jararweh, "Low-latency vehicular edge: A vehicular infrastructure model for 5G," *Simul. Model. Pract. Theory*, vol. 98, Jan. 2020, Art. no. 101968.

[28] I. Al Ridhawi, Y. Kotb, M. Aloqaily, Y. Jararweh, and T. Baker, "A profitable and energy-efficient cooperative fog solution for IoT services," *IEEE Trans. Ind. Informat.*, vol. 16, no. 5, pp. 3578–3586, May 2020.

[29] G. Li, J. Wu, J. Li, Z. Zhou, and L. Guo, "SLA-aware fine-grained QoS provisioning for multi-tenant software-defined networks," *IEEE Access*, vol. 6, pp. 159–170, 2018.

[30] *Fog Computing and Networking Architecture Framework IEEE Communications Society—IEEE Standard for Adoption of OpenFog Reference Architecture for Fog Computing*, IEEE Standard 1934-2018, Aug. 2018.

[31] N. Siasi and A. Jaesim, "Priority-aware SFC provisioning in fog computing," in *Proc. IEEE 17th Annu. Consum. Commun. Netw. Conf. (CCNC)*, Las Vegas, NV, USA, Jan. 2020, pp. 1–6.

[32] B. Ngo and H. Lee, "Analysis of a pre-emptive priority M/M/c model with two types of customers and restriction," *Electron. Lett.*, vol. 26, no. 15, pp. 1190–1192, Jul. 1990.

[33] N. Siasi, M. A. Jasim, J. Crichigno, and N. Ghani, "Container-based service function chain mapping," in *Proc. SoutheastCon*, Huntsville, AL, USA, Apr. 2019, pp. 1–6.

[34] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, Ottawa, ON, Canada, May 2015, pp. 98–106.

[35] O. Soualah, M. Mechtri, C. Ghribi, and D. Zeghlache, "Energy efficient algorithm for VNF placement and chaining," in *Proc. 17th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGRID)*, Madrid, Spain, May 2017, pp. 98–106.

[36] D. Boru, D. Kliazovich, F. Granelli, P. Bouvry, and A. Y. Zomaya, "Energy-efficient data replication in cloud computing datacenters," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Shenzhen, China, Dec. 2013, pp. 385–402.

NAZLI SIASI (Member, IEEE) received the B.S. degree in information technology and computer engineering from Tehran Polytechnic University, the M.S. degree in information technology and computer engineering from Tabriz State University, and the M.S. and Ph.D. degrees in electrical engineering from the University of South Florida, Tampa, FL, USA. She is currently an Assistant Professor of cybersecurity with Christopher Newport University, Newport News, VA, USA. Her research interests include networking, virtualization, and fog and cloud computing.

MOHAMMED JASIM (Member, IEEE) received the bachelor's degree in electrical engineering from Applied Science University, Jordan, the master's degree in electrical engineering from Brunel University London, U.K., and the Ph.D. degree in electrical engineering from the University of South Florida, USA. He was with Valparaiso University, Valparaiso, IN, USA. He is currently an Assistant Professor of electrical engineering with the University of Mount Union, Alliance, OH, USA. His research interests include millimeter wave communications, beamforming, fog and cloud computing, and network function virtualization.

ADEL ALDALBAHI (Member, IEEE) received the B.S. degree in electrical engineering from Virginia Commonwealth University, Richmond, VA, USA, in 2011, and the M.S. and Ph.D. degrees in electrical engineering from the New Jersey Institute of Technology, Newark, NJ, USA, in 2013 and 2017, respectively. He joined with the Electrical Engineering Department, King Faisal University, as an Assistant Professor, in 2018. His research projects are funded by King Faisal University and Ministry of Higher Education (MOHE). His current research interests include visible light communication, wireless networks, modeling, analysis, and millimeter wave for 5G and beyond. He served as a TPC for ICWMC 2019 and 2020, as a Reviewer for IEEE Access, the IEEE Photonics, and the IEEE ISSPIT 2018. He served as a Session Chair for WIMOB 2018 and 2019, ICCAIS' 2019. He served as a Publicity Chair for WIMOB 2020.

NASIR GHANI (Senior Member, IEEE) received the bachelor's degree in computer engineering from the University of Waterloo, the master's degree in electrical engineering from McMaster University (supervisor Dr. Simon Haykin, Life Fellow, IEEE), and the Ph.D. degree in computer engineering from the University of Waterloo (supervisor Dr. Jon. W. Mark, Life Fellow, IEEE). He was an Associate Chair with the ECE Department, University of New Mexico. He was a Faculty Member with Tennessee Tech University. He is currently a Professor with the Electrical Engineering Department, University of South Florida. He is also a Research Liaison with Cyber Florida. He also spent several years working with large Blue Chip organizations (IBM, Motorola, Nokia) and several hi-tech startups. His research has been supported by the National Science Foundation, Defense Threat Reduction Agency, Department of Energy, Qatar Foundation, and Sprint-Nextel. He has published over 200 peer-reviewed publications. His research interests include cyberinfrastructure networks, cybersecurity, cloud computing, disaster recovery, and cyber-physical systems. He received the NSF CAREER Award, in 2005, and the Best Paper Awards at IEEE PIMRC 2017 and IEEE ANTS 2010. He has chaired symposia for the IEEE GLOBECOM, the IEEE ICC, the IEEE ICCCN, and workshops for the IEEE INFOCOM. From 2007 to 2010, he was also the Chair of the IEEE Technical Committee on High Speed Networking (TCHSN). He has served as an Associate Editor for the IEEE/OSA Journal of Optical and Communications and Networking, the IEEE Systems, and the IEEE Communications Letters. He has also edited special issues of the IEEE Network and the *IEEE Communications Magazine*.