


# Routing in Fog-Enabled IoT Platforms: A Survey and an SDN-Based Solution

Feyza Yildirim Okay and Suat Ozdemir 

**Abstract**—Fog computing is a promising technology that helps overcome to the difficulties of handling a huge amount of Internet of Things (IoT) data by distributing its applications and services closer to the network edge. In fog computing, distributed fog servers' proximity to the end devices enables upcoming and outgoing data to be forwarded efficiently. However, as the size of fog-enabled IoT platforms increases, efficient routing mechanisms are required in fog computing to forward the ubiquitous IoT data to related servers with low latency, low bandwidth, and/or high security. Therefore, software defined networking (SDN) can be employed to optimize the routing process in fog-enabled IoT platforms. In this paper, we first identify the routing requirements of fog computing. Then, we present an extensive survey of routing in fog-enabled IoT platforms. Lastly, we propose a hierarchical SDN-based fog computing architecture for routing in fog-enabled IoT platforms where fog controllers take actions locally for frequent events while the cloud controller takes actions globally for rare events. The proposed framework is evaluated by varying the number of controllers. The initial results show that the proposed hierarchical SDN-based framework has the potential to reduce routing delay and data transmission overhead. Also, throughput is increased as the number of controllers increases.

**Index Terms**—Fog computing, Internet of Things (IoT), routing, software defined networking (SDN).

## I. INTRODUCTION

INTERNET of Things (IoT) is an emerging paradigm that brings people, machines, and objects together. To improve life quality, it is expected to grow in the future as well [1]. According to Cisco, 50 billion devices are expected to be connected to the Internet by 2020 [2]. Considering the data generated by billions of IoT devices, how to cope with this pervasive and ubiquitous tremendous amount of data in real-time has become the key challenge. In this regard, over the past decades, cloud computing technology has been offered as a perfect candidate with its high computation power and storage capability, on-demand access to a virtually shared pool of computing and storage resources available at a centralized cloud. However, sending all data to a centralized cloud for computing and storing processes may result in congestion, high latency, and poor quality of service (QoS) as the number of IoT devices

increases [3]. Therefore, fog computing concept has emerged to compensate for these deficiencies of cloud computing by moving computing, storage and networking capabilities to the edge of the network. The characteristics of fog computing and its improvements are mainly due to the fog server's proximity to the end users and their distributed structure. Rather than a centralized cloud server, a huge amount of data is processed in a distributed manner by a number of fog servers located at the edge of the network. These fog servers have enough capability of computing, processing, and temporary storage; so they can provide services and applications to end users with low-latency, location awareness, and high mobility. In addition, they collaborate with the cloud server and transmit the data to the corresponding cloud server for further computing, processing, and storage tasks [4]–[6].

Fog computing may generally pose a three-tier architecture where different types of communications occur among tiers and peers. The priority of communication depends on the status of the devices, the application to be performed, the link quality between devices, and the quality of the requested services. Fog servers are located closer to end devices and generally connected directly to these end devices. However, for large-scale networks, fog servers may communicate with some end devices over a number of hops. As a result, routing algorithms play an important role in determining how data generated by a specific end device is transmitted to its own fog server or to a neighboring fog server. There are several ways in which data can be transmitted from a certain end device to a neighboring fog server. Two ways of data transmission can be explained as follows.

- 1) The data is first transmitted to neighboring end devices to reach the coverage area of neighboring fog server. Then, this data is transmitted to the neighboring fog server through one or few hops over end devices.
- 2) The data is first transmitted to the respective fog server of the end device through one or few hops. Then, this data is transmitted to the neighboring fog server.

Software defined networking (SDN) is an auxiliary paradigm for fog computing by decoupling network traffic as control plane and data plane to take routing decisions easily and flexibly. It has a global view on network-wide distributed forwarding elements such as fog servers and end devices, and it creates a flexible, scalable, and programmable decision-making progress and routing mechanism [7]. However, as the network size increases, it may be hard to manage and control all forwarding elements by only one controller. Therefore, in this paper, we propose a distributed SDN concept where there

Manuscript received March 22, 2018; revised July 12, 2018 and October 14, 2018; accepted November 12, 2018. Date of publication November 21, 2018; date of current version January 16, 2019. This work was supported by the Scientific and Technological Research Council of Turkey under Grant 118E212. (Corresponding author: Suat Ozdemir.)

The authors are with the Computer Engineering Department, Gazi University, 06500 Ankara, Turkey (e-mail: feyzaokay@gazi.edu.tr; suatozdemir@gazi.edu.tr).

Digital Object Identifier 10.1109/IIOT.2018.2882781

are multiple controllers, generally, hierarchically distributed in fog and cloud layers. In our proposed architecture, controllers are distributed in fog and cloud layers. Our initial simulation results indicate that the distributed SDN concept is a good candidate for routing requirements of fog-enabled IoT platforms and services. A fog controller manages its related fog network which includes one or more fog server and several end devices located underneath whereas the cloud controller manages the whole network. The contributions of this paper are threefold.

- 1) Fog computing presents many opportunities for many IoT services in which different routing requirements must be satisfied. We first deeply analyze services with respect to their routing needs and present a comprehensive survey of the related work.
- 2) We propose a hierarchical SDN-based fog computing architecture by using SDN as a promising technology to provide substantial improvements in network utilization. In addition, we benefit from SDN's high performance on decision-making and traffic management capability. By distributing controllers, the network is controlled locally in the fog level and globally in the cloud level. While local controllers reduce latency, a global controller monitors the network and takes important decisions that affect all forwarding devices to provide an efficient routing policy management and scalability.
- 3) The proposed hierarchical SDN-based fog computing architecture for fog-enabled IoT services is evaluated with a different number of controllers in terms of delay, data transmission overhead, and throughput over different scenarios. During experiments, first, the delay is measured in terms of the time required for transmitting the first packet from source host to destination host. Then, the data transmission overhead is calculated by the number of transmitted OpenFlow (OF) packets among controllers and switches. Lastly, throughput is measured for different duration times, scenarios, and the number of controllers.

The rest of this paper is organized as follows. Section II gives detailed information about the fog computing architecture and defines different paradigms that are related to fog computing. Section III identifies the routing needs for fog-enabled IoT platforms and summarizes the important related works in the literature. Section IV provides background information about SDN and summarizes the existing works on SDN-based fog-enabled IoT services. Section V presents our hierarchical SDN-based fog computing architecture and evaluates experimental results over different scenarios. Section VI emphasizes the open research areas and future directions of this paper. Finally, Section VII concludes this paper.

## II. FOG COMPUTING

Fog computing presents additional features to traditional cloud approaches by moving computing, storing, and networking capabilities from the central structure to the edge of the network. Its key opportunities are its proximity and geographically distributed nature that both improve its capabilities

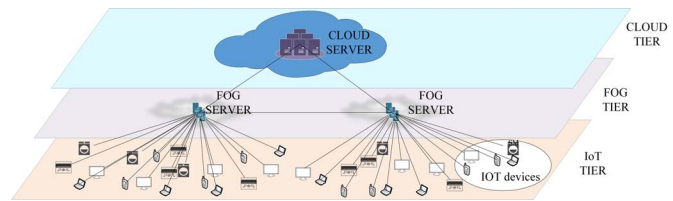


Fig. 1. Fog computing architecture.

vertically (less time to reach fog servers) and horizontally (less amount of data to compute or storage). In other words, fog computing removes the necessity of uploading/downloading all data to a distant and centralized cloud server for every request of cloud users. Instead, fog servers can process data and local queries in a distributed manner. Therefore, the amount of transmitted data and time for data transmission are significantly reduced [8].

Fog computing and cloud computing are two paradigms work as complementary systems. After fog servers collect and process the data coming from IoT devices locally, they send those data to the cloud for further processing and permanent storage [9]. This relationship between fog and cloud computing requires a hierarchical structure. Many studies in the literature point to the three-tiered structure as illustrated in Fig. 1. According to this structure, a cloud server is located at the top and connected to a number of fog servers. Fog is located in the middle of the cloud server and end devices. Each fog server also interconnects with end devices located at the bottom.

Heterogeneous IoT devices can be static or mobile and, thus, location awareness of these devices becomes an important matter for cloud computing. However, due to its centralized structure, cloud computing may be insufficient to provide location-based services. In the case of cloud-based processing of a large amount of mobile user's data, high computation cost and increased delay are inevitable [10]. For example, let a tourist group visiting a new city be hungry and they start walking around to find the closest restaurant. In the cloud computing scenario, they first connect to the cloud to check nearby restaurants. Then, restaurant information for the entire city is received. According to the location of the tourist group, the location of the closest restaurant is calculated and returned to the users. Unfortunately, they get this content through a long-distance link from the cloud which is quite costly due to expensive cellular bandwidth, especially when the restaurant and mobile users are physically close to each other. Fog computing simplifies this scenario by providing location-based services. It does not need to move the data to the cloud, instead, it just process the data locally over a wireless connection such as WiFi, 4G, and 5G which are cheaper and faster. If we adapt the same scenario for fog computing, the tourist group connect to the nearest base station to search the location of the closest restaurant. Then, the base station seeks for restaurants in a specific area in the city where the users are located (consists of a much smaller number of restaurants when compared to cloud computing). After finding the closest restaurant, location information is returned to the users.

This also occurs in a shorter time period compared to cloud computing. As a result, fog computing may provide both time and cost savings.

There are some computing paradigms similar to fog computing. Although most of them share the same computing, processing, and networking capabilities, they are still different from each other in terms of several aspects. Studies [11]–[15] in the literature focus on these paradigms by comparing, examining, and discussing in different perspectives, including communication, computation, and security. The interested readers may refer to [11]–[15] for more detailed infer. In what follows, we explain each term briefly.

- 1) *Cloud Computing*: Over the past decades, cloud computing technology has been offered as a perfect candidate with its high computation power, storage capability, on-demand access to a virtually shared pool of computing and storage resources available at a centralized cloud center. As the amount of data rises, however, sending all data to a centralized cloud for computing and storing results in bandwidth bottleneck, high latency, network congestion, and QoS degradation [16]. Besides, most of the time it is enough to take decisions locally as explained in the above example.
- 2) *Cloudlet*: A cloudlet can be a resource-rich computer or a cluster of such computers. It can be considered as a micro-cloud, located closer to the end users so that it can offer its services quickly while maintaining Internet connectivity [17]. Cloudlets generally serve mobile users through WiFi access points. This means, cloudlets have a small coverage area and they may not support ubiquitous computing needs of IoT devices [1]. It can work in both stand-alone modes and with cloud connection.
- 3) *Mobile Cloud Computing (MCC)*: It is another trending topic based on a structure where the applications run outside handheld devices [18]. Since most of the mobile devices have energy, storage, and computational resource constraints, services and resources are provided to mobile users by remote cloud data centers (CDCs). Hence, MCC allows mobile users to benefit from high computational power and storage capacity of data centers [19].
- 4) *Mobile Edge Computing (MEC)*: MEC framework refers to a distributed MCC where the applications running at the edge of the network, close to the mobile device [20]. It aims to smooth and seamless integration of cloud capabilities into the mobile network [21]. As an edge-centric computing paradigm, it is very close to the fog computing concept. Its main difference is that the computation and storage tasks are realized in client edge devices, near the things and data sources, whereas fog computing performs same tasks on network edge devices, one or few hops away from client edge devices [22], [23].

### III. FOG-ENABLED IoT SERVICES WITH RESPECT TO ROUTING

Fog-enabled IoT services and their applications, generally, are deployed in large-scale platforms to perform different

tasks and services, such as load balancing and mobility. In accordance with this purpose, fog components are specialized for these tasks by considering the needs and interests of the network. When performing these services, all data must be sent to the appropriate locations (or fog servers) via communication channels. However, as the network size increases, transmission of these heterogeneous data can be challenging and, thus, proper routing mechanisms are needed to determine how to send these data to their destinations by taking into account to the demands of the applications, such as latency, bandwidth, or security.

According to the hierarchical structure of fog computing (by assuming the three-tiered structure), data is transmitted over four types of communication channels between tiers and peers: 1) end to end; 2) end to fog; 3) fog to fog; and 4) fog to cloud. Communications in these channels are bi-directional. IoT layer generates data by sensing the environment and transmits them to the fog layer. However, for some situations, the data cannot be transmitted directly to the upper layer due to a bandwidth constraint or a security reason. Instead, the data is transmitted over a neighboring device, called a relay node, which is capable of data forwarding.

In what follows, some of the computation, networking, or storage services that may need an effective routing mechanism during and after task deployment are listed. Then, a comprehensive survey is carried out by categorizing each service with respect to its routing needs.

- 1) *Mobility*: End devices can be static or mobile. In mobility assistant services, mobile devices can move away from their respective fog server's coverage area and move in another fog server's coverage area. For example, if an electric car is driven to a new city and needs to be charged, the owner needs to charge it using a random power outlet using its ID information. The smart meter of the power outlet should communicate with the fog server or smart meter of the car owner and send the charging information to it [24].
- 2) *Content Delivery and Caching*: Content delivery services are now shifting to the edge with the emerging fog computing technology. Fog servers can be used as content delivery nodes and they may serve the cached content to local users within low latency. Caching techniques can be classified into two subclasses as reactive and proactive caching. Especially, proactive caching allows users to obtain the desired information before they request it to reduce communication bandwidth and latency so that they may require higher network performance in the future. Routing algorithms help content delivery systems accomplish these goals effectively [23].
- 3) *Computation Offloading*: Due to the limited capacity of end devices, they may need to offload their computation tasks to the available fog server which is more resourceful. These tasks can be routed over an available fog server or end device. For example, an onboard fog server can be installed on a bus and during its traveling it can request available computing resources from roadside static fog servers. Therefore, by offloading some of

TABLE I  
ROUTING SCHEMES FOR FOG-ENABLED IOT SERVICES

Service	Reference	Contributions	SDN Usage	Routing Algorithm	Routing Requirements			
					Latency	Bandwidth	Security	Throughput
Mobility	Lu <i>et al.</i> [28]	- Presents a highly dynamic routing protocol used in VANET for city scenario. - Proposes VFC by taking advantages of fog computing in terms of geographical distribution and mobility.		IGR	✓			✓
	Fang <i>et al.</i> [29]	- Provides secure anonymity and secure transmitting. - Ensures energy efficiency by using Cyclic Redundancy Check (CRC-4).		SAL-AODV			✓	✓
	Soo <i>et al.</i> [30]	- Provides seamless task migration between fog devices by proactive fog services.		Adaptive routing protocol	✓			
	Ivanov <i>et al.</i> [31]	- Provides user-aware multicasting concept by taking into account user presence. - Resilient to the heterogeneity of sensor load.		GRP	✓	✓		
	Gupta <i>et al.</i> [32]	- Exhibits a steady average video quality. - Provides to a better rate of data delivery for the health-care scenario.	✓	QoS aware routing policy	✓			
Content Caching and Delivery	Jingtao [33]	- Finds the minimum weighted path to retrieve resources from fog cluster.		Steiner tree scheme	✓			
	Amin <i>et al.</i> [34]	- Differentiates IP and ICN flows. - Ensures reduced RTT, better throughput, and reduced upstream bandwidth. - Provides secure content sharing over two channel: HTTPS channel and ICN based data channel		DCAR	✓	✓	✓	✓
		- Enables flexible allocation of data flows. - Adapts changes on data flows dynamically.		LASR	✓			
Computation Offloading	Teranishi <i>et al.</i> [35]	- Ensures low-latency processing of visual data.						
	Trinh <i>et al.</i> [36]	- Improves throughput performance sustainability. - Allows flexible policy specification.		SPIDER	✓	✓		✓
	Huang <i>et al.</i> [37]	- Provides centralized management by SDN. - Finds a possible path between two vehicles. - Chooses the highest lifetime connection path among many possible paths. - Repairs broken paths.	✓	LT-NSR LT-PR				✓
		- Prolongs the network lifetime by balancing energy consumption on the basis random number of CH selection.						
Load Balancing	Naranjo <i>et al.</i> [38]	- Ensures fairness by giving no privileges to select a CH. - Enables energy heterogeneity. - Optimizes minimum distance between CHs and related fog devices.		P-SEP	✓			
	Hakiri <i>et al.</i> [39]	- Performs global and optimal path selection by using hybrid OLSR data forwarding and OpenFlow. - Ensures better QoS metrics like lower latency, overhead and bandwidth utilization. - Performs flexible load balancing to offload network load and select best routing path.	✓	OLSR	✓	✓		✓
	Betzler <i>et al.</i> [40]	- Balances network load to avoid congestion. - Ensures CPU aware routing by alleviating load on routing tasks and granting power to perform computing tasks. - Redirects data flow from external interferences.	✓	New routing policy				✓
		- Minimizes average completion time.						
Task Scheduling	Liu <i>et al.</i> [41]	- Provides synchronization of arrival time for each task that constitutes a job.		MJSLP	✓	✓		
Resource Allocation and Sharing	Lin <i>et al.</i> [42]	- Provides efficient use of computing resources. - Confirms the relationships between traffic density and CPU utilization.		TTFP	✓			

its computation load, it may be able to perform the tasks that require high computing power [25].

- 4) **Load Balancing:** Fog servers generally consist of routers, switches, base stations, gateways, and access points. They also have computing skills like encryption and aggregation. But still, they may have limited memory and computation resources to perform applications that require higher performance or permanent storage. Load balancing helps all devices work more efficiently by lightening the burden on a specific server. It aims to avoid being overloaded and improve resource and storage utilization by distributing the workload [26], [27]. The workload on a particular fog server should be rerouted to the available servers in order to prolong the lifetime of each device in the network. For example, calculating GPS positions or transcoding of multimedia may be considered as power-intensive tasks and they must be offloaded to more resourceful fog servers in terms of memory, battery, or computing skills.
- 5) **Task Scheduling:** Task scheduling algorithms determine the tasks of the components in each layer. In other words, after deciding which component should be assigned a task, they specify which component should perform which tasks. According to their task specification, related data should be routed to the corresponding component.
- 6) **Resource Allocation and Sharing:** When performing a specific task due to resource constraints, IoT systems must efficiently perform the task by considering user requirements such as pricing, latency, etc. Resource allocation and sharing algorithms try to achieve the best matching between a pool of tasks and a pool of resources. Besides, for further computational needs, they estimate computing resources to be allocated and define some policy in terms of user characteristics and desired QoS.

Hereinafter, above-mentioned fog-enabled IoT services that need efficient routing mechanisms are surveyed in detail. The survey covers fog-enabled networks, such as wireless sensor networks (WSNs), mobile ad-hoc networks (MANETs), vehicular area networks (VANETs), mobile ad-hoc social networks (MASNs), etc. It should be noted that routing methods have already been proposed in some studies to solve different problems for various fog-enabled IoT platforms. We summarize these existing routing solutions in Table I. However, unlike other studies, we focus on more on the necessity of routing methods in IoT services. Accordingly, Table II summarizes fog-enabled IoT services with respect to routing.

#### A. Mobility

Mobility-assisted services are generally proposed in cooperation with routing algorithms in the literature. Fog computing helps in determining the location of moving devices. Movements of these devices are tried to be optimized with various routing algorithms under different metrics.

Lu *et al.* [28] proposed an improved geographic routing (IGR) protocol for fog-enabled urban-based vehicular networks. Generally, geographic routing protocols assume that each node is aware of its location and delivers its packets according to its location. IGR exploits vehicular fog computing (VFC) for the effective use of communication and computational resources. Also, it determines the best routing path by taking into account the link errors and vehicle intensity. The authors adopt geographic routing for VANET in three modes which are junction selection, improved greedy routing, and repair mode. According to the results, the proposed model outperforms greedy perimeter stateless routing and greedy perimeter coordinator routing (GPCR).

Fang *et al.* [29], first, applied ad-hoc on-demand distance vector (AODV) routing protocol to fog-enabled MANETs, then propose source anonymity-based lightweight secure

TABLE II  
SUMMARY OF FOG-ENABLED IoT SERVICES WITH RESPECT TO ROUTING

Service	Reference	Proposed Protocol(s)	Network Type	Performance Metric(s)	Evaluation Method	Benchmark Protocol(s)
Mobility	Lu <i>et al.</i> [28]	IGR	Urban area vehicular network	Received packet rate End-to-end delay	Simulation	GPSR GPCR
	Fang <i>et al.</i> [29]	SAL-AODV	MANET	Energy consumption Throughput BPUE comparison Security	Simulation	AODV SAODV
	Soo <i>et al.</i> [30]	Adaptive routing protocol based on flooding-based scheme	MASN	Deployment time Transmission time Latency Hand-off time	Simulation, Prototype	-
	Ivanov <i>et al.</i> [31]	GRP	WSN	Delivery ratio Hop count Power consumption Delay	Simulation	RBMulticast SPT
Content Caching and Delivery	Al-Turjman [44]	CCFC	ICSN	Availability Efficiency Trust analysis	Simulation	FC CC LC
	Song <i>et al.</i> [45]	SCC SCC-E	ICN	Number of packets Transmission latency	Analytical	IP network ICN flooding
	Jingtao <i>et al.</i> [33]	Steiner based tree scheme	Fog network	Cost	Simulation	Shortest path scheme
	Hua <i>et al.</i> [49]	SE SE-U SE-B	Fog network	Latency Cache hit ratio	Simulation	Edge caching
	Amin <i>et al.</i> [34]	DCAR DISCS	5G network	Processing overhead Throughput Efficiency in data transfer	Prototype	SCP
	Lee <i>et al.</i> [50]	Online computational caching	Fog network	Latency	Simulation	No computational caching
Computation Offloading	Oueis <i>et al.</i> [52]	EDF-PC EDF-LAT CS-LAT	WSN	Power consumption Latency User satisfaction ratio	Simulation	No clustering Static clustering
	Ye <i>et al.</i> [53]	Genetic algorithm	Bus network	Total cost Dropping rate	Simulation	Non-offloading
	Hassan <i>et al.</i> [54]	Offloading method	Mobile network	Efficiency	Prototype	Cloud computing
	Fricker <i>et al.</i> [55]	Offloading method	Fog network	Blocking rate	Analytical	-
	Sardellitti <i>et al.</i> [56]	Successive convex approximation algorithm	Cellular networks	Energy consumption Latency Power budget	Analytical	Disjoint optimization algorithms
	Mushunuri <i>et al.</i> [57]	Optimization libraries within the ROS	Sensor-actuator network	Battery usage Latency	Simulation	-
	Yu <i>et al.</i> [58]	DSL	MEC network	System cost	Simulation	No offloading Random offloading Total offloading Multi-label linear classifier based offloading
	Teranishi <i>et al.</i> [35]	LASR	Edge network	Latency Overhead	Simulation	Master-slave method TBPS
	Trinh <i>et al.</i> [36]	SPIDER	MANET	Latency Energy efficiency Mobility Failure	Simulation	GEAR AODV HWMP
	Liu <i>et al.</i> [59]	Semi-smooth newton algorithm	Social-aware mobile network	Execution cost Delay	Simulation	SCA LODCO
	Cardellini <i>et al.</i> [60]	Distributed non-cooperative strategy	Mobile network	Efficiency	Analytical	Social optimum
	Liu <i>et al.</i> [61]	Multi-item auction game Cognitive game	Mobile network	Offloading ratio Completion time	Analytical	ADE OTS
	Naranjo <i>et al.</i> [38]	P-SEP	WSN	Energy efficiency Number of alive nodes Transmission rate	Simulation	SEP LEACH-DCHS M-SEP EM-SEP
	Ningning <i>et al.</i> [62]	Graph repartitioning theory	Fog network	Number of migration Run time	Simulation	HDLB
Load Balancing	Puthal <i>et al.</i> [63]	Secured and sustainable load balancing solution	Fog network	Efficiency Scalability	Simulation	Static Random allocation Proportional allocation
	Beraldi <i>et al.</i> [64]	CoolLoad	Edge network	Blocking probability Latency	Analytical	-
	Yu <i>et al.</i> [65]	SLDB	MEC network	Memory efficiency Update speed Data plane throughput	Simulation	Hash table
	Chen <i>et al.</i> [66]	Socially trusted collaborative edge computing platform	Ultra-dense network	Cost Security	Simulation	Non-cooperative scheme Cloud-offloading minimization Centralized collaborative scheme
	Yousefpour <i>et al.</i> [67]	AFP	Fog network	Service delay	Analytical Simulation	LFP NFP
						FCFS Priority task scheduling Multi objective task scheduling
Task Scheduling	Verma <i>et al.</i> [68]	RTES	Fog network	Turn-around time	Simulation	-
	Li <i>et al.</i> [69]	Minimum bandwidth codes Minimum latency codes	Fog network	Communication load Computation latency	Analytical	-
	Deng <i>et al.</i> [70], [71]	Workload allocation formulation	Fog network	Power consumption Latency	Simulation Analytical	Cloud computing
	Habak <i>et al.</i> [72]	FemtoCloud	Edge network	Computational throughput Compute resource utilization Network utilization	Prototype Simulation	PreOb
	Habak <i>et al.</i> [73]	Adaptive workload management mechanism	Edge network	Job completion time Job admission ratio	Prototype Simulation	PTR CPOP
	Wang <i>et al.</i> [74]	Lighweith mobility aware task scheduling scheme	Mobile network	Task execution delay	Simulation	Local execution Op without mobility Op with mobility and max ex. time
	Singh <i>et al.</i> [75]	RT-SANE	Edge network	Success ratio Throughput	Simulation	FCFS Cdc-only scheduling algorithm
	Liu <i>et al.</i> [76]	Optimal task scheduling policy	MEC network	Delay	Simulation	Greedy offloading Local execution Cloud execution
	Liu <i>et al.</i> [41]	MISLP	FC-mDC	Completion time Number of frequency slots	Analytical Simulation	FCFS Scheduling-only Routing-only
						Greedy approach
Resource Sharing and Allocation	Abedin <i>et al.</i> [77]	Refined Irving's matching algorithm	Peer-to-peer IoT network	Utility	Simulation	Greedy approach
	Aazam <i>et al.</i> [78], [79]	Fog-based IoT management model	WSN	Resource estimation Pricing	Simulation	-
	Imharawijitr <i>et al.</i> [80]	Fog model	5G cellular network	Blocking probability	Simulation	-
				Response time Bandwidth utilization Power consumption The amount of data traffic Latency		
	Agarwal <i>et al.</i> [81]	Efficient resource allocation algorithm	Fog network		Simulation	RDLB ORT
	Sun <i>et al.</i> [82]	Crowd-funding algorithm	Fog network	Efficiency SLA violation rate	Simulation	MM MBFD
	Zhang <i>et al.</i> [83]	Stackelberg sub-game Moral hazard model Student project allocation matching sub-game	Fog network	Transmission delay Utility	Simulation	Random matching model Independent payment model Stochastic independent payment model Technologically independent payment model Single bonus payment model
	Plachy <i>et al.</i> [84]	Dynamic resource allocation method	Mobile network	Delay Energy consumption	Simulation	SO Wang's algorithm PSwH
	Amjad <i>et al.</i> [85]	DTMO	Wireless network	Fulfillment percentage	Simulation	Traditional scheme
	Tan <i>et al.</i> [86]	Location-aware prediction algorithm	MEC network	Prediction accuracy Average prediction error	Simulation	Location unaware prediction method
	Lin <i>et al.</i> [42]	GPS TTFP	Mobile network	Latency CPU utilization	Analytical Testbed Simulation	FIFO PS
	Xu <i>et al.</i> [87]	Zenith	Edge network	Utilization Response time Success rate	Simulation	CEC Cloud

AODV protocol to provide different advantages, such as source node anonymity from being traced, secure transmitting with data integrity, and reduced computation and energy consumption. Simulation results show that the proposed protocol has less energy consumption and high bits-per unit of energy (BPUE) ratio when compared to secure

AODV (SAODV) in tradeoff negligible delay. Also, it ensures security more than other AODV protocols by a random delay transmitting mechanism.

In another study, Soo *et al.* [30] proposed a proactive fog service discovery and process migration framework to establish a seamless connection with fog services in MASN. As a mobile user moves, the system continuously monitors and gathers information from its peers. Each MASN node keeps a routing table by listening and forwarding requests of surrounding peers. When a mobile device loses its connection with its fog server, it automatically connects to the available fog server by using the distributed hash table. More specifically, the mobile device sends a request closest MASN peers, they reply to this request with the location of available fog server. Then a connection is established and tasks are migrated between fog servers. The performance of the proposed model is simulated on a simulator, called ONE, where routing and task migration performances are evaluated in terms of transmission time, delivery time, deployment time, and handoff time.

Ivanov *et al.* [31] proposed a gravity routing protocol which priorities user presence for a WSN-based alert system in a fog environment. Each sensor is capable of generating alerts when a specific event happens, and this alert is delivered to users. Users are grouped with respect to their preferences. The authors propose a user-group aware multicast protocol, called gravity gradient routing (GGR), which aims to hand-off alerts spontaneously to the user's handheld device. The GGR protocol is supported by fog computing to avoid extra delay incurred while transmitting data to the cloud. The proposed method is compared with two well-known strategies which are RBMulticast and shortest path tree (SPT). The results show that GGR exhibits higher efficiency due to the use of data fusion, in addition, it is adaptable to various traffic heterogeneities by resulting in the enhanced quality of alert delivery in tradeoff increased price.

## B. Content Delivery and Caching

Content delivery and caching services aim to keep the most valuable contents closer to the end devices to deliver them quickly in order to reduce latency, congestion, and to improve energy efficiency. With the emergence of fog computing technology, fog servers become responsible for caching since they are located closer to the end devices compared to cloud-based systems. However, as the fog network size grows, the delivery of the requested content to the end user may require an efficient routing scheme. In [43], different caching policies for edge caching and content delivery are discussed from various aspects, including policies, strategies, algorithms, file types, impact on system performance, and application areas.

Al-Turjman [44] proposed a cognitive caching approach for the future fog for information centric sensor networks (ICSNs) to assign a value for each content by considering some parameters, such as age, popularity, delay, and fidelity. This value determines whether it should be removed from the cache content of a particular end device. The proposed approach outperforms existing caching techniques which are

node functionality-based caching (FC), content-based caching (CC), and location-based caching (LC) in terms of cache size, data popularity, cache ratio, and network connectivity. Besides, a trust analysis is realized to provide security. In this framework, data is categorized into delay-sensitive data, demand-based data, and age-based data. For example, delay sensitive data should be delivered and cached quickly to the related devices when requested by the user. Performance of the proposed model is evaluated by adopting a use case scenario.

In another solution for caching in information centric networking (ICN), Song *et al.* [45] introduced a smart collaborative caching (SCC) scheme by enabling ICN principles in fog computing. It provides quick and robust connectivity between IoT devices and cooperative caching for different situations, such as resource pooling, the content storing, and node locating. Experiments are conducted with two proposed solutions, including ICN SCC and ICN SCC-E (enhanced version of ICN SCC) and two traditional solutions including IP network and ICN flooding in two different scenarios by changing the number of responses. According to the analytical results, the proposed solutions outperform the traditional solutions in terms of a number of packets and transmission latency.

To highlight the problem of how to share or cache resources among specific fog servers in a fog cluster, Su *et al.* [33] introduced a Steiner tree-based caching scheme. In this scheme, a Steiner tree is employed to find the shortest path with minimum cost in the aim of minimizing cost for caching among fog servers. Instead of storing all resources or communicating with other devices to share its resources, each device in a fog cluster is able to obtain the necessary resources from its related fog cluster. The proposed scheme is compared to the shortest path scheme, and the results show that the proposed scheme outperforms the shortest path scheme in terms of efficiency.

Tandon and Simeone [46] and Sengupta *et al.* [47] presented a cache-aided fog radio access network to minimize content delivery latency by introducing a new metric as normalized delivery time (NDT) via placement and delivery strategies. These schemes focus on what to cache and how to transmit it over the fronthaul and wireless channel. Roushdy *et al.* [48] also proposed the same architecture with partial connectivity by considering NDT. They exploit maximum distance separable codes for placement strategy and real interference alignment for delivery strategy.

Hua *et al.* [49] introduced semi-edge caching (SE) for fog networks to remove the dependence on content or network types as conventional caching strategies. SE is more generic and adaptable to different scenarios. The authors also derive SE as unicast SE (SE-U) and broadcast SE (SE-B) to adapt different fog topologies and scenarios. According to simulation results, SE strategies decrease latency while increase cache hit ratio in the comparison of the edge caching. Also, they observe that SE-U is slightly better than SE-B.

Amin *et al.* [34] proposed first an edge router called dual-mode content aware router (DCAR) to cache content sharing objects by differentiating IP and ICN flows. Then, by leveraging ICN technology, they develop DCAR-based information-centric secure content sharing (DISCS) architecture to provide



secure content sharing for fog services. Security features of this architecture are analyzed in terms of authentication and privacy. Also, evaluation results show that the DCAR has a better performance when compared with secure copy protocol (SCP) in terms of processing overhead, throughput, and efficiency in data transfer.

Since fog nodes (FNs) are not able to predict future operations properly due to limited memory size, determining which intermediate computation result should be stored and reused become a challenging issue in fog networks. To address this issue, Lee *et al.* [50] presented an online computational caching algorithm to minimize transmission and computation latency. The online computational caching algorithm helps FNs to select proper intermediate computation results for storing and reusing in the aim of decreasing latency. Based on the simulation results, the proposed algorithm has reduced total latency when compared with the no computational caching method.

### C. Computation Offloading

Offloading mechanisms usually target mobile applications to help them execute their tasks since mobile devices are resource constrained and they do not have enough capacity to execute their tasks. How to send offloaded tasks to more powerful fog servers is a routing issue and requires efficient routing methods. Wei *et al.* [51] proposed a new computation offloading structure called MVR for MEC platforms. The proposed technique is basically a mobile federation that offers fine-grained offloading execution for edge cloud platform by taking into account the questions of why to offload, what to offload, when to offload, and how to offload.

Oueis *et al.* [52] address the small cell cloud (SCC) architecture where a small cell acts as a fog server between cloud and mobile user. They aim to form fog clusters, where each cluster represents a group of small cells which shares resources to reduce power consumption. In this regard, the authors formulate the clustering problem for multiple users as an optimization problem by forming clusters and allocating computational and communicational resources under the latency constraint of each user. To this aim, a use case scenario is modeled on the 3GPP framework. Different variants of proposed customizable algorithms (EDF-PC, EDF-LAT, and CS-LAT) are created by changing metrics, scheduling rules, and optimization objectives. Then, the authors compare their strategies with static and no clustering strategies. Their overall results indicate that their proposal outperforms in terms of user satisfaction rate. Also, the joint distribution of computation and radio resources minimize the latency and power consumption of each mobile user.

By considering the limited capacity of the cloudlets, Ye *et al.* [53] proposed a scalable fog computing concept with service offloading (SO) in bus networks. Because cloudlets are insufficient in order to perform the mobile user's intensive tasks which cause an undesired delay, roadside cloudlets should offload their computation tasks to the fog server located on the bus in motion and, thus, computing capability of

cloudlets is improved to achieve those tasks. Also, an allocation strategy using a genetic algorithm is employed in order to reduce cost and increase user satisfaction. Performance of the proposed strategy is simulated by using traffic data set about city buses and compared with a nonoffloading strategy. Also, the total cost is measured for a different number of users. The results show that the total cost is reduced considerably with high user satisfaction.

Another study applied by Hassan *et al.* [54] aims to show that fog computing is more effective than cloud computing in terms of application offloading and storage expansion for mobile applications. In accordance with this purpose, they formulate these two services for resource-constrained mobile devices. These problems are formulated as constrained graph partitioning problem. In this graph, functions are modeled as nodes, while interactions among these functions are modeled as edges. Implemented prototypes show that fog computing is more effective and mostly better than cloud computing in terms of computation offloading and storage due to utilizing nearby resources.

Fricker *et al.* [55] proposed an offloading mechanism among data centers located at the network edge. According to offloading probability, with a certain amount of blocked request at a small data center are forwarded to a big neighboring data center. To evaluate system performance, a numerical analysis is performed. The numerical results show that the proposed mechanism improves the loss rate and reduces the request blocking rate at the small data center significantly.

The offloading problem is reformed as a joint optimization problem of the radio resources for multicell mobile-edge computing by Sardellitti *et al.* [56] to minimize energy consumption in tradeoff latency and power budget. The problem is formulated in a single-user case where the global solution in closed form is obtained, and a multiuser case where successive convex approximation (SCA) technique is adopted to converge local optimal solutions. According to the numerical results, the proposed algorithm surpasses disjoint optimization schemes.

Mushunuri *et al.* [57] exploited the robot operating system (ROS) for robotic sensor-actuator devices to distribute intensive tasks among edge devices or smart gateways to lighten the load on a single device with the aim of alleviating latency and battery consumption. The authors integrate optimization libraries with the ROS to be deployed in a wide range of application areas, such as physical robots, single board computers, and smartphones.

Deep learning techniques can be employed as an offloading approach as well. Yu *et al.* [58] presented a dynamic offloading framework named deep supervised learning (DSL) by leveraging a deep approach. Their primary aim is to minimize offloading cost in MEC networks. To this aim, they first adapt offloading problem into a multilabel classification problem. Then, they decrease the total offloading cost considerably compared with no offloading case and random offloading, total offloading, and a multilabel linear classifier-based offloading.

Teranishi *et al.* [35] extended the distributed topic-based pub/sub (TBPS) approach by adding the index notion to allow flexible allocations of data flows. Also, they propose

locality-aware stream routing (LSAR) that employs extended TBPS by adapting to changes on data streams dynamically in edge computing. Simulation results show that LSAR is able to alter the data flow destination quickly compared to a master-slave method and TBPS in a tradeoff small overhead.

Facial recognition applications in MANETs are studied by Trinh *et al.* [36]. To analyze the tradeoffs between computing policy and their impact on energy consumption of visual data over low latency, they propose offload decision-making algorithm. They also present sustainable policy-based intelligence-driven edge routing (SPIDER) to allow flexible policy specifications. The proposed algorithm is evaluated in the context of a disaster incident response scenario. It leverages satellite images to learn the geo-information about existing physical obstacles. The results show that the proposed algorithm ensures flexibility to offloading visual data processing job to an edge cloud or core cloud. In addition, compared to other well-known existing solutions such as geographic and energy aware routing or AODV, the proposed routing algorithm is shown to be more sustainable.

There are also several game-theory-based solutions for the computation offloading problem in fog computing related networks. Liu *et al.* [59] proposed socially aware dynamic computation offloading algorithm, named semi smooth Newton algorithm, to solve generalized Nash equilibrium problem (GNEP) in a fog computing architecture by taking advantage of game theory approaches and energy harvesting techniques to model the relationships in social groups and to minimize execution cost. Also, different queue models are performed during the offloading process to analyze delay performance in detail. Comparison results with energy efficient computing framework and Lyapunov optimization-based dynamic computation offloading (LODCO) algorithm show that the proposed algorithm has better performance than others in terms of execution cost.

GNEP is also applied in the work of Cardellini *et al.* [60]. The authors reformulate GNEP to examine user interactions by presenting a game theory approach. Unlike single user-scenarios, the authors present a decentralized solution involving a noncooperative usage scenario where selfish nodes are able to prefer their computation offloading resources from either a close range resource-constrained cloudlet or a long distance resourceful cloud server. Comparative results supported by numerical experiments show the efficiency of the proposal with respect to social optimum approach.

Liu *et al.* [61] proposed two decentralized data offloading games which are multi-item auction game mechanism to maximize the revenue of mobile subscribers and congestion game to minimize the payment of these mobile subscribers. Overall numerical results show that the proposed mechanisms improve the effectiveness of the system performance by comparing to always delayed offloading (ADE) and on-the-spot offloading (OTS) schemes in terms of offloading ratio and completion time to receive entire data from the cloud.

Besides the above studies, Mach and Becvar [21] performed a comprehensive survey of offloading schemes for MEC. They investigate offloading schemes in the literature in the perspective of offloading type, objective, and the number of the

user equipment. They also consider resource allocation and mobility management in the MEC.

#### D. Load Balancing

Load balancing services balance the workload among network components. Routing algorithms help load balancing methods distribute overload among other available members of the network. Multipath routing is important to perform load balancing by selecting less busy channels as the next-hop when the network is under heavy traffic.

Naranjo *et al.* [38] proposed a modified stable election protocol (SEP), called prolong SEP (P-SEP), based on cluster head (CH) selection to prolong network life by balancing energy consumption in WSNs. In each round of cycle, a random number of CH is selected by aiming at reduced energy consumption. This network includes two types of nodes which are normal and advanced nodes, and they have no priority over each other to become a CH. In addition, P-SEP utilizes energy heterogeneity, optimizes the minimum distance between CHs and fog servers, and nominates some CHs to transmit aggregated data to the fog server. According to the performance evaluation, P-SEP has a better network lifetime in comparison with SEP, low-energy adaptive clustering hierarchy with deterministic CH selection (LEACH-DCHS), modified SEP (M-SEP) and an efficient M-SEP (EM-SEP) under various initial energy and heterogeneity parameters.

Ningning *et al.* [62] proposed a dynamic load balancing mechanism based on graph partitioning theory. They adopt fog computing paradigm into cloud atomization technology that maps physical resources into virtual resources. They measure the average number of moving nodes along with the number of joining or quitting nodes. Also, they compare operating times along with the increased number of tasks. Performance of the proposed method is evaluated by comparing with classical hybrid load balancing scheme under a time constraint. Simulation results show that the proposed mechanism has a lower migration cost, besides its flexible nature.

Authentication may be critical if the deployment is done in an unattended environment. To provide secure load balancing, Puthal *et al.* [63] present a novel approach which considers both authentication and load balancing process together for edge data centers (EDCs). In this approach, first, an adaptive EDC authentication mechanism is used to authenticate all system components beginning from the cloud data center. Then, a sustainable load balancing mechanism is employed to find the less loaded EDC. The proposed solution is compared with other allocation techniques, including static allocation, random allocation, and proportional allocation. According to the numerical and experimental analysis, the proposed method shows higher performance results than others in terms of efficiency and scalability.

A cooperative load balancing scheme between two data centers called CooLoad is proposed by Beraldi *et al.* [64]. The proposed technique is responsible for transferring overloaded data from one data center to another to lighten the load of data centers. Evaluation results clearly show that the proposed



scheme improves QoS by reducing the blocking probability at each data center and execution delay of the tasks.

Yu *et al.* [65] introduced a new load balancer, named scalable and dynamic load balancer (SDLB), for MEC and fog computing. The proposed scheme employs parallel Othello group (POG) data structure that is based on minimal perfect hashing. The proposed scheme has a dynamic structure and uses the SDN paradigm. The SDLB exhibits a higher update speed and a lower memory usage when compared with commonly used hash table-based load balancer design for the cloud. The superiority of the proposed protocol comes from the fact that the new data structure POG supports networks dynamics.

A socially trusted collaborative edge computing platform where small cell base stations (SBSs) are willing to join in edge computing process by an intensive payment model is proposed by Chen and Xu [66]. In this paper, a coalitional game theory-based SBS coalition formation algorithm is also used to decrease the number of computation tasks offloaded to the cloud. Lastly, to minimize the security risks, a trust management model is applied to ensure that there is a secure collaboration between the two SBSs. The performance improvements of the proposed framework are evaluated extensively by comparing it with the noncooperative scheme, cloud-offloading minimization, and centralized collaborative scheme. The study proves that SBSs can be incentivized to form a coalition which also supports cooperation by following the payment-based incentive mechanism. In this respect, the authors present a new perspective for socially trusted cooperation among densely deployed edge devices by offering a trust management model for the security risks of this cooperation.

A cooperative delay-minimization task offloading policy to reduce service delay by load sharing among FNs is developed by Yousefpour *et al.* [67]. The proposed policy considers both queue length and different request types. The authors call their proposed policy as all fog processing (AFP) and they compared it with the other two methods, namely light fog processing (LFP) and no fog processing (NFP). The feasibility of the proposed method is confirmed in terms of reducing the service delay. The superiority of this policy is its applicability to any architecture and its dynamic structure which is not limited to a number of nodes, type of nodes, or topology.

### E. Task Scheduling

Task scheduling algorithms determine which component should realize which tasks. After, deciding the responsibilities of each component in the network, generated data should be delivered to the corresponding component quickly, securely, or without congestion guaranteed by routing algorithms. Therefore, routing constraints need to be considered when performing task scheduling.

Verma *et al.* [68] proposed a real-time efficient task scheduling algorithm (RTES) for load balancing in fog-enabled architectures. The authors present three-layered architecture and aim to balance the load between the client and the fog layer to satisfy client's requirements with minimum latency. Then, if clients do not have demanded resources, their requests

are forwarded to the cloud. The authors compare their algorithm with existing algorithms such as first come first serve (FCFS) and priority tasks scheduling and multiobjective tasks scheduling algorithms. The performance results show that their proposed algorithm has minimum execution time. In other words, it provides fast response time to its clients.

Li *et al.* [69] discussed two coding schemes, minimum bandwidth codes and minimum latency codes, that enable redistribution of tasks and injection of redundant tasks to a tradeoff between communication load and computation latency. The authors aim to demonstrate the impact of these two coding schemes on fog networks. They indicate that coding schemes are able to reduce bandwidth usage and latency by leveraging abundant computing resources. The authors also elaborate on a tradeoff between computation latency and communication load.

To assess the tradeoff between power consumption and delay, Deng *et al.* [70], [71] defined the workload allocation problem. The authors formalize this problem mathematically and solve it by decomposing into three subproblems. Extensive simulations show that fog computing reduces communication latency by collaborating with cloud computing. This paper confirms that the fog is able to complement cloud with decreased communication latency.

In another study, Habak *et al.* [72] proposed a femtocloud system architecture which enables mobile devices to be organized in a cloud computing service at the edge. To assess the feasibility of the proposed system, including computational throughput, compute resource utilization, and network utilization, they first implement a femtocloud client service to be installed on each mobile device so that each user are able to create a user profile to connect a cloudlet or a controller through WiFi network. Therefore, intensive tasks are assigned to the controller by leveraging cloudlets' computational capacity. The authors also present the task scheduling problem as a part of designing such a system and provide a heuristic solution based on an iterative greedy approach to this problem by aiming to maximize: 1) the efficiency of utilizing the communication channel; 2) the effective computation; and 3) the amount of tasks executed by a cluster. The results show that the proposed architecture decreases the computational load from the central cloud compared to the presence time oblivious scheduler (PreOb). Habak *et al.* [73], also, extended their previous work to satisfy the requirements for workload management functions in FemtoClouds. The authors also propose adaptive workload management mechanisms for resource management and churn masking. To this aim, they implement a prototype to evaluate the performance and simulate to assess the efficiency of the system. The proposed mechanisms show higher performance when compared with two traditional task assignment mechanisms, namely per-task risk-controlled assignment (PTR) and critical path on a processor (CPOP) in terms of job completion time and job admission ratio.

Wang *et al.* [74] proposed a lightweight mobility-aware task scheduling scheme by aiming to increase resource utilization and reduce execution delay. Then, a delay estimation scheme is utilized to make accurate offloading decisions on mobile

devices. While performing evaluations, four different cases of mobility and execution time scenarios with or without an optimized assignment. Overall simulation results show how the proposed scheme decreases notably the execution time of tasks with respect to the others.

In another important work, Singh *et al.* [75] proposed real-time security aware scheduling on the network edge (RT-SANE) algorithm that enables the integration of micro data centers (MDC) and CDCs to reduce latency and improve security/privacy. Basically, the proposed algorithm decides jobs (private, semi-private, or public) should be executed in either MDC or CDC according to MIPS load, deadline factor, and delay factor. According to the results, success ratio and throughput results of the proposed algorithm is shown to be better than both the FCFS and the CDC-only task scheduling algorithms.

Liu *et al.* [76] proposed an optimal task scheduling policy to minimize the execution delay caused by two main delays: 1) the delay to decide a task to execute locally or in an MEC server and 2) the delay for transmission. The authors first analyze the average delay for each task and average power consumptions at each device. After evaluating the analysis results, they develop a 1-D search algorithm to find an optimal task scheduling policy. The proposed optimal task scheduling policy shows reduced execution delay compared to conventional scheduling techniques, such as greedy offloading, local execution, and cloud execution.

A joint optimization problem multiple jobs scheduling and lightpath provisioning (MJSLP) is formulated by Liu *et al.* [41]. The proposed algorithm aims to minimize average completion time for fog computing micro data centers (FC-mDC). This algorithm consists of two subalgorithms which minimize completion time and reduce frequency slots by providing synchronization of arrival time for each task that constitutes a job. The extensive simulation results show that MJSLP algorithm provides superiority in terms of average completion time and the number of frequency slots compared to FCFS, scheduling-only, and routing-only methods.

### F. Resource Allocation and Sharing

Resource allocation and sharing services allocate network resources by taking into consideration of user requirements. After deployment and sharing resources among network components, computation tasks and contents should be transmitted to the related components. Routing algorithms help this transmission in terms of different performance metrics. Thus, similar to task scheduling, routing constraints need to be considered when performing resource allocation and sharing among components.

Based on Irving's matching game, Abedin *et al.* [77] proposed an efficient pairing scheme. The Irving's matching game enables to compute resource sharing among resource-constrained mobile devices inside the same fog domain. By refining Irving's algorithm for IoT nodes, the proposed system provides better utility in terms of communication efficiency between peer devices in the same fog domain when compared to the greedy algorithm.

In the first study of Aazam and Huh [78], a resource estimation and allocation model for existing and new customers by accounting for different criteria such as device type, customer type, service type, and price, fluctuating relinquish probability of the customer, and variance of relinquish probability is proposed. Then, the authors extend their previous work by including price estimation for new customers [79]. In both studies, each fog server acts as a microdata center to provide both effective resource management and user satisfaction. Each IoT device is categorized according to its type and mobility. Also, customers' traits and characteristics are considered during resource estimation, and pricing to make the proposed model more flexible and scalable. Overall performance is evaluated in terms of resource estimation for existing and new customers, and pricing for new customers. According to the results, the proposed method is effective in determining the number of required resources by avoiding resource waste.

To minimize blocking probability for 5G cellular networks, Intharawijitr *et al.* [80] presented a fog server selection system for resource delivery. The authors aim to examine blocking probability under three policies: 1) random policy where a fog server is randomly selected; 2) lowest latency policy where a fog server which has lowest total latency is selected; and 3) maximum capacity policy where a fog server which has maximum remaining resource is selected to execute the workload. Simulation results show that the lowest latency policy provides better performance in accordance with accessing resources quickly.

Agarwal *et al.* [81] proposed an efficient resource allocation architecture by distributing the workload among cloud and fog servers. A fog server manager located in the fog layer checks the available resources depending on the user's request. After determining available resources, the manager can execute all tasks or part of them, and postpone the execution of others. Simulation results show that the proposed algorithm shows better performance when it is compared to reconfiguring dynamically with load balancing (RDLB) and optimize response time policy (ORT) in terms of response time, request and processing time, loading time, and total cost.

Sun and Zhang [82] proposed a resource sharing method that utilizes an incentive mechanism based on crowd-funding and repeated game. This scheme supports resource owners to spare resources, while monitoring resource supports to execute tasks positively. The proposed scheme provides better working efficiency and reduced SLA violation rate when compared to the min-min (MM) algorithm.

Game theory may offer a good solution for load balancing as well other fog-enabled IoT services. Zhang *et al.* [83], [88] aimed to adapt two-layered resource allocation problem between data service operator (DSO) and authorized data service subscribers (ADSSs) into the three-layered problem by taking into account FNs between DSOs and ADSSs. To this aim, they first apply three subgames which are Steckelberg game, moral hazard game, and student project allocation matching game to manage the interactions between DSO-ADSS, DSO-FN, and FN-ADSS, respectively. Then, they analyze these relations from different aspects to achieve optimal strategy with stable and optimal payoffs. More specifically, the

authors propose SPA-(S,P) algorithm to find optimal matching between FNs and ADSSs and compare this algorithm with random matching strategy. The results show that the proposed method exhibits low transmission delay and high resource utility. Besides, the authors propose a payment model with a higher utility than independent, stochastic independent, technologically independent, and single bonus payment models.

Plachy *et al.* [84] proposed a collaborative resource allocation algorithm for dynamic virtual machine (VM) migration and the communication path selection to find the optimal path between mobile users and VMs that are allocated as computational resources in the cloud. To handle the mobility of users, they first predict suitable routes of the user's movement. Then, a VM is allocated dynamically and the optimal communication path is selected by leveraging these predictions. According to the simulation results, the proposed algorithm reduces offloading delay more than others and preserve consumed energy at the same level compared to other well-known approaches, such as Wang's online approximation algorithm, path selection with handover (PSwH), and SO algorithm.

Amjad *et al.* [85] presented an edge computing-based dynamic resource allocation system for IoT. They offer dynamic tracking, monitoring, and orchestration (DTMO) of cloud resources to the end-users in order to provide an efficient decision-making process in resource allocation. The authors also consider computation offloading to process capacity shortage and reduce latency. Their proposed framework is examined under four different scenarios, and its feasibility is evaluated by comparing it with the traditional scheme. According to the results, the proposed framework achieves a higher percentage of fulfillment in resource allocation requests.

For EDCs, Tan *et al.* [86] presented a vector auto regression-based location-aware load prediction algorithm. During the load prediction phase, the proposed algorithm uses information from its neighbor EDCs besides its historical data. They simulate the proposed scheme by using real-world mobility traces obtained in San Francisco. The simulation results show that the proposed method outperforms unaware load prediction methods in terms of prediction accuracy and prediction error. In other words, the proposed algorithm is able to provide more efficient resource allocation and cost saving. The authors also examine the impact of a number of neighbors and practicability of its algorithm on real-world load prediction.

Lin *et al.* [42] adopted the generalized-processor-sharing model (GPS) in C-RAN architectures with the help of an MEC system to analyze the processor queuing behavior. The authors demonstrate the superiority of the GPS over first-in-first-out (FIFO) and processor sharing (PS) models. According to comparison results, the authors also propose two-thresholds forwarding policy (TTFP) to arrange data traffic by taking into account the current traffic state. Simulation results show that the TTFP requires low waiting time even in high traffic volume proving that employing near real-time traffic data reduces latency in MEC.

Zenith, a new resource allocation method in edge computing networks, is proposed by Xu *et al.* [87]. This scheme decouples the management of edge computing infrastructure

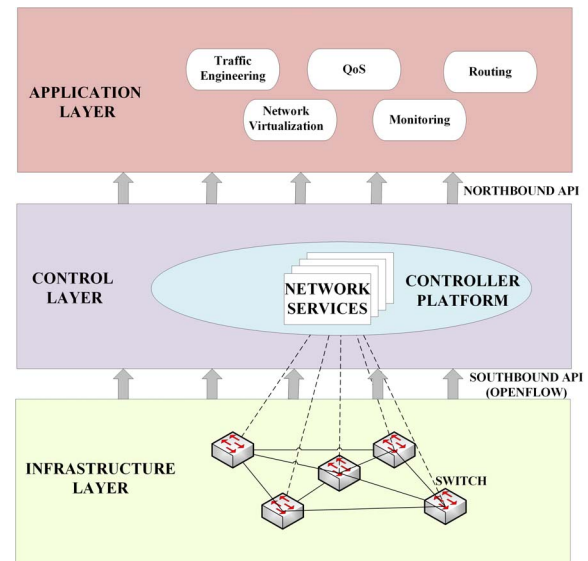


Fig. 2. Generic SDN framework.

by allowing edge service providers to establish resource sharing contract with edge infrastructure providers in the aim of truthfulness and utility-maximization. They also develop a latency aware task scheduling mechanism that allows proper allocation of resources based on contracts. The proposed method is compared with coupled edge computing (CEC) and cloud-based solutions to show its effectiveness in terms of utilization, response time and success rate.

#### IV. SDN

SDN is a rapidly emerging new networking paradigm and its most prominent feature is the decoupling of network control logic (control plane) from underlying routers and switches (data plane) which is responsible for forwarding data properly. This separation makes it possible to transmit the data by simple switches and control these devices with software applications through standardized interfaces. In SDN, forwarding decisions are made according to flow tables. The transmission process in the network is managed by the match criteria in the flow table and the actions corresponding to these criteria [89].

Compared to traditional infrastructures, SDN provides centralized control and management of the entire network from a global point of view. Therefore, it adds programmability, flexibility, and scalability to the network in some scenarios, such as decision-making, forwarding packets, and managing network configurations [90]. In addition, SDN is recently used for maintaining session continuity and connectivity and ensuring mobility management [91], [92].

As shown in Fig. 2, SDN has a three-tiered structure which consists of an infrastructure layer, a control layer, and an application layer. The infrastructure layer (data plane) is located at the bottom layer and consists of forwarding elements. According to flow tables that are based on predefined instruction sets, forwarding elements such as switches or routers that reside in this layer take an action on the packet they received.

TABLE III  
SDN-BASED FOG-ENABLED IOT SERVICES

Service	Reference	Proposed Protocol(s)	Network Type	Performance Metric(s)	Evaluation Method	Benchmark Protocol(s)
Mobility	Gupta <i>et al.</i> [32]	QoS aware routing policy (SDFog)	Health smart home	QoS Structural similarity index	Simulation	Best effort routing policy
	Sun <i>et al.</i> [97]	EdgeIoT, novel VM migration scheme	Cellular network	Traffic load	Analytical	-
Content Delivery and Caching	Yang <i>et al.</i> [98]	SDN-enable fog-cloud interoperation	5G Wireless network	Latency Fog processed requests	Analytical	Popularity unaware caching
Computation Offloading	Chen <i>et al.</i> [99]	Efficiency software defined task offloading scheme	Ultra-dense network	Task duration Energy cost	Simulation	Random offloading scheme Uniform offloading scheme
	Huang <i>et al.</i> [37]	LT-NSR LT-PR	VANET	Throughput	Simulation	GD-NSR
Load Balancing	Hakiri <i>et al.</i> [39]	Hybrid method based on OLSR and OpenFlow	WMN	Latency Throughput Bandwidth utilization	Simulation	Traditional OpenFlow routing
	Betzler <i>et al.</i> [40]	OLSR	WMN	Throughput Channel load CPU load	Testbed	-
	He <i>et al.</i> [100]	MPSO-CO	Vehicular network	Latency	Simulation Analytical	LBMM, PSO-CO Greedy-LB
Task Scheduling	Zeng <i>et al.</i> [101]	Joint task scheduling (Three-stage heuristic algorithm)	Fog network	Transmission time I/O time Computation time	Simulation	Server greedy task scheduling Client greedy task scheduling
Resource Allocation and Sharing	He <i>et al.</i> [102]	A big data deep reinforcement learning approach	Smart city	Total utility	Simulation	Static scheme

These actions can be forwarded to a specific port or controller, dropping, or rewriting some header. Here, instructions are defined by southbound interfaces that allow interaction and communication between control plane and data plane. The most common southbound interface is OF [93]. It is a protocol that installs flow table entries in switches through the OF controller by allowing them to forward traffic according to these entry rules. Also, ForCES [94] and SoftRouter [95] are other standardized southbound interfaces which separate control plane and data plane as well. All forwarding elements are controlled and managed by one or more controller located at the control layer (control plane). While they can communicate with the data plane through southbound interfaces, they also can communicate with the application layer (management plane), located at the upper layer through northbound interfaces. Application layer adds new features to the network, such as programmability, security, and assisting network configuration. With northbound interfaces, applications running on application layer do not have to know all detail about data plane like network topology and thus, it facilitates network management [96].

#### A. SDN-Based Fog-Enabled IoT Services

SDN can be integrated into fog computing in order to increase network performance. In one of the earliest works [103], an architecture combining fog computing and SDN concepts (FSDN) is proposed for VANETs. In this paper, the SDN controller is located in the middle of fog and cloud layers and it is responsible for fog orchestration and resource management. The benefits of this hybrid system are illustrated by presenting two use case scenarios: 1) data streaming as nonsafety service and 2) lane-change assistance as a safety service. Table III comparatively summarizes the other studies in which FSDN concepts are used together.

In the first study, Gupta *et al.* [32] proposed an SDN-based fog computing system called SDFog. The proposed system performs QoS-aware orchestration by managing and controlling flows between services. By taking advantages of SDN and network function virtualization technologies, the authors aim to show the effectiveness of the proposed system over a health smart home use case scenario. The proposed QoS-aware

routing policy is compared to the best effort routing by measuring the quality of video delivery. According to the obtained results, QoS-aware approach exhibits a steady average quality and better rate of data delivery, while the best effort routing exhibits low quality due to increased network congestion.

Sun and Ansari [97] introduced EdgeIoT which is a hierarchical structure of fog computing and cloud to handle the data effectively produced by IoT devices at the edges. In this structure, the SDN-based cellular core is located at top of fog servers and it is responsible for data forwarding among fog servers. The cooperation of FSDN provides efficient collection, classifying, and analyzing IoT data streams. The authors also propose a novel proxy VM migration scheme to minimize the traffic in the network. They consider user mobility with future movement predictions to optimize the migration process.

SDN is also employed in the study of Yang *et al.* [98] to make fog-cloud interoperation feasible and improved QoS. To this aim, they analyze the proposed SDN-based architecture through two case scenarios which are crowdsourcing task scheduling and popularity-aware content caching. According to the case scenarios, this architecture improves QoS by reducing backhaul pressure and providing agile network management.

Chen and Hao [99] considered the task offloading problem in software-defined ultradense networks. With the integration of SDN, a macro cell base station has global information about mobile devices, and it can decide whether a device should perform a task locally or offload to the edge cloud. The authors adapt the task allocation problem into a mixed integer nonlinear problem and then solve it by decomposing it into two subproblems which are task-placement and resource allocation. According to the solutions, the authors present an efficient software defined task offloading scheme in comparison with random and uniform offloading schemes. Overall results show that the proposed algorithm decreases the task duration and energy cost.

A centralized offloading control scheme for VANETs is developed by Huang *et al.* [37] by leveraging SDN. In this paper, using SDN technology inside the MEC platform, each vehicle in the network informs about its context. Then SDN controller searches for possible paths between two vehicles as

a vehicle-to-vehicle VANET offloading path. If there is more than one path, the controller chooses the path with the longest connection lifetime. Life-time-based network state routing (LT-NSR) algorithm is employed to find the path which has the longest lifetime. Also, in the case of disconnection of a path, life-time-based path recovery (LT-PR) algorithm is responsible for recovering this path. The performance results show that the proposed offloading scheme has better throughput compared to conventional greedy routing (GD-NSR) scheme for middle vehicle-density situation. However, the proposed algorithms do not show any superiority for high or low-density situations.

Hakiri *et al.* [39] introduced fog computing for wireless mesh networks (WMNs) and define fog mobile networks. In addition, SDN is proposed to give the network a central point of view during managing and monitoring network components so as to eliminate the partial visibility due to distributed routing protocols used by WMNs. However, according to the authors, SDN OF is incapable of ensuring the functionalities of wireless fog networks. Therefore, they propose a hybrid architecture that combines optimal link state routing (OLSR) protocol for data forwarding at the bottom layer by taking advantages of existing protocols and OF for global and optimal path selection at the upper sublayer. The performance analyses show that the proposed hybrid routing protocols ensure global and optimal path selection, efficient load balancing for offloading, and best path selection. Also, the results indicate that the proposed approach has lower latency, lower bandwidth utilization, and high throughput.

In another study, Betzler *et al.* [40] proposed two path forwarding policies by taking advantages of SDN features for WMNs. These policies consider the network state and available computing resources. A testbed is used to evaluate the performance of the proposed policies over three experimental scenarios. The performance results show that the proposed policies ensure flow balancing to avoid network congestion and maintain CPU-aware routing which aims to perform computing tasks rather than routing tasks by redirecting the flows.

To alleviate the burden of the cloud in the concept of Internet of Vehicles, He *et al.* [100] applied FSDN technologies in software defined cloud/fog networking architecture. In addition, they propose an SDN-based modified constrained optimization particle swarm optimization (MPSO-CO) algorithm to provide an efficient load balancing mechanism so that latency-sensitive tasks can be processed efficiently. The proposed model is evaluated theoretically and experimentally by comparing it with constrained optimization particle swarm optimization (PSO-CO), max-min load balancing algorithm (LBMM), and greedy load balancing algorithm (Greedy-LB). The results clearly show that the proposed algorithm exhibits better performance by providing decreased latency and enhanced QoS.

A load balancing scheme with fog computing supported SDN embedded system for effective use of network resources is presented by Zeng *et al.* [101]. They aim to design an effective task scheduling and resource management system to satisfy user experiences. The problem is formulated as mixed-integer nonlinear programming problem and evaluated

extensively in three-stage task completion time under transmission time, I/O time, and computation time constraints. The simulation results show the efficiency of the proposed method in minimizing task completion time.

He *et al.* [102] reformulated resource allocation problem as a joint optimization problem and solve it by deep reinforcement learning approach to improve the effectiveness of smart city applications. Convergence performance is evaluated under different scenarios and different learning rates by comparing the static scheme.

## V. PROPOSED HIERARCHICAL SDN-BASED ROUTING SCHEME

As the network size increases, in the management of forwarding elements by a single controller, several difficulties may arise in terms of interaction between the controller and forwarding devices, response time, scalability, infrastructure support, and availability [104]. For this reason, large-scale networks such as fog-enabled networks can be partitioned into multiple controller domains and distributed computing can be ensured locally by multiple SDN controllers [105]. The hierarchical SDN architecture contains a set of controllers to achieve a certain level of performance and scalability. Each controller in the network can be physically centralized or physically distributed. In the physically centralized architecture, data plane is controlled and managed by a central controller. However, the tendency among new studies shifts to the physically distributed architecture due to concerns, such as single point of failure and scalability concerns of physically centralized architecture. Physically distributed SDN architectures can also be categorized as logically centralized or logically distributed. The logically centralized architecture is suggested especially to be free of the single point of failure. It contains multiple controllers, but, they are all replicated, and act as a single controller. They have the same management responsibilities and they always share information with each other to be aware of other's changes. On the other hand, logically distributed SDN architecture has multiple controllers which have different responsibilities and views of the network [106].

In this paper, due to the separate needs for local and global control of network view separately, we propose a physically and logically distributed SDN architecture for fog-enabled IoT platforms to provide an efficient routing. Abstract view of the proposed architecture is presented in Fig. 3. In this architecture, there are three tiers with different responsibilities. Each tier's components and duties are explained below in detail.

- 1) *IoT Tier*: IoT tier consists of different types of end devices which are connected to each other. They generate heterogeneous and ubiquitous IoT data with their sensing capability of events in a neighborhood area. Besides, some end devices are capable of relaying the neighbor data. These devices (relay nodes) have no power limitations to transmit the received data. Due to the limited battery or memory constraints, however, some devices are not able to relay data. It should be noted that only relay nodes act as a forwarding element and the routing process is realized over them.

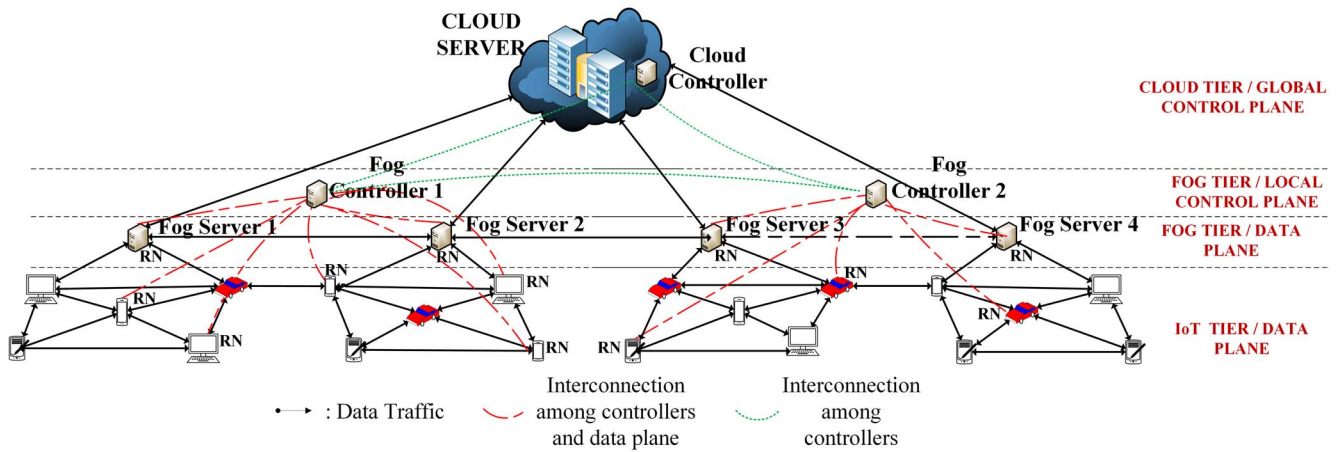


Fig. 3. Hierarchical SDN-based fog computing architecture (RN: relay node).

- 2) *Fog Tier*: Fog tier includes fog servers and fog controllers (some of the fog servers are assigned as a fog controller). Fog servers are responsible for collecting and processing IoT data with less latency and storing that data temporarily. They forward their processed data to the cloud tier for further processing and permanent storage. Fog servers and end devices are OF-enabled and controlled by fog controllers. Fog controllers are also located at the fog tier with different responsibilities. They manage and control fog servers and end devices connected to them. Hence, they have a local view of the entire network. With a local managing mechanism, routing decisions can be applied to the forwarding devices faster which result in low latency.
- 3) *Cloud Tier*: Cloud tier includes a cloud server and a cloud controller. The cloud server is responsible for collecting processed data from fog servers. As it is a resource-rich server, it is able to process the data with high performance and storage capability. Also, the cloud controller ensures global controlling and managing of the entire network. It is connected to fog controllers. Since fog controllers cannot have the full visibility of the entire network, they ask the cloud controller about rare events to make a global decision. For frequent local events, they do not have to connect with the upper layer, they can make their local decisions.

As shown in Fig. 3, our proposed model assumes that not all end devices are directly connected to the fog servers. For example, in the case of a smart home scenario, the smart home may have several IoT devices in which one of them is the smart meter. All IoT devices in such home communicate with their respective fog server over the smart meter. In some cases, end devices may need to send their data to neighboring devices so that they can forward that data to the fog server which may be few hops away. Similarly, due to battery power and wireless connection limitations, an IoT device may need to send its data to its regarding smart meter over multihop routes. Moreover, an IoT device may have to communicate with a neighboring fog server due to a congested network condition or security requirements. In all cases, reaching from an IoT device to a fog

server requires a routing scheme. Considering the properties of fog-enabled IoT services, in our proposed model, we assume that the solution of this routing scheme may face the following problems.

- 1) *Finding the Shortest Path*: This constraint is faced when fog or cloud services need to serve their end users with minimum latency. For example, in health monitoring or surveillance applications due to the timeliness of the collected data shortest path routing is required.
- 2) *Ensuring the Required Bandwidth Along the Path*: From source to destination, a path that satisfies the bandwidth needs of the data flow traffic must be ensured. For example, the amount of data traffic in the fog-to-fog communication is generally higher than in fog-to-end device communication. If the amount of data traffic on the link that connects a pair of fog servers reaches to the link capacity, a routing scheme should guarantee the required bandwidth. Even though the established path is the shortest path, another path that ensures the amount of traffic does not reach to link capacity over the path must be formed.
- 3) *Guaranteeing the Security Along the Path*: Not all fog servers or IoT devices may be trustworthy. Due to application dependent requirements of certain data (such as health etc.) may need to be sent over trusted nodes [107].

Our proposed hierarchically SDN architecture is able to mitigate these routing issues in fog computing. Fig. 4 presents a detailed view of the proposed architecture. There are fog controllers at the fog tier and a cloud controller at the cloud tier. For frequent events, fog controllers have the capability of taking actions locally without asking to the cloud. Since not all data are transferred to the cloud layer during the decision-making process, the latency is considerably reduced. For rare events, the cloud controller takes action with the full control of the network.

Our proposed architecture can be used to solve previously specified routing issues efficiently by defining the routing rules at the local fog controllers as flow tables. Depending on data type or link condition, if a forwarding device needs to take an action, it first checks its flow entries to look at what to



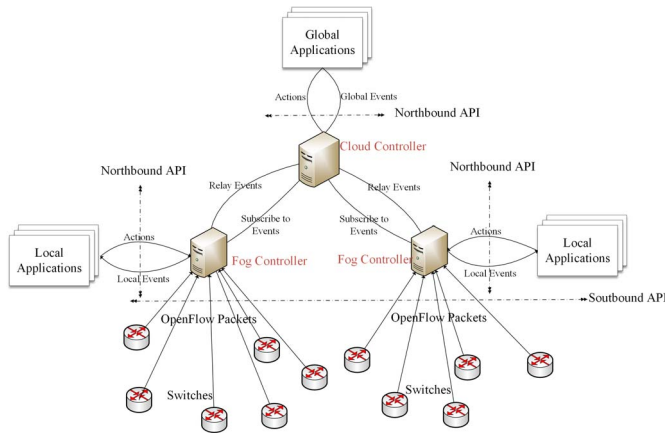


Fig. 4. Hierarchical SDN architecture.

do by evaluating the results of incoming packets and finding the highest priority match. If there is a match, packets are processed and forwarded to the destination address. Then, if there is no rule for this action, it communicates with its local fog controller and sends a request for further processing. For example, if a secure routing is needed and the forwarding device does not have the secure route to the destination, it asks its local fog server for this information. Then, it applies the action according to instructions coming from the fog controller. It also updates its flow table entries for further actions. By employing frequency-based rules and actions for frequent events (data types) are stored in local fog servers. However, if a rare event happens in the network, the forwarding device first asks its local fog controller. If its related fog controller has no predefined action according to its flow table and then, it asks the cloud controller. According to the cloud controller's response, the fog controller relays this response and the forwarding device takes the action. The proposed model is expected to reduce traffic and latency due to routing needs in fog-enabled IoT services. In what follows, due to page limits, we briefly show how our proposed model behaves under the shortest path constraint.

#### A. Experimental Results and Discussions

In the previous section, we explained how the distributed SDN structure is well suited to the fog computing to increase routing capabilities of IoT services. In this section, the performance of the proposed SDN-based fog computing architecture is evaluated in Mininet [108], which is an open-source SDN emulator containing virtual hosts, switches, and controllers to create a network on a single OS kernel and uses the real network stack to process packets and connect to real networks [109]. Our proposed framework supports POX [110] as Python-based SDN controller and OF [93] protocol to ensure communication between the controller and switches. Also, Dijkstra's algorithm is employed to find the shortest path from source to destination. It must be noted in this survey paper; we only provide a primitive evaluation of the proposed protocol to show the feasibility of it.

To evaluate the proposed architecture, a simple SDN tree topology is generated with 127 switches, 2 communicating

TABLE IV  
SIMULATION SETTINGS

	Scenario 1	Scenario 2
Upper link bandwidth	100 Mbps	10 Mbps
Lower link bandwidth	10 Mbps	5 Mbps
Upper link delay	1-2 ms	5-10 ms
Lower link delay	5-10 ms	10-20 ms
Number of switches	127	
Number of hosts	2	
Number of controllers	1,3,5,7,9	
Transmitted packet types	ICMP, TCP	
Controller	POX	
Testing tools	Ping, Iperf	

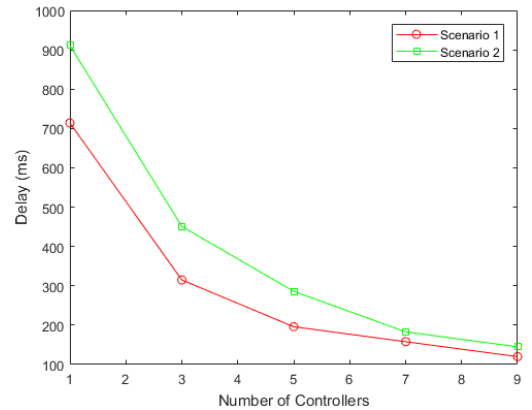


Fig. 5. Delay versus number of controllers.

hosts, and varying number of controllers in Mininet. Tree topology is chosen due to well fitting into our proposed architecture. We assume that the shortest path is always required between communicating hosts, but the hosts do not have the shortest path information. We created two scenarios by varying the number of controllers and some other parameters in the network to show the efficiency of our proposed model. While transmitting the data, Scenario 1 has a better link capacity and lower latency than Scenario 2. In both scenarios, upper links are denoted as the links among fog servers. Similarly, lower links are denoted as the links connecting IoT devices to fog servers. The bandwidth and the delay vary according to link states. The details of settings are defined in Table IV. We measure the routing delay, data transmission overhead, and throughput over two different scenarios by comparing each other.

All the simulations are performed ten times and the average values are considered to validate the proposed architecture. In the first experiment, the delay denotes the time required for the first ICMP packet to arrive the destination host. Fig. 5 shows the delay according to the number of controllers. In the second experiment, the total number of OF packets are calculated according to the number of controllers over these two scenarios. Fig. 6 shows the results for total number of OF packets for 1000 packet transmission between hosts according to varying number of controllers. On the other hand, it is observed that the delay time is affected by the change of link parameters

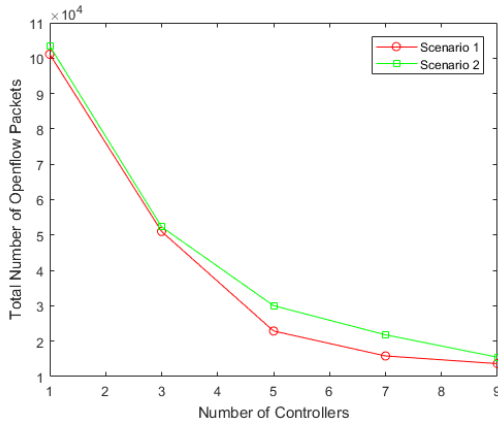


Fig. 6. Total number of OF packets versus number of controllers.

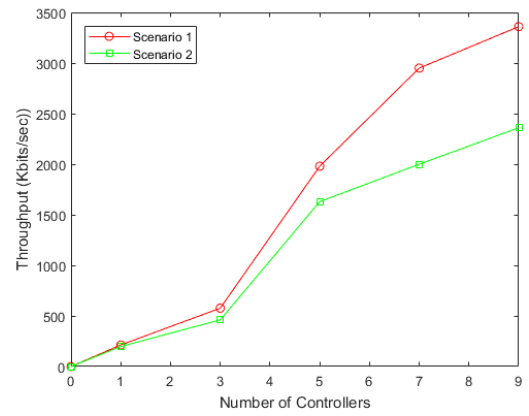


Fig. 8. Average throughputs for 60 s.

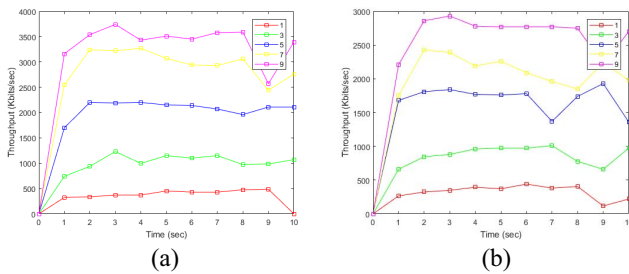


Fig. 7. Throughput during the first 10 s in (a) Scenario 1 and (b) Scenario 2 for the different number of controllers (1, 3, 5, 7, 9) in the network.

especially as the number of controllers decreases. However, the total number of OF packets is affected very slightly by the change of link parameters.

In the proposed architecture, as the number of hierarchical layers increases in the network, more controllers are assigned to manage the network. Increasing the number of controllers in the experiments refers that the network is more distributed so that a smaller portion of the network is managed and controlled by a single controller. Accordingly, the time and data transmission amount required for local IoT services are expected to be reduced. In fact, this can be observed in Fig. 5. As seen in Fig. 5, the time required for processing an IoT service is decreased as local decisions are taken quickly. A similar effect is also observed in Fig. 6. As the number of controllers increases, local fog servers cover a smaller portion of the network topology and deal with fewer OF packets resulting in reduced data transmission overhead.

Lastly, the throughput is measured for the TCP packets for the first 10 s and 60 s durations. Fig. 7(a) and (b) shows the throughput during the first 10 s for Scenario 1 and Scenario 2, respectively. Each line represents the throughput change with a different number of controllers (e.g., 1, 3, 5, 7, 9). As expected, the number of controllers has a positive effect on throughput in both scenarios. Scenario 1 has slightly better throughput than Scenario 2. In addition, average throughput values are given in Fig. 8 to show the difference as the number of controllers increases over two scenarios. Both figures confirm that the throughput increases as the number of controllers increases thereby showing the efficiency of the proposed architecture.

## VI. OPEN RESEARCH AREAS AND FUTURE DIRECTIONS

As presented in the previous sections, routing is a critical issue for data transmission with security, reduced latency, and bandwidth usage in fog-enabled IoT services. Due to the dynamic nature of the fog-enabled applications, satisfying routing needs of these services is not a trivial task and there are still several open research problems to be addressed. In what follows, we summarize these issues and open research problems.

- 1) Fog and cloud are complementary systems where they reside in different layers. Both layers have different tasks and distribution of these tasks directly affect the routing performance. This integration brings the problem of who will do which task. The requirements of the tasks must be clearly defined and sent to the appropriate computing layer. For example, tasks which require real-time processing or low computational cost are processed with fog computing. On the other hand, tasks which require massive storage space and high computational cost are processed with cloud computing. For service orchestration, incorporation with semantics can be utilized for many issues arising from task distribution conflict.
- 2) The collaboration of fog computing with SDN facilitates a better network routing management; however, it may also cause an extra latency. Dynamic conditions caused by mobility and unreliable wireless connections pose a challenging issue for this integration. The requirements of both FSDN concepts should be taken into account for a smooth integration. Also, an efficient routing analysis should be realized to avoid excessive routing delay.
- 3) Policies should be specified for the components of fog-cloud infrastructure to distribute tasks and services. For example, a robust policy management is required for a smart vehicle scenario with real-time data analytics and secure communication requirements. In this case, a different policy for each requirement (such as analytics or security) should be defined clearly by considering the overlaps and conflicts between fog and cloud entities.
- 4) In accordance with predefined policies, the location and responsibility of each controller should be determined. Planning the controller locations is an instance of the joint

optimization problem and it can be solved by linear programming, single-objective, or multiobjective heuristic algorithms. Therefore, by solving this problem low communication latency among controllers and their switches, high reliability and resiliency of the network, minimum deployment cost, and minimum energy consumption can be ensured.

- 5) Fog servers are not always resource-rich. When they are employed in scenarios that have various application types, it can be very challenging to maintain QoS due to the resource fragmentation. Traditional optimization techniques offer appropriate design and management of fog resources by efficiently determining the number and location of fog servers to ensure QoS requirements for user satisfaction.
- 6) Limited bandwidth and latency may degrade the performance of offloaded tasks. Fog computing is used for computation offloading for resource-constrained end devices, however, a decision should be made carefully about whether and where to offload the intensive tasks. Game theoretic approaches are good candidates to tackle the decentralized computation offloading problem of fog computing. These approaches may ensure the efficient computation offloading performance and scalability.
- 7) Fog servers can be customized and employed for a certain service such as healthcare or air-quality monitoring. In such cases, the relevant information generated from heterogeneous IoT devices must be routed over the network to the fog server that is specialized for that service. To this aim, content-centric approaches can be used to forward data to the related fog servers. Content-based delivery services may help fog computing achieve reduced download delays, less bandwidth usage, improved content availability, and reduced cost. In addition, named data networking can be integrated with fog computing to realize content-based routing.

## VII. CONCLUSION

In large-scale fog enabled IoT platforms, data transmission from end devices to fog servers requires a routing scheme that satisfies several constraints, such as latency, bandwidth, and security. In this paper, we present a comprehensive survey of routing in fog-enabled IoT platforms. We analyzed and categorized IoT services based on their routing needs and their integration with SDN technology. After identifying requirements, we propose a hierarchical SDN-based fog computing architecture to solve the routing issues in fog networks. Distributing SDN controllers into the fog and cloud tiers enables workload sharing and local and global management of the network. Our initial experimental results clearly show that the proposed hierarchical SDN-based fog computing architecture reduces routing delay and data transmission overhead while increasing throughput.

## REFERENCES

- [1] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: Architecture, key technologies, applications and open issues," *J. Netw. Comput. Appl.*, vol. 98, pp. 27–42, Nov. 2017.
- [2] "Cisco global cloud index: Forecast and methodology, 2015–2020," San Jose, CA, USA, Cisco, White Paper, 2016. Accessed: Aug. 15, 2018. [Online]. Available: <http://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.pdf>
- [3] "Fog computing and the Internet of Things: Extend the cloud to where the things are," San Jose, CA, USA, Cisco, White Paper, 2016. Accessed: Apr. 8, 2018. [Online]. Available: [http://www.cisco.com/c/dam/en\\_us/solutions/trends/iot/docs/computing-overview.pdf](http://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf)
- [4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed. Workshop Mobile Cloud Comput. (MCC)*, 2012, pp. 13–16.
- [5] L. M. Vaquero and L. Roderio-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, 2014.
- [6] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proc. Workshop Mobile Big Data (Mobidata)*, 2015, pp. 37–42.
- [7] S. Agarwal, M. Kodialam, and T. V. Lakshman, "Traffic engineering in software defined networks," in *Proc. IEEE INFOCOM*, 2013, pp. 2211–2219.
- [8] S. K. Datta, C. Bonnet, and J. Haerri, "Fog computing architecture to enable consumer centric Internet of Things services," in *Proc. IEEE Int. Symp. Consum. Electron. (ISCE)*, 2015, pp. 1–2.
- [9] M. Yannuzzi, R. Milito, R. Serral-Gracià, D. Montero, and M. Nemirovsky, "Key ingredients in an IoT recipe: Fog computing, cloud computing, and more fog computing," in *Proc. IEEE 19th Int. Workshop Comput.-Aided Model. Design Commun. Links Netw. (CAMAD)*, 2014, pp. 325–329.
- [10] T. H. Luan *et al.*, "Fog computing: Focusing on mobile users at the edge," *ArXiv Preprint*. [Online]. Available: <https://arxiv.org/pdf/1502.01815v3.pdf>
- [11] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog *et al.*: A survey and analysis of security threats and challenges," *Future Gener. Comput. Syst.*, vol. 78, pp. 680–698, Jan. 2018.
- [12] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.
- [13] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog computing: A taxonomy, survey and future directions," in *Internet of Everything*. Singapore: Springer, 2018, pp. 103–130.
- [14] Y. Ai, M. Peng, and K. Zhang, "Edge cloud computing technologies for Internet of Things: A primer," *Digit. Commun. Netw.*, vol. 4, no. 2, pp. 77–86, 2017.
- [15] K. Wang, H. Yin, W. Quan, and G. Min, "Enabling collaborative edge computing for software defined vehicular networks," *IEEE Netw.*, vol. 32, no. 5, pp. 112–117, Sep./Oct. 2018.
- [16] "Cisco global cloud index: Forecast and methodology, 2014–2019," San Jose, CA, USA, Cisco, White Paper, Oct. 2015.
- [17] M. Firdhous, O. Ghazali, and S. Hassan, "Fog computing: Will it be the future of cloud computing?" in *Proc. 3rd Int. Conf. Informat. Appl. (ICIA)*, 2014, pp. 8–15.
- [18] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wireless Commun. Mobile Comput.*, vol. 13, no. 18, pp. 1587–1611, 2013.
- [19] B. Liang, "Mobile edge computing," in *Key Technologies for 5G Wireless Systems*. Cambridge, U.K.: Cambridge Univ. Press, 2017.
- [20] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," Sophia Antipolis, France, ETSI, White Paper, pp. 1–16, 2015.
- [21] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.
- [22] E. Ahmed and M. H. Rehmani, "Mobile edge computing: Opportunities, solutions, and challenges," *Future Gener. Comput. Syst.*, vol. 70, pp. 59–63, May 2017.
- [23] M. T. Beck, M. Werner, S. Feld, and S. Schimper, "Mobile edge computing: A taxonomy," in *Proc. 6th Int. Conf. Adv. Future Internet*, 2014, pp. 48–55.
- [24] F. Y. Okay and S. Ozdemir, "A fog computing based smart grid model," in *Proc. Int. Symp. Netw. Comput. Commun. (ISNCC)*, 2016, pp. 1–6.
- [25] L. Gao, T. H. Luan, S. Yu, W. Zhou, and B. Liu, "FogRoute: DTN-based data dissemination model in fog computing," *IEEE Internet Things J.*, vol. 4, no. 1, pp. 225–235, Feb. 2017.

- [26] K. A. Nuaimi, N. Mohamed, M. A. Nuaimi, and J. Al-Jaroodi, "A survey of load balancing in cloud computing: Challenges and algorithms," in *Proc. IEEE 2nd Symp. Netw. Cloud Comput. Appl.*, 2012, pp. 137–142.
- [27] E. J. Ghomi, A. M. Rahmani, and N. N. Qader, "Load-balancing algorithms in cloud computing: A survey," *J. Netw. Comput. Appl.*, vol. 88, pp. 50–71, Jun. 2017.
- [28] T. Lu, S. Chang, and W. Li, "Fog computing enabling geographic routing for urban area vehicular network," *Peer-to-Peer Netw. Appl.*, vol. 11, no. 4, pp. 749–755, 2018.
- [29] W. Fang, W. Zhang, J. Xiao, Y. Yang, and W. Chen, "A source anonymity-based lightweight secure AODV protocol for fog-based MANET," *Sensors*, vol. 17, no. 6, p. 1421, 2017.
- [30] S. Soo, C. Chang, and S. N. Srirama, "Proactive service discovery in fog computing using mobile ad hoc social network in proximity," in *Proc. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber Phys. Soc. Comput. (CPSCom) IEEE Smart Data (SmartData)*, 2016, pp. 561–566.
- [31] S. Ivanov, S. Balasubramaniam, D. Botvich, and O. B. Akan, "Gravity gradient routing for information delivery in fog wireless sensor networks," *Ad Hoc Netw.*, vol. 46, pp. 61–74, Aug. 2016.
- [32] H. Gupta, S. B. Nath, S. Chakraborty, and S. K. Ghosh, "SDFog: A software defined computing architecture for QoS aware service orchestration over edge devices," *ArXiv Preprint*, 2017. [Online]. Available: <https://arxiv.org/pdf/1609.01190.pdf>
- [33] J. Su, F. Lin, X. Zhou, and X. Lu, "Steiner tree based optimal resource caching scheme in fog computing," *China Commun.*, vol. 12, no. 8, pp. 161–168, Aug. 2015.
- [34] S. O. Amin, R. Ravindran, and G. Wang, "DISCS: Secure content distribution in IP using information centric networking," in *Proc. IEEE Fog World Congr. (FWC)*, 2017, pp. 1–6.
- [35] Y. Teranishi, T. Kimata, H. Yamanaka, E. Kawai, and H. Harai, "Dynamic data flow processing in edge computing environments," in *Proc. IEEE 41st Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 1, 2017, pp. 935–944.
- [36] H. Trinh *et al.*, "Energy-aware mobile edge computing and routing for low-latency visual data processing," *IEEE Trans. Multimedia*, vol. 20, no. 10, pp. 2562–2577, Oct. 2018.
- [37] C.-M. Huang *et al.*, "V2V data offloading for cellular network based on the software defined network (SDN) inside mobile edge computing (MEC) architecture," *IEEE Access*, vol. 6, pp. 17741–17755, 2018.
- [38] P. G. V. Naranjo, M. Shojafar, H. Mostafaei, Z. Pooranian, and E. Baccarelli, "P-SEP: A prolong stable election routing algorithm for energy-limited heterogeneous fog-supported wireless sensor networks," *J. Supercomput.*, vol. 73, no. 2, pp. 733–755, 2017.
- [39] A. Hakiri, B. Sellami, P. Patil, P. Berthou, and A. Gokhale, "Managing wireless fog networks using software-defined networking," in *Proc. 14th Int. Conf. Comput. Syst. Appl. (AICCSA)*, 2017, pp. 1149–1156.
- [40] A. Betzler, F. Quer, D. Camps-Mur, I. Demirkol, and E. Garcia-Villegas, "On the benefits of wireless SDN in networks of constrained edge devices," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, 2016, pp. 37–41.
- [41] Z. Liu, J. Zhang, Y. Li, L. Bai, and Y. Ji, "Joint jobs scheduling and lightpath provisioning in fog computing micro datacenter networks," *J. Opt. Commun. Netw.*, vol. 10, no. 7, pp. B152–B163, Jul. 2018.
- [42] D.-Y. Lin, Y.-L. Hsu, and H.-Y. Wei, "A novel forwarding policy under cloud radio access network with mobile edge computing architecture," in *Proc. IEEE 2nd Int. Conf. Fog Edge Comput. (ICFEC)*, 2018, pp. 1–9.
- [43] S. Wang *et al.*, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.
- [44] F. Al-Turjman, "Cognitive caching for the future sensors in fog networking," *Pervasive Mobile Comput.*, vol. 42, pp. 317–334, Dec. 2017.
- [45] F. Song *et al.*, "Smart collaborative caching for information-centric IoT in fog computing," *Sensors*, vol. 17, no. 11, p. 2512, 2017.
- [46] R. Tandon and O. Simeone, "Cloud-aided wireless networks with edge caching: Fundamental latency trade-offs in fog radio access networks," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2016, pp. 2029–2033.
- [47] A. Sengupta, R. Tandon, and O. Simeone, "Fog-aided wireless networks for content delivery: Fundamental latency tradeoffs," *IEEE Trans. Inf. Theory*, vol. 63, no. 10, pp. 6650–6678, Oct. 2017.
- [48] A. Roushdy, A. S. Motahari, M. Nafie, and D. Gündüz, "Cache-aided fog radio access networks with partial connectivity," in *Proc. Wireless Commun. Netw. Conf. (WCNC)*, 2018, pp. 1–6.
- [49] Y. Hua, L. Guan, and K. Kyriakopoulos, "Semi-edge: From edge caching to hierarchical caching in network fog," in *Proc. 1st Int. Workshop Edge Syst. Anal. Netw.*, 2018, pp. 43–48.
- [50] G. Lee, W. Saad, and M. Bennis, "Online optimization for low-latency computational caching in fog networks," in *Proc. IEEE Fog World Congr. (FWC)*, 2017, pp. 1–6.
- [51] X. Wei *et al.*, "MVR: An architecture for computation offloading in mobile edge computing," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, 2017, pp. 232–235.
- [52] J. Oueis, E. C. Strinati, and S. Barbarossa, "The fog balancing: Load distribution for small cell cloud computing," in *Proc. Veh. Technol. Conf. (VTC Spring)*, 2015, pp. 1–6.
- [53] D. Ye, M. Wu, S. Tang, and R. Yu, "Scalable fog computing with service offloading in bus networks," in *Proc. IEEE 3rd Int. Conf. Cyber Security Cloud Comput. (CSCloud)*, 2016, pp. 247–251.
- [54] M. A. Hassan, M. Xiao, Q. Wei, and S. Chen, "Help your mobile applications with fog computing," in *Proc. 12th Annu. IEEE Int. Conf. Sens. Commun. Netw. Workshops (SECON Workshops)*, 2015, pp. 1–6.
- [55] C. Fricker, F. Guillemin, P. Robert, and G. Thompson, "Analysis of an offloading scheme for data centers in the framework of fog computing," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 1, no. 4, p. 16, 2016.
- [56] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015.
- [57] V. Mushunuri, A. Kattepur, H. K. Rath, and A. Simha, "Resource optimization in fog enabled IoT deployments," in *Proc. 2nd Int. Conf. Fog Mobile Edge Comput. (FMEC)*, 2017, pp. 6–13.
- [58] S. Yu, X. Wang, and R. Langar, "Computation offloading for mobile edge computing: A deep learning approach," in *Proc. IEEE 28th Annu. Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, 2017, pp. 1–6.
- [59] L. Liu, Z. Chang, and X. Guo, "Socially aware dynamic computation offloading scheme for fog computing system with energy harvesting devices," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1869–1879, Jun. 2018.
- [60] V. Cardellini *et al.*, "A game-theoretic approach to computation offloading in mobile cloud computing," *Math. Program.*, vol. 157, no. 2, pp. 421–449, 2016.
- [61] D. Liu, L. Khoukhi, and A. Hafid, "Decentralized data offloading for mobile cloud computing based on game theory," in *Proc. IEEE 2nd Int. Conf. Fog Mobile Edge Comput. (FMEC)*, 2017, pp. 20–24.
- [62] S. Ninging, G. Chao, A. Xingshuo, and Z. Qiang, "Fog computing dynamic load balancing mechanism based on graph repartitioning," *China Commun.*, vol. 13, no. 3, pp. 156–164, Mar. 2016.
- [63] D. Puthal *et al.*, "Secure and sustainable load balancing of edge data centers in fog computing," *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 60–65, May 2018.
- [64] R. Beraldi, A. Mtibaa, and H. Alnuweiri, "Cooperative load balancing scheme for edge computing resources," in *Proc. 2nd Int. Conf. Fog Mobile Edge Comput. (FMEC)*, 2017, pp. 94–100.
- [65] Y. Yu, X. Li, and C. Qian, "SDLB: A scalable and dynamic software load balancer for fog and mobile edge computing," in *Proc. Workshop Mobile Edge Commun.*, 2017, pp. 55–60.
- [66] L. Chen and J. Xu, "Socially trusted collaborative edge computing in ultra dense networks," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, 2017, p. 9.
- [67] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On reducing IoT service delay via fog offloading," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 998–1010, Apr. 2018.
- [68] M. Verma, N. Bhardwaj, and A. K. Yadav, "Real time efficient scheduling algorithm for load balancing in fog computing environment," *Int. J. Inf. Technol. Comput. Sci.*, vol. 8, no. 4, pp. 1–10, 2016.
- [69] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coding for distributed fog computing," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 34–40, Apr. 2017.
- [70] 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)*, 2015, pp. 3909–3914.
- [71] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1171–1181, Dec. 2016.
- [72] K. Habak, M. Ammar, K. A. Harras, and E. Zegura, "Femto clouds: Leveraging mobile devices to provide cloud service at the edge," in *Proc. IEEE 8th Int. Conf. Cloud Comput. (CLOUD)*, 2015, pp. 9–16.

- [73] K. Habak, E. W. Zegura, M. Ammar, and K. A. Harras, "Workload management for dynamic mobile device clusters in edge femtoclouds," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, 2017, p. 6.
- [74] Z. Wang *et al.*, "User mobility aware task assignment for mobile edge computing," *Future Gener. Comput. Syst.*, vol. 85, pp. 1–8, Aug. 2018.
- [75] A. Singh, N. Auluck, O. Rana, A. Jones, and S. Nepal, "RT-SANE: Real time security aware scheduling on the network edge," in *Proc. 10th Int. Conf. Utility Cloud Comput.*, 2017, pp. 131–140.
- [76] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2016, pp. 1451–1455.
- [77] S. F. Abedin, M. G. R. Alam, N. H. Tran, and C. S. Hong, "A fog based system model for cooperative IoT node pairing using matching theory," in *Proc. 17th Asia-Pac. Netw. Oper. Manag. Symp. Manag. Very Connect World (APNOMS)*, 2015, pp. 309–314.
- [78] M. Aazam and E.-N. Huh, "Dynamic resource provisioning through fog micro datacenter," in *Proc. 12th IEEE Int. Workshop Manag. Ubiquitous Commun. Services*, 2015, pp. 105–110.
- [79] M. Aazam and E. N. Huh, "Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT," in *Proc. IEEE 29th Int. Conf. Adv. Inf. Neww. Appl. (AINA)*, 2015, pp. 687–694.
- [80] K. Intharawijitr, K. Iida, and H. Koga, "Analysis of fog model considering computing and communication latency in 5G cellular networks," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun.*, Sydney, NSW, Australia, 2016, pp. 5–8.
- [81] S. Agarwal, S. Yadav, and A. K. Yadav, "An efficient architecture and algorithm for resource provisioning in fog computing," *Int. J. Inf. Eng. Electron. Bus.*, vol. 8, no. 1, pp. 48–61, 2016.
- [82] Y. Sun and N. Zhang, "A resource-sharing model based on a repeated game in fog computing," *Saudi J. Biol. Sci.*, vol. 24, no. 3, pp. 687–694, 2017.
- [83] H. Zhang, Y. Zhang, Y. Gu, D. Niyato, and Z. Han, "A hierarchical game framework for resource management in fog computing," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 52–57, Aug. 2017.
- [84] J. Plachy, Z. Becvar, and E. C. Strinati, "Dynamic resource allocation exploiting mobility prediction in mobile edge computing," in *Proc. IEEE 27th Annu. Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, 2016, pp. 1–6.
- [85] A. Amjad, F. Rabby, S. Sadia, M. Patwary, and E. Benkhelifa, "Cognitive edge computing based resource allocation framework for Internet of Things," in *Proc. IEEE 2nd Int. Conf. Fog Mobile Edge Comput. (FMEC)*, 2017, pp. 194–200.
- [86] C. N. L. Tan, C. Klein, and E. Elmroth, "Location-aware load prediction in edge data centers," in *Proc. IEEE 2nd Int. Conf. Fog Mobile Edge Comput. (FMEC)*, 2017, pp. 25–31.
- [87] J. Xu, B. Palanisamy, H. Ludwig, and Q. Wang, "Zenith: Utility-aware resource allocation for edge computing," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, 2017, pp. 47–54.
- [88] H. Zhang *et al.*, "Fog computing in multi-tier data center networks: A hierarchical game approach," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2016, pp. 1–6.
- [89] D. Kreutz *et al.*, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 102, no. 1, pp. 14–76, Jan. 2015.
- [90] B. A. A. Nunes, M. Mendonca, X.-N. Nyugen, K. Obraczka, and T. Turtletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1617–1634, 3rd Quart., 2014.
- [91] L. Suresh, J. Schulz-zander, R. Merz, A. Feldman, and T. Vazao, "Towards programmable enterprise WLANs with Odin," in *Proc. ACM 1st Workshop Hot Topics Softw. Defined Netw.*, 2012, pp. 115–120.
- [92] K. K. Yap *et al.*, "OpenRoads: Empowering research in mobile networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 1, pp. 125–126, 2010.
- [93] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.
- [94] A. Doria *et al.*, "Forwarding and control element separation (forces) protocol specification," Internet Eng. Task Force, Fremont, CA, USA, RFC 5810, 2010.
- [95] T. V. Lakshman, T. Nandagopal, R. Ramjee, K. Sabnani, and T. Woo, "The softrouter architecture," U.S. Patent 7715 382, pp. 1–6, 2010.
- [96] W. Braun and M. Menth, "Software-defined networking using OpenFlow: Protocols, applications and architectural design choices," *Future Internet*, vol. 6, no. 2, pp. 302–336, 2014.
- [97] X. Sun and N. Ansari, "EdgeIoT: Mobile edge computing for the Internet of Things," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 22–29, Dec. 2016.
- [98] P. Yang, N. Zhang, Y. Bi, L. Yu, and X. S. Shen, "Catalyzing cloud-fog interoperability in 5G wireless networks: An SDN approach," *IEEE Netw.*, vol. 31, no. 5, pp. 14–20, Sep. 2017.
- [99] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 587–597, Mar. 2018.
- [100] X. He, Z. Ren, C. Shi, and J. Fang, "A novel load balancing strategy of software-defined cloud/fog networking in the Internet of Vehicles," *China Commun.*, vol. 13, no. 2, pp. 140–149, 2016.
- [101] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system," *IEEE Trans. Comput.*, vol. 65, no. 12, pp. 3702–3712, Dec. 2016.
- [102] Y. He, F. R. Yu, N. Zhao, V. C. M. Leung, and H. Yin, "Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 31–37, Dec. 2017.
- [103] 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. Manag. (IM)*, Ottawa, ON, Canada, 2015, pp. 1202–1207.
- [104] H. Selvi, G. Gür, and F. Alagöz, "Cooperative load balancing for hierarchical SDN controllers," in *Proc. IEEE 17th Int. Conf. High Perform. Switching Routing (HPSR)*, 2016, pp. 121–126.
- [105] S. Schmid and J. Suomela, "Exploiting locality in distributed SDN control," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, 2013, pp. 121–126.
- [106] O. Bial, M. B. Mamoun, and R. Benaini, "An overview on SDN architectures with multiple controllers," *J. Comput. Netw. Commun.*, vol. 2016, Apr. 2016, Art. no. 9396525.
- [107] S. Shen *et al.*, "Multistage signaling game-based optimal detection strategies for suppressing malware diffusion in fog-cloud-based IoT networks," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1043–1054, Apr. 2018.
- [108] *Mininet*. Accessed: Jun. 13, 2018. [Online]. Available: <http://mininet.org/>
- [109] S.-Y. Wang, "Comparison of SDN OpenFlow network simulator and emulators: EstiNet vs. mininet," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, 2014, pp. 1–6.
- [110] *POX Wiki*. Accessed: Jun. 13, 2018. [Online]. Available: <https://openflow.stanford.edu/display/ONL/POX+Wiki>

Authors' photographs and biographies not available at the time of publication.