Addressing the Node Discovery Problem in Fog Computing

Vasileios Karagiannis
Distributed Systems Group, TU Wien, Austria
v.karagiannis@dsg.tuwien.ac.at

Nitin Desai
Mälardalen University, Västerås, Sweden
nitin.desai@mdh.se

Stefan Schulte
Distributed Systems Group, TU Wien, Austria
s.schulte@dsg.tuwien.ac.at

Sasikumar Punnekkat
Mälardalen University, Västerås, Sweden
sasikumar.punnekkat@mdh.se

Abstract

In recent years, the Internet of Things (IoT) has gained a lot of attention due to connecting various sensor devices with the cloud, in order to enable smart applications such as: smart traffic management, smart houses, and smart grids, among others. Due to the growing popularity of the IoT, the number of Internet-connected devices has increased significantly. As a result, these devices generate a huge amount of network traffic which may lead to bottlenecks, and eventually increase the communication latency with the cloud. To cope with such issues, a new computing paradigm has emerged, namely: fog computing. Fog computing enables computing that spans from the cloud to the edge of the network in order to distribute the computations of the IoT data, and to reduce the communication latency. However, fog computing is still in its infancy, and there are still related open problems. In this paper, we focus on the node discovery problem, i.e., how to add new compute nodes to a fog computing system. Moreover, we discuss how addressing this problem can have a positive impact on various aspects of fog computing, such as fault tolerance, resource heterogeneity, proximity awareness, and scalability. Finally, based on the experimental results that we produce by simulating various distributed compute nodes, we show how addressing the node discovery problem can improve the fault tolerance of a fog computing system.

2012 ACM Subject Classification Computer systems organization → Cloud computing; Computer systems organization → Fault-tolerant network topologies

Keywords and phrases Fog computing, Edge computing, Internet of Things, Node discovery, Fault tolerance

Digital Object Identifier 10.4230/OASIcs.Fog-IoT.2020.5

Funding The research leading to this paper has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 764785, FORA – Fog Computing for Robotics and Industrial Automation.

1 Introduction

The IoT paradigm envisions a world in which everyday objects (i.e., wearables, dumpsters, phones, etc.) connect to the Internet [14]. Such objects may use this connectivity to exchange, store, and process data in order to sense and to affect the surrounding environment [12]. Since the computational resources of the everyday objects alone may not be sufficient for handling the required computational efforts to achieve this, the IoT devices commonly make use of cloud-based computational resources [24].

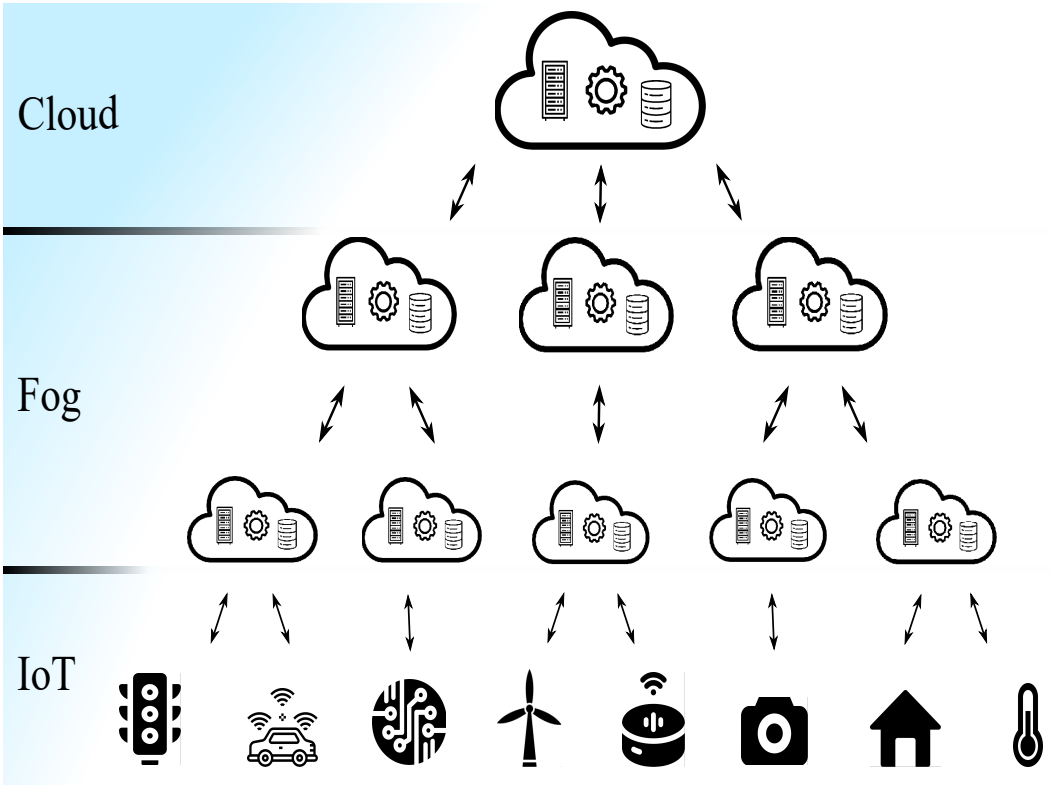


Figure 1 A fog computing system consisting of various compute nodes that span from the cloud to the edge of the network.

However, despite the aid of the cloud, the traffic from a large number of Internet-connected devices can still lead to bottlenecks which increase the communication latency, and may even limit the expansion of the IoT [19]. Moreover, there are concerns related to preserving the privacy of the aggregated IoT data, and reducing the communication cost [20]. To cope with such issues, novel computing paradigms have emerged, two of the most popular being fog computing, and edge computing.

One distinguishing characteristic to separate fog computing from edge computing, is that the fog envisions a hierarchy of computational resources which span from the cloud to the edge of the network [3]. For example, Fig. 1 shows a fog computing system that includes various interconnected cloud and fog compute nodes which spread to the network edge where the IoT devices reside. Edge computing on the other hand, aims at pushing the computations towards the edge of the network wherever there are available computational resources (e.g., cloudlets or fog nodes) without explicitly including interactions with the cloud [25].

The research efforts applied in the context of these two paradigms have resulted in architectures, models, and frameworks for performing computations in the proximity of the IoT devices. Due to such efforts, fog compupting and edge computing systems have been observed to provide significant benefits for use cases like data stream processing [5], preserving privacy in the IoT [20], performing analytics of IoT data [1], online storage [21], and others [17].

To implement such architectures, compute nodes are provisioned at strategic positions throughout the network in order to distribute the computations, avoid bottlenecks, and reduce the communication latency [13]. A lot of research has been conducted in this context,

resulting in multiple computing systems which aim at leveraging the edge of the network in order to satisfy the application requirements (e.g., regarding latency and bandwidth) and to improve the user experience [15].

Despite the popularity of fog computing and edge computing in the distributed systems research community, computing at the edge of the network is still a relatively recent research topic. For this reason, there are still various important open research problems and challenges, which require further investigation [22]. In this paper, we focus on the node discovery problem [4, 23].

Typically, fog computing and edge computing research assumes that compute nodes are already discovered and integrated in the system [11]. However, this can be a complicated task because the current node discovery approaches usually used in cloud-based systems, are not applicable to fog computing since the problem is very different when dealing with compute nodes at the edge of the network [29]. For instance, fog computing systems are expected to leverage on the proximity of the compute nodes while also considering compute nodes with very diverse resource capacities. Such aspects which have not been considered in the context of cloud computing, make novel node discovery techniques–tailored to fog computing–necessary. For this reason, in this paper we analyze the node discovery problem in fog computing, and we discuss the various related aspects that need to be taken into account. Furthermore, we identify the related research questions which need to be addressed, in order to tackle this problem efficiently.

The rest of this paper is organized as follows: Section 2 discusses related work from the literature. Afterwards, in Section 3, we analyze the node discovery problem in fog computing, and we identify related research questions. Subsequently in Section 4, we present the preliminary evaluation results that we produce based on simulations which show some of the benefits of addressing the proposed problem (regarding fault tolerance). Finally, Section 5 concludes this work, and describes our plans for further research on this topic.

2 Related work

The majority of related work, assumes that the various compute nodes of a fog computing system are already discovered and integrated in the system [11]. Typically, these systems follow a hierarchical architecture whereby the nodes are organized in layers [26]. For instance, Bellavista et al. [2] discuss the execution of services on compute nodes at the edge of the network using a three-layer architecture, and Deng et al. [6] discuss the provisioning of services in distributed edge nodes. However, none of these approaches discuss how the compute nodes are discovered and placed in appropriate positions in the hierarchy.

Kolcun et al. [18] present a distributed platform that allows IoT devices from wireless sensor networks, to send data to cloud and local compute nodes. By shifting the computations from the cloud to the local nodes, this approach reduces the network traffic. Furthermore, the authors propose a node discovery algorithm which aids in finding an appropriate compute node for each IoT device.

Similarly, Tomar and Matam [27] present a framework that allows the data from the IoT devices to be processed in local compute nodes thereby lowering the dependency on the cloud. This framework also includes a node discovery algorithm for finding appropriate compute nodes for the IoT devices.

Finally, Venanzi et al. [31] address the same problem of node discovery for IoT devices although, the focus of this approach is to prolong the lifespan of these devices by considering energy efficiency aspects.

Notably, these approaches focus on the problem of selecting appropriate compute nodes for processing the IoT data. In contrast, the work at hand focuses on the problem of discovering new compute nodes that join a fog computing system. Even though these problems seem similar, they require different solutions. The former problem relies on the wireless communication of the IoT devices to discover potential compute nodes (i.e. the compute nodes that reside within wireless range). In the latter problem, which is the problem we address in our work, the compute nodes that span from the cloud to the edge of the network may not integrate wireless communication. Therefore, the aforementioned solutions that address the node discovery problem in the IoT, do not apply to the node discovery problem in fog computing.

Further related work can be found in approaches that aim at creating fog computing systems for handling applications related to safety. For instance, Dobrin et al. [8] discuss safety-critical applications while focusing on the problem of having unexpected failures, and Desai et al. [7] discuss various safety aspects (with a focus on safety-critical applications) that need to be considered in fog computing systems.

In our work, we also address fault tolerance. However, these works consider fault tolerance as an independent problem which makes it hard to cope with. In our work, we consider fault tolerance at a very early stage, i.e., during the node discovery phase, which increases our options regarding finding appropriate solutions, and based on this, we present promising results.

Therefore, the papers discussed so far either briefly mention the node discovery problem in fog computing, or assume that the compute nodes are already discovered and integrated in the system. Thus, they do not provide an analysis of the problem, or any concrete ways to solve it. On the contrary, in our work we analyze different aspects of this problem, we propose related research questions, and we also present promising results towards addressing the node discovery problem in fog computing efficiently.

3 The Node Discovery Problem

The node discovery problem refers to the way that new compute nodes are detected by the system, as well as the process of integrating these nodes (this is also referred to as the discovery phase). For instance, in Fig. 1 we show a fog computing system consisting of one cloud compute node, and eight fog compute nodes (e.g., cloudlets, base stations, routers, etc.), which are organized in three layers. If a new compute node becomes available, how is this node detected by the system, and with which nodes should the new node communicate? In other words, where should the new node be placed in the hierarchy. There are several options because a new node can be placed in each one of the three layers, and connect to different nodes from the adjacent layers. However, every option has a different impact on the performance of the system. Since fog computing systems are expected to scale massively [9], new compute nodes are likely to join the system frequently. Thus, node discovery is an essential part of fog computing systems.

To address this problem, we analyze the different aspects of a fog computing system that are affected by the manner whereby nodes are discovered and integrated in the system. To this end, the following sections discuss the reason that the node discovery problem affects different aspects of fog computing, and why these aspects are important. Specifically, Section 3.1 discusses fault tolerance, Section 3.2 addresses the potential resource heterogeneity of the nodes, Section 3.3 discusses the importance of proximity awareness, and Section 3.4 addresses scalability. Finally, Section 3.5 presents the research questions that need to be answered in order to address the node discovery problem efficiently.

3.1 Fault Tolerance

In fog computing, some of the participating compute nodes may be unreliable, and might fail unexpectedly at any moment, which can divide a fog computing system into disjoint parts [16], and affect the system's reliability [32]. For this reason, mechanisms for handling node failure become essential. However, this can be especially challenging in fog computing because when a node fails, moving the computations to neighbor nodes or to the cloud, may affect the performance of the system (e.g., might increase the communication latency) [30].

Nevertheless, it is possible to cope with this problem by integrating efficient mechanisms for handling potential future node failures, at the discovery phase, i.e., when a new node joins the system. This can be achieved by having each new node store additional nodes which may not reside in proximity, and are not necessarily used for processing the IoT data, but can be used for maintaining connectivity in case the neighbors fail (cf. Section 4).

3.2 Resource Heterogeneity

Fog computing systems consist of various resource-heterogeneous compute nodes [28]. This means that the participating compute nodes may have very different resource capacities, e.g., regarding CPU and memory, but they may also have different capabilities, e.g., regarding hosted services and applications. This diversity should be taken into account during the discovery phase, because different nodes need to be treated differently. For example, upon discovery, a cloud compute node which is able to provide a huge amount of computational resources should go to the top of the hierarchy. This way, the nodes of lower layers will be able to send the IoT data to that node (for processing) by forwarding the data upwards the hierarchy (cf. Fig 1). On the contrary, a compute node at the edge of the network should be placed close to the IoT devices (cf. Fig 1) in order to leverage on the low communication latency. Therefore, the resource heterogeneity of the compute nodes needs to be considered during the discovery phase in order to ensure the efficient operation of a fog computing system.

3.3 Proximity Awareness

Since processing data in nearby compute nodes improves the communication efficiency [9], fog computing systems leverage on the proximity among the various compute nodes, and the IoT devices, in order to process the IoT data with low communication latency. Most approaches assume that the participating compute node are already discovered and integrated in the system based on proximity (as discussed in Section 1). However, in order to take into account the proximity among the nodes, new nodes need to take proximity measurements (e.g., using round-trip time or hop count), and then connect to the neighbors of the closest proximity.

Taking into account the proximity among the nodes during the discovery phase is a challenging task in fog computing, because proximity measurements may have conflicts with other aspects, e.g., with the resource heterogeneity aspect (cf. Section 3.2). This can happen for instance, upon discovery of a new compute node which integrates a big amount of computational resources, and should be placed in a high layer so that many nodes of lower layers can use these resources. At the same time, this new node may be in the proximity of nodes in lower layers. This means that according to proximity, the new node should be placed in a low layer. Thus, during the discovery phase, there may be conflicts based on the different goals of the discovery problem.

3.4 Scalability

As discussed in Section 1, fog computing systems can include compute nodes that span from the cloud to the edge of the network and thus, they may need to scale to a large degree [9]. This means that during the discovery phase, there can be a huge number of possible positions for a new node. Examining all the possible options means taking proximity measurements for a very large number of potential neighbors. However, this may not be possible since this process generates a considerable amount of network traffic which is part of the overhead of the discovery phase. Furthermore, more messages need to be exchanged in order to discover and store additional nodes for fault tolerance, and in order to examine the resource heterogeneity of the other nodes, as discussed in Sections 3.1 and 3.2. Since generating a significant amount of overhead can compromise the scalability of the system, the overhead of the discovery phase needs to be considered, especially because in fog computing new compute nodes may be discovered at any time [16].

3.5 Research Questions

There are many aspects of fog computing that can be improved by considering the node discovery problem (cf. Sections 3.1 – 3.4). For this reason, and in order to be able to solve this problem efficiently, we identify the following research questions (RQ):

RQ1 To what degree can fog computing systems be fault-tolerant, by storing additional nodes during the discovery phase, which are used in case of node failures?
RQ2 How should the proximity and the resource heterogeneity of the compute nodes, affect the position of a new node that joins a fog computing system?
RQ3 How to make sure that the overhead from new compute nodes joining, does not compromise the scalability of a fog computing system?

When we are able to answer these research questions, then we will be in the position to design efficient discovery mechanisms that aid in improving various aspects of fog computing.

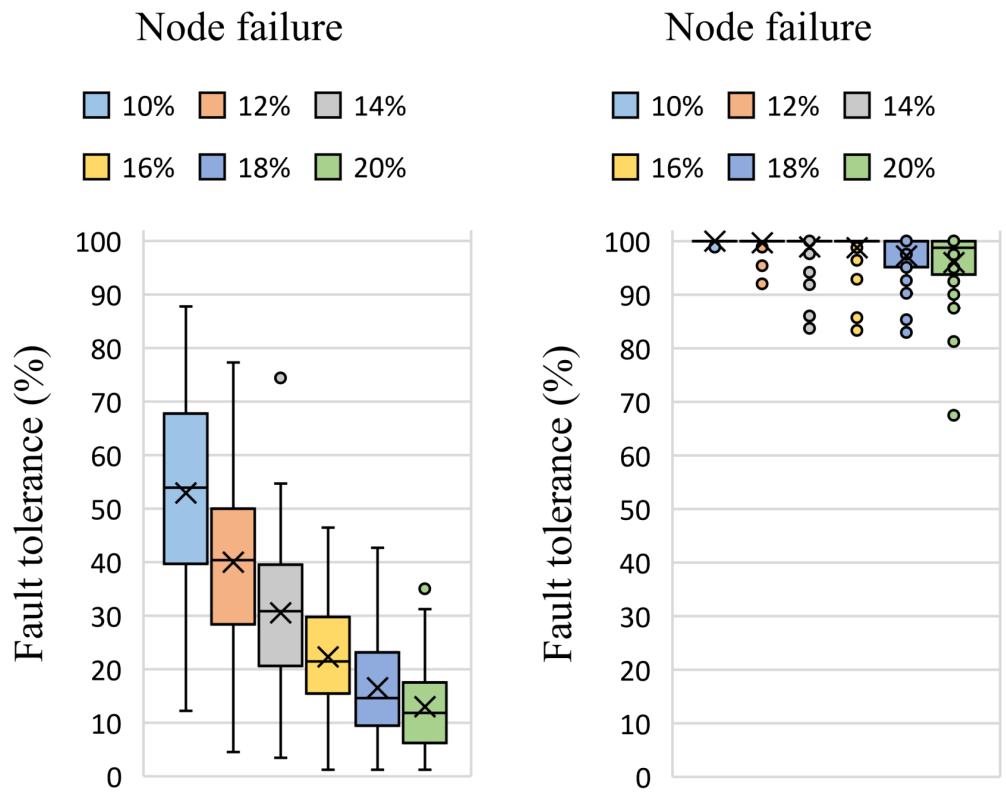
4 Evaluation

In this section, we report the preliminary results of our efforts to tackle the node discovery problem in fog computing. The setup we use in order to produce these results is described in Section 4.1. Afterwards in Section 4.2, we perform various experiments which focus on the fault tolerance aspect of the node discovery problem, and we present our results.

4.1 Evaluation Setup

In order to perform experiments, and examine the fault tolerance of a fog computing system, we have built a simulator using Java. The reason we do not use a simulator developed in the scope of related work from the literature (e.g., iFogSim [10]), is that alternative simulators lack the necessary functionality to address the proposed problem (e.g., compute nodes that fail or become unavailable temporarily).

By using our simulator, we are able to simulate hierarchical fog computing systems consisting of compute nodes that span from the cloud to the edge of the network. The number of the participating compute nodes in these systems is configurable, but the layout is always hierarchical. In the hierarchy, every parent node selects as neighbors up to three children nodes, as shown in Fig. 1. For this evaluation, we perform 50 experiments with



(a) When each compute node stores only nearby neighbors.

(b) When each compute node stores neighbors and additional nodes to be used in case of failures.

Figure 2 Fault tolerance of a fog computing system.

100 nodes. The reason we have selected these specific numbers, is that after experimenting extensively with this simulator, we found these numbers to produce results which can be considered representative of the general case.

In each one of the 50 experiments, we select various percentages of the participating compute nodes to become unresponsive, and then we examine the percentage of the responsive nodes that remain connected. Since node failure can divide a fog computing system into disjoint parts (as discussed in Section 3.1), with this experiment we aim at measuring the fault tolerance of the system. The specific nodes that fail are chosen randomly using the uniform distribution. Using this evaluation setup, we examine two node discovery mechanisms.

In the first mechanism, each new node requests to join from a preexisting node of the system (i.e., a contact node), and stores only nearby neighbors which are found through the contact node. In the second, the new node requests to join through the contact node again, but apart from storing the nearby neighbors, it also stores the neighbors of the contact node. The neighbors of the contact node may not reside nearby so they might not be suitable for processing data with low communication latency. However, these nodes are used in case the other neighbors fail.

4.2 Evaluation Results

In Fig. 2, we show the results of our experiments. For Fig. 2a, the nodes store only neighbors, i.e., using the first node discovery mechanism (cf. Section 4.1). In this experiment, we induce node failure of 10%, 12%, 14%, 16%, 18%, and 20% of the nodes, and we measure the corresponding percentages of the responsive nodes that remain connected. Each box plot includes 50 values from the 50 experiments we have conducted. Notably, the average percentage of responsive compute nodes that remain connected is approximately 53% with 10% node failure, and the fault tolerance of the system decreases, while the percentage of node failures increases.

For Fig. 2b, we repeat the same experiment, but we change the node discovery mechanism. Instead of storing only neighbors (as done for Fig. 2a), in this experiment every node stores additional nodes to be used in case of failures, i.e., the second node discovery mechanism (cf. Section 4.1). Thus, when a responsive node detects (e.g., using heartbeat messages) that the neighbors have failed, this node tries to connect to the system using the additional nodes. Notably, the average percentage of responsive compute nodes that remain connected in this experiment, is approximately 99% with 10% of node failure. Again, the fault tolerance of the system decreases, while the node failures increase although, until the node failures reach 20%, the average fault tolerance remains always above 90%.

Based on Fig. 2a, we note that creating a fog computing system whereby each node stores only its neighbors, is not an efficient approach with regard to fault tolerance. This is claimed because, when various nodes fail, the percentage of remaining responsive nodes which remain connected decreases radically.

However, according to Fig. 2b, we note that the fault tolerance of a fog computing system can be increased significantly, by storing additional nodes during the node discovery phase. Similarly, we believe that addressing the node discovery problem can aid in improving various aspects of fog computing systems, as discussed in Section 3.

5 Conclusion

In this paper, we present the node discovery problem in fog computing systems. To this end, we analyze various aspects of fog computing that can be affected from the way new nodes are discovered and integrated in the system, such as: fault tolerance, resource heterogeneity, proximity awareness, and scalability. Furthermore, we identify related research questions which need to be addressed in order to tackle the proposed problem efficiently. Finally, we simulate fog computing systems, and we perform experiments with various compute nodes which integrate a node discovery mechanism that focuses on improving the fault tolerance of the system. By analyzing the results, we show that when each new node that joins, stores additional nodes during the discovery phase, the fault tolerance of a fog computing system improves significantly.

Due to the promising results, in the future we plan to focus on node discovery mechanisms that improve fog computing systems. Specifically, we plan to design node discovery mechanisms tailored to fog computing systems by considering not only the fault tolerance of the system, but also aspects related to proximity awareness, resource heterogeneity, scalability, and others.

References

1 Hamid Reza Arkian, Abolfazl Diyanat, and Atefe Pourkhalili. Mist: Fog-based data analytics scheme with cost-efficient resource provisioning for iot crowdsensing applications. *Journal of Network and Computer Applications*, 82:152–165, 2017.

2 Paolo Bellavista, Alessandro Zanni, and Michele Solimando. A migration-enhanced edge computing support for mobile devices in hostile environments. In *International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 957–962. IEEE, 2017.

3 Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Workshop on Mobile Cloud Computing (MCC)*, pages 13–16. ACM, 2012.

4 Rustem Dautov, Salvatore Distefano, Dario Bruneo, Francesco Longo, Giovanni Merlino, Antonio Puliafito, and Rajkumar Buyya. Metropolitan intelligent surveillance systems for urban areas by harnessing iot and edge computing paradigms. *Software: Practice and Experience*, 48(8):1475–1492, 2018.

5 Marcos Dias de Assuncao, Alexandre da Silva Veith, and Rajkumar Buyya. Distributed data stream processing and edge computing: A survey on resource elasticity and future directions. *Journal of Network and Computer Applications*, 103:1–17, 2018.

6 Shuiguang Deng, Zhengzhe Xiang, Jianwei Yin, Javid Taheri, and Albert Y Zomaya. Composition-driven iot service provisioning in distributed edges. *IEEE Access*, 6:54258–54269, 2018.

7 Nitin Desai and Sasikumar Punnekkat. Safety of fog-based industrial automation systems. In *Workshop on Fog Computing and the IoT (IoT-Fog)*, pages 6–10. ACM, 2019.

8 Radu Dobrin, Nitin Desai, and Sasikumar Punnekkat. On fault-tolerant scheduling of time sensitive networks. In *Workshop on Security and Dependability of Critical Embedded Real-Time Systems (CERTS)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

9 Pedro Garcia Lopez, Alberto Montresor, Dick Epema, Anwitaman Datta, Teruo Higashino, Adriana Iamnitchi, Marinho Barcellos, Pascal Felber, and Etienne Riviere. Edge-centric computing: Vision and challenges. *ACM SIGCOMM Computer Communication Review*, 45(5):37–42, 2015.

10 Harshit Gupta, Amir Vahid Dastjerdi, Soumya K Ghosh, and Rajkumar Buyya. ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Software: Practice and Experience*, 47(9):1275–1296, 2017.

11 Cheol-Ho Hong and Blesson Varghese. Resource management in fog/edge computing: a survey on architectures, infrastructure, and algorithms. *ACM Computing Surveys (CSUR)*, 52(5):1–37, 2019.

12 Vasileios Karagiannis. Building a Testbed for the Internet of Things. *Alexander Technological Educational Institute of Thessaloniki*, pages 1–92, 2014.

13 Vasileios Karagiannis. Compute node communication in the fog: Survey and research challenges. In *Workshop on Fog Computing and the IoT (IoT-Fog)*, pages 1–5. ACM, 2019.

14 Vasileios Karagiannis, Periklis Chatzimisios, Francisco Vazquez-Gallego, and Jesus Alonso-Zarate. A survey on application layer protocols for the internet of things. *ICAS Transaction on IoT and Cloud Computing*, 3(1):11–17, 2015.

15 Vasileios Karagiannis and Apostolos Papageorgiou. Network-integrated edge computing orchestrator for application placement. In *International Conference on Network and Service Management (CNSM)*, pages 1–5. IEEE, 2017.

16 Vasileios Karagiannis, Stefan Schulte, Joao Leitao, et al. Enabling fog computing using self-organizing compute nodes. In *International Conference on Fog and Edge Computing (ICFEC)*, pages 1–10. IEEE, 2019.

17 Vasileios Karagiannis, Alexandre Venito, Rodrigo Coelho, Michael Borkowski, and Gerhard Fohler. Edge computing with peer to peer interactions: Use cases and impact. In *Workshop on Fog Computing and the IoT (IoT-Fog)*, pages 1–5. ACM, 2019.

18 Roman Kolcun, David Boyle, and Julie A McCann. Optimal processing node discovery algorithm for distributed computing in iot. In *2015 5th International Conference on the Internet of Things (IOT)*, pages 72–79. IEEE, 2015.

19 Yang Liu, Jonathan E Fieldsend, and Geyong Min. A framework of fog computing: Architecture, challenges, and optimization. *IEEE Access*, 5:25445–25454, 2017.

20 Rongxing Lu, Kevin Heung, Arash Habibi Lashkari, and Ali Akbar Ghorbani. A lightweight privacy-preserving data aggregation scheme for fog computing-enhanced iot. *IEEE Access*, 5:3302–3312, 2017.

21 Ivan Lujic, Vincenzo De Maio, and Ivona Brandic. Efficient edge storage management based on near real-time forecasts. In *International Conference on Fog and Edge Computing (ICFEC)*, pages 21–30. IEEE, 2017.

22 Carla Mouradian, Diala Naboulsi, Sami Yangui, Roch H. Glitho, Monique J. Morrow, and Paul A. Polakos. A comprehensive survey on fog computing: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials*, 20(1):416–464, 2017.

23 Ilir Murturi, Cosmin Avasalcai, Christos Tsigkanos, and Schahram Dustdar. Edge-to-edge resource discovery using metadata replication. In *International Conference on Fog and Edge Computing (ICFEC)*, pages 1–6. IEEE, 2019.

24 Carlo Puliafito, Enzo Mingozzi, Francesco Longo, Antonio Puliafito, and Omer Rana. Fog computing for the internet of things: A survey. *ACM Transactions on Internet Technology*, 19(2):18, 2019.

25 Mahadev Satyanarayanan. The emergence of edge computing. *Computer*, 50(1):30–39, 2017.

26 Vitor Barbosa Souza, Xavi Masip-Bruin, Eva Marín-Tordera, Sergi Sànchez-López, Jordi Garcia, Guang-Jie Ren, Admela Jukan, and Ana Juan Ferrer. Towards a proper service placement in combined fog-to-cloud (F2C) architectures. *Future Generation Computer Systems*, 87:1–15, 2018.

27 Nitendra Tomar and Rakesh Matam. Optimal query-processing-node discovery in iot-fog computing environment. In *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 237–241. IEEE, 2018.

28 Luis M Vaquero and Luis Rodero-Merino. Finding your way in the fog: Towards a comprehensive definition of fog computing. *ACM SIGCOMM Computer Communication Review*, 44(5):27–32, 2014.

29 Blesson Varghese, Nan Wang, Sakil Barbhuiya, Peter Kilpatrick, and Dimitrios S Nikolopoulos. Challenges and opportunities in edge computing. In *International Conference on Smart Cloud (SmartCloud)*, pages 20–26. IEEE, 2016.

30 Prateeksha Varshney and Yogesh Simmhan. Demystifying fog computing: Characterizing architectures, applications and abstractions. In *International Conference on Fog and Edge Computing (ICFEC)*, pages 115–124. IEEE, 2017.

31 Riccardo Venanzi, Burak Kantarci, Luca Foschini, and Paolo Bellavista. MQTT-driven node discovery for integrated IoT-fog settings revisited: The impact of advertiser dynamicity. In *Symposium on Service-Oriented System Engineering (SOSE)*, pages 31–39. IEEE, 2018.

32 Zhenyu Wen, Renyu Yang, Peter Garraghan, Tao Lin, Jie Xu, and Michael Rovatsos. Fog orchestration for internet of things services. *IEEE Internet Computing*, 21(2):16–24, 2017.