Router-based Brokering for Surrogate Discovery in Edge Computing

Julien Gedeon, Christian Meurisch, Disha Bhat, Michael Stein, Lin Wang, Max Mühlhäuser
Telecooperation Lab, Technische Universität Darmstadt
Email: {gedeon,meurisch,stein,wang,max}@tk.tu-darmstadt.de

***Abstract*—In-network processing pushes computational capabilities closer to the edge of the network, enabling new kinds of location-aware, real-time applications, while preserving bandwidth in the core network. This is done by offloading computations to more powerful or energy-efficient *surrogates* that are opportunistically available at the network edge. In mobile and heterogeneous usage contexts, the question arises how a client can discover the most appropriate surrogate in the network for offloading a task. In this paper, we propose a brokering mechanism that matches a client with the best available surrogate, based on specified requirements and capabilities. The broker is implemented on standard home routers, and thus, leverages the ubiquity of such devices in urban environments. To motivate the feasibility of this approach, we conduct a coverage analysis based on collected access point locations in a major city. Furthermore, the brokering functionality introduces only a minimal resource overhead on the routers and can significantly reduce the latency compared to distant, cloud-based solutions.**

I. Introduction

The traditional approach to overcome limitations in an individual device's computing power is to offload computational tasks to a distant cloud computing infrastructure [1], [2]. These infrastructures offer vast computational resources at affordable costs. However, cloud computing has issues regarding latency, mobility support, and location awareness [3], [4]. In addition, high network bandwidth utilization and possible security and privacy concerns are inherent to this approach. The new paradigm of in-network processing, also known as *fog computing* [5], [3] or *edge computing* [6], moves these capabilities to the network edge. Here, devices referred to as *surrogates* act as micro-clouds or *cloudlets* [7]. Compared to distant clouds, cloudlets are small-scale, decentralized, and opportunistic, as they can be hosted on any kind of networked device. Other clients can exploit these cloudlets to perform computations.

In order for client devices to find appropriate surrogates, a local and decentralized discovery mechanism is required. The discovery mechanism should introduce only a small latency, which is important for new emerging applications, such as real-time monitoring, emergency response, and augmented reality, where even small delays have a considerable impact on the perceived quality of service. We look at the problem of surrogate discovery in the context of an urban area, where we are faced with a high mobility of devices and users.

As a mechanism to enable the discovery and selection of available surrogates, we propose a novel brokering architecture. Available surrogates advertise themselves to the broker and provide metadata, e.g., on their current state and capabilities. The broker maintains information on all available surrogates in its proximity and matches requests from clients in order to find the best surrogate.

To this end, we implement the brokering functionality on standard home routers, because we argue that they are ubiquitously available and can cover large parts of urban areas. Furthermore, the computing capacities of these devices are underutilized most of the time and current router models often feature multi-core processors as well as extra memory and storage capacities that we can leverage. However, we still need to be careful not to impede their normal functions and therefore our brokering application introduces only a small resource overhead on the routers.

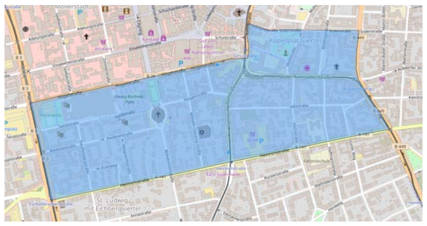
The contributions of this paper are as follows:

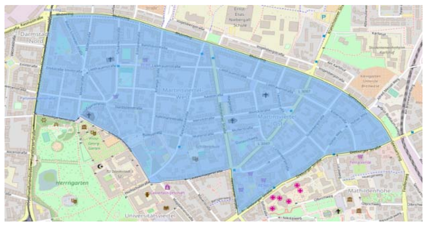
- We propose a novel brokering mechanism that matches client requests with the best available surrogates, given certain capabilities and requirements. Multiple brokers are interconnected using Distributed Hash Tables (DHTs) in order to exchange information.
- We implement the proposed approach on off-the-shelf home routers, introducing only a small resource overhead on the devices.
- Given Wifi access points in a major city, we show that this approach can lead to high coverage, even when only a few routers are available.

The remainder of this paper is organized as follows: Section II provides a case study for the analysis of router coverage in an urban area to further motivate our approach. Section III describes the design of our system. Our implementation is evaluated in Section IV. Section V reviews related work and finally, conclusions and an outlook on future work are given in Section VI.

II. Router Coverage in Urban Areas: A Case Study

The discovery mechanism we propose is implemented on standard home routers. These devices are ubiquitous and a large number of them is available with high density, especially in urban areas. Ideally, this would lead to a high coverage, which is an important metric for the quality of service offered by wireless networks [8] and in our case reflects if a user has a discovery-enabled router in his vicinity. Different aspects can incentivize private users to upgrade their routers in order to provide this added functionality. First, there are already


(a) Area 1


(b) Area 2

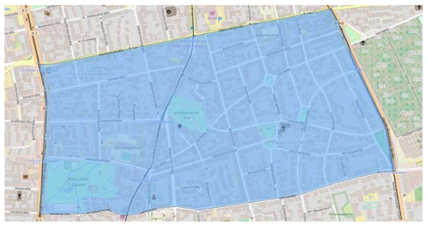

(c) Area 3

Figure 1. Selected areas of the city

Table I
STATISTICS ON THE CITY AREAS

Area	Size in m^2	Number of Routers	Router Density
1	458,814	1127	2.46
2	644,317	1755	2.72
3	1,029,600	2004	1.95

initiatives for people who are willing to share their Wifi connections. We think that providing computational capabilities on the routers is the next logical step. Second, the incentive can also stem from a business perspective, e.g., service providers may control what additional applications are deployed on the router in return for a reduced monthly bill. Therefore to further motivate the use of home routers, we perform an analysis of the coverage that can be achieved when only a subset of potential routers are actually enabled to provide this functionality. To do so, we analyzed real-world data from the city of Darmstadt, Germany. Using an Android application, volunteers walked around the city and collected the signals from Wifi access points. In total, 23,744 data points were captured. The position of the access points is estimated via triangulation from multiple measurements and the RSSI of the discovered access points. We perform some basic filtering on the data, such as the elimination of duplicate MAC addresses. Furthermore, by doing a lookup on the MAC addresses, we eliminate all manufacturers that do not produce routers. It is important to note, that while this data might include some wrong data and uncertainty regarding the exact position of the access points, it still gives a good impression of how many potential routers could be available. More importantly, the access points were collected while walking through the city and not inside buildings or private locations, therefore reflecting the same usage context a mobile user who wishes to perform opportunistic offloading has. Because the usage patterns of our collection app were not uniform throughout the city, we restrict our measurements to three areas of the city. Figure 1 shows these areas within the city. Table I summarizes information about the selected areas in terms of their size and the number of routers present after filtering. It also gives an indication of the density of routers, measured in their number per 1,000 square meters.

Given this data, we perform a coverage analysis as follows: For each percentage value from 1 to 100, we randomly select the respective number of routers that equals that percentage.

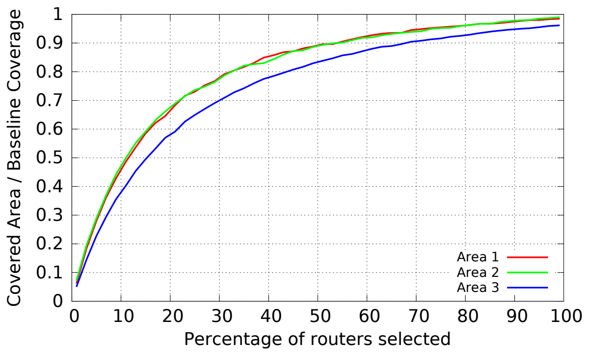


Figure 2. Evaluation of the coverage

We assume a uniform communication range for each router and set the radius of that range to be 30 meters, as we think this is a realistic and conservative estimation. We calculate the area of the union of all sensing ranges and are therefore able to compute the coverage that is achieved, given by the ratio between the union of the communication ranges of the selected routers (i.e., the area that is covered by that selection of routers) and the total size of the area (i.e., our baseline coverage). Because we choose the routers randomly, we run each experiment 20 times and average the results, although the standard deviation is very low, with a maximum of 0.018. The results of our analysis can be seen in Figure 2. For Areas 1 and 2, only about 30 percent of all routers are necessary to achieve a coverage of close to 80 percent. For Area 3, this is a bit lower (70%), since the density of routers is smaller. Nevertheless, in this area we can also achieve 80 percent coverage with just about 45 percent of all total routers. For all Areas, 70 percent of the routers are sufficient to get over 90 percent coverage relative to the size of the area. We therefore conclude that this approach could be viable in practice, i.e., with a relatively small number of upgraded routers we can reach a high coverage.

III. SYSTEM DESIGN

In this section, we describe the design of our system, its components, the brokering mechanism that determines the best surrogate for a given client request and an implementation thereof on home routers. Figure 3 shows the general architecture of our system and the communication paths

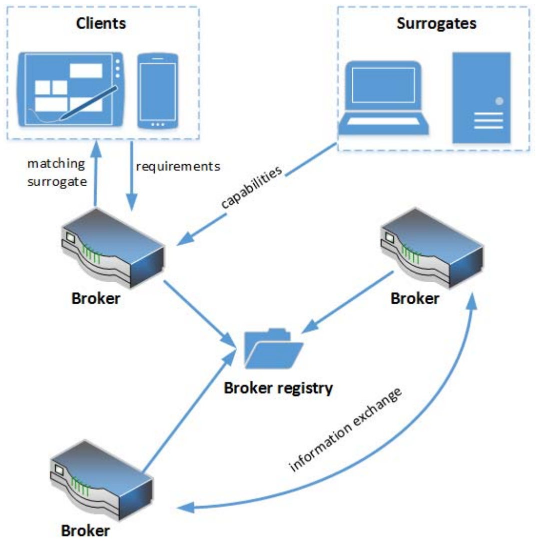


Figure 3. System Design

between entities. To exchange messages, we use Protocol Buffers[1] as a lightweight message format. In the following, we will describe the different entities present in our system and explain their role.

Client Clients are the devices that wish to perform offloading of computations, either because they do not have the processing capabilities or in order to save battery, as to prolong their lifetime.

Surrogate Surrogates are devices that have extra computational power available and can therefore be used by clients to offload computations. It is important to note that surrogates can be various kinds of devices and, thus, are heterogeneous regarding their capabilities.

Broker The broker is the main entity of our system. It maintains a local directory of all surrogates that have advertised themselves to the broker. A heartbeat mechanism ensures that the broker periodically checks whether a surrogate is still available. Whenever a client issues a request to the broker, it returns the best available surrogate for the request. This mechanism is described in detail in Section III-A.

Registry Server All brokers in the system register themselves to a central registry server. The registry server is required to implement parts of our distributed approach, which is discussed in detail in Section III-B. While the registry server might introduce a single point of failure, it is required for the management of the routers (e.g. in case one router becomes unavailable). In a real-world deployment scenario, this registry server could be managed by service providers, guaranteeing a high availability.

[1]https://developers.google.com/protocol-buffers/

A. Surrogate Selection

The main functionality the broker provides is to select the best available surrogate for an incoming client request. To do so, the broker considers a number of attributes that surrogates and clients send as capabilities and requirements, respectively. The attributes we consider can be categorized as follows:

Network information This includes information about the connectivity of the devices, such as their supported network type and the available bandwidth.

Hardware capabilities These are the hardware properties of the devices, such as CPU speed, memory and storage capacity.

Distance In some cases, the physical distance between the devices and the broker might be known and for use cases like processing locally relevant information or sharing data, this might be considered to make the offloading decision.

When a client request arrives at the broker, the following steps are performed:

I. First, the available network connectivities of the client and the surrogate are compared. Clients and surrogates might use different network interfaces (e.g. Wifi, Wifi Direct, Bluetooth, cellular). All surrogates that do not provide at least one common interface with the client are discarded.

II. For each of the remaining surrogates, a score is computed. This score reflects how well a surrogate is able to match the requirements of the client. For each of the attributes, a surplus capability is computed. In addition, for each capability a client can provide a weight, indicating the importance of the respective attribute. The surrogate with the highest score (i.e., the sum of all weighted surplus capabilities) is then selected and its connection information returned to the client as the best match. The client can then initiate a connection to the selected surrogate and perform the desired offloading. We can also define this in a more formal way: Given a set C of clients, a set S of surrogates and n different attributes of which we consider the surplus score. Then, the function $\lambda : C \rightarrow S$ computes the best available surrogate as follows: $\lambda(c) = \arg\max_{s \in S}(\sum_{i=1...n} \alpha_i \cdot \beta_{isc})$, where α_i is the weight and β_{isc} the surplus capability of the surrogate $s \in S$ for the attribute i given a request from the client $c \in C$.

III. When a surrogate is selected for offloading, the broker updates the information about the selected surrogate, i.e., resources that are now consumed by the client are deducted from the initial capabilities.

B. Distributed Brokers

While a single broker is able to provide the desired functionality to devices locally connected to it, we consider potential application scenarios where both clients and surrogates are mobile and, thus, are likely to be connected to one given broker for only a short amount of time. This of course limits the viability of using only an isolated single broker, as information about a surrogate will only be available on the broker it has registered to. We therefore extend our approach to interconnect multiple brokers. Connected brokers exchange information about the surrogates registered to them and can send updates when this data has changed. A naive way would be to flood

Table II
HARDWARE SPECIFICATIONS OF ROUTERS

Model	Platform	Target Toolchain	Instruction Set	CPU Clock Rate	Flash Memory	RAM
Netgear R7500	Qualcomm Atheros IPQ8064	ipq806x	ARMv7	2x1400 MHz	128 MB	256 MB
Linksys WRT 1900AC v1	Marvell Armada XP MV78230	mvebu	ARMv7	2x1200 MHz	128 MB	256 MB
Linksys WRT 3200ACM v1	Marvell Armada 385 88F6820	mvebu	ARMv7	2x1866 MHz	256 MB	512 MB

the messages to all the brokers, however, this introduces a high message overhead in the network. Instead, we use an approach based on Distributed Hash Tables (DHT), more specifically, we implement the Chord protocol [9]. To do so, we need a registry server, to which each broker registers. The registry server computes the finger table for each broker that determines to which other brokers updates should be sent. The entries for the finger table are computed based on a unique ID, generated by hashing the broker's MAC address. Once the broker has received its finger table from the registry server, it then sends updates to the other brokers contained in its finger table. Similarly, brokers receive updates from other brokers in whose finger tables they are contained. This approach also satisfies some desirable principles of distributed systems in general, such as transparency to the client (regardless to which of the brokers the available surrogates have connected) or failure resilience (in case of failure of one broker, others maintain the latest state information). To summarize, the distributed broker approach allows us to efficiently exchange information about locally registered surrogates.

C. Devices

As motivated in Section II, we want to run the brokering functionality on standard home routers. While nearly all of these devices come with a pre-installed, vendor-specific operating system, projects like OpenWrt[2] provide an open alternative. With the appropriate toolchain, we are therefore able to develop own applications for the devices. To test our approach, we chose three routers with hardware specifications shown in Table II. We can clearly see that modern routers for private use are equipped with enough resources to perform other tasks besides routing. All the routers feature a dual-core ARM-based CPU and at least 256MB of RAM. They also have extra flash storage available, allowing the deployment of custom applications. As operating system we chose the latest development snapshot of OpenWrt, except for the Linksys WRT 3200ACM, which is not yet supported. Instead, we used LEDE[3], a fork of OpenWrt with a strong community support.

IV. EVALUATION

In this section, we analyze the viability of our approach with respect to latency, the performance and resource overhead on the devices and the benefits of our distributed approach.
First, it is crucial for our brokering application to introduce only a small resource consumption on the routers, as to not impede their normal function. This also serves as a motivation

[2]https://openwrt.org
[3]https://lede-project.org

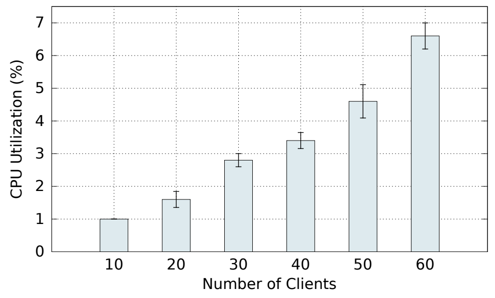


Figure 4. CPU utilization

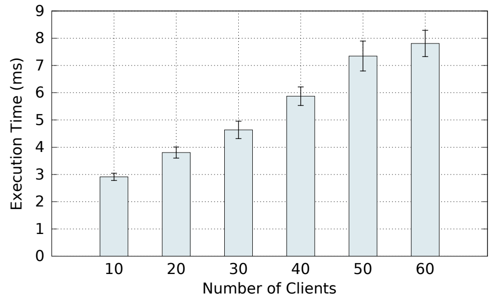


Figure 5. Execution time

to make one's home router available. Figure 4 shows the CPU utilization on the Netgear R7500 when a fixed number of 50 surrogates are registered to the broker. We measure the CPU utilization in percent when a varying number of parallel clients request for a surrogate and perform 5 experiments in total. From the graph, we can see that even with a high number of 60 client requests, the CPU utilization is under 7 percent. Given the hardware specifications of the device (cf. Table II), this is a very low utilization and ensures the owner is still able to use the device as originally intended. The memory utilization on the routers is negligible with a near-constant 2%. Next, we take a look at the execution time, i.e., the time the broker needs to compute the best surrogates for client requests. Again, we assume that 50 surrogates have advertised themselves to the broker. Figure 5 shows the execution time for a varying number of threaded client requests. Even with 60 clients, the execution time is under 9 ms. In practice, we can fairly assume this number to be much less. To put this into perspective, with up to 20 client requests, the execution time remains below 4 ms. Our main motivation for deploying the brokering mechanism on close-by routers are the benefits in terms of latency we can achieve compared to a distant,

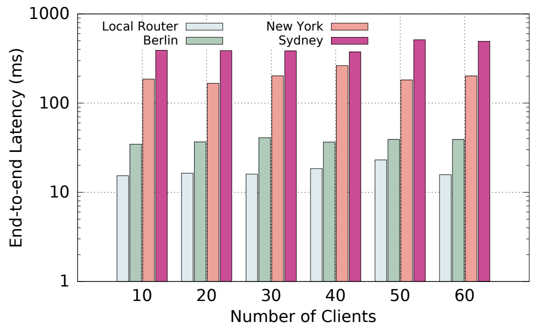


Figure 6. Latency measurements

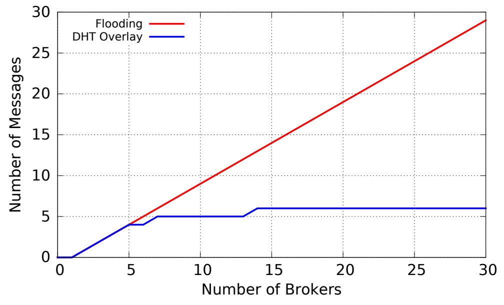


Figure 7. Message overhead

cloud-based broker. To show the benefit of our approach, we compare the end-to-end latency in Figure 6, depending on where the brokering functionality is deployed. Clearly, our approach outperforms the other scenarios, where the brokering functionality is placed on distant servers, either in Berlin, New York or Sydney. The beeline distances to these places from our location are 443 km, 6218 km and 16,500 km. To evaluate our distributed broker approach, we first look at the benefit of using a DHT overlay instead of flooding the update messages to all brokers in the network. Given a single round of update messages, Figure 7 shows how many messages are sent using either flooding or a DHT overlay for a varying number of brokers from 1 to 30. For flooding, the number of messages equals $n-1$, where n is the number of brokers. Using a DHT, n is equal to the number of other brokers a broker has in its finger table. On average, this results in 80% less messages being sent compared to flooding. Our distributed approach furthermore enables to find surrogates that have registered to other brokers than the client is connected to. This is made possible by the exchange of surrogate information between the brokers. Without this, a given client request might not be satisfied by the locally available surrogates, and therefore would have to be rejected by the broker. We compare the number of rejections in an example setup of 10 brokers with numbers of surrogates and clients as listed in Table III, each of them having different requirements and capabilities. Figure 8 shows the client rejections for the 10 brokers, i.e., how many client request have to be rejected in each approach. Overall,

Table III
EXPERIMENTAL SETUP FOR THE COMPARISON OF THE REJECTION RATE

Broker #	Number of surrogates	Number of clients
1	10	20
2	7	15
3	5	10
4	2	5
5	3	10
6	8	15
7	1	5
8	4	10
9	6	15
10	9	20

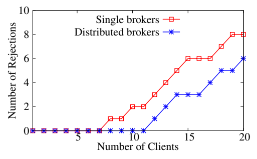
in this setup, we could decrease the number of rejected clients by 57%.

V. Related Work

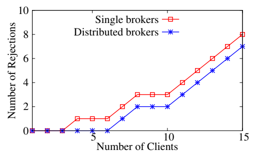
The problem of service discovery has been studied extensively since the proliferation of distributed computing. In the context of surrogate discovery for cyber foraging, according to [10], we can distinguish between *directory-based* and *non directory-based* discovery mechanisms. The latter eliminate the need to maintain an explicit list of known surrogates [11], [12]. Directory-based approaches can either be *local*, i.e., the available surrogates are stored on mobile devices [13], [14] or *remote*, i.e., the available surrogates are maintained in a distant repository [15], [16]. Existing approaches however typically operate in less dynamic environments. We consider scenarios with a high mobility of both clients and surrogates and introduce a way to exchange information about available surrogates in a distributed manner in order to provide a high number of potential surrogates for offloading. Leveraging standard home routers has been examined previously [17], [18], however, this existing work uses the devices as a computing entity, which depending on their capabilities might not lead to a high performance. We instead propose to use routers as proxies to find appropriate surrogates in an urban usage context. Other work has examined the ideal placement of access points from the user perspective [19]. We however assume we cannot control the placement of routers and analyze real-world data collected in a major city to estimate the coverage that can be achieved.

VI. Conclusion and Future Work

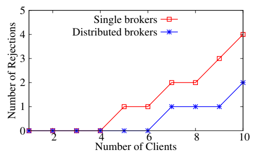
In this paper, we proposed a brokering system that enables clients to find suitable surrogates for computational offloading. This is based on the use case of in-network processing, where highly mobile users need to discover close-by surrogates they can leverage. Based on metadata from both the surrogates and the clients that describe their capabilities and requirements, the broker finds the best available surrogate for a given client request. Furthermore, we enable the interconnection between multiple brokers through a DHT overlay. To demonstrate our approach, we implement the brokering application on standard home routers, thus leveraging existing devices that are ubiquitously present in urban areas. We have shown that



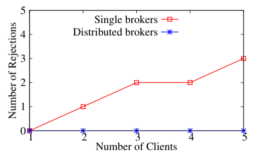
(a) Broker #1



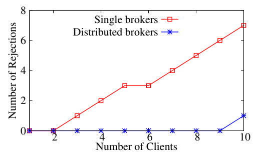
(b) Broker #2



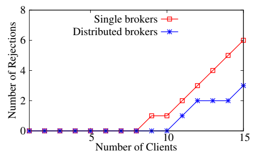
(c) Broker #3



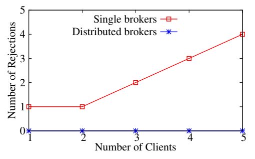
(d) Broker #4



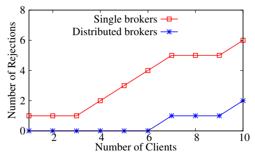
(e) Broker #5



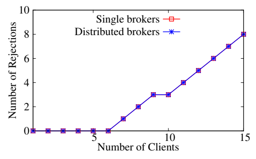
(f) Broker #6



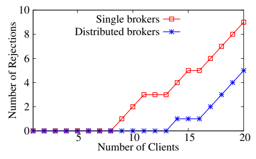
(g) Broker #7



(h) Broker #8



(i) Broker #9



(j) Broker #10

Figure 8. Number of rejected client requests on the brokers

placing the brokering functionality on routers that are close to the end users is beneficial in terms of the network latency. In addition, our approach introduces only a small overhead on the devices and therefore does not impede their normal function. The analysis of Wifi access points collected in a major city has shown that even if only a small number of people are willing to upgrade their devices in order to provide this added functionality, we can achieve a high coverage.

In future work, we will examine the role of brokers as proxy nodes to relay data between clients and surrogates. Based on its own resource availability and processing power, a router also might opportunistically decide to perform certain computations on its own if it can meet the client's requirements, thus eliminating the need for a client-surrogate connection and further lowering the end-to-end latency. Regarding the coverage analysis, instead of only considering the locations of access points we plan to correlate them with mobility traces from users, thereby giving us an even more accurate estimation of the coverage this approach can achieve.

Acknowledgement

This work has been cofunded by the German Federal Ministry for Education and Research (BMBF, Software Campus project DynamicINP, grant no. 01IS12054), by the German Research Foundation (DFG) as part of the Collaborative Research Center (CRC) 1053 - MAKI and by the LOEWE initiative (Hessen, Gemany) within the NICER project.

References

[1] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: Elastic execution between mobile device and cloud," in *Proceedings of the Sixth Conference on Computer Systems*, ser. EuroSys '11. ACM, 2011, pp. 301–314.

[2] H. Flores, P. Hui, S. Tarkoma, Y. Li, S. Srirama, and R. Buyya, "Mobile code offloading: From concept to practice and beyond," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 80–88, 2015.

[3] S. Yi, C. Li, and Q. Li, "A Survey of Fog Computing: Concepts, Applications and Issues," *Proceedings of the 2015 Workshop on Mobile Big Data - Mobidata '15*, pp. 37–42, 2015.

[4] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog Computing: Platform and Applications," *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, pp. 73–78, 2015.

[5] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog Computing and Its Role in the Internet of Things," *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13–16, 2012.

[6] A. Chandra, J. Weissman, and B. Heintz, "Decentralized edge clouds," *IEEE Internet Computing*, vol. 17, no. 5, pp. 70–73, 2013.

[7] M. Satyanarayanan, P. Bahl, R. Cáceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.

[8] B. Wang, "Coverage problems in sensor networks: A survey," *ACM Computing Surveys*, vol. 43, no. 4, pp. 32:1–32:53, Oct. 2011.

[9] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, Aug. 2001.

[10] G. A. Lewis and P. Lago, "A catalog of architectural tactics for cyber-foraging," in *Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures*, ser. QoSA '15. ACM, 2015, pp. 53–62.

[11] S. Ou, K. Yang, and L. Hu, "Cross: A combined routing and surrogate selection algorithm for pervasive service offloading in mobile ad hoc environments," in *Proceedings of the IEEE global telecommunications conference (GLOBECOM '07')*, Nov 2007, pp. 720–725.

[12] K. Yang, S. Ou, and H. H. Chen, "On effective offloading services for resource-constrained mobile devices running heavier mobile internet applications," *IEEE Communications Magazine*, vol. 46, no. 1, pp. 56–63, Jan 2008.

[13] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, *Cuckoo: A Computation Offloading Framework for Smartphones*. Springer, 2012, pp. 59–79.

[14] R. K. Balan, M. Satyanarayanan, S. Y. Park, and T. Okoshi, "Tactics-based remote execution for mobile computing," in *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services*, ser. MobiSys '03, 2003, pp. 273–286.

[15] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *2012 Proceedings IEEE INFOCOM*, March 2012, pp. 945–953.

[16] Y. Xiao, P. Simoens, P. Pillai, K. Ha, and M. Satyanarayanan, "Lowering the barriers to large-scale mobile crowdsensing," in *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*, ser. HotMobile '13, 2013, pp. 9:1–9:6.

[17] C. Meurisch, A. Seeliger, B. Schmidt, I. Schweizer, F. Kaup, and M. Mühlhäuser, "Upgrading wireless home routers for enabling large-scale deployment of cloudlets," in *International Conference on Mobile Computing, Applications, and Services*. Springer, 2015, pp. 12–29.

[18] P. Liu, D. Willis, and S. Banerjee, "Paradrop: Enabling lightweight multi-tenancy at the network's extreme edge," in *2016 IEEE/ACM Symposium on Edge Computing (SEC)*, Oct 2016, pp. 1–13.

[19] E. Bulut and B. K. Szymanski, "Rethinking offloading wifi access point deployment from user perspective," in *2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct 2016, pp. 1–6.