Analyzing the Deployment Challenges of Beacon Stuffing as a Discovery Enabler in Fog-to-Cloud Systems

Zeineb Rejiba*, Xavier Masip-Bruin*, Eva Marín-Tordera*
*Advanced Network Architectures Lab (CRAAX), Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
{zeinebr, xmasip, eva}@ac.upc.edu

***Abstract*—In order to meet the needs of emerging IoT applications having tight QoS constraints, new computing paradigms have been proposed, bringing computation resources closer to the edge of the network, where IoT resides. One of such paradigms is Fog-to-Cloud (F2C), defined as a framework where the combined use of fog and cloud resources is coordinated and managed in an optimized manner to achieve the desired service requirements. Unlike cloud computing, the fog provides a heterogeneous set of resources, possibly within fixed deployments provided by city managers, or that could even be contributed by end users. This brings in many challenges yet to be addressed such as resource discovery, which is the focus of this paper. This paper digs into the utilization of 802.11 beacon stuffing as a possible solution allowing the discovery of nearby devices in an F2C system, particularly dealing with specific design and implementation details of the proposed solution and more importantly its real applicability to an F2C system through the analysis of several experiments carried out on a real world testbed.**

***Index Terms*—fog computing, F2C computing, resource discovery, device discovery, 802.11 beacon stuffing**

I. Introduction

New IoT-based services are gaining growing interest from the general public and businesses alike, resulting in higher QoS expectations from the customers on the one hand, and increasing the pressure on the current networking infrastructure on the other hand. As conventional cloud computing fails to cope with such requirements, fog computing [1] has recently emerged as a new computing paradigm, bringing computation resources to the edge of the network, i.e., closer to data sources and end users. The adoption of fog computing does not exclude the use of cloud resources, though. Instead, both paradigms should be used in conjunction, to optimally serve application demands. This is the main rationale behind the Fog-to-Cloud (F2C) concept [2]. As shown in Fig. 1, besides the conventional cloud layer, two main fog-level entities are proposed in a F2C system. More specifically, any device with enough capacities to contribute to the F2C system becomes an agent and distinct agents are set into areas managed by one agent that is selected to serve as a leader. The policies to decide the clustering into areas and the leader selection are currently ongoing work.

One of the top concerns in such a novel architecture is an efficient control, management and coordination of the resources spanning different layers of the hierarchy. To that end, a suitable resources discovery mechanism has to be

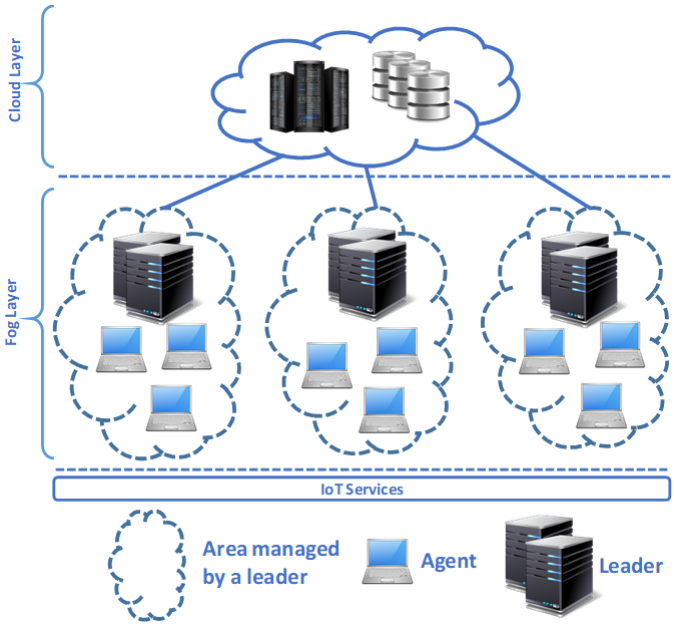

Fig. 1. F2C Architecture

implemented, to make such resources available for use by the envisioned applications, whenever needed. More specifically, there should be a mechanism allowing an agent in the F2C architecture to discover a leader's presence in its vicinity in order to be able to contribute resources to it. This task is not trivial, though, given that an agent and a leader do not initially belong to a common domain, unlike resources in a datacenter in a cloud computing environment or a mobile device and a cloudlet which belong to the same WLAN.

That is why, discovery solutions generally found in the literature such as using zero configuration (zeroconf) techniques (Multicast DNS in conjunction with DNS Service Discovery) [3] or centralized directory-based solutions [4] may fail in addressing the requirements brought by the inherent fog characteristics.

As a result, 802.11 beacon stuffing, and more specifically embedding of discovery information within the beacon's Vendor-Specific Information Element (VSIE) has been already proposed [5] as a potential solution allowing an agent to discover the presence of a nearby F2C leader, even when no

978–1–5386–1478–5/18/$31.00 © 2018 IEEE

prior knowledge about such a presence was acquired.

In this paper, we leverage the work in [5] and contribute in two directions as follows:

- we propose design and implementation directions for a discovery solution in F2C systems.
- we conduct a set of experiments in a real testbed environment to validate the whole solution.

The remainder of this paper is organized as follows. Section II surveys related work in the areas of edge-based resource discovery. Section III describes the design details that have been adopted in the proposed solution. Section IV presents the used experimentation environment and the obtained discovery times and power consumption. Furthermore, the pros and cons of the use of the proposed approach along with the research opportunities it brings are discussed in Section V. Finally, Section VI concludes the paper.

II. Related work

As fog/edge computing paradigms gain in popularity, increasing attention is being drawn to the open issues, which are generally associated with them. Within this context, recent fog/edge computing studies started to highlight the need for adequate resource discovery mechanisms that comply with fog-specific requirements. For instance, authors in [6] state that the existing methods used in the cloud will not be practical for the discovery of edge nodes. Within such a decentralized setup, the use of manual discovery mechanisms is discarded, as it will not scale to the number of devices at the edge. The used discovery mechanism needs to ultimately contribute to a seamless integration of nodes spanning different levels of the hierarchy while guaranteeing the expected user experience. Aligned to our vision, authors in [7] point out that the distribution of resources over multiple entities and locations requires suitable discovery and monitoring schemes to help in the orchestration of resources in a decentralized setup. In [8], FogOS is proposed as a computing architecture for IoT services. Within this architecture, discovery of edge devices may be proactive or reactive. Following the proactive option, when an edge device enters a FogOS-administred network, it notifies FogOS of its intention to join. However, in a reactive scheme, edge resources are queried on demand, resulting in an up-to-date resource information. Authors indicate that a proactive scheme is preferred in a large scale deployment as it provides faster responses. However, the authors do not specify how each of the approaches could be implemented in a real scenario.

In [9], authors propose Edge-as-a-Service (EaaS) as a platform allowing edge nodes to offer their services to other user devices in their proximity through a master node (located either at the edge or elsewhere). The latter is listening on a specific port, allowing edge nodes willing to contribute services to discover it. Since the IP and the port number of the master node are assumed to be known in advance to the edge node, this approach does not seem practical given its static nature that may fail to cope with a dynamic edge environment. Authors in [3] propose a cloudlet discovery mechanism based on Zero Configuration Networking, where a client wishing to discover a cloudlet submits a DNS-SD query to the cloudlet's multicast address. The cloudlet then replies with its IP address and port number. This context is different than the envisioned F2C scenario in that we do not consider F2C devices to be previously connected to the same local network. Pytos is proposed in [4] as a framework for mobile computation offloading. In Pytos, a cloud server is expected to host a repository of available cloudlets for each location. Therefore, each time the client needs to discover available cloudlets, it queries the cloud-side server. This approach is not edge-centric in the sense that cloud servers are still used, thus maintaining the burden on the core network. Additionally, as the requests are always serviced from the cloud, this approach does not consider the users' mobility at the edge.

Therefore, observing a lack of implementations of discovery solutions that meet the requirements of fog/edge computing systems, our goal was to design and implement such a feasible solution with an edge-centric design in mind and to gradually improve it. To this end, in [5], we proposed leveraging the widespread presence of the Wi-Fi technology in modern devices and more specifically their capacity to carry custom information within 802.11 beacon frames, to allow nearby devices in a Fog-to-Cloud (F2C) setup to become aware of each other, thus coining the name F2C-Aware. This strategy is based on the beacon stuffing concept defined in [10], where custom information could be embedded in either the Service Set Identifier (SSID), Basic Service Set Identifier (BSSID) or the Vendor-Specific Information Element (VSIE) fields of the beacon frame (depicted in Fig 2). The IEEE 802.11 standard specifies the VSIE field in the beacon frame in order to carry customized information. Therefore, our solution uses this option to advertise F2C-related information.

III. Design directions

As pointed out in Section 1, a key challenge in an F2C system deals with leaders' detection. More in detail, an agent (device, see Fig. 1) in the fog, wanting to contribute resources to the F2C system, has to first detect an F2C leader to be able to join a fog area and to interact with the system, at the edge level. To achieve this, embedding F2C-related discovery information within VSIEs in 802.11 beacons to be sent by leaders as advertisements is used, thus allowing agents to become aware of the existence of leaders in their vicinity.

As dictated by the standard VSIE format (shown in Fig. 2), the first byte represents the information element ID, which should be 221. It is followed by one byte to represent the length of the subsequent fields. Then, the Organizationally Unique Identifier (OUI) is included (in our case FF:22:CC). Finally comes the actual vendor-specific content, where up to 252 bytes of information can be carried. We represent this information as a list of consecutive TLVs (Type-Length-Value).

Table I provides the proposed hexadecimal encodings for the list of TLV attributes defined in [5]. This list could eventually be extended if such a need arises.

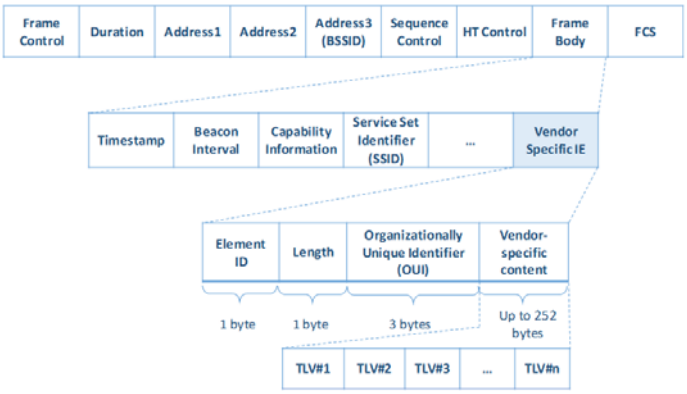

Fig. 2. Structure of a 802.11 beacon frame including the proposed use of TLVs

TABLE I
USED TLV TYPES

TLV type	Hexadecimal Value
Leader ID	0x01
Service type	0x02
Urgency level	0x03

Within a TLV attribute, the first byte represents the type of the attribute being advertised (second column of Table I), the second byte represents the length in bytes of the attribute's value. The latter constitutes the last part of the TLV.

Leveraging the work in [5], we propose a list of hexadecimal codes to encode the different values of each attribute, thus perfectly fitting within one single byte of the TLV format. Table II and Table III list the proposed encodings for the "Service type" attribute and the "Urgency level" attribute, respectively. These lists could be extended, as well.

TABLE II
SERVICE TYPE CODES

Service type	Hexadecimal Code
High storage	0x01
Medium storage	0x02
Low storage	0x03
High bandwidth	0x04
Medium bandwidth	0x05
Low bandwidth	0x06
High processing	0x07
Medium processing	0x08
Low processing	0x09

TABLE III
URGENCY LEVELS CODES

Urgency level	Hexadecimal Code
High	0x01
Medium	0x02
Low	0x03

IV. EVALUATION RESULTS

In this section, we show through real-world experiments a performance evaluation of the proposed F2C-Aware discovery solution. We further compare the F2C-Aware solution (Solution A), in terms of discovery times and the energy consumption, to two solutions that address the discovery problem in the cloudlets field, as an example of an edge-centric environment having similarities with the envisioned F2C system. The first one is based on Zero Configuration Networking (as in [3], solution B) while the second one relies on the use of a centralized directory hosting resource information (as in [4], solution C). It is worth highlighting that although the compared approaches are based on different layers of the networking stack and consider assumptions different than ours (e.g. LAN connection already established), they only constitute a baseline for comparison.

In [5], we developed a limited implementation of the proposed discovery strategy on standard Linux machines, both from the leader's side and the agent's side, and that was restricted to a static scenario. However, this paper extends the implementation in [5] by also considering mobile devices (running the Android OS) as agents as they require special attention to efficiency aspects given their battery limitations and resource-constrained nature.

In the following, we provide an overview of the tools used to create an experimentation environment for each of the 3 considered solutions. Then, we present and analyze the obtained results.

A. Experimentation environment

In order to advertise F2C-specific discovery information within 802.11 beacons, we use the vendor_elements field in the hostapd configuration file (see [5]). This is done through a Python script that takes as an input a configuration file where a list of F2C attributes have been defined. The script then encodes those attributes into the proper hexadecimal encoding expected in the vendor_elements field. Finally, it adds this field in the hostapd configuration file along with other parameters and starts the broadcast. The default 100ms beacon interval is used. From the mobile device side, as information elements are not included within the scan results provided by Android's Wi-Fi API, the "nlscanner" binary developed by [11][1] has been used. Its role is to pass information elements from kernel space to user space without requiring root privileges. Such privileges are needed for newer Android versions, though. Both processes are depicted in Fig 3. One discovery call is performed per minute (therefore one underlying Wi-Fi scan is performed every minute). This scan frequency could be increased in certain contexts in order not to miss leader finding opportunities, or reduced in others where it is less likely to find leaders. However, we do not address this specific question in this paper.

In order to create a test app allowing the discovery of services advertised by other devices on the same local network, i.e. to mimic the discovery proposed in [3], we used the open source library RxDNSSD[2], since we experienced unexpected crashes using Android's Network Service Discovery (NSD). On the other hand, we used Python zeroconf library[3] to advertise a leader service "_leader._tcp" on the same wireless local network.

[1]https://github.com/lows/lows
[2]https://github.com/andriydruk/RxDNSSD
[3]https://github.com/jstasiak/python-zeroconf

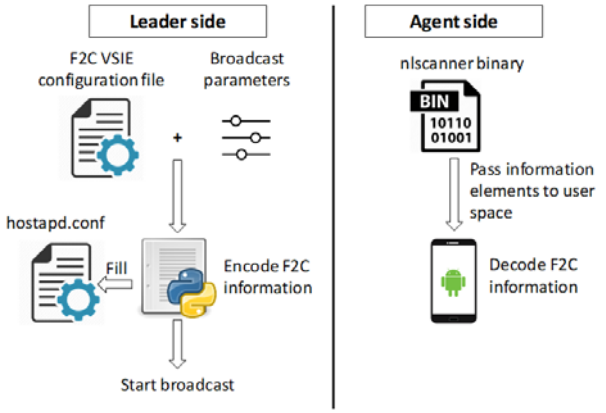

Fig. 3. F2C-Aware implementation details

Finally, in order to emulate a centralized directory-based discovery solution (as in [4]), we assume the mobile device periodically queries a cloud-based directory server for newly-registered resources. To obtain such a behavior, we simply instruct the device to perform an HTTP GET request every minute to a web server hosted within Heroku[4] cloud service. The mobile device is provided with an Internet connection via Wi-Fi during the test.

Note that for these 3 setups, we use a smartphone running Android 4.1.2 and a laptop running Ubuntu 16.04 and acting as a leader.

B. Discovery times

The discovery time is an important indicator of the efficiency of a given discovery solution especially for time-sensitive fog computing applications where discovery is performed reactively at runtime.

Besides the time natively consumed by the discovery process, other network-related operations also affect the discovery timing performance, such as the WLAN discovery (consisting in the scan time and generally taking 1,560 ms in an Android device according to [12]) and WLAN association (generally lasting 100ms [13]), which are necessary to make discovery possible. Fig 4 illustrates how these aspects affect the overall discovery process.

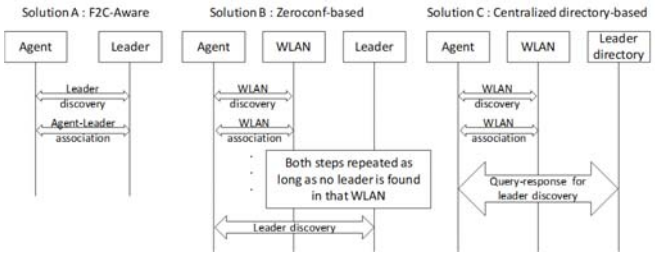

Fig. 4. Comparison of the considered discovery processes

In fact, in F2C-Aware, "Leader discovery" represents the delay between the time at which the device starts listening for beacons and the time when the content of the received F2C beacon is correctly decoded. The use of beacons implies that Leader discovery and WLAN discovery are inherently merged into a single step. The "Agent-Leader association" occurs afterwards and does not affect the discovery process. Solutions B and C have the phases of WLAN discovery and WLAN association in common as they are a prerequisite for the discovery. However, in Solution B, those two phases could occur consecutively multiple times until a leader is discovered within a WLAN, thus potentially impacting the overall timing and power consumption. Finally, in solution C, the time between when the request was issued and the time at which the response was received is considered.

Table IV shows the native discovery times for different runs of the three considered approaches.

TABLE IV
DISCOVERY TIMES

	F2C-Aware	Zeroconf-based	Centralized directory-based
Run #1	854 ms	141 ms	717 ms
Run #2	832 ms	119 ms	158 ms
Run #3	866 ms	162 ms	888 ms
Run #4	830 ms	109 ms	259 ms
Run #5	837 ms	575 ms	117 ms
Run #6	863 ms	122 ms	673 ms
Run #7	873 ms	217 ms	97 ms
Run #8	858 ms	214 ms	187 ms
Run #9	916 ms	234 ms	254 ms
Run #10	835 ms	251 ms	333 ms
Average	**856.4 ms**	**214.4 ms**	**368.3 ms**

As it can be seen, we obtained a discovery time of 856.4 ms on average for our proposal, which is quite reasonable given that an underlying 802.11 scan is performed to enable discovery. Authors in [12] have checked the Android source code and found that the dwelling time on each Wi-Fi channel is set to 120 ms, consequently resulting in an overall 1,560 ms delay in order to go through the 13 consecutive channels of the 2.4 GHz spectrum. As such, the obtained discovery time falls within the standard range. In addition, it is worth mentioning that the use of the Wi-Fi technology makes the discovery time dependent on several factors including the used wireless chip, the type of the underlying scan (active or passive) in addition to the processing delay of beacon information.

Shorter discovery times have been observed with the other considered approaches. For example, for the Zeroconf-based approach an average of 214.4 ms discovery delay has been obtained. However, for this discovery to successfully take place, a connection to a LAN has to be initially established. If this aspect was taken into consideration in the measured time, it would have resulted in an increase in the overall delay (as shown in Fig. 4). Additionally, a device in the presence of multiple WLANs has to consecutively connect to each of these WLANs and initiate a DNS-SD request looking for advertised services, which would be time-consuming since it has no prior knowledge whether a specific WLAN hosts the service or not. Since our proposal provides a pre-association discovery, this extra time will be avoided.

Similarly, for the centralized directory-based discovery, an average discovery time of 368.3 ms was obtained. As stated previously, the mobile device was provided with a Wi-Fi-based

[4]https://www.heroku.com/

Internet access to query the directory server. The delay for gaining such an access was not accounted for either (first two steps in Fig. 4). Our solution apparently causes more delay but compared to this centralized directory-based approach, it alleviates the burden experienced by the core network, since it takes place at the edge of the network. [5]

C. Battery power consumption

In order to compare the energy consumed by each one of the discovery strategies, the different energy profilers studied in [14] were considered and the Trepn energy profiler was used, since it provided stable measurements of the average battery power consumed by each individual app.

Experiments lasted for a duration of 15 minutes and were repeated 5 times, starting with a fully-charged battery in each case. The apps described above were used, each of them starting a discovery service running in the background to guarantee that UI usage is not contributing to the measured energy consumption.

For F2C-Aware and Zeroconf-based discovery approaches, we consider the two cases when advertisements are enabled or disabled (via beacons or service advertisements, respectively) to evaluate the impact of unsuccessful discovery calls on the energy consumption. The obtained results are plotted in Fig. 5.

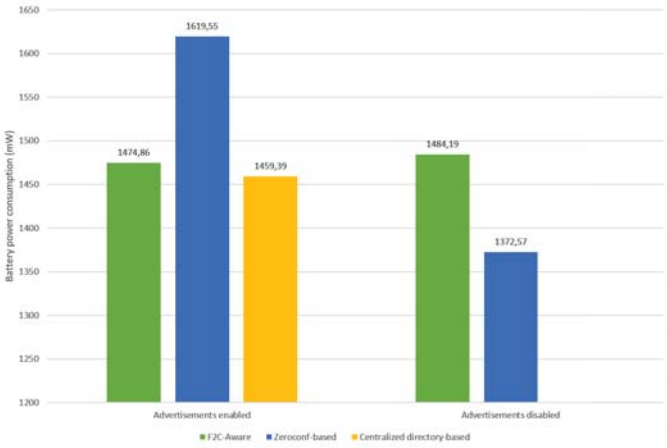

Fig. 5. Power consumption of different discovery solutions

As it can be seen, the obtained results were close, especially for our F2C-Aware approach and the centralized directory-based approach, while the increase for the Zeroconf approach is due to the default behavior that was kept, thus resulting in leaders being discovered more often, and not every minute like in the other two cases. Similar to discovery times explained in Section IV-B, if the power consumption resulting from network operations that made the discovery possible for the Zeroconf and the centralized approaches was captured in the results, the overall consumption of these approaches would have become higher.

In addition, it is clear that the average power consumption of 1474.86mW obtained with the proposed F2C-Aware for a "beacons-enabled" scenario is consistent with the usually observed values. In fact, a Wi-Fi scan generally generates a power consumption ranging from 60mW to 150mW according to [15] and since 15 discovery calls were performed in our case, a resulting consumption of 98.32mW fits into the expected aforementioned range.

Finally, it is worth noting that for the "advertisements disabled" scenario, the obtained power consumption for our approach is close to the "advertisements enabled" case, since the 802.11 scan is still made, without F2C information to decode, though[6]. This proves that the decoding process does not contribute to the power consumption. In the Zeroconf case, the consumption is reduced since there are no advertisements to be resolved by the client once the discovery starts. In both approaches, the discovery calls were not useful for the device, as there was no opportunity of finding a leader, thus resulting in an unnecessary energy overhead. These discovery calls could then have been avoided using a dynamic strategy that determines the optimal timings at which discovery should be performed. We aim to address this aspect in a future work.

V. Discussion

This section is intended to open up a discussion highlighting the benefits but also the potential issues brought by the proposed F2C-Aware solution. The discussion ends pointing out the set of challenges and research avenues demanded by a potential deployment of the proposed approach.

The main advantage of the proposed F2C-Aware approach is that it enables pre-association discovery, unlike many solutions in the literature. In fact, as the F2C discovery information is included inside the beacon, which is received regardless of the existence of a network connection, no manual intervention from the agent side is required to discover whether a leader exists in its network, thus resulting in both time and energy savings. Moreover, F2C-Aware is specially designed for F2C systems, what certainly optimizes its utilization. Indeed, it allows discovery of F2C-enabled devices only, unlike other approaches allowing discovery of all the devices (i.e. neighbors) within the radio range. It is also worth mentioning that F2C-Aware, as an example of an edge-centric solution, strongly leverages proximity as an important factor. Finally, F2C-Aware does not rely on the use of a broker (alternatively presented as a directory or a registry), which generally becomes a bottleneck as the number of discovery-related messages it has to process increases. Additionally, the problem of the broker discovery is avoided.

However, there are some limitations that must be carefully considered for a realistic deployment of our solution. In fact, kernel space modifications would be needed to ensure a comprehensive support across a variety of operating systems and hardware equipment. To avoid this, the 802.11 standard

[5]We do not conduct experiments regarding this aspect, as it is clear that when discovery is occurring at the edge, no traffic is flowing to the core network.

[6]Unexpectedly, we obtained a slightly higher value for the "advertisements disabled" scenario compared to the "advertisements enabled" scenario, which may be due to an outlier.

may play a major role in supporting this feature, thus allowing devices to send and receive custom information elements with low complexity. Also crucial in a real deployment would be to consider that the leader's wireless card must support the Wi-Fi master mode, allowing it to broadcast F2C-enhanced beacons. As a result, it cannot simultaneously operate in the managed (client) mode, unless such a combined use (client and AP) is supported by the wireless card driver. However, this should not be a problem if the leader is equipped with an additional network interface. Finally, although the Wi-Fi technology, which is the basis of our solution, is a connectivity option of choice in a number of fog/edge deployments, our solution should be complemented with a more generic discovery framework with support for other communication technologies and other layers of the hierarchy.

All in all, F2C-Aware may be seen as an attractive discovery approach for F2C systems, opening up some research avenues. Next, a tentative list of challenges driven by the adoption of the beacon-based discovery strategy is presented, including some interesting directions for future work.

- The beacon interval in conventional Wi-Fi is set to 100ms. However, for improved efficiency, we foresee that an adaptive beacon interval would be needed to better adapt to F2C systems. For example, the interval might be dynamically adjusted based either on estimations or on history of agents' arrival patterns.
- Once an agent finally decides to associate with a leader, it will need information on how to connect to it. To that end, an option would be to send connectivity information within the beacon (as the SSID is sent in regular beacons). However, for security reasons, this should be avoided since the beacon is sent in clear over the air, thus making it possible for unauthorized parties to get it. Another option may leverage the fact that each F2C agent goes through a fully trustful process to get system-level credentials (in an Eduroam-like fashion), thus setting a secure connection with a leader. If this process is trustful enough, the leader can guarantee that agents connecting to it are not malicious.
- A different challenge focuses on real/fake beacons. Indeed, the agent should have built-in verification mechanisms to detect fake beacons, so that it can trust the device that originated the beacon when sending the aforementioned credentials.

VI. Conclusion

This paper has provided additional design and implementation details of F2C-Aware, a solution that we previously proposed to address the problem of discovering nearby devices in an F2C system and that is based on the use of 802.11 beacon stuffing. We also presented a set of experiments that we conducted in order to measure the impact of using this solution. F2C-Aware has been shown to be comparable to baseline solutions, yet addressing other F2C characteristics (e.g. devices not necessarily belonging to the same LAN, need for core network traffic reduction, etc). Future work is expected to further investigate the issues that have been highlighted. In particular, since the use of the wireless technology makes our discovery solution energy-consuming, we intend to optimize this behavior by adopting an adaptive scan mechanism that performs discovery only at appropriate times.

Acknowledgments

This work was partially supported by the H2020 EU mF2C Project ref. 730929 and by the Spanish Ministry of Economy and Competitiveness and by the European Regional Development Fund, under contract TEC2015-66220-R (MINECO/FEDER).

References

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing - MCC '12*, p. 13, ACM, 2012.

[2] X. Masip-Bruin, E. Marín-Tordera, G. Tashakor, A. Jukan, and G. J. Ren, "Foggy clouds and cloudy fogs: A real need for coordinated management of fog-to-cloud computing systems," *IEEE Wireless Communications*, vol. 23, no. 5, pp. 120–128, 2016.

[3] G. Lewis, S. Echeverría, S. Simanta, B. Bradshaw, and J. Root, "Tactical cloudlets: Moving cloud computing to the edge," in *Proceedings - IEEE Military Communications Conference MILCOM*, pp. 1440–1446, IEEE, 2014.

[4] E. A. S. Mendoza, A. F. D. Conceicao, A. H. Aliaga, and D. Vieira, "Pytos: A Framework for Mobile Computation Offloading in Python," in *Proceedings - 11th International Conference on Signal-Image Technology and Internet-Based Systems, SITIS 2015*, pp. 262–269, IEEE, 2016.

[5] Z. Rejiba, X. Masip-Bruin, A. Jurnet, E. Marín-Tordera, and G.-J. Ren, "F2C-Aware: Enabling discovery in Wi-Fi-powered Fog-to-Cloud (F2C) systems," in *6th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (Mobile Cloud 2018)*, p. To Appear, 2018.

[6] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, "Challenges and opportunities in edge computing," in *Smart Cloud (SmartCloud), IEEE International Conference on*, pp. 20–26, IEEE, 2016.

[7] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.

[8] N. Choi, D. Kim, S. J. Lee, and Y. Yi, "A Fog Operating System for User-Oriented IoT Services: Challenges and Research Directions," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 44–51, 2017.

[9] B. Varghese, N. Wang, J. Li, and D. S. Nikolopoulos, "Edge-as-a-Service: Towards Distributed Cloud Architectures," *ArXiv e-prints*, 2017.

[10] R. Chandra, J. Padhye, L. Ravindranath, and A. Wolman, "Beacon-stuffing: Wi-Fi without associations," *Proceedings - 8th IEEE Workshop on Mobile Computing Systems and Applications, HOTMOBILE 2007*, pp. 53–57, 2007.

[11] S. Zehl, N. Karowski, A. Zubow, and A. Wolisz, "LoWS: A complete Open Source solution for Wi-Fi beacon stuffing based Location-based Services," *2016 9th IFIP Wireless and Mobile Networking Conference, WMNC 2016*, pp. 25–32, 2016.

[12] K. A. Nguyen, Z. Luo, and C. Watkins, "On the Feasibility of Using Two Mobile Phones and WLAN Signal to Detect Co-Location of Two Users for Epidemic Prediction," in *Progress in Location-Based Services 2014*, pp. 63–78, Springer, 2014.

[13] C. Xu, W. Jin, Y. Han, G. Zhao, and H. Tianfield, "MP-SDWN: A novel multipath-supported software defined wireless network architecture," in *International Conference on Communicatins and Networking in China*, pp. 119–128, Springer, 2016.

[14] A. Bakker, "Comparing Energy Profilers for Android," in *21st Twente Student Conference on IT*, vol. 21, 2014.

[15] N. Brouwers, M. Zuniga, and K. Langendoen, "Incremental wi-fi scanning for energy-efficient localization," in *Pervasive Computing and Communications (PerCom), 2014 IEEE International Conference on*, pp. 156–162, IEEE, 2014.