Mobile Interactions and Computation Offloading in Drop Computing

Radu-Ioan Ciobanu and Ciprian Dobre

Abstract In recent years, the amount of data consumed by mobile devices has grown exponentially, especially with the advent of the Internet of Things and all its connected devices. For this reason, researchers are looking for methods of alleviating the congestion and strain on the network, generally through various means of offloading, or by bringing the data and computations closer to the devices themselves through edge and fog computing. Thus, in this paper we propose an extension to the Drop Computing paradigm, which introduces the concept of decentralized computing over multilayered networks. We present a novel offloading technique to be employed by Drop Computing nodes for increasing processing speed, reducing deployment costs and lowering mobile device battery consumption, by using the crowd of mobile nodes belonging to humans and the edge devices as opportunities for offloading data and computations. We compare our method with the initial Drop Computing implementation and with the default scenario for mobile applications and show that it is able to improve the overall network performance. We also perform an analysis of human interactions with two monitoring nodes located in an academic environment, to obtain realistic data and to extract behavior patterns regarding human habits and interactions, that aid us in developing an efficient offloading solution.

1 Introduction

Because the number of mobile devices connected to the Internet has been growing lately, new paradigms such as edge and fog computing have garnered plenty of

Radu-Ioan Ciobanu
Faculty of Automatic Control and Computers, University Politehnica of Bucharest, Romania, e-mail: radu.ciobanu@cs.pub.ro

Ciprian Dobre
National Institute for Research and Development in Informatics, Bucharest, Romania, e-mail: ciprian.dobre@ici.ro

attention. They alleviate the strain placed on the network by the large amounts of data exchanges by bringing information closer to where it is needed, at the edge of the network. Through offloading, edge devices are able to aid the mobile nodes (smart devices, Internet of Things sensors, etc.) by fulfilling requests faster.

In this paper, we aim to improve the performance of Drop Computing [1], a paradigm which offers decentralized computing for mobile nodes in networks that combine cloud and wireless technologies over a social crowd formed between mobile and edge devices. In the initial Drop Computing proposal, mobile nodes that had some computation tasks to solve would either offload them directly to a cloud, or would request help from other nodes in the network in an opportunistic device-to-device fashion. In this paper, we propose enhancing Drop Computing by adding edge devices to the paradigm, which can act as more powerful nodes that can perform computations, or as relays for the mobile nodes, helping them offload their computations to the cloud without needing to use up their data plan by employing mobile broadband communication. For this purpose, we present an experiment where we used two mobile nodes for collecting Wi-Fi interaction data. We analyze this data to observe offloading opportunities using edge devices and, based on the conclusions obtained, we propose an extended Drop Computing variant.

The remainder of this paper is structured as follows. In Sect. 2, we present the original Drop Computing paradigm and other similar solutions that perform data and computation offloading in mobile networks for various purposes, highlighting the benefits that our framework can bring. In Sect. 3, we perform an analysis on Wi-Fi data collected using two local static nodes, showing the behavior of humans in an academic environment and the possibilities for offloading that come up in such a scenario. Section 4 presents the edge-based offloading mechanism that we propose for Drop Computing, while the results we obtained are analyzed in Sect. 5. Finally, we extract some conclusions and discuss future work in Sect. 6.

2 Related Work

The solution we propose here is an extension of the Drop Computing paradigm [1], which advances the concept of decentralized computing over multilayered networks, combining cloud and wireless technologies over a social crowd formed between mobile and edge devices. The need for Drop Computing arises from the insufficiency of the classic cloud model, which is not suitable anymore for the Internet of Things. When a large number of small devices communicate, they all need to send requests to the cloud, receive them back, and then process them. In Drop Computing, mobile devices and people interconnect to form ad-hoc dynamic collaborations to support the equivalent of a crowd-based edge multilayered cloud of clouds, where the capabilities of any mobile device are extended beyond the local technology barriers, to accommodate external resources available in the crowd of other devices. Thus, instead of every data or computation request going directly to the cloud, Drop Computing offloads requests towards the mobile crowd formed of edge nodes and

devices in close proximity for quicker and more efficient access. Devices in the mobile crowd are leveraged for requesting already downloaded data or performing computations, and the cloud acts as the second (or even third) option.

There are two main approaches to traffic offloading in mobile networks [2, 3], namely through Wi-Fi access points or in a terminal-to-terminal (T2T) fashion, where nearby nodes are used as a cache (if they already have the data stored) or as a relay. Several existing solutions employ a hybrid approach [4, 5], where the two types of offloading are used concurrently. This is also what Drop Computing does, through the crowd of nodes and edge devices.

Several solutions that are somewhat similar to the idea of Drop Computing have been proposed in the literature, but they all have some drawbacks that we aim to address. One such solution is a framework for creating virtual mobile cloud computing providers [6], where users with common goals collaborate by computing parts of a task and merging their results. However, this solution does not account for user mobility and for heterogeneous devices. Another example is proposed in [7], where mobile devices owned by different people help each other perform computational tasks, but it has the drawback that nodes are only able to communicate with each other one hop away, which limits the collaboration opportunities considerably. Furthermore, in case offloading to another device cannot be performed, there is no cloud infrastructure or edge device that can take over, as is the case with Drop Computing. Further solutions include mCloud [8] (which runs resource-intensive applications on cooperating mobile devices while using the cloud as a backup, but only allows nodes to communicate in a single-hop fashion), cloudlets [9] (where trusted nodes are located in the vicinity of a mobile user and can help with computations, but the solution has some limitations with regard to keeping track of all the cloudlets and how to access them, while communication between nodes is only performed through Wi-Fi and 4G), and a code offloading platform that performs offloading through various wireless channels (Wi-Fi, 3G, Bluetooth, Wi-Fi Direct), employing a multi-criteria offloading decision-making algorithm, which takes into account energy consumption, execution time reduction, resource availability, network conditions, and user preferences. The main limitation of this last solution is that it does not allow nodes to communicate further than one hop. The issues presented here are addressed by the Drop Computing paradigm, as shown in Sect. 4 and Sect. 5.

3 Offloading Opportunities Analysis

In this section, we present a data collection experiment that we ran at our faculty in order to analyze the behavior of mobile nodes in terms of contacts with static access points that can be employed as edge devices in our proposed Drop Computing framework. We deployed two static nodes in two different locations in the campus that had their Wi-Fi interfaces set to monitor mode. The nodes were part of the MONROE platform [10, 11], which is an open access hardware-based framework for large-scale experiments in mobile broadband and wireless scenarios. It offers a

large set of static and mobile nodes that have one Wi-Fi and three MBB interfaces each, located in several countries. The platform allows users to schedule Linux-based containerized experiments through a Web interface. A MONROE node has a 1 GHz 64-bit quad-core AMD Geode APU, 4 GiB of RAM and a 16 GiB SSD.

We chose two locations for the MONROE nodes that would be different from each other in terms of mobile device interactions, in order to have a better view of the offloading opportunities, and also to be able to compare various situations. The first node was placed in a large amphitheater (of approximately 180 seats), where courses are being held daily. Furthermore, the hallway next to the amphitheater connects two large parts of the faculty building, so there is a constant flux of students passing nearby. The second MONROE node was placed in the Pervasive and Mobile Services Laboratory, where generally 4-5 people work from the morning until the evening. However, the laboratory also has a constant flux of traffic, because students and other faculty members come to discuss their work, research, etc.

We performed data collection on both nodes from the 11th of May (on a Friday) at 8 AM until the 21st of May (on a Monday) at 4 PM, so there is more than a full week that can show us how the contacts fluctuate based on the day of the week. Furthermore, we have collected data from two full Saturdays, Sundays and Mondays, in order to assess the predictability of human behavior on a weekly basis. It should also be noted that the collection was performed during the semester, when classes were being held and students were attending them.

Figure 1 shows the distribution of total and unique contacts for each four-hour interval of the data collection period. At first look, it can be observed that many of the contacts that the two MONROE nodes observe per four-hour intervals are unique, which means that other devices do not pass many times near the nodes. They either arrive in their range and stay for a longer period (for example, students attending classes for the amphitheater node, or the researchers working in the laboratory for the other scenario), or they simply pass through. The contacts that are not unique most likely belong to professors that have multiple courses in the amphitheater or to researchers moving about the laboratory. Repeating contacts are extremely useful

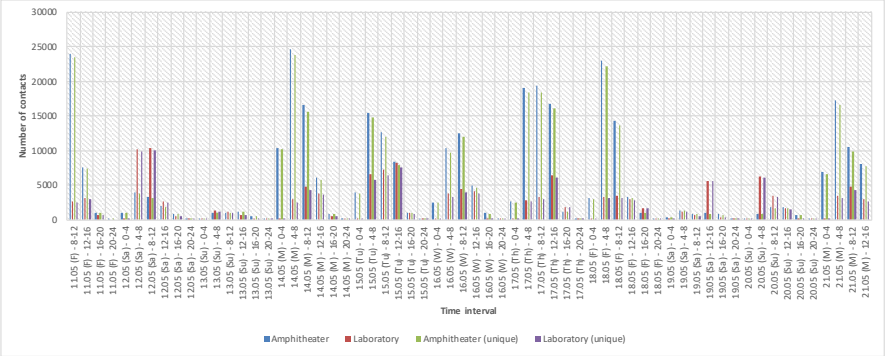


Fig. 1 Contacts of the two MONROE nodes with other devices.

for data offloading through device-to-device (D2D) communication, and is something we plan on investigating in more depth in the future. In Fig. 1, we computed the uniqueness of contacts per four-hour interval, but devices might repeat with larger inter-contact times, which is something we also wish to analyze in the future.

Another interesting conclusion that can be drawn from Fig. 1 is that, as expected when talking about a university scenario, there are far fewer contacts on the weekend. Furthermore, the time interval with the most contacts for the amphitheater node is between 4 AM and 8 AM. Most likely, this happens because the most courses in our faculty are in the morning, and that is the period when students arrive. On the other hand, the interval between 12 PM and 4 PM is the most popular one for the laboratory node, because the researchers working there arrive later. For data and computation offloading, it is important to know these intervals and the way they fluctuate. As shown in Fig. 1, this can be achieved by analyzing historical data, because there is a certain predictability in terms of number of contacts per day of the week (i.e., the number and distribution of contacts on 11 and 18 May are very close, and this is also true for 14 and 21 May). It has been previously shown [12] that opportunistic contacts in several types of environments can be modeled as Poisson processes, and in the future we wish to analyze the data collected here to see if this conclusion is true for this scenario. The difference in this case is that the MONROE nodes are static, as opposed to fully mobile nodes in opportunistic networks.

4 Mobile Data Offloading

The offloading mechanism that we propose in this paper is based on a network configuration as shown in Fig. 2. There are many mobile devices with mobile broadband connectivity that roam around the network area (or even exit it for periods of time). They can communicate with each other through close-range protocols such as Bluetooth or Wi-Fi Direct when they are in proximity, and they can also access the external cloud through 4G. Furthermore, there are several edge devices in the network that the nodes can connect to through Wi-Fi.

Fig. 2 Extended Drop Computing paradigm. The left arrow represents opportunistic communication between mobile nodes (where one node helps the other with computations), the middle arrow shows a mobile node communicating directly with the cloud, while the two arrows on the right show an edge device relaying a mobile node's requests to the cloud.

The mobile nodes run applications that require resources (such as CPU or battery) that they might not always posses. For this reason, in classic mobile applications, the main processing is performed in the cloud. The application is basically only a thin client for complex computations that are performed in the cloud and brought back to the user. However, such a scenario can prove somewhat costly, because the application developers have to maintain (or pay for) a cloud infrastructure that needs to serve all customers. In [1], we have shown that the Drop Computing paradigm is able to improve this situation by allowing other mobile nodes in range to help with the computations. Thus, large computations can be split into smaller tasks that can be passed on to encountered nodes, which can either process the tasks themselves or spread them even further in an opportunistic fashion [13]. This way, the cloud usage decreases because the number of requests is reduced. At the same time, the battery lifetime of the mobile devices increases, since they do not need to connect to the cloud through the mobile broadband interface, which is a notorious battery consumer [14] (especially when compared to the close-range protocols that Drop Computing employs for device-to-device communication).

In this paper, as shown in Fig. 2, we take the Drop Computing paradigm one step further, through the addition of edge devices that can either relay the computation requests to the cloud (and bring the results back), or that can even process some requests themselves. For the first case, the benefit would be brought by fast processing (since the transfer speed of Wi-Fi is higher than that of mobile broadband) and lower battery consumption. In addition, if the edge devices actually perform computations themselves, then the transfer latency between the edge device and the cloud would be gone, further reducing the total processing speed. Thus, a node that wants to compute some tasks has four possibilities: perform the computations itself, relay them to a device in range, send them to an edge node if there is one close by, or upload the tasks directly to the cloud. Naturally, the best option would be to use an edge node, as discussed above, but this can be done only when the node arrives in the edge device's Wi-Fi range.

The extended Drop Computing offloading mechanism is presented in detail in Algorithm 1. At certain intervals, the mobile nodes in the network generate tasks of various sizes and store them in a list. Until a mobile device encounters another node (either a mobile one or an edge device), it computes the tasks itself, starting from the beginning of the list (lines 44-50). When an edge node is encountered, the mobile device picks the first task from its list and sends it to the edge device, waiting for the answer (lines 14-16). When the task is computed, the edge device sends it back to the node if it is still in range. If that is the case, then the node picks the next task from the list, and repeats this operation until all tasks are successfully completed or the mobile node moves out of the edge device's range. When that happens, the node resumes computing itself the first task from its list. It should also be noted that, prior to sending tasks to the edge device, a mobile node first exchanges information about completed tasks with it. Thus, it first looks if the edge device has any completed tasks belonging to the current node, and then the mobile device also sends to the edge node information about the tasks it has completed itself for other nodes or that it has received from any encountered peers along the way (lines 8-11).

Algorithm 1 Extended Drop Computing offloading mechanism.

```
1:  A - current node, L_A - list of tasks of node A (its own or belonging to others), F_A - completed tasks of other nodes
2:
3:  for all time ticks do
4:      if A should generate tasks then
5:          generate tasks and add them to L_A
6:      end if
7:
8:      if edge device E in range then
9:          receive from E completed tasks belonging to A
10:         send F_A to E
11:     end if
12:
13:     while edge device E in range and L_A contains tasks do
14:         send first task T from L_A to E
15:         wait for completion of T
16:         mark T as completed and remove from L_A
17:
18:         if A is not T 's owner then
19:             add T to F_A
20:         end if
21:     end while
22:
23:     if mobile node B in range of A then
24:         receive from B completed tasks belonging to A
25:         send to B completed tasks from F_A belonging to B
26:
27:         if A and B are familiar then
28:             send F_A to B
29:             receive F_B from B and add to F_A
30:         end if
31:
32:         if L_A and L_B are not balanced then
33:             balance L_A and L_B
34:         end if
35:     end if
36:
37:     for all expired tasks T in L_A that belong to A do
38:         send T to cloud
39:         wait for completion of T
40:         receive T from cloud
41:         mark T as completed and remove from L_A
42:     end for
43:
44:     get first task T from L_A
45:     if T is completed at this tick then
46:         mark T as completed and remove from L_A
47:         if A is not T 's owner then
48:             add T to F_A
49:         end if
50:     end if
51: end for
```

Whenever another mobile node is encountered, the first step is to exchange information about completed tasks. Thus, the current node will receive the results for the tasks it had generated in the past that have been computed by other nodes (lines 24-25). Then, if the two nodes are familiar (i.e., are socially connected or they have encountered each other for at least a given number of times), they also exchange information about the completed tasks of other nodes (lines 27-30). In this way, they can help to opportunistically deliver the computation results to other nodes they

may encounter. As shown by lines 32-34, for the next step the two nodes check if the tasks each has to execute are balanced (i.e., if the total estimated durations of their computations are roughly equal). If the nodes are not balanced, an attempt to balance them is made, taking into account the processing power of each node and the owner of each task. If the two nodes are familiar and they need to balance their tasks, they simply exchange tasks (i.e., a node that offloads a task will no longer have that task in its computation list). On the other hand, if the two nodes are not familiar but they need to balance their tasks, replicas of the tasks are made (so each node keeps its previous tasks and potentially adds some more).

There is also the possibility that a node has too many tasks to process and does not encounter any edge or mobile device for a long period of time. In this case, the only solution is to employ the cloud, as done in the default case (lines 44-50). In Sect. 5, we will show the benefits that our extended Drop Computing mechanism brings over simply offloading everything to the cloud.

5 Results

For testing the solution proposed in this paper, we have implemented it in MobEmu[1], which is an opportunistic network simulator that can run a user-created routing or dissemination algorithm on a real-life mobility trace or on a synthetic mobility model [15]. We have updated MobEmu with extra functionality that allows us to simulate the scenario presented previously in Fig. 2. For our tests, we considered four situations: 1) nodes exclusively use the cloud for performing computations for a mobile application (referred to as "cloud only"); 2) they compute the tasks themselves or opportunistically with the help of other nodes, with no access to an Internet infrastructure ("opportunistic only"); 3) they employ Drop Computing as presented in [1] and Sect. 2 ("Drop Computing"); 4) finally, we also considered the extended Drop Computing paradigm presented in Sect. 4 ("Ext. Drop Computing").

We have tested our Drop Computing implementation on the two scenarios presented in Table 1. The first scenario uses the HCMM mobility model [16] to simulate node behavior and interactions. We simulated a scenario with 40 nodes grouped in four communities, with 10 travelers (i.e., nodes that move between communities). The nodes move in area of 400x400 meters with speeds between 1.25 and 1.5 m/s. We set the transmission radius of the devices to 10 meters, since we assume that data exchanges between nodes are done on Bluetooth. For the second testing scenario, we used a real-life mobility trace collected in an office environment (entitled "UPB 2015") in Bucharest using the HYCCUPS Tracer app[2], which was installed and run by the employees of an IT company. We selected a sample from this trace which contains contacts collected from 6 nodes during a 7-hour window in a workday.

[1] https://github.com/raduciobanu/mobemu.

[2] http://www.smartrdi.net/2017/11/08/getting-started/.

Configuration	Scenario 1	Scenario 2
Mobility model/trace	HCMM mobility model	UPB 2015 mobility trace (office)
Duration (hours)	24	7
Number of nodes	40	6
Contacts	134034	324
Bluetooth transfer speed	3 MB/s	
Wi-Fi transfer speed	10 MB/s	
4G transfer speed	3 MB/s	
Mobile nodes	One core, 2.3 GHz	
Edge nodes	One core, 2.3 GHz	
Cloud VMs	One core, 3.3 GHz	
Types of tasks	small (1 Mcycle), medium (1000 Mcycles), large (10000 Mcycles)	
Task size	5 MB	
Task expiration limit	1 ms for small tasks, 1 s for medium tasks, 10 s for large tasks	
Task generation	1-10 small, 0-10 medium, 0-10 large	

Table 1 Testing parameters.

Both the HCMM mobility model and the UPB 2015 trace only contain contacts between mobile devices. However, our scenario also assumes the existence of static edge nodes that mobile devices can connect to through Wi-Fi and use as data relays and for task computation. For this reason, we have used the Wi-Fi data presented in Sect. 3 to simulate two edge nodes for each scenario. We have selected the nodes with the most contacts with the edge devices and mapped them onto the nodes from the simulation scenario (i.e., 40 nodes for HCMM and 6 for UPB 2015).

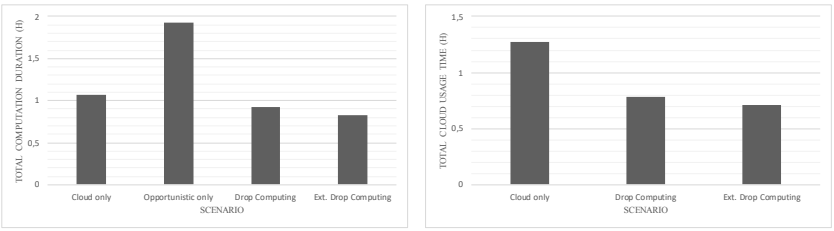
Figure 3 shows the results for the UPB 2015 scenario. It can be observed in Fig. 3(a) that Drop Computing manages to reduce the total computation time when compared to the case when devices only use the cloud or each other. For the cloud-only case, the total duration is 1.06 hours, whereas the opportunistic-only case naturally has a much higher duration (about 1.92 hours), since the devices have slower CPUs than the cloud virtual machines, while also having a limited battery life. By carefully selecting between using the cloud and collaborating with nearby devices, Drop Computing manages to reduce the computation time to about 0.92 hours, for a reduction of 13.5% when compared to the cloud-only case, and 52% when compared to the opportunistic-only case. Furthermore, it can be observed that the extended version of Drop Computing presented here, that takes advantage of two edge devices to further offload computations, manages to reduce the total computation duration ever more, by decreasing it with a further 11% compared to the initial Drop Computing solution.

However, the benefits of employing Drop Computing do not stop here. Figure 3(b) shows the cloud usage time for the three cloud-based scenarios analyzed above (the opportunistic-only scenario assumes that nodes do not have access to the cloud). Firstly, it can be observed that both Drop Computing versions manage to reduce the total cloud computation time. This means that not only does Drop Computing perform computations faster, but it also reduces the costs of the application developers by decreasing the cloud usage. When compared with the cloud-only

case, Drop Computing decreases the cloud usage with 38%, while the extended version presented in Sect. 4 obtains an even better reduction of 44.5%. This shows that, by carefully placing the edge devices in the network, the Drop Computing paradigm can bring clear benefits regarding several metrics.

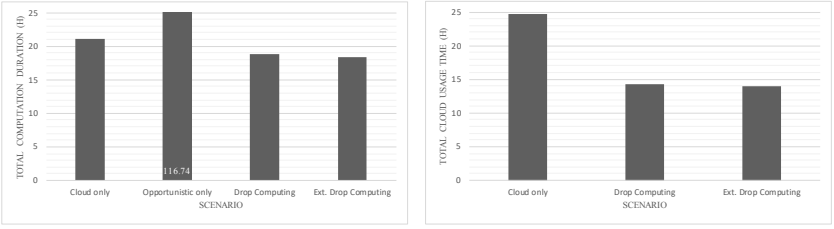
A similar situation can be observed in Fig. 4 for the HCMM scenario. The first observation that can be made is that the opportunistic-only scenario performs much worse here than for the UPB 2015 trace. This happens because there are more nodes (40, as opposed to 6 at UPB 2015), which generate more tasks, making it more difficult for a node to compute both its own tasks and those of other peers. For the same reason, the total computation time for HCMM is much worse than for UPB 2015, with values as high as 116 for the opportunistic-only case. Again, Drop Computing manages to decrease both the total computation duration (as shown in Fig. 4(a)) and the cloud usage (seen in Fig. 4(b)). This means that computations are performed faster for the users and cheaper for the developers. When compared to the cloud-only case, the two Drop Computing versions improve the computation time by 11% and 12.8% respectively, while at the same time decreasing the cloud usage time by 41.5% and 43% respectively).

The reason that the improvement brought by extended Drop Computing when compared to the original version are not as high as for the UPB 2015 scenario is that there are 40 nodes instead of 6, and only two edges devices (corresponding to the two MONROE nodes that we used for collecting data, as presented in Sect. 3). We believe that, by strategically placing additional edge devices in the network, we can



(a) Total computation time (b) Cloud usage time

Fig. 3 Results for the UPB 2015 scenario.



(a) Total computation time (b) Cloud usage time

Fig. 4 Results for the HCMM scenario.

further improve the behavior of the extended version of Drop Computing presented in this paper, and we plan on verifying this assumption in future work.

6 Conclusions and Future Work

In this paper, we addressed the topic of computation offloading in mobile networks. More specifically, we discussed the Drop Computing paradigm, which introduces the concept of decentralized computing over multilayered networks. We presented a data collection experiment and analyzed it in terms of offloading opportunities from mobile to edge devices, and then we proposed an extension to Drop Computing that takes advantage of edge devices and of other mobile nodes to offload computation. Through experimental analysis on mobility and contact traces (including the one we presented in this paper), we showed that the solution proposed here is able to reduce the processing time of mobile node tasks, while at the same time reducing the total cloud usage. This leads to lower costs for app developers and a higher satisfaction for users.

In the future, we wish to perform a more in-depth analysis on the data collected using the two MONROE nodes acting as edge devices. It would be interesting to see not only the number of contacts per time interval, but also their recurrence, as well as the duration of the encounters between devices. Furthermore, an analysis on the type of devices observed based on the MAC would also yield interesting results. Finally, we wish to continue improving the Drop Computing solution presenting here, by adding more edge devices, as well as making more informed decision when offloading data. Using the MONROE nodes, we also want to add mobile broadband traces to our simulator, in order to have more realistic experiments with mobile nodes equipped with 4G capabilities.

Acknowledgements This work is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 644399 (MONROE) through the open call project "Traffic and Data Offloading in Mobile Networks: TTOff". The views expressed are solely those of the authors. This research is also supported by University Politehnica of Bucharest, through the "Excellence Research Grants" program, UPB - GEX 2017, identifier UPB- GEX2017, ctr. no. AU 11.17.02/2017.

References

1. R.-I. Ciobanu, C. Negru, F. Pop, C. Dobre, C. X. Mavromoustakis, and G. Mastorakis, "Drop computing: Ad-hoc dynamic collaborative computing," *Future Generation Computer Systems*, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17305678
2. F. Rebecchi, M. D. de Amorim, V. Conan, A. Passarella, R. Bruno, and M. Conti, "Data offloading techniques in cellular networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 17, no. 2, pp. 580–603, Secondquarter 2015.

3. A. Aijaz, H. Aghvami, and M. Amani, "A survey on mobile data offloading: technical and business perspectives," *IEEE Wireless Communications*, vol. 20, no. 2, pp. 104–112, April 2013.
4. S. Dimatteo, P. Hui, B. Han, and V. O. Li, "Cellular traffic offloading through wifi networks," in *2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems*, Oct 2011, pp. 192–201.
5. M. Pitkanen, T. Karkkainen, and J. Ott, "Opportunistic web access via wlan hotspots," in *2010 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, March 2010, pp. 20–30.
6. G. Huerta-Canepa and D. Lee, "A virtual cloud computing provider for mobile devices," in *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*, ser. MCS '10. New York, NY, USA: ACM, 2010, pp. 6:1–6:5. [Online]. Available: http://doi.acm.org/10.1145/1810931.1810937
7. N. Fernando, S. W. Loke, and W. Rahayu, "Dynamic mobile cloud computing: Ad hoc and opportunistic job sharing," in *Proceedings of the 2011 Fourth IEEE International Conference on Utility and Cloud Computing*, ser. UCC '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 281–286. [Online]. Available: http://dx.doi.org/10.1109/UCC.2011.45
8. E. Miluzzo, R. Cáceres, and Y.-F. Chen, "Vision: Mclouds - computing on clouds of mobile devices," in *Proceedings of the Third ACM Workshop on Mobile Cloud Computing and Services*, ser. MCS '12. New York, NY, USA: ACM, 2012, pp. 9–14. [Online]. Available: http://doi.acm.org/10.1145/2307849.2307854
9. T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, "Cloudlets: Bringing the cloud to the mobile user," in *Proceedings of the Third ACM Workshop on Mobile Cloud Computing and Services*, ser. MCS '12. New York, NY, USA: ACM, 2012, pp. 29–36. [Online]. Available: http://doi.acm.org/10.1145/2307849.2307858
10. Ö. Alay, A. Lutu, R. García, M. Peón-Quirós, V. Mancuso, T. Hirsch, T. Dely, J. Werme, K. Evensen, A. Hansen, S. Alfredsson, J. Karlsson, A. Brunstrom, A. S. Khatouni, M. Mellia, M. A. Marsan, R. Monno, and H. Lonsethagen, "Measuring and assessing mobile broadband networks with monroe," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2016 IEEE 17th International Symposium on A*. IEEE, 2016, pp. 1–3.
11. Ö. Alay, A. Lutu, M. Peón-Quirós, V. Mancuso, T. Hirsch, K. Evensen, A. Hansen, S. Alfredsson, J. Karlsson, A. Brunstrom, A. Safari Khatouni, M. Mellia, and M. Ajmone Marsan, "Experience: An open platform for experimentation with commercial mobile broadband networks," in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*. ACM, 2017, pp. 70–78.
12. R. I. Ciobanu and C. Dobre, "Predicting encounters in opportunistic networks," in *Proceedings of the 1st ACM Workshop on High Performance Mobile Opportunistic Systems*, ser. HP-MOSys '12. New York, NY, USA: ACM, 2012, pp. 9–14. [Online]. Available: http://doi.acm.org/10.1145/2386980.2386983
13. R.-C. Marin, R.-I. Ciobanu, and C. Dobre, "Improving opportunistic networks by leveraging device-to-device communication," *IEEE Communications Magazine*, vol. 55, no. 11, pp. 86–91, november 2017.
14. J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4g lte networks," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '12. New York, NY, USA: ACM, 2012, pp. 225–238. [Online]. Available: http://doi.acm.org/10.1145/2307636.2307658
15. R.-I. Ciobanu, R.-C. Marin, and C. Dobre, "Mobemu: A framework to support decentralized ad-hoc networking," in *Modeling and Simulation in HPC and Cloud Systems*. Springer, 2018, pp. 87–119.
16. C. Boldrini and A. Passarella, "Hcmm: Modelling spatial and temporal properties of human mobility driven by users' social relationships," *Comput. Commun.*, vol. 33, no. 9, pp. 1056–1074, Jun. 2010.