Using Edge-Clouds to Reduce Load on Traditional WiFi Infrastructures and Improve Quality of Experience

Pedro M. Pinto Silva, João Rodrigues, Joaquim Silva,
Rolando Martins, Luís Lopes, Fernando Silva
Faculty of Science, University of Porto & CRACS/INESC-TEC
Rua do Campo Alegre, 1021, 4169-007 Porto, Portugal
e-mail: firstname.lastname@dcc.fc.up.pt

Abstract—**Crowd-sourcing the resources of mobile devices is a hot topic of research given the game-changing applications it may enable. In this paper we study the feasibility of using edge-clouds of mobile devices to reduce the load in traditional WiFi infrastructures for video dissemination applications. For this purpose, we designed and implemented a mobile application for video dissemination in sport venues that retrieves replays from a central server, through the access points in the WiFi infrastructure, into a smartphone. The fan's smartphones organize themselves into WiFi-Direct groups and exchange video replays whenever possible, bypassing the central server and access points. We performed a real-world experiment using the live TV feed for the Champions League game Benfica-Beşiktaş with the help of a group of volunteers using the application at the student's union lounge. The analysis of the logs strongly suggests that edge-clouds can significantly reduce the load in the access points at such large venues and improve quality of experience. Indeed, the edge-clouds formed were able to serve up to 80% of connected users and provide 56% of all downloads requested from within.**

Keywords: Mobile edge-clouds; P2P; WiFi offloading.

I. Introduction

By the end of 2016 more than 2 billion consumers worldwide will own a smartphone, accounting for 80% of mobile data traffic [1]. From e-commerce, e-learning and gaming to healthcare, mobile applications are growing in number, complexity and diversity, and have significantly impacted our social lives [16]. One of the causes is that applications are becoming more context-aware and offer a wider range of multi-user functionalities, such as shared editing of documents and online multi-player gaming. To achieve this, many applications integrate internal resources and sensing with the cloud.

Traditional Mobile Cloud Computing (MCC) focuses on moving processing and storage of data generated by mobile devices to centralized cloud datacenters. This offloading of computation and data benefits the users by decreasing battery consumption in the devices and allows them to access highly reliable infrastructure with seemingly unlimited computational and storage resources. However, due to the distance (both physical and logical) that separates a device at the edge of the network from the cloud, a major technical challenge prevails: how can MCC provide for applications with low-latency and/or high-bandwidth requirements? Moreover, MCC can not be directly put into use in cases where the infrastructure is unavailable or bandwidth limitations are part of the setting, for example, in realistic disaster scenarios and in dense environments such as sports or music events.

To address these issues, new paradigms such as mobile edge-clouds strategically combine traditional cloud infrastructure with the resources provided by devices, enabling proximity-aware applications (c.f., Section II). In this scenario, smartphones are not viewed as just "thin clients", but rather as devices capable of performing relevant computational tasks and act as "thin servers" [17]. An edge-cloud is composed of available nearby devices that work together to form a pool of computing resources with sustained operation under poor connectivity and access to crowd-sourced information which otherwise might be unavailable. Applications can use the edge-cloud to perform data caching and mining; image and video processing or streaming; enforce security and control mechanisms (authentication, content flagging, etc); or simply query data. More importantly, computational tasks are performed locally, i.e. there is no offloading of computation or data to a traditional cloud infrastructure. However, various challenges still need to be addressed in order to build an edge-cloud infrastructure capable of: supporting a variety of heterogeneous devices; handling churn, i.e. coping with the dynamic movement of devices entering and leaving the edge-cloud; deciding when to leverage one wireless technology and/or network topology over another (e.g., peer-to-peer vs. client-server, WiFi through AP or WiFi-Direct) [9].

Presently, and unlike MCC, few applications have been proposed that use and demonstrate the value of edge-clouds. A real world scenario that illustrates their value is the dissemination of video contents, e.g. soccer replays, to club fans in a sports venue by offloading part of the WiFi traffic to the edge-cloud. There is a growing market of apps that provide users within (and outside) the stadium with almost real-time statistics and multimedia contents like the number of kilometers a player has run or video replays for goals or interesting events [2, 3]. If the fan chooses to watch a video replay, the content is downloaded from the central servers through stadium installed access points (WiFi or cellular), and then played on the device. If, however, the venue is crowded, the large number of requests can stress the infrastructure [8, 12].

One way to solve this problem is to use mobile edge-clouds. In this way, the fan's device and its neighbours in the stadium can form a local cache for the server contents - users of the service can be encouraged to share in several ways, e.g., sweepstakes of team merchandise or lower rates for the service. For example, before asking the server for a video replay, the application might ask the other phones in the edge-cloud whether they have a copy of the video. If a copy of the replay is located, the fan can retrieve it directly from a neighbouring phone.

Although this is not the scope of this paper, we are aware that such a scenario raises privacy and security issues, but there is research on this subject. For privacy, for example, one can use use the native container/sandbox [7] infrastructure provided by Android coupled with a careful management of permission schemas [10]. For security, one can, for example, envision a curation mechanism that would involve a team of people visualizing the videos submitted and digitally signing them before allowing them to be streamed by the application.

In this paper we address this problem and make two fundamental contributions. First, we design and implement a prototype application for Android devices that dynamically switches between the WiFi infrastructure (to access a central server) and an edge-cloud composed of WiFi-Direct groups (to access cached copies) to disseminate video contents. To our knowledge there are no similar applications nor have edge-clouds been used in such a scenario. Second, we performed a real-world experiment using the live TV feed for the Champions League game Benfica-Beşiktaş with the collaboration of a group of volunteers that installed and used the application at the student's union lounge. The analysis of the logs from the experiment strongly suggests that edge-clouds can indeed significantly reduce the load in the access points at such large venues and improve quality of experience.

The remainder of the paper has the following structure. Section II describes state-of-the-art work on edge-clouds and their uses. Section III describes our application scenario in detail and the analysis of the logs from the experiment. Finally, Section V puts forward our conclusions and thoughts for future work.

II. Related Work

Content Delivery Networks (CDN) emerged with the objective of improving the distribution of content to end-users by providing high data availability, reduced latency and increased network bandwidth. There is a substantial body of research on CDNs, namely using hybrid server/P2P-based architectures (e.g., [5, 18] and more recently [13]), focusing mostly on traditional networks. Here, we are interested in providing these infrastructures for mobile edge-clouds and in verifying whether some of the performance gains reported in the literature can be transposed to this context.

A mobile edge-cloud is a cluster of mobile devices that collectively pool their storage and computation resources over a local communication network, e.g., WiFi, Bluetooth, ad-hoc. While each individual mobile device in the edge-cloud might not be resource-rich, collectively the pool of these mobile devices can present the opportunity for significant computational resources that might collaboratively support proximity-aware applications [17].

Mobile Message Passing Interface (MMPI) [4], Hyrax [14] and Honeybee [15] were amongst the first approaches to use mobile devices for computation. MMPI used a Bluetooth Piconet to run MPI jobs over a set of mobile devices. This was implemented in Java ME and did not tolerate churn. Hyrax used a different strategy, by porting Hadoop to Android devices they were able to schedule computations using only mobile devices or heterogeneous networks of mobile devices and servers over WiFi local networks. The performance was not good, but the system was capable of handling some churn using Hadoop fault tolerance algorithms. Honeybee [15], can use both Bluetooth or WiFi-Direct to form the local network and provides distributed computation using work sharing and work stealing approaches. This framework works on small networks, and provides techniques to partially handle churn using the master or group owner node to periodically ping other nodes checking their availability, and rescheduling failed jobs if necessary. A similar approach to Honeybee was proposed by mCloud [6], but there was no implementation of this system.

FemtoClouds introduce a concept of cloud computing very similar to what is described as an edge-cloud [11]. Using mobile devices, the authors create a network, that is capable of receiving jobs, and scheduling them through the network. For this, they propose to use a special device, called "control device" that is responsible for managing the mobile devices, as well as scheduling the jobs. The authors propose a client software that interacts with the control device, by providing relevant information for job allocation, as well as receiving or generating work.

The aforementioned edge-cloud implementations use the combined processing and storage resources of devices to store data and perform traditional distributed computations. Our focus is on streaming and caching of video contents by the edge-cloud, complementing a central infrastructure by removing load in the servers and access points and improving user experience.

III. Soccer Replays: A Case Study for Edge-Clouds

As previously mentioned, services for video replays in current sports venues rely on the retrieval of multimedia contents through a central server, accessible to users by one of possibly many installed WiFi access points (AP). Yet, the infrastructure is easily flooded in very crowded and user-active spaces, as each AP cannot handle more than a few tens of connections simultaneously. Therefore, equipping sport centres and venues with the amount of wireless infrastructure required to provide adequate WiFi coverage comes at a high cost.

On the other hand, fan's smartphones can be leveraged to empower the existing network. Using device to device communication, further layers of connectivity can be orchestrated in a way that the number of users under the same AP can be significantly increased. We call edge-cloud to a set of devices operating on a second layer of connectivity, established with peer-to-peer networking through, in this case, WiFi-Direct. We show how such an infra-structure can be used to extend traditional WiFi infrastructure and alleviate load on access points.

A. *Edge-Cloud Architecture*

In this paper, we consider a scenario (Figure 1) in which the replays are generated remotely and placed in a centralized server or cloud. Moreover, nearby devices use WiFi-Direct exclusively as the enabling technology for device to device communication. The edge-cloud is thus made up of several WiFi-Direct groups, each one with a group owner (GO) that acts as an AP for other member devices. As a device can not connect to two WiFi APs at the same time, the GO is the only device that is connected to the infrastructure AP and hence to the application server. It is important to note, however, that the devices maintain access to the Internet while in such a configuration. This is made possible through a proxy or a VPN installed in the respective GO.

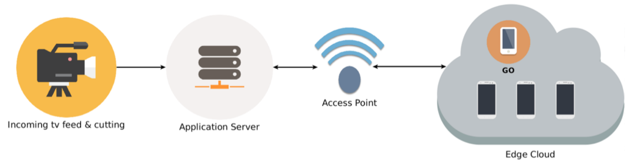


Fig. 1. The experimental scenario.

Data and Metadata: A replay is composed of video content, and metadata - human-readable information about the video such as title, duration and thumbnail. We consider that a user can only decide whether or not a replay is worth downloading by first inspecting its metadata. Thus, a crucial component of the edge-cloud architecture is the dissemination of metadata. The architecture must specify how a device obtains video metadata, given a set of data links. Information about the origin of videos, whether generated remotely or at the edge, can also be used to determine which types of links should be prioritized.

Similarly, the device must first obtain information about the existence and availability of a nearby edge-cloud before it decides whether it can be part of the edge-cloud and what will be its role in it. To that extent, a discovery-advertisement mechanism must be in place that enables devices to capture the edge-cloud metadata necessary to make a decision.

WiFi-Direct Groups: Devices that create groups become owners and are responsible for registering group's metadata on the application server upon creating and destroying the group. Group metadata contains the required information for other devices to discover the group's WiFi network, connect to it and communicate with the GO via a TCP socket. This information consists of: the SSID of the network, its password, group size, the owner's IP, and the TCP port the owner is listening on. Furthermore, a metadata version number is included so that conflicting views of the same group are handled in a straightforward fashion. This value can only be issued and modified by the GO, which increments it whenever there is an update to the metadata, for instance a member joining or leaving the group.

Finally, the GO is responsible for keeping track of the members of the group, their connection information, i.e. IP and TCP listening port, and finally for indexing the content held by each. Reciprocally, when devices join a group they immediately send a hello message to the group owner with their connection information and list of owned video replays.

Centralized Server: The server is responsible for keeping the metadata of both videos and groups, along with the content of the replays themselves. Devices periodically try to retrieve video metadata. If the devices are GOs, or do not integrate any group, they retrieve metadata from the server using a HTTP request made to a known URL. Otherwise, they retrieve it from the GO using a TCP socket.

Replay Requests: In the case of replays' content being requested within the edge-cloud, in-group transfers are prioritized over available infrastructure APs. If the requesting device is a GO, then it can search for a local replay owner directly within the content-by-device map kept in memory. Otherwise, the device will query the GO for content holders, receiving a list of candidates with their connection info. If the list's size is greater than one, then the device will randomly remove candidates one by one and attempt to download the content over TCP, breaking on success. However, if no one in the group owns the content at the time of request, the replay will be downloaded from the server through the GO which forwards the incoming data stream directly to the device that requested it.

B. *Edge-Cloud Orchestration*

Orchestration is the decision process that devices go through in order to determine their role in the edge-cloud. In our scenario, devices obtain edge-cloud metadata through the server. To that extent, devices need to be connected to the Internet at the start of the application. If connectivity to the server is available, the device requests from the server a list of active groups operating under the AP (bssid) to whom it is currently connected to. If the set of groups is not empty and at least one group is available, i.e. not operating at maximum capacity, then the device will try to join the group. Otherwise, it will create a new group.

We have limited the number of devices in a group to 4 and the number of attempts to connect to a group to 1 for the following reasons. First, we experimented with different devices configured to work as GO and they were just able to handle 5-7 connections at same time. This limitation is imposed by Android and by the network hardware and drivers in the devices. As a result, we assumed that 4 devices in a group was a safe number in order to reduce the connection attempt errors. Second, we also limited the group size to avoid stressing the GO during the game, which would result in a bottleneck and the dismantlement of the group. Third, the number of attempts to connect to a group is just 1 because a connection to a group, as well as to a WiFi AP, takes a considerable amount of time (on average more than 6 seconds in our tests) and we wanted to keep the user online as much as possible.

Algorithm 1 illustrates the decision process that each device has to perform to join a group. When the application starts, the algorithm is always performed to decide which role the device will play in the edge-cloud. Due the dynamic nature of edge clouds, it is not, at all times, possible to figure out the role of the device on the edge, thus in these cases it waits an exponential back-off time, based on the number of retries, after which the algorithm is restarted. When the device is on the exponential waiting process, it tries to reconnect to the last connected WiFi network in order to keep in touch with the server, allowing it to see metadata updates and to get the videos if necessary.

Algorithm 1 Application main orchestration loop

```
1:  procedure GROUPFORMATION(bssid, tries)
2:      if $amGroupOwner \land noMembers$ then
3:          $restart \leftarrow true$
4:      else
5:          $groups \leftarrow$ GET_GROUPS(bssid)
6:          $gp\_avail \leftarrow$ GET_GROUP_AVAILABLE($groups$)
7:          if $\#groups = 0 \ \lor \neg gp\_avail$ then
8:              $restart \leftarrow \neg$ BECOME_SOFT_AP( )
9:          else
10:             $restart \leftarrow \neg$ CONNECT_TO_GROUP($groups$)
11:         end if
12:     end if
13:     if $restart$ then
14:         $timeout \leftarrow$ GET_WAIT_TIME(tries)
15:         RESTART_FORMATION($timeout$)
16:     end if
17: end procedure
```

The algorithm has a few self healing capabilities for situations such as group disconnection and empty groups. When a group disconnection happens the device will try to reconnect to the last WiFi network and re-run the algorithm. In case of a recently created group, or an emptied one, the GO will check if it has new members after one minute. If not it will destroy the group and run the algorithm again aiming to reduce the number of groups. In this case, there is no need to wait an exponential time because the device that was a GO has always an active connection to the WiFi AP.

IV. Experiments and Results

In order to validate the proposed implementation we designed an experiment to employ real users. We invited a number of students to watch a soccer match openly broadcasted over television and offered them pizza and soft beverages as incentives. In exchange, they would use an Android application developed by our team. The participants could either install the application on their smartphones or use one of 15 provided tablets (Nexus 9). The application would give them access to a list of replays, generated during the game, that once successfully downloaded could be viewed.

A. Experimental Setup

The replays were generated by two of the authors located on a separate room. As the game was being transmitted, it could be captured via DVB-T (we used a RTL2838 DVB-T dongle) and manipulated on a Linux workstation. Using several scripts built on top of the ffmpeg software tools, we were able to cut the original feed into smaller chunks that could be concatenated to form short videos of soccer replays. Finally, the videos would be uploaded to the application server via HTTP.

In the student's lounge, the application would periodically request the list of replays either through the GO, if the device was part of a group, or through the server if the device was a GO or not part of any group. Moreover, there was a WiFi access point at the lounge which provided access to the Internet and therefore to the application server. Four other access points were accessible from the lounge, part of the University's wireless coverage. Students were asked to prioritize connectivity to the room's access point.

The minimum supported Android version was API version 16, named Jellybean or Android 4.1. We increased the heterogeneity by varying the Android versions of the tablets provided to the students (most were Lollipop, several Marshmallow and one was Nougat).

The application registered user and device activity on a SQL Lite database as a single array of characters structured in JSON format. Furthermore, whenever connectivity to the server was available, the application would try to transmit the contents of the database to the server via HTTP. We found the following events relevant for logging:

- a successful connection to the edge-cloud (i.e., to the GO of a WiFi-Direct group) or to the WiFi infrastructure (i.e., a regular WiFi access point);
- a successful disconnection from the edge-cloud or from the WiFi infrastruture;
- the creation of a WiFi-Direct group;
- the destruction of a WiFi-Direct group;
- the completion of the download of a video replay.

Each event was logged with a timestamp and user-id along with relevant information for that particular event. This information made it possible later on to trace back the activity of each device within the edge-cloud, and therefore the origin of download requests and providers.

As each device has its own clock, we cannot build the global timeline of events directly using the registered timestamps. Thus, we used the application server as the reference clock as it was the common denominator to all devices. Therefore, a calibration step was performed at the start of the application, which estimated the difference δ between the timestamps of the devices and that of the remote server. A request for synchronization was sent from the device to the server via HTTP, for which the server responded with its timestamp. Assuming routing and latency symmetry for request and response packets, the device was able to estimate δ as the difference between the returned server timestamp and the mean value between the timestamps at which it sent and received the synchronization packet. To improve robustness of estimation, 10 synchronization operations were performed in sequence, and the final value for δ was considered as the median of all successfully performed operations. This value was added to each logged event so that it could be used later for relating device activity.

B. Log Analysis

The experiment took place on the 13th of September 2016, from 6:45 p.m to 8:50 p.m GMT time, during the Champions League match between Benfica-Beşiktaş. The logs show that 43 different users interacted with the application, resulting in 3926 distinct entries successfully recorded at the remote server. Of the total number of logged messages, 1222 were events of WiFi connection/disconnection, 1526 events of group creation/destruction and 1178 attempts of download, with 860 successes and 318 failures.

General Considerations: To begin with, device timestamps were corrected based on the median of registered δs per device. Then, all events were sorted by timestamp and evaluated sequentially. For a given event of type E, registered on device D at timestamp t, we classify D's state as E between t and $t + 1$. We ignore events of download at this stage as they do not alter device state. Therefore, we classify D as a connected device if E_{t-1} is a successful connection to a WiFi network, either a traditional or a WiFi-Direct access point. In case of the later, the device is also classified as a member of a group. Alternatively, if D is already connected and E_{t-1} corresponds to a create group event then that device is also a GO. Similarly, a device is classified as disconnected after an event of disconnection.

With this information we computed the size of the edge-cloud at t as the sum of all connected GOs and group members. Its size was incremented on events of group creation or connection and decremented on events of disconnection, group destruction or application destruction. On the other hand, the set of connected users were solely incremented and decremented by events of WiFi connection and disconnection.

The ratio between sizes of the edge-cloud and the set of connected users represents the *scope* of the edge-cloud, whereas the fraction of group members over group owners represents its *depth*. A scope ratio of 1 indicates that all nearby connected devices are actively participating in the edge-cloud. A depth ratio is optimal if it tends towards the maximum number of elements allowed in each group, representing the edge-cloud operating at its maximum capacity.

However, this approach is only possible if for each device the number of increments equals the decrements. In other words, a device can not have different numbers of connection and disconnection events. A connection can only occur if the device is in a disconnected state, and vice-versa. Therefore, such properties should be verified before proceeding with computations, else the obtained results will not depict the accurate unfolding of events.

During pre-processing we found that several events were unsuccessfully registered either at the time of occurrence or later. Consequently, we applied a semi-automatic approach which aimed at discovering the missing events and inserting them if no conflicts were caused as a result. Otherwise, these would be signalled for human-eye review.

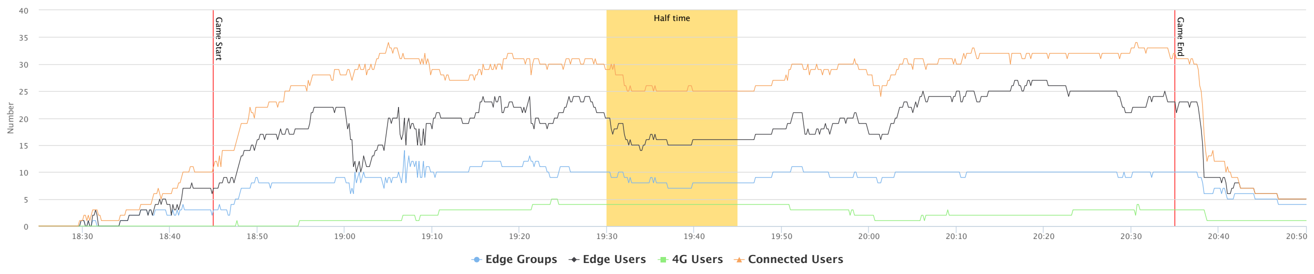
Run-time Evolution of the Edge-Cloud: Figure 2 depicts the activity of the edge-cloud over the period of the experiment. Activity is mainly represented in Figure 2a by the number of active groups ("Edge Groups") and devices within the edge-cloud ("Edge Users"). As expected, these values increase gradually after the beginning of the game and stabilize near the middle of the 1st half of the game. Furthermore, there is a decrease in activity around the half time, when pizzas were distributed and users were allowed to leave the room. Similar behavior is observed during the second half until the game ends and users leave the room, disbanding the edge-cloud as a consequence. We also present the total number of users connected through WiFi APs ("Connected Users"), which comprises all APs servicing the lounge, including the edge-cloud itself. Some users that connected to the server using 4G are also represented ("4G Users").

During the first half we had a short malfunction of the AP which resulted in a substantial amount of failed downloads. The loss of the connections of the GO to the AP triggered the destruction of the groups and the edge-cloud had to reconstruct itself using the self-healing algorithm as described in Section III-B.

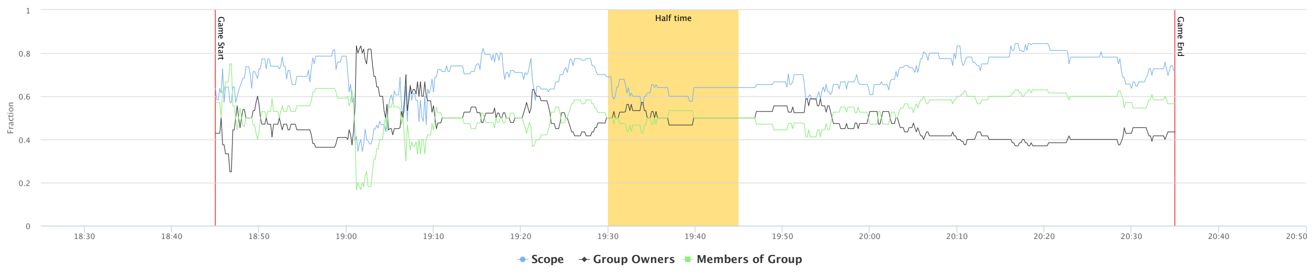
In Figure 2b we can observe the measure of scope ("Scope") for the duration of the game. We expected that nearly all users would participate in the edge-cloud, however the observed fraction is rarely over 0.8. The reason lies in the exponential back-off time used when devices can not join a WiFi-Direct group or create one. During this period the devices are just connected to the AP. Furthermore, the average *depth* score ("Depth") for the edge-cloud was 1.2, corresponding to 30% of the maximum group capacity. Therefore, as most active groups contained few members and were operating at a suboptimal level, alternatives and optimizations can be explored for improving the orchestration of the edge-cloud. Additionally, Figure 2b depicts the fractions of devices that are GO ("Group Owners") and members of groups ("Members of Group").

TABLE I
DESCRIPTION OF DOWNLOAD COUNTS (FIRST TWO COLUMNS) AND SPEEDS IN MB/S (LAST THREE COLUMNS) BY CATEGORY.

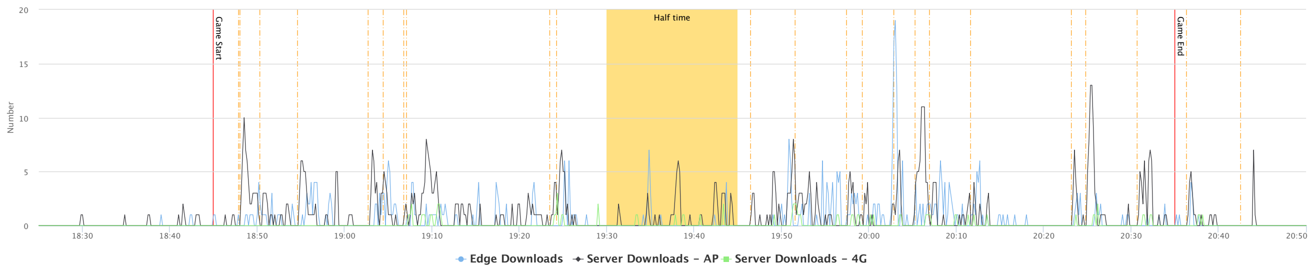
Category	Successful	Failed	Median	IQR	1Q
EE	369	6	2.40	2.46	1.30
ES	287	138	0.79	0.85	0.42
RS	155	131	0.81	0.83	0.54



(a) Number of 4G, wifi-connected and edge-cloud users.



(b) Scope of the edge-cloud along with the average percentage of owners and members in active groups.



(c) The number of successful downloads over time.

Fig. 2. Network and download activity over the course of the experiment.

In complement, Figure 2c provides the count of downloads successfully delivered via edge-cloud and server: AP, and 4G. Vertical lines indicate when new replays were generated and made available on the server. It is interesting to notice that when a replay is generated it is usually followed by a spike in downloads performed through the server and, only then is followed by a spike in downloads performed through the edge. This observation is expected because the content is only available at the server when it is generated. As groups do not communicate with each other, new downloads will be obtained at least once from the server per group.

Download Speed and Bandwidth: Regarding the analysis of download activity, we are particularly interested in estimating the amount of traffic that was reduced from the access point. This was possible by fixing how many replays the edge-cloud was able to deliver with and without the use of the access point. To that extent, we divided downloads by categories based on where they were requested and provided from: Edge-Edge (EE) if the download was requested from within the edge-cloud and provided by the edge-cloud; Edge-Server (ES) if instead it was provided by the server via the access point; Remote-Server (RS) if the download was requested outside of the edge-cloud and provided by the server via the access point.

Table I shows distribution of downloads and their speeds by category. About 56.3% of the downloads requested from within the edge-cloud were processed internally without the need for an AP. Taking into account the 3rd category of downloads, this value decreases to 45.5%. Nevertheless, it is a significant shift in load from the AP to the users' mobile devices. Especially when we consider that as soon as a video was published it had always to be retrieved from the server at least once for each active group. Furthermore, no aggressive caching techniques or optimization strategies were used to improve the performance of the edge-cloud.

Download speeds are particularly useful when surveying quality of experience. Higher speeds indicate that content is transferred faster, so the user spends less time waiting for its completion hence increasing quality of experience. Looking at the median, the Inter-Quartile Range (IQR) and the First Quartile (1Q) values for each category of downloads it is clear that it has benefits for edge computing. Not only do EE downloads contain a median 3 times greater than ES or RS, but also a 3rd quartile at nearly 4MB/s and several benign outliers (Figure 3). In part, this result was expected,

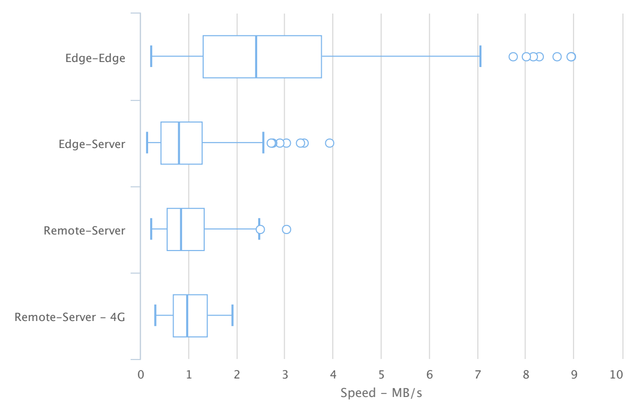


Fig. 3. Descriptive summary of download speeds by category.

as content travels over a single hop within groups, even when it is transferred over two members given that we use TDLS (Tunneled Direct Link Setup) in conjunction with WiFi-Direct. Moreover, devices that support 802.11ac (5GHz ISM Band) are capable of higher throughput.

However, there is room for improvement. One possibility is to differentiate data and metadata traffic, using Bluetooth for metadata dissemination and WiFi-Direct for actual video streaming [9]. Although Bluetooth connections have low data rates and range of operation when compared to WiFi, this is more than enough for metadata. Moreover, since devices can maintain several simultaneous connections, Bluetooth could be used to disseminate metadata messages a few KBs in size over large networks in crowded scenarios.

V. Conclusion

The landscape for mobile networking is currently shifting towards device to device communication with growing investment being made on edge-clouds and new technologies such as 5G and LTE-Direct.

To our knowledge there are no previous studies that tried to engage real users and acquire data based on edge-cloud implementations. On one hand, there have been few solutions that have been coined under the edge-cloud computing terminology. On the other hand, the existing operating systems for mobile devices limit the peer to peer networking capabilities of their phones to the extent that it is very difficult to perform wireless ad-hoc networking on large numbers of connected devices.

Still, we were able to develop a new methodology for structuring and orchestrating an edge-cloud. The current scenario makes extensible use of a remote application server, in particular to keep the required edge metadata. Nevertheless, this experiment has showed that edge-clouds can be used to alleviate load in an infrastructure and provide users with faster downloads. The edge-cloud was able to serve up to 80% of connected users and provide 56 % of all downloads requested from within. An average of 9.7 groups were active during the game operating at an average capacity of 30%. Moreover, the speed of downloads at the edge was 3 times greater than the ones made through the AP, indicating that the edge-cloud enhances quality of experience.

This work is the first step towards the development of a comprehensible edge-cloud solution which runs on top of off-the-shelf smartphones and that can be used to extend, or substitute under particular conditions, traditional WiFi infrastructure.

In the future we envision combining multiple wireless technologies to empower the architecture of the edge-cloud, in particular for managing groups more efficiently. Caching techniques and improved orchestration methods can also be devised to optimize the performance of the edge-cloud.

Acknowledgments

This work has been sponsored by projects HYRAX (CMUP-ERI/FIA/0048/2013), funded by FCT, and SMILES (NORTE-01-0145-FEDER-000020), funded by NORTE 2020, under PORTUGAL 2020, and through the ERDF fund. Our thanks to the “Associação de Estudantes da FCUP” and the student volunteers.

References

[1] Global Mobile Statistics 2016, Q2 Report. http://mobiforge.com/.
[2] YinzCam. http://www.yinzcam.com/.
[3] HuaweiVoice: Agile Stadiums Bring Digital Content To Sports Fans. http://www.forbes.com/, Mar 2015.
[4] D. Doolan, S. Tabirca and L. Yang. MMPI a Message Passing Interface for the Mobile Environment. In *Proceedings MoMM'08*, pages 317–321, New York, NY, USA, 2008. ACM.
[5] D. Xu, H-K. Chai, C. Rosenberg and S. Kulkarni. Analysis of a Hybrid Architecture for Cost-Effective Streaming Media Distribution. *SPIE, Multimedia Computing and Networking*, 5019:87–101, 2003.
[6] E. Miluzzo, R. Cáceres and Y-F. Chen. Vision: MClouds - Computing on Clouds of Mobile Devices. In *Proceedings of the MCS'12*, pages 9–14. ACM, 2012.
[7] Google. Android Security White Paper, 2015.
[8] J. Erman and K. K. Ramakrishnan. Understanding the Super-sized Traffic of the Super Bowl. In *IMC'13*, pages 353–360. ACM, 2013.
[9] J. Rodrigues, J. Silva, R. Martins, L. Lopes, U. Drolia, P. Narasimhan, F. Silva. Benchmarking Wireless Protocols for Feasibility in Supporting Crowdsourced Mobile Computing. In *Proceedings of DAIS'16*, pages 96–108. Springer, 2016.
[10] J. Tan, U. Drolia, R. Martins, R. Gandhi and P. Narasimhan. CHIPS: Content-Based Heuristics for Improving Photo Privacy for Smartphones. In *Proceedings of WiSec'14*, pages 213–218. ACM, 2014.
[11] K. Habak, M. Ammar, K. A. Harras and E. Zegura. Femto Clouds: Leveraging Mobile Devices to Provide Cloud Service at the Edge. In *Proceedings of CLOUD'2015*, pages 9–16, June 2015.
[12] P. Kapustka and C. Stoffel. State of the Stadium Technology Survey. Technical report, 2014.
[13] M. Ghareeb, S. Rouibia, B. Parrein, M. Raad and C. Thareau. P2PWeb: A Client/Server and P2P Hybrid Architecture for Content Delivery Over Internet. In *Proceedings of ICCIT'13*, pages 162–166, June 2013.
[14] E. E. Marinelli. Hyrax: Cloud computing on mobile devices using mapreduce. Master's thesis, Carnegie Mellon University, 2009.
[15] N. Fernando, S. Loke and W. Rahayu. Honeybee: A Programming Framework for Mobile Crowd Computing. In *MobiQuitous'12*, pages 224–236. Springer, 2012.
[16] The Internet Society. Internet Society Global Report 2015 - Mobile Evolution and Development of the Internet. http://www.internetsociety.org/globalinternetreport/assets/download/IS_web.pdf, 2015.
[17] U. Drolia, R. Martins, J. Tan, A. Chheda, M. Sanghavi, R. Gandhi and P. Narasimhan. The Case for Mobile Edge-Clouds. In *Proceedings of ATC'13*, pages 209–215, Dec 2013.
[18] V. Padmanabhan, H. Wang, P. Chou, K. Sripanidkulchai. Distributing Streaming Media Content Using Cooperative Networking. In *Proceedings of NOSSDAV'02*, pages 177–186. ACM, 2002.