Towards Enabling Hyper-Responsive Mobile Apps Through Network Edge Assistance

Miguel Báguena
Universitat Politècnica de València
mibaal@upvnet.upv.es

Andreas Pamboris
University of Cyprus
pamboris.andreas@cs.ucy.ac.cy

Peter Pietzuch
Imperial College London
prp@doc.ic.ac.uk

George Samaras
University of Cyprus
cssamara@cs.ucy.ac.cy

Mihail L. Sichitiu
NC State University
mlsichit@ncsu.edu

Pietro Manzoni
Universitat Politècnica de València
pmanzoni@disca.upv.es

***Abstract*—Poor Internet performance currently undermines the efficiency of hyper-responsive mobile apps such as augmented reality clients and online games, which require low-latency access to real-time backend services. While *edge-assisted execution*, i.e. moving entire services to the edge of an access network, helps eliminate part of the communication overhead involved, this does not scale to the number of users that share an edge infrastructure. This is due to a mismatch between the scarce availability of resources in access networks and the aggregate demand for computational power from client applications.**

Instead, this paper proposes a *hybrid edge-assisted deployment model* in which only part of a service executes on LTE edge servers. We provide insights about the conditions that must hold for such a model to be effective by investigating in simulation different deployment and application scenarios. In particular, we show that using LTE edge servers with modest capabilities, performance can improve significantly as long as at most 50% of client requests are processed at the edge. Moreover, we argue that edge servers should be installed at the core of a mobile network, rather than the mobile base station: the difference in performance is negligible, whereas the latter choice entails high deployment costs. Finally, we verify that, for the proposed model, the impact of user mobility on TCP performance is low.

I. Introduction

Recent technological trends such as the emergence of wearable devices (e.g. Google Glass), the availability of ubiquitous wireless networks (e.g. 3G, 4G/LTE and WiFi) and new cognitive algorithms that surpass human abilities in tasks such as computer vision, speech recognition and natural language translation, have brought a new class of hyper-responsive mobile apps to our doorstep. For example, augmented reality apps [1] retrieve digital information of users' surroundings, which is overlaid on a live view of the physical world in real-time; apps for hyper-local advertising [2] retrieve and display location-aware adverts promptly; and advanced online gaming clients [3] interact seamlessly with each other in shared virtual game worlds. For such applications to work as intended, clients require *ultra low latency access* to *real-time backend services*, ideally with at most tens of milliseconds of delay [4] in order to attain the speed of human perceptual and cognitive processing.

The underlying network that interconnects clients and services is often a major hindrance to achieving the seamless responsiveness users expect from smartphones and wearable devices. Backend services are typically hosted at distant cloud-based machines, which in many cases translates to hundreds of milliseconds of round trip time (RTT) delay between clients and services. An alternative is to re-locate entire services to the edge of the access network, so as to avoid high Internet delays. However, in light of the increasing number of mobile clients, it remains unclear to what extent this infrastructure can replace cloud-scale data centers due to the limited availability of resources at the network edge [5], [6].

We envision a *hybrid edge-assisted service deployment model*, which allows for moving only part of the backend logic into the access network, as the predominant model for supporting real-time services. Its advantage is that it can reduce network transmission delays for multiple mobile clients while respecting edge resource limitations. This paper explores the conditions that must hold in order to support hyper-responsive mobile apps at large scale using edge assistance. We first quantify the impact of 4G/LTE networks on application performance in terms of end-to-end latency and jitter, considering different types of networked application workloads. We then evaluate through simulation how performance can be improved using our proposed model, taking into account: (a) different placements of edge servers within the LTE infrastructure; (b) the number of clients sharing edge resources; (c) the ratio of user requests served by edge servers over the total requests made by a particular mobile client; (d) different processing capabilities of edge servers; and (e) user roaming.

Our key insights from this work are summarised below:
(1) The proposed model can improve collectively the performance of mobile clients by avoiding part of the communication over slow Internet network links and, at the same time, it relieves edge servers from increased computational load.
(2) The difference in application responsiveness between the two alternative placements of edge servers, i.e. at the eNodeB or the Packet Data Network Gateway (PGW), is negligible. Therefore, in light of the associated deployment costs involved, the PGW should be preferred.
(3) Due to the limited processing capabilities of edge servers and the increasing number of clients sharing edge infrastructure, performance gains are only achieved when up to 50% of user requests are processed at the network edge.
(4) TCP performance is not affected by user roaming, therefore, no adaptation to the transport protocol is required.

The remainder of this paper is organised as follows: §II describes 4G/LTE networks and discusses previous work on cloud-assisted execution as well as the current support for deploying services at the network edge; §III introduces the hybrid edge-assisted service deployment model; §IV evaluates the proposed model in simulation; and §V concludes the paper.

II. Background

LTE networks. 4G/LTE networks consist of three main components: *User Entities (UEs)*, commonly referred to as mobile devices; *eNodeBs*, i.e. base stations communicating directly with UEs; and the *Evolved Packet Core (EPC)*, which comprises two main components for data transmission: the *Serving Gateway (SGW)*, which connects eNodeBs to the EPC; and the *Packet Data Network Gateway (PGW)*, which provides access to external IP networks. The primary communication protocol used is the GPRS Tunnelling Protocol (GTP). To handle user mobility, the two main types of handover events supported are the X2- and S1-based handovers. This work considers the latter, which is the most time-consuming of the two and therefore more likely to degrade TCP performance.

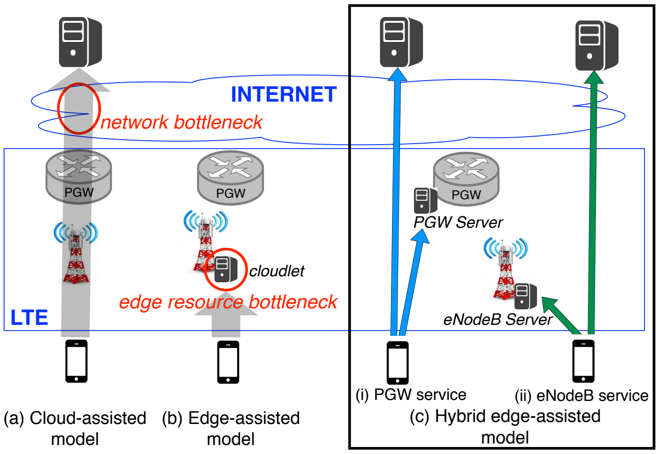
Cloud-assisted execution. Systems that offload CPU-intensive computation from mobile devices to the cloud are the predominant remedy to work around resource and power limitations of today's mobile devices. Existing approaches either treat mobile devices as thin clients by offloading all of an application's logic to the cloud [7], or employ a more fine-grained application partitioning for increased performance gains [8], [9], [10], [11].

Despite the increasing popularity of code-offloading approaches, their efficiency remains restricted by network performance. Therefore, it has been proposed to leverage edge infrastructure, referred to as *cloudlets*, to replace distant cloud-based nodes when possible [12]. Besides improving application responsiveness, the ability to host applications at the network edge has been exploited in different ways, e.g. to relieve mobile networks from unused traffic using edge proxies that filter the data sent to mobile devices [13].

Cloudlet-based approaches presuppose the availability of enough resources at the edge of the network to accommodate mobile users that share the edge infrastructure. Especially in densely populated urban environments, moving backend services in their entirety to the edge of a mobile network requires edge resources that are comparable to what data centres offer today. However, this assumption is impractical as it requires significant changes to current mobile network infrastructure and therefore entails high operational costs.

Support for edge-assisted execution. The need for fast access to real-time backend services has led to re-visiting the concept of edge computing. This is evidenced by the *Mobile Edge Computing (MEC)* initiative [14], a new Industry Specification Group within the European Telecommunications Standards Institute (ETSI) that aims at providing application developers and content providers with cloud-computing capabilities at the edge of the mobile network.

The release of new edge computing platforms such as *IBM ASPN* [15] is a step towards this direction, which allows for running and managing applications directly within mobile networks, provided that the required edge infrastructure is in place. Nevertheless, the capabilities of *edge servers* installed in mobile networks are still limited by form-factor and placement constraints [16]. An example is the new *Radio Applications Cloud Server (RACS)* by Nokia Networks [6], designed explicitly to fit in base stations, albeit with limited processing and storage capabilities. In this paper, we investigate how we can work around such limitations to take advantage of the available edge infrastructure to benefit many users collectively.



(a) Cloud-assisted model (b) Edge-assisted model (c) Hybrid edge-assisted model

Fig. 1. Hybrid edge-assisted deployment model

III. Proposed Hybrid Deployment Model

This section positions the proposed *hybrid edge-assisted service deployment model* in relation to extreme approaches that either deploy services in their entirety at cloud-scale data centres or at the edge of an access network. A comparison of all three deployment models is shown in Figure 1.

Currently the prevalent deployment model for real-time backend services is the *cloud-assisted model* illustrated in Figure 1(a). Services execute at distant cloud-based servers with virtually unlimited resources. Although in terms of computational resources, this model scales well to an increasing number of mobile clients, it suffers from high Internet delays during the communication between mobile clients and services.

At the other end of the spectrum, a *pure edge-assisted model*, i.e. Figure 1(b), moves entire services closer to users, thus eliminating the communication overhead over the Internet. While this model requires no changes to existing services, it is constrained by the fact that resources at the edge are significantly less plentiful than what is available in large data centres. Therefore, this approach cannot scale to the demand of increasing numbers of clients sharing the edge infrastructure.

The *hybrid edge-assisted deployment model*, shown in Figure 1(c), sits between the aforementioned approaches. The intuition behind this model is that it prioritises client requests based on latency requirements and splits services accordingly to execute across a cloud and an edge environment. This allows for improving app responsiveness for many clients while reducing the resource burden on edge servers.

We investigate the effectiveness of this hybrid model in the remainder of this paper. Note that how to realise the actual split of services into remote and edge components is specific to each service and thus not the focus of our work. Service providers, however, can base such decisions on the insights drawn from this study.

IV. Experimental Study

Through experiments in simulation, we evaluate the performance of edge-assisted applications in 4G/LTE networks. We first describe the simulation set-up and our experimental methodology. We then analyse the results, highlighting the benefits and drawbacks of different configurations for the proposed hybrid deployment model.

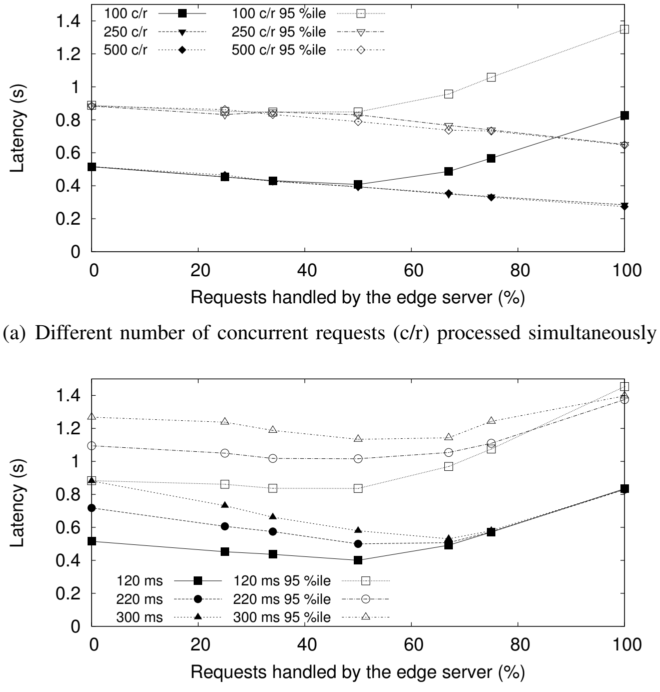
A. Simulation setup and methodology

We perform experiments in a simulated 4G/LTE network using the *OMNeT++* framework. We extended three publicly available models to simulate a complete 4G access network: INET[1] provides the tools for simulating a wired network and related protocols; SimuLTE[2] models the physical wireless link in a 4G network, as well as the link-level protocols between mobile devices and eNodeBs; and 4GSim[3] implements the network gateways and the control protocols that allow mobile devices to interact with control devices.

Simulation scenario. We focus on a 4G/LTE access network that contains two eNodeBs with several mobile devices attached to them. The eNodeBs are connected with the Evolved Packet Core (EPC) through a node that allows us to inject arbitrary delays in the access network, thus modelling the heterogeneity in current access networks. *Edge servers* can be placed at the eNodeB or the EPC, i.e. next to the PGW.

Edge servers have limited processing, disk and memory resources due to their form-factor limitations compared to servers located outside of the access network. We model this by enforcing a maximum number of requests that an edge server can handle concurrently; additional requests are queued until a new processing slot becomes available. We model the probability that a request is served by the edge server (and not a cloud-based server) according to a variable service rate.

Application network traces. We use network traces to evaluate the performance of the proposed hybrid model that are based on three distinct app traffic patterns with different characteristics: (1) the first pattern is characterised by large response packet sizes and low processing delays. This is common for apps such as augmented reality clients that perform image recognition on the mobile device and retrieve increasing amounts of data from a remote server to be superimposed on the device's camera output; (2) the second pattern exhibits the opposite behaviour: apps receive smaller packets but the backend logic is more CPU-intensive, thus leading to high processing delays. A representative example is a role-playing online game [17] in which users exchange information such as their location within a shared virtual world and various combat actions, which need to be processed remotely to determine how these affect other clients; and (3) the third pattern presents a balance between the two previous traces, thus representing any other client app with intermediate delays and packet sizes.



(a) Different number of concurrent requests (c/r) processed simultaneously

(b) Different Internet latencies

Fig. 2. Average latency and 95th percentile for different percentages of requests processed by the PGW edge server

TABLE I. Characterisation of workload traces

Representative application	Inter-request delay	Request size	Processing delay	Response size
Augmented reality	50-100 ms	20-30 B	50-100 ms	700-900 B
Online games	200-250 ms	20-30 B	200-250 ms	100-200 B
Other	100-150 ms	20-30 B	100-150 ms	400-600 B

For all traces, we assume non-concurrent request/response interactions between individual clients and services—only one request per client is pending a response at any point in time. For mobile client apps, typically the size of a single request is small (tens of bytes) compared to the size of responses, which ranges from hundreds to even thousands of bytes. A detailed analysis of the traces, including packet sizes, processing delays and inter-request delays (due to application-level delays or user idle states), is given in Table I. The LTE latency and delays due to hardware constraints are added by our simulator.

Due to space constraints, and given that the evaluation results obtained for all three traffic patterns exhibit similar trends, we only present the results that correspond to the online gaming use case, which is the most compute-intensive of them all. However, we highlight any significant differences observed for the other two patterns when applicable.

B. Results

Next, we evaluate the performance of apps with edge assistance for different configurations. We consider the average request/response latency and its 95th percentile as the primary metrics for performance. This latency includes the transmission delays for both requests and corresponding responses and the server-side processing delay.

1) Edge server collocated with PGW: First, we assume that edge servers are attached to the PGW gateway. To identify the conditions that must hold for this configuration to achieve better performance, we vary the ratio of requests that are served

[1] http://inet.omnetpp.org

[2] http://simulte.com/

[3] https://github.com/4gsim/4Gsim

entirely at the network edge and either (i) the number of concurrent requests that can be processed simultaneously by the edge server; or (ii) the Internet latency.

Variable edge server capability. Figure 2(a) plots the average and the 95th percentile of the latency per request for different fractions of requests processed at the network edge and different capabilities of the PGW edge server. We assume 500 cooperating clients. The access network and Internet latencies are set to 25 ms [18] and 120 ms [19], respectively.

When most requests are directed to the backend server, the average application latency is above 500 ms and dominated by the request/response transmission delays. *By allowing more requests to be served by the edge server, high Internet delays are avoided and latency gradually decreases. However, when the processing capacity of the edge server saturates (at around 50% of requests), due to server queuing, latency increases to a point where edge-assisted execution is outperformed by the original configuration.* The 95th percentile curves show that by increasing the number of requests processed at the edge server, jitter (inferred by the difference between the average and 95th percentile values) also increases due to CPU contention.

Similar trends are observed for the other types of traces. However, the time to process a request is less, therefore queues are less congested, which reduces queuing delays.

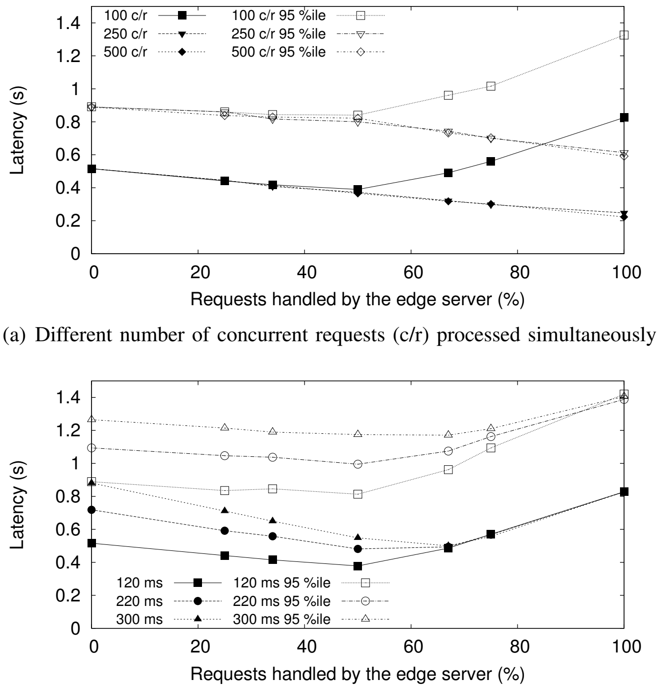
Variable Internet latency. Next, we focus on the impact of Internet latency on the performance of client apps. We consider three distinct Internet latency values: 120 ms, 220 ms and 300 ms, which correspond to the latency between a client in Spain and a server in the UK, the USA and Japan, respectively. These values were empirically obtained using ICMP Echo messages, which were sent at random times during the day. As before, we set the access network latency to 25 ms, while the processing capability of the edge server is now fixed to 100 concurrent requests, with a total load of 500.

Figure 2(b) shows that the performance degrades significantly when more requests are sent to the backend server for processing. With edge-assisted execution, however, savings in latency of approximately 300 ms for the worst of conditions are possible. *In summary, the higher the Internet latency, the more beneficial edge-assisted execution becomes. This trend is common for all configurations until the processing capacity of the edge server becomes a bottleneck, i.e. when more than 50% of client requests are handled by the edge server.*

2) Edge server collocated with eNodeB: We repeat the same experiments and now assume that the edge server is collocated with the eNodeB rather than the PGW gateway.

Variable edge server capabilities. Figure 3(a) shows the application latency when the processing capacity of the edge server is limited to 100, 250 and 500 concurrent requests. As before, edge-assisted execution improves application responsiveness up to the point at which queuing at the edge server is a major source of delay, i.e. when more than 50% of requests are processed at the edge.

Variable Internet latency. Figure 3(b) shows the achievable performance when varying the Internet latency for this configuration. The results obtained are almost identical to the previous results from Figure 2(b)—moving the edge server from the PGW gateway to the eNodeB base station reduces network delays by a negligible amount, thus improving performance only marginally.



(a) Different number of concurrent requests (c/r) processed simultaneously

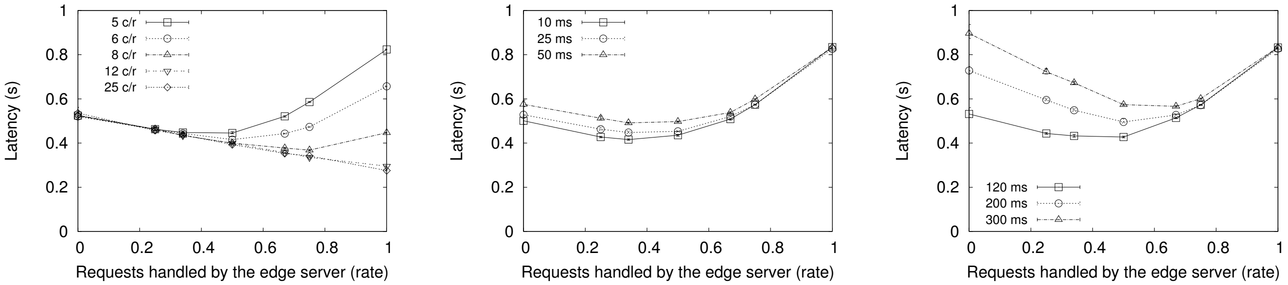
(b) Different Internet latencies

Fig. 3. Average latency and 95th percentile for different percentages of requests processed by the eNodeB edge server

In general, there are few differences between the two alternative choices for placing the edge server, i.e. the eNodeB or the PGW. For traffic patterns that cause less congestion at the edge server, a reduction in latency by at most 50 ms (approximately 18% using the online gaming app as reference) is possible by moving the edge server from the PGW to the eNodeB. However, compared to other sources of delay such as the 300 ms Internet latency, the 200 ms average processing delay and queuing delays when the edge server has limited resources, a 50 ms latency reduction is negligible. In addition, one needs to take into account the cost of deploying edge servers at base stations. In any case, this marginal performance gain can be negated during handover events (see Section IV-B4).

3) Inter-user delay: We also investigate the effect of the hybrid model on cooperating clients that share a particular service. For example, in online games such as first-person shooter games, multiple clients interact with each other in a shared virtual game world. Each client request thus leads to a broadcast of the corresponding response to all clients. For this experiment, we measure the inter-user delay, i.e. the time for a request to be sent and processed plus the time to propagate any state changes to all other affected users. We set the number of cooperating clients to 25 in order to avoid extra delays due to network saturation. The remaining configuration parameters are set according to the values used in the previous section.

Figure 4(a) shows the average time for clients to receive updates due to a request made by others, when the server is placed at the PGW. As before, we vary (1) the number of concurrent requests that can be processed by the edge server; and (2) the ratio of requests served entirely at the network



(a) Different edge server processing capabilities

(b) Different access network latencies

(c) Different Internet latencies

Fig. 4. Latency of broadcast events for different percentages of requests processed by the edge server

edge. The results follow closely those described in the previous experiments, suggesting that *this scenario does not create any asymmetries that change the average delay drastically*. The error bars indicate the maximum and minimum delays experienced, showing that variability in inter-user delay is low.

A similar trend is observed in Figure 4(b). We assume that the edge server is capable of processing only a maximum of 20% of the overall concurrent requests issued by the clients. In Figure 4(c), we repeat the same experiment but for varying Internet latencies (the access network latency is set to 25 ms). *The results indicate that the higher the Internet latency, the more beneficial edge-assisted execution becomes.*

4) Inter-cell mobility and handover events: Next we investigate the impact of mobility on application latency under the hybrid deployment model. We run a set of experiments that simulate a single handover event to understand how the additional delays introduced affect performance.

We assume that a mobile client roams between adjacent cells at 20 m/s and waits for 20 s in each cell before moving again. We assume a total of 500 mobile clients, initially split equally among two cells. A client can be in either of two states during simulation: (1) the *pre-handover state* in which a client is connected to the originating cell before the handover event takes place; and (2) the *post-handover state* in which the client is connected to an adjacent cell after the handover finishes. We set the access network latency to 25 ms and the Internet latency to 120 ms. We vary both the number of concurrent requests that an edge server can handle in parallel and the percentage of requests that are processed at the edge server.

Figure 5 shows the TCP behavior when simulating multiple handover events. Figure 5(a) displays handover events as vertical lines and RTO timeouts as distinct points in the graph. Several retransmissions occur close to most handover events. These are mostly due to the bad quality of the channel at the boundaries of a cell, with many of them occurring before the handover initiates. Figure 5(b) shows the TCP window during a handover event, which is initiated 72.5 s in the experiment. The vertical lines illustrate the handover's start and end times. As shown in the figure, the server's TCP window increases for almost half a second after the handover starts, therefore, it is unlikely that this is what causes the subsequent timeout. *We infer that the impact of handovers on TCP performance is small, with the impact of packet losses due to bad reception at the boundaries of a cell being far more important.*

Figure 5(c) shows the latency before and after a handover event for the online game traces. We observe a smooth transition between the two base cases, i.e. when all requests are either processed at the edge server or they are directed to the backend server for processing. For the latter, both edge server placements yield similar results in the pre-handover state. In the post-handover state, however, the eNodeB placement exhibits a larger degradation in performance, which is due to the fact that each request must traverse the access network three times: once for the request to reach the SGW; once to be forwarded to the original eNodeB; and one last time to reach the backend server. In contrast, when the server is placed at the EPC, packets go through the access network only once.

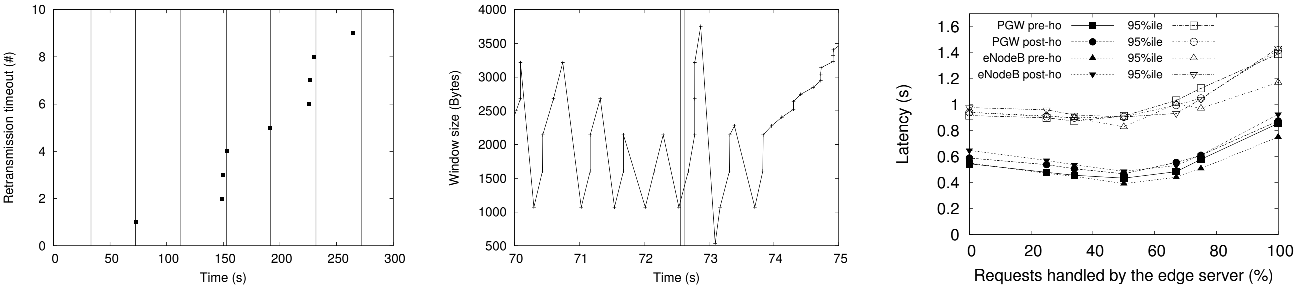
As shown in previous sections, when all requests are handled at the network edge, better performance is achieved in pre-handover conditions for the eNodeB placement. However, in post-handover conditions, performance is worse for some applications because packets must traverse the access network twice to reach the edge server on the other eNodeB.

The above results reinforce the observation that there is no significant gain in placing the edge server at the eNodeB instead of the PGW. In fact, the marginal performance improvement comes at the cost of degraded performance when handovers occur.

5) Latency breakdown: Figure 6 shows the breakdown of end-to-end application latency for the online gaming traces, when the edge server is able to process 100 concurrent requests in parallel and 50% of the client requests are processed at the network edge. The backhaul and Internet latencies are set to 25 ms and 120 ms, respectively.

When requests are processed at the edge server, the eNodeB placement slightly outperforms the PGW placement because it avoids sending data through the access network. When requests are processed by the backend server, the application latency is significantly higher because requests and responses are sent to/from the distant backend server.

Figure 6 shows that with a hybrid edge-assisted deployment model, processing delays become by far the largest contributors to application latency. This is not the case for the cloud-assisted model, for which network delays due to request/response delivery times dominate performance. Overall, an improvement of 40-45% in performance is possible when approximately 50% of client requests are served at the network edge, despite multiple users competing for the comparatively limited computational resources available there.



(a) Retransmission timeouts during handover events

(b) Window evolution in a handover event

(c) Average latency and 95th percentile for different percentages of requests

Fig. 5. TCP behaviour during handover events

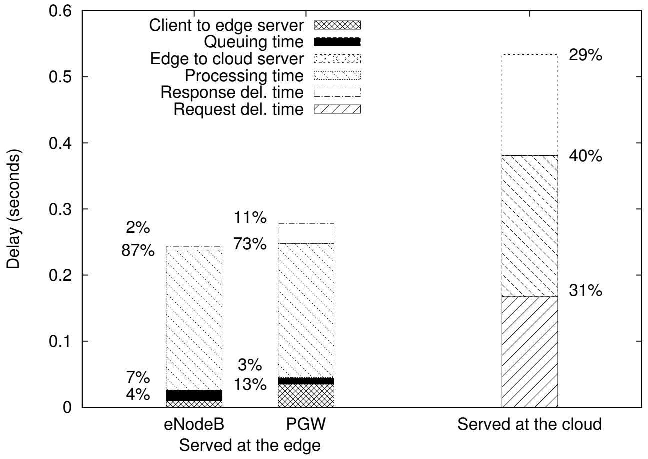


Fig. 6. Breakdown of application latency

V. Conclusions

In this paper, we investigated the feasibility of a hybrid edge-assisted service deployment model, which supports the hosting of real-time services jointly on cloud- and edge-based servers. This model can help reduce application latencies over LTE for many mobile clients that share an edge infrastructure, despite the comparatively limited resources available at the edge. Our simulation-based experiments provide insights regarding the conditions that must hold for this model to yield performance gains with many mobile clients.

Contrary to existing wisdom in mobile edge cloud proposals, we showed that there is no advantage in hosting services at servers collocated with the eNodeB as opposed to the PGW because the performance gain is not enough to compensate for additional deployment costs involved. Moreover, we showed that a combination of both locally and remotely served requests is required in order to realise benefits in performance—to avoid contention on resource-constrained edge servers, up to 50% of the total requests should be processed at the network edge. Finally, we confirmed that the impact of user mobility on TCP is negligible in such a scenario.

Acknowledgments

This work was supported by the *Ministerio de Economía y Competitividad* Spain, under Grant TEC2014-52690-R, and the *Ministerio de Educación, Cultura y Deporte* Spain, under Grant AP2010-4397. Further support came from *EIT Digital* as part of the 13130 Multi-Cloud Data Management (MC-Data) activity under the Future Cloud action line. The work is a result of a mobility stay funded by the Erasmus Mundus Programme of the European Commission under the Transatlantic Partnership for Excellence in Engineering (TEE) Project.

References

[1] Digital Trends. *Best Augmented Reality Apps*, 2014. http://www.digitaltrends.com/mobile/best-augmented-reality-apps/.

[2] Marketing Land. *How Hyperlocal Mobile Advertising Changes Everything*, 2014. http://marketingland.com/hyperlocal-mobile-advertising-changes-everything-92979.

[3] Phone Arena. *15 Best FPS/TPS Games for Android, iPhone, iPad and Windows Phone*, 2015. http://goo.gl/7M4o6d.

[4] Nokia Networks. Cloudlets: At the Leading Edge of Mobile-Cloud Convergence, 2014.

[5] Shiqiang Wang, Guan-Hua Tu, et al. Mobile Micro-Cloud: Application Classification, Mapping, and Deployment. In *AMITA*, 2013.

[6] Nokia Networks. Nokia Siemens Networks, Intel work together to transform mobile broadband experience. http://goo.gl/vC1anR, 2013.

[7] Eric Y. Chen and Mistutaka Itoh. Virtual Smartphone Over IP. In *WoWMoM*, 2010.

[8] Eduardo Cuervo, Aruna Balasubramanian, et al. Maui: Making Smartphones Last Longer with Code Offload. In *MobiSys*, 2010.

[9] Sokol Kosta, Andrius Aucinas, et al. ThinkAir: Dynamic Resource Allocation and Parallel Execution in the Cloud for Mobile Code Offloading. In *INFOCOM*, 2012.

[10] Mark S. Gordon, D. Anoushe Jamshidi, et al. COMET: Code Offload by Migrating Execution Transparently. In *OSDI*, 2012.

[11] ByungGon Chun, Sunghwan Ihm, et al. CloneCloud: Elastic Execution between Mobile device and Cloud. In *EuroSys*, 2011.

[12] Mahadev Satyanarayanan, Paramvir Bahl, et al. The Case for VM-Based Cloudlets in Mobile Computing. *Pervasive Comput.*, 2009.

[13] Andreas Pamboris and Peter Pietzuch. EdgeReduce: Eliminating Mobile Network Traffic Using Application-Specific Edge Proxies. In *MobileSoft*, 2015.

[14] MEC. *Executive Briefing Mobile Edge Computing (MEC) Initiative*, 2014. https://goo.gl/muBg5J.

[15] IBM. *IBM ASPN*, 2013. http://goo.gl/ZHJtxs.

[16] Intel. *Increasing Mobile Operators Value Proposition With Edge Computing*, 2014. https://goo.gl/NVcPnA.

[17] Xiaofei Wang, Hyunchul Kim, et al. Measurement and Analysis of World of Warcraft in Mobile WiMAX Networks. In *NETGAMES*, 2009.

[18] Markus Laner, P. Svoboda, et al. A comparison between one-way delays in operating HSPA and LTE networks. WiOpt, 2012.

[19] W. Matthews and L. Cottrell. The PingER project: active Internet performance monitoring for the HENP community. *IEEE Commun. Mag.*, 2000.