Distributed User-centric Service Migration for Edge-Enabled Networks

Lucas Pacheco*†, Denis Rosário†, Eduardo Cerqueira†, Leandro Villas‡,
Torsten Braun*, and Antonio A. F. Loureiro§

* University of Bern, Bern – Switzerland
† Federal University of Pará (UFPA), Belém – Brazil
‡ University of Campinas (UNICAMP), Campinas – São Paulo – Brazil
§ Federal University of Minas Gerais (UFMG), Belo Horizonte – Minas Gerais – Brazil
Email: {lucas.pacheco, torsten.braun}@inf.unibe.ch, {denis, cerqueira}@ufpa.br, leandro@ic.unicamp.br, loureiro@dcc.ufmg.br

***Abstract*—Edge-enabled scenarios are composed of all sorts of heterogeneous devices, networks, and services. Such devices can cooperate in providing ubiquitous service availability for users in both fixed and mobile networks. A multi-tier edge-enabled network is a suitable scheme for provisioning cloud services ubiquitously with QoS support. However, in mobile scenarios, the network may need to migrate services and applications to follow user mobility and avoid disruptions, posing a great challenge for the network and cloud mobility management solution. This paper proposes an intelligent service migration and resource management algorithm for intra- and inter-tier communication in edge-enabled environments, called Migration On Edge-Enabled Scenarios (MOES). MOES considers mobility prediction, service requirements, and monetary costs to perform an intelligent service migration decision. MOES also manages service migrations as devices leave the coverage area of their networks. MOES was implemented in a realistic edge-enabled heterogeneous scenario in which simulation results show its efficiency in delivering the necessary service requirements by up to 72% in terms of latency, at a cost 58% lower in terms of the number of migrations compared to state-of-the-art algorithms.**

I. Introduction

Modern urban scenarios are characterized by computing and communication's joint presence, providing a set of new digital services.The worldwide adoption of smart city technologies will be accelerated in the next few years, which will change how new services will be produced and offered by the industry, transportation, health, educational, security, and entertainment systems [1]. Hence, smart city applications heavily depend on the ubiquitous presence of mobile devices, cloud services, and wireless communication networks. In such scenarios, mobility management is a crucial factor in guaranteeing good connectivity and QoS for users. The mobility management algorithms in place must support requirement matching, resource control and be executed proactively since these are critical issues for mobile applications' success in edge-enabled networks [2], [3].

In urban, intelligent service providers and administrators should consider mobility at a management level, given mobile devices' heavy presence in these scenarios. Besides, traditional cloud-based deployments cannot support all applications with the required QoS level due to, for instance, high response time to a particular service requisition. When these services are kept closer to users at the network edge, learning and intelligent algorithms can provide personalization and optimize the user experience and service provisioning. For instance, it is essential to consider the application requirements individually rather than a one-size-fits-all solution. In this context, an edge-enabled network environment organized in multiple tiers is the most prominent approach to provide such requirements to a wide range of mobile applications for such scenarios and to support a wide range of services [4]. Edge nodes can be deployed almost anywhere within the mobile user-to-cloud path, leading to a hierarchical deployment of the edge nodes [5]. Hence, edge-enabled environments can offer high bandwidth and low latency, which will be essential to many mobile applications. However, such approaches' computing power is still inferior to traditional cloud computing and must be carefully managed. While some tiers can closely follow user mobility, others are static and have a specific coverage area in which they better function [6].

To best provide user needs, services must be migrated between intra- and inter-tier to keep the services as close as possible to the mobile users, considering the trade-off between minimal QoS requirements for mobile users and costs for using the edge-node [7]. Service migration consists of transferring the service running on a virtual machine or container or the service session to a new edge node. Also, each tier may charge the different monetary cost of its utilization per node. It is essential to avoid unnecessary costs to deploy and maintain a service, which may not be fully supported by its node. An efficient service migration must occur proactively to solve these issues, taking into account service requirements, mobility, and edge node information to reduce the chance of service disruptions and disconnections. Some studies in the literature [8], [9] consider a centralized service allocation and migration controller. However, this might not scale appropriately in large IoT scenarios, ranging from thousands to millions of devices [10]. Distributed control is essential to accomplish a service migration at this scale.

II. Related Works

Location data is essential to perform a correct migration decision. Some studies investigate how the user location can

978-3-903176-32-4

be securely transmitted and used by a migration framework without being compromised by any third parties. Wang *et al.* [9] considered the risk of leakage for the location data and user-perceived delay in a Markov Decision Process. However, other metrics are essential to provide services with a satisfactory experience, such as cost, QoS, and application requirements. Another vital factor, typically not considered, is predicted locations, which can provide a better decision framework.

Other optimization mechanisms are investigated, as well. Liang *et al.* [11] proposed a swarm-based migration scheme that considers service migrates to new edge nodes to better provide application requirements by predicting queue times. It shows the importance of employing local edge controllers instead of a centralized model for scalability reasons. However, the optimization strategy can be complex in some cases, making an online decision to live migrations difficult. In such scenarios, a test evaluation and decision are crucial to induce latency or disconnections to the user services.

The computing resources in the edge node are typically constrained and, thus, must be carefully allocated. To solve this problem, Zhai *et al.* [12] considered a mechanism to offload tasks from overloaded edge nodes to less requested ones as a load balancing strategy at the edge. While this has advantages, edge nodes are not homogeneous from a user's point of view, and task allocation is influenced by a wide range of variables that vary over time.

Wu *et al.* [13] employed a user-centric location prediction based on Location-based Social Networks (LSBNs) in edge-enabled scenarios. A factor graph model joins user locations and the correlation between user movements to perform optimizations to the migration optimizations. Other solutions have proposed using LSBN [13] data for location prediction in the past, but it must deal with problems such as privacy and the sparsity of the data. Several strategies can be applied to solve these issues. However, when it comes to real-time location prediction for network management, we need to advance further.

III. MOES

A. System Model

We consider a scenario comprised of a set of n mobile devices $d \in D = \{1, 2, \ldots, n\}$, such as vehicles, smartphones, sensors, wearables, and others. We consider a set of k edge nodes $u \in U = \{1, 2, \ldots, k\}$ deployed anywhere in the network, organized in tiers between the mobile devices (at the bottom) and the cloud (at the top) [5]. Each edge node u has heterogeneous characteristics in computing power, costs to deploy and maintain a given service, and throughput and latency it can achieve. Specifically, the cloud node offers a high throughput and computing capabilities but does not support low-latency services. On the other extreme, we have the mist tier, in which user devices can form a personal cloud to serve that user's applications. Figure 1 illustrates the scenario described.

B. Mobility Prediction

To perform a seamless service migration, MOES relies on a mobility prediction algorithm to forecast handovers and

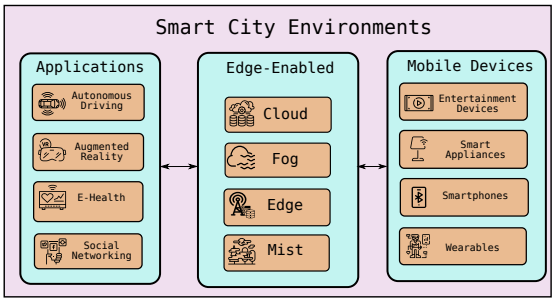

Figure 1: Heterogeneous Multi-tier Edge-enabled Scenarios Overview

computes the best edge server for each service and device. Different strategies are proposed in the state-of-the-art, such as those based on user trajectory [14] and signal strength [15], which provide reliable handover forecasting. We consider a trajectory-based forecasting scheme [14], given its general reliability and reduced complexity. The forecasting consists of predicting the cell coverage area a given mobile device d will be in the foreseeable future. MOES performs position prediction and signal estimation, when executing a traditional handover algorithm, such as the Strongest Cell handover algorithm [16]. We consider a mobility prediction based on ARIMA, a robust and presents a low complexity forecaster, as it learns statistical properties of time series. Previous studies have established ARIMA's efficiency in assisting predictive network functions, such as handover and service migrations [17].

An ARIMA model is described as a 3-tuple (p, d, q), where p corresponds to past measurements weighted in the estimation, d consists of the number of differentiation applied to the series to make statistically stationary (with an approximately constant average), and q corresponds to the number of moving average terms considered. The basic formulation of the model is given by Eq. 1. We denote past terms as y, past moving averages as ϵ, while θ and ϕ are individual weights for each term and trained by the model. We found that the configuration $ARIMA(5, 1, 0)$ yields the best results in our scenario through a grid-search, and different scenarios may require different configurations.

$$\begin{aligned} y_t = \theta_0 + \phi_1 y_{t-1} + \phi_2 y_{t-1} + \phi_3 y_{t-3} + \cdots + \phi_p y_{t-p} \\ \epsilon_0 + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-1} + \theta_3 \epsilon_{t-3} + \cdots + \theta_q \epsilon_{t-q}. \end{aligned} \quad (1)$$

ARIMA is used to forecast a single-variable time series, and, thus, it has to perform a training step separately for the latitude and longitude measurements. The ARIMA model can be used for device mobility prediction $L(x_i, y_i, t+1)$. In this sense, the model must be trained for each mobile device separately for each coordinate (*i.e.*, x, and y).

We use the predicted position as input to a path loss model to estimate the most potent cell at a given moment. In a signal-based handover algorithm, this has relatively high accuracy, as cells tend to be not so close to each other in a way that signal fluctuations have a significant affect —- the propagation model used by MOES is the CI, or Close In, propagation loss model

[18]. We use the CI propagation loss model according to Eq. 2, where the computed PL is the path loss value, FSPL(f, 1m) is the loss at a one-meter distance from the transmitter, n is the loss exponent, and X_σ is a Gaussian noise component with average 0 and standard deviation σ.

$$PL(f,d) = FSPL(f,1m) + 10n\log_{10}(d) + X_\sigma \quad (2)$$

C. Intelligent Migration Decision

MOES considers three fundamental parameters to decide about service migration in a multi-tier edge-enable scenario, namely, service requirement (req), mobility prediction ($mobil$), and cost to deploy and maintain a given service s in an edge node u ($cost$). We consider the ARIMA model for mobility prediction. Service requirement req specifies the set of the edge node's service requirements u. This is done by assessing the application's minimal requirements (*e.g.*, latency and bandwidth) and what each edge node can provide to make a better match between them. In contrast, mobility prediction $mobil$ means the predicted staying time in an edge node u, *i.e.*, the amount of time the edge node u will be available based on device mobility.

We chose the Analytical Hierarchical Problem (AHP) [19] method to adjust their weights of each parameter at runtime since it considers a pairwise comparison between the numerical values and their relative degrees of importance. For each mobile device d, MOES constructs a matrix $A = (C_{i,j})_{nxn}$ to compare all pairs of metrics, where n denotes the number of elements to be compared, as shown in Eq. 3. We denote $c_{i,j}$ as how important the $i-th$ element is compared with the $j-th$ element. The comparison matrix M indicates which parameters have higher priority than others, as shown in Eq. 3. The application requirement metric Mobility is the most important, *e.g.*, mobility is two times more important than the mobility parameter Cost, and four times compared to Requirements. As a result, we obtain the eigenvector $E = [0.571\ 0.286\ 0.144]$, which means that requirements, mobility prediction, and edge node cost have weights 0.571, 0.286, and 0.144, respectively. Note that the matrix M's consistency ratio is 0% (were lower than 10% is acceptable).

$$M = \begin{array}{c} \\ Mobil \\ Cost \\ Req \end{array} \begin{array}{c} \begin{array}{ccc} Mobil & Cost & Req. \end{array} \\ \begin{pmatrix} 1 & 2 & 4 \\ 1/2 & 1 & 2 \\ 1/4 & 1/2 & 1 \end{pmatrix} \end{array} \rightarrow \quad (3)$$

$$\rightarrow [0.571\ 0.286\ 0.144]$$

Finally, MOES calculates a product as the eigenvector and a vector that stores the measured values, obtaining each available edge node's score. MOES calculates the product between the The complete functioning of MOES is described in Algorithm 1. MOES must know the services to be run and their requirements. For each service, MOES evaluates if its respective node meets its requirements. The algorithm performs the handover prediction, and if any is predicted, it performs a migration in advance. Suppose the application is not critical or can support disruptions. In that case, it is preferred to wait for the handover to happen than to maintain two instances of the service during the migration. The algorithm describes the service migration process, in which each node collects the parameters, and the AHP score is calculated. If the chosen node does not have free computing resources, the second-best node is selected, and so on until a node is selected.

Algorithm 1: MOES Algorithm

```
Data: Services Being Run
for service ∈ services do
    get service requirements;
    probe latency and throughput;
    if latency and throughput do not meet requirements
     or handover detected then
        Query available nodes;
        for node in nodes do
            get the node performance metrics;
            calculate the node's AHP score
        while Node has not been chosen do
            if node with the highest score has free
             computing resources then
                migrate to the node with the best AHP
                 score;
            else
                Pop node from the stack, and migrate
                 to the second-best node;
```

IV. Evaluation

A. Scenario description and methodology

We implemented MOES, as well as three state-of-the-art service migration algorithms, using the NS-3.31[1] simulator, which implements the LTE protocol stack for mobile communication. In our scenario, we consider a 2.5 km $\times$ 2.5 km area containing 60 ENodeBs randomly allocated, as well as 100 user devices, respectively. The simulation considers the Nakagami path loss model, which is considered suitable for urban scenarios [20]. We conducted 33 simulations with different randomly generated seeds. Results show the values with a confidence interval of 95%.

For traffic and mobility simulation, we employed the Luxembourg mobility trace[2]. This realistic mobility trace takes place in the city of Luxembourg. Pedestrian mobility models have also been implemented following a random walk mobility model.

Besides MOES, we also implemented three service migration algorithms. Two additional algorithms are implemented as a baseline: (i) a greedy algorithm where a migration happens whenever a handover is triggered; Furthermore, (ii) no-migration, where the services are fixed in place, and no migration is triggered for the simulation duration. In this approach, the services are rerouted from the nodes to the

[1]http://www.nsnam.org/

[2]https://github.com/lcodeca/LuSTScenario

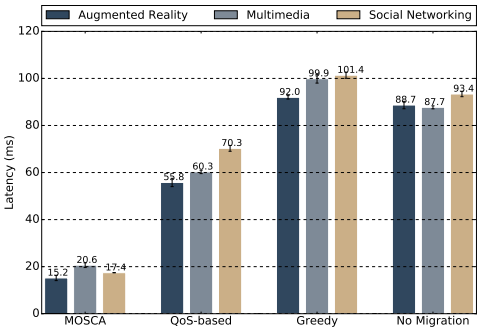

Figure 2: Average Latency For all Applications Under Each of the Tested Algorithms

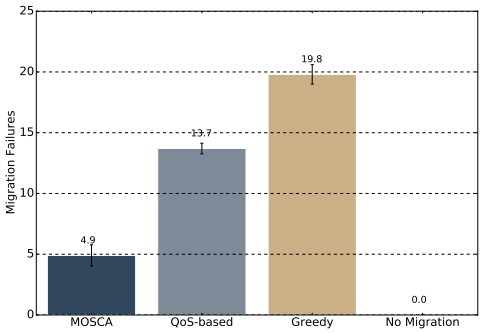

Figure 3: Migration Failures Under Each of the Tested Algorithms

user as the network topology changes. As a state-of-the-art representation, (iii) we also compared MOES with the algorithm presented by Wu *et al.* [21], discussed in Section II, and referenced here as QoS-based migration. We simulate three applications: Augmented Reality (AR), Multimedia entertainment considering a video flow, and Social Networking.

B. Results

Figure 2 shows the average latency achieved by each algorithm and for each application. In the case of AR, the maximum recommended latency is about 20 ms [22], and MOES is the only algorithm to provide this requirement with a 15 ms average latency, 72% lower than the other algorithms. For entertainment services, the latency was kept below 50 ms only by MOES, whereas the Location-based, Greedy, and No-migration algorithms achieved 68 ms, 96 ms, and 87 ms, respectively. MOES still maintains a lower latency for social networking than the competing algorithms, except for the QoS-based one. This is because MOES's execution services are preferably allocated to the nodes that best suit their requirements in a predictive way so that the network can find the best nodes and perform the migration without disrupting the service.

This might happen due to insufficient resources in the target node, or user mobility may cause the user to leave the node's coverage before the transferring is complete. Figure 3 shows the number of migration failures, on average, under each algorithm. The no-migration approach did not perform any migrations, so it has no failures. The greedy strategy has the highest number of failures due to the lack of mobility prediction and resource awareness. MOES has 58% fewer migration failures than the QoS-based approach, and 73% fewer than the Greedy Migration, maintaining it in about 8 failures per simulation. This happens because the QoS-based approach does not include any resource checking in its decision.

The cost of edge node utilization is also taken into account in the simulations. We apply real edge node prices with the Amazon AWS TCO Calculator[3]. We notice that due to having most services closer to their respective user devices, in the mist, fog, and edge layers, MOES displays the most

[3]https://awstcocalculator.com/

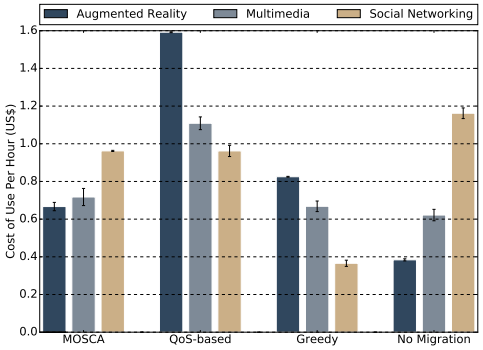

Figure 4: Cost of Utilization for Each of the Tested Algorithms

significant use cost among the tested algorithms, especially for highly demanding applications as AR. However, the cost is close to a state-of-the-art proposal serving as a baseline. In the AR case, it is necessary to provide the lowest latency nodes for the application, maintaining the minimum Quality of Service expected by users. For the entertainment and social networking applications, MOES uses a strategy that avoids over-provisioning, maintaining lower cost than the concurring algorithms.

V. Conclusion

This paper introduced a mobility-, requirements-, and cost-based service migration algorithm in an intelligence edge-enabled environment and highly heterogeneous scenarios, called MOES. MOES takes into account individual service requirements, user mobility, and node cost to make a migration decision. MOES performs a mobility prediction of the node to define when migration becomes necessary. Then, it executes a Multiple-Criteria Decision-Making scheme to find the most appropriate node for the application/service. Based on simulation results, MOES improves the delivery of applications with different QoS requirements, such as AR, entertainment, and social networking. MOES delivers demanding applications with up to 72% improvement in latency while causing 58% fewer migration failures than state-of-the-art algorithms. As future work intends to explore real-time learning and intelligent-edge computing possibilities more profoundly, enabling further personalization for end-users.

REFERENCES

[1] T. L. Inn, "Smart city technologies take on covid-19," *World Health*, 2020.

[2] Y.-S. Chen and Y.-T. Tsai, "A mobility management using follow-me cloud-cloudlet in fog-computing-based rans for smart cities," *Sensors*, vol. 18, no. 2, p. 489, 2018.

[3] Y. Deng, Z. Chen, X. Yao, S. Hassan, and J. Wu, "Task scheduling for smart city applications based on multi-server mobile edge computing," *IEEE Access*, vol. 7, pp. 14 410–14 421, 2019.

[4] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li, "A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet," *ACM Computing Surveys (CSUR)*, vol. 52, no. 6, pp. 1–36, 2019.

[5] D. Rosário, M. Schimuneck, J. Camargo, J. Nobre, C. Both, J. Rochol, and M. Gerla, "Service migration from cloud to multi-tier fog nodes for multimedia dissemination with qoe support," *Sensors*, vol. 18, no. 2, p. 329, 2018.

[6] L. Pacheco, H. Oliveira, D. Rosário, E. Cerqueira, L. Villas, and T. Braun, "Service migration for connected autonomous vehicles," in *2020 IEEE Symposium on Computers and Communications (ISCC)*, 2020, pp. 1–6.

[7] L. Pacheco, D. Rosário, E. Cerqueira, and L. Villas, "Service migration in edge computing environments for connected autonomous vehicles," in *Proceedings of the 38th Brazilian Symposium on Computer Networks and Distributed Systems*. Porto Alegre, RS, Brasil: SBC, 2020, pp. 533–546. [Online]. Available: https://sol.sbc.org.br/index.php/sbrc/article/view/12307

[8] M. Zhang, H. Huang, L. Rui, G. Hui, Y. Wang, and X. Qiu, "A service migration method based on dynamic awareness in mobile edge computing," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–7.

[9] W. Wang, S. Ge, and X. Zhou, "Location-privacy-aware service migration in mobile edge computing," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2020, pp. 1–6.

[10] T. V. Doan, Z. Fan, G. T. Nguyen, D. You, A. Kropp, H. Salah, and F. H. Fitzek, "Seamless service migration framework for autonomous driving in mobile edge cloud," in *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2020, pp. 1–2.

[11] L. Liang, J. Xiao, Z. Ren, Z. Chen, and Y. Jia, "Particle swarm based service migration scheme in the edge computing environment," *IEEE Access*, vol. 8, pp. 45 596–45 606, 2020.

[12] Z. Zhai, K. Xiang, L. Zhao, B. Cheng, J. Qian, and J. Wu, "Iot-recsm—resource-constrained smart service migration framework for iot edge computing environment," *Sensors*, vol. 20, no. 8, p. 2294, 2020.

[13] Q. Wu, X. Chen, Z. Zhou, and L. Chen, "Mobile social data learning for user-centric location prediction with application in mobile edge service migration," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7737–7747, 2019.

[14] R. Arshad, H. ElSawy, S. Sorour, T. Y. Al-Naffouri, and M. Alouini, "Cooperative handover management in dense cellular networks," in *IEEE Global Communications Conference (GLOBECOM)*, Dec 2016, pp. 1–6.

[15] A. Miyim, M. Ismail, R. Nordin, and G. Mahardhika, "Generic vertical handover prediction algorithm for 4g wireless networks," in *IEEE International Conference on Space Science and Communication (IconSpace)*. IEEE, 2013, pp. 307–312.

[16] 3GPP, "Evolved universal terrestrial radio access (e-utra); radio resource control (rrc); protocol specification," *3GPP TS 36.331 V9.4.0 (2010-09), Evolved Universal Terrestrial Radio Access (E-UTRA);Radio Resource Control (RRC);Protocol specification(Release 9)*, 2011.

[17] A. Costa, L. Pacheco, D. Rosário, L. Villas, A. A. Loureiro, S. Sargento, and E. Cerqueira, "Skipping-based handover algorithm for video distribution over ultra-dense vanet," *Computer Networks*, vol. 176, p. 107252, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128620302188

[18] S. Sun, T. S. Rappaport, S. Rangan, T. A. Thomas, A. Ghosh, I. Z. Kovacs, I. Rodriguez, O. Koymen, A. Partyka, and J. Jarvelainen, "Propagation path loss models for 5g urban micro-and macro-cellular scenarios," in *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*. IEEE, 2016, pp. 1–6.

[19] T. L. Saaty, "Decision making—the analytic hierarchy and network processes (ahp/anp)," *Journal of systems science and systems engineering*, vol. 13, no. 1, pp. 1–35, 2004.

[20] L. Rubio, J. Reig, and N. Cardona, "Evaluation of nakagami fading behaviour based on measurements in urban scenarios," *AEU-International Journal of Electronics and Communications*, vol. 61, no. 2, pp. 135–138, 2007.

[21] Q. Wu, X. Chen, Z. Zhou, and L. Chen, "Mobile Social Data Learning for User-Centric Location Prediction with Application in Mobile Edge Service Migration," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7737–7747, 2019.

[22] F. Zheng, T. Whitted, A. Lastra, P. Lincoln, A. State, A. Maimone, and H. Fuchs, "Minimizing latency for augmented reality displays: Frames considered harmful," in *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2014, pp. 195–200.