Weathering the Reallocation Storm: Large-Scale Analysis of Edge Server Workload

Lauri Lovén, Ella Peltonen, Erkki Harjula and Susanna Pirttikangas
University of Oulu
Oulu, Finland
Email: {first.last}@oulu.fi

***Abstract*—Efficient service placement and workload allocation methods are necessary enablers for the actively studied topic of edge computing. In this paper, we show that under certain circumstances, the number of superfluous workload reallocations from one edge server to another may grow to a significant proportion of all user tasks – a phenomenon we present as a reallocation storm. We showcase this phenomenon on a city-scale edge server deployment by simulating the allocation of user task workloads in a number of scenarios capturing likely edge computing deployments and usage patterns. The simulations are based on a large real-world data set of city-wide Wi-Fi network connections in 2013–2014, with more than 47M connections over ca. 800 access points. We identify the conditions for avoiding the reallocation storm for three common edge-based reallocation strategies, and study the latency–workload trade-off related to each strategy. As a result, we find that the superfluous reallocations vanish when the edge server capacity is increased above a certain threshold, unique for each reallocation strategy, peaking at ca. 35% of top ES workload. Further, while a reallocation strategy aiming to minimize reallocation distance consistently resulted in the worst reallocation storms, the two other strategies, namely, a random reallocation strategy, and a bottom-up strategy which always chooses the edge server with the lowest workload as a reallocation target, behave nearly identically in terms of latency as well as the reallocation storm in dense edge deployments. Since the random strategy requires much less coordination, we recommend it over the bottom-up one in dense ES deployments.**

I. Introduction

Load balancing is a mission-critical topic for any computing service from cloud to local networking capabilities. In everyday language known as a "rock festival phenomena", certain events – whether predictable or not – can cause serious lacks in the quality of service or even exterminate the service for an amount of time. Even though certainly studied in the cloud computing context, it is not clear if the similar load-balancing strategies fit for the resource-constrained edge computing world, due to many differences related to e.g. coherence of the network topologies and variance in server capabilities. In this paper, we highlight how *horizontal and vertical workload* become the bottleneck of edge computing services, how *reallocation strategies* designed for clouds fail in the edge computing environment, and how *reallocation storms*, defined in the next section, can be avoided.

Edge computing has become the de facto strategy for bringing computational capabilities into the local environments to reduce latency between clients and traditional cloud services [1]. Harnessing the local computational capabilities does not only remove networking load but also enables several real-time applications characterized by hard time-constraints and other critical resources. Such application areas include, among others, augmented reality [2], vehicular computing [3], [4], connected healthcare [5], and Industry 4.0 systems and services [6]. With increasing interest in edge intelligence, or EdgeAI, bringing artificial intelligence and machine learning on the edge [7], [8], there will be a set of applications critically reliable of edge resources and availability.

Workload management is a widely studied topic in the context of server clusters of (cloud) data centers [9] and grid computing [10]. In edge computing, workload management must, however, deal with e.g. user mobility and higher variance in server and network topologies and capacities, thus making it a distinct research topic. Workload management on the edge can be handled with different strategies, such as the physical placement of edge servers [11]–[13] or reallocating services on the software-side with different optimization algorithms [6]. Reallocation can rely on known edge server features, such as capacity, or their current state, such as load. The simplest option may be to reallocate workload to the nearest available edge server, or, if a number of candidates are produced, one selected randomly amongst these [14]. More detailed hybrid models can combine edge server placement, resource allocation and run-time reallocation, also optimizing for proximity [15]. Some of these heuristic reallocation algorithms can minimize both computing and network delay, penalizing for longer computing times of over-capacity edge servers [16], [17].

Furthermore, cloud offloading from the edge environment [18], [19] is sometimes a viable option. In some cases, however, cloud offloading cannot take place due to, for example, requirements for low latency, such as on the vehicular edge [20] or with mobile augmented reality [2], where either critical services or user's quality of experience cannot be compromised by increased latency.

In this paper, we identify a phenomenon we name as *reallocation storm*, where reallocations trigger new reallocations, and the number of these superfluous reallocations grows to a significant proportion of all user tasks executed on the edge. We discuss the circumstances where this phenomenon happens, what are its consequences, and how it can be avoided. Our contributions can be summarized as follows:

- We identify the phenomenon on a real-world data set of city-wide Wi-Fi network connections with more than 47M connections over ca. 800 access points.
- We demonstrate how popular reallocation strategies can lead to the reallocation storm, with up to 15-20% of all tasks reallocated needlessly.
- We show that the reallocation storm is highly linked to

the capacity of the edge servers and vanishes when the edge server capacity is increased above a certain threshold, unique for each reallocation strategy, peaking at ca. 35% of top workload.

II. Definitions and Scoping

Workload. Workload is the computational burden on edge servers (ES) or cloud, caused by user tasks accumulating on those servers [11], [21]. Workload may be the result of, for example, user applications or their components offloaded on the edge servers [22]; by edge applications following users, migrating from one ES to another [23], [24]; or by cloud applications being onloaded to edge servers for low-latency interaction with users or environment [22], [25].

Workload allocation. In offline edge server placement, expected workload from user tasks is allocated on the edge servers as a part of the process of finding the optimal placement for the servers. Workload allocation designates each access point (AP) (at least) one edge server, which acts as the default server for the workload from the users of that AP when the edge deployment is online [11]. ESs serve two types of workload: *vertical* workload is caused by user tasks on the APs allocated to the ES, while *horizontal* workload is reallocated from other ESs. The grid computing paradigm can be seen analogous to edge computing in the sense that the grid data centers serve both vertical and horizontal workload [26].

Reallocation. Reallocation is an online process, where the allocation of an AP is changed [6]. Reallocation may be caused by the allocated edge server being unavailable due to, for example, exceeded capacity. Depending on the edge computing architecture, reallocation may be the responsibility of an orchestrator [27].

Fig. 1 illustrates an example of reallocation. ES 1 is over capacity, and cannot accept the workload from AP 1.1. Consequently, AP 1.1 has to reallocate new user workload to either ES 2 or ES 3, or else offload the workload to cloud. Depending on the type of the edge application, reallocation may require the migration of data from one ES to another. For example, if the edge application in question maintains connection-specific session or state, that state needs to migrate.

Reallocation strategy. Reallocating workload, the target ES needs to be decided (Fig. 1). While the details vary, proposed strategies often fall into the following broad categories: **Cloud strategy** always offloads the workload to cloud, if the allocated ES is over capacity [18], [19]. This is the default strategy the edge strategies may revert to if they, for some reason, fail. **Edge strategies** try to find another ES in the edge deployment as a reallocation target. There are a number of subcategories:

- **Proximity strategy** reallocates workload to the nearest ES with available capacity [14]–[16].
- **Bottom-up strategy** reallocates workload to the ES with the lowest current workload [6], [26], [28]. This strategy does not minimize communication latency. Depending on the edge architecture, it may require constant communication between the ESs and an orchestrator to maintain up-to-date book-keeping of ES workloads, queue sizes, or minimum estimated task completion times for each ES.
- **Random strategy** reallocates workload to a random ES with available capacity. This strategy is also a possible fallback if another strategy fails or produces a number of possible candidates [14].

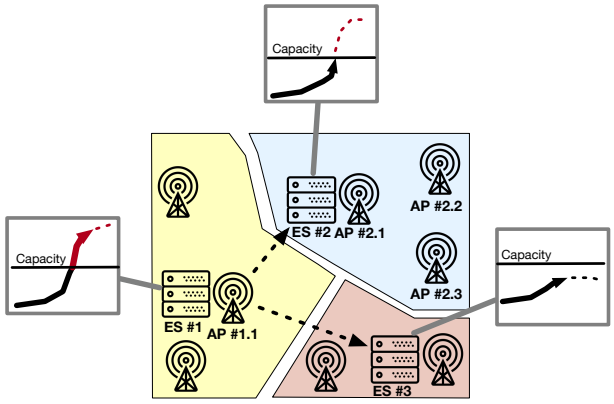

Fig. 1. **Reallocation of workload.** ES 1 is over capacity, so AP 1.1 may decide to reallocate new workload from its users to ES 2, the nearest ES (proximity strategy), ES 3, with the lowest workload (bottom-up strategy), or offload the workload to cloud (cloud strategy).

Superfluous reallocation. A poor choice of target ES with the chosen edge strategy may trigger another reallocation. In Fig. 1, if AP 1.1 chooses (or is orchestrated) to follow the proximity strategy and reallocate workload to ES 2, the nearest one, ES 2 will exceed its capacity. Any subsequent vertical workload from any of ES 2's allocated APs (2.1, 2.2 or 2.3) must thus also be reallocated.

Reallocation storm. Under certain conditions, the number of superfluous reallocations may grow to a high proportion of all user connections. This happens when a high proportion of ESs are above or nearly above capacity, i.e., when the ES network as a whole is under high workload. In such conditions, superfluous reallocations may trigger yet another round of reallocations if other nearby ESs are also near capacity, et cetera. In this paper, we study the conditions leading to reallocation storms, their frequency and impact, and find ways to avoid them. We simulate the reallocation strategies, analyse the workload–latency trade-off for each of them, study how the reallocation storm is realized for each strategy, and discuss their merits and pitfalls.

III. Methods and Experimental Setup

Data. We use a real-world Wi-Fi network connection log as a basis for our analysis. The log data set is collected in the PanOULU public Wi-Fi network in the city of Oulu, Finland, in 2007–2015 [29]. We select the starting timestamps and the APs of all the connections during our observation period of 2013–2014, chosen to include the last two full years in the data. The total number of individual Wi-Fi connections during the observation period was 47.656.939, with the number of quarterly connections shown in Fig. 2. There are a total of 898 APs in the data set, 855 of which were active in the observation period. These APs are depicted in Fig. 3.

Assumptions. The topology of the underlying fixed network is not available in the data set. As a number of related work (see [11]), we assume the underlying network is homogeneous in relation to AP density, and approximate the latency between the APs with the Euclidean distance. We assume an edge deployment where a number of ESs of identical capacity are

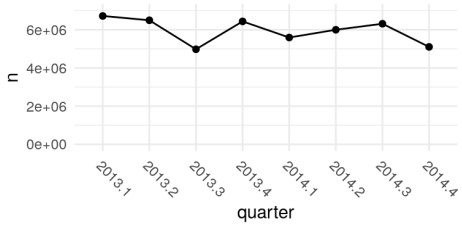

Fig. 2. **Number of quarterly connections.** The number of connections fluctuates slightly over the observation period 2012–2014, remaining between 5M and 7M connections quarterly.

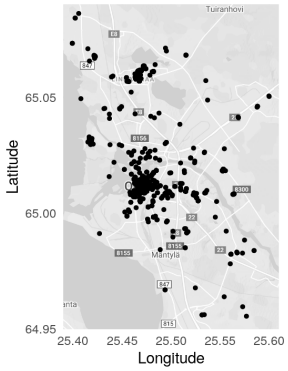

Fig. 3. **Access points.** Depicted on the map are the PanOULU public Wi-Fi network APs, active during the observation period of 2010–2014, located in the city of Oulu, Finland.

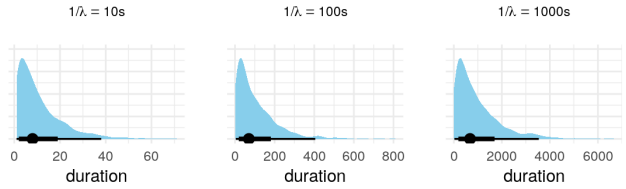

Fig. 4. **Empirical densities of the durations of simulated user tasks.** We sample the durations from the exponential distribution with three different means ($1/\lambda$): 10s, 100s and 1000s.

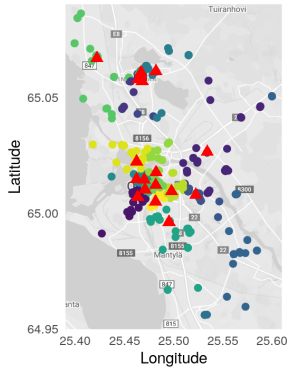

Fig. 5. **Example of edge server placement and allocation.** The example is derived with duration mean 100s and 20 edge servers (red triangles). APs (round dots) allocated to the same ESs are all painted in the same colour.

deployed at some of the access points in Fig. 3. The ESs serve user tasks of variable workload, one per user connection. To simplify the analysis, we use a pre-determined numeric value for the *capacity* of an ES, compared online to the combined workload of the tasks allocated to it. An ES may either be below the capacity, where it is assumed to function normally, or have exceeded its capacity, where it is no longer functioning normally, and will reallocate vertical workload (i.e., the workload of its allocated APs) and refuse horizontal workload (i.e., workload reallocated by other ESs). This reflects a scenario where the workload of an ES exceeds a threshold such that the processing time of a task becomes unacceptable.

Data augmentation. We enrich the base data set with simulated data on the duration of the user tasks and the normalized workload they incur on the edge servers during that time. The workloads are sampled from the log-normal distribution, with mean 1. The task durations are sampled from the exponential distribution with three alternative means ($1/\lambda$), corresponding to three different dominant application usage patterns: 1) $1/\lambda = 10$s, reflecting e.g. dominant messaging application usage; 2) $1/\lambda = 100$s, reflecting e.g. dominant mobile game usage; and 3) $1/\lambda = 1000$s, reflecting e.g. dominant AR/VR or other constant or near-constant app usage. Indeed, Hintze et al. [30] measure comparable session durations on mobile phone and tablet, and the empirical density of their measured durations roughly follows the shape of the sampled exponential densities (Fig. 4).

We split the enriched data to a training set we use for offline edge server placement, and a testing set we use for simulation of the online operation. The training–testing split is set to 0.8:0.2 respectively along the temporal axis, with the larger training set comprising ca. 38M user connections between Jan 1st, 2013, and July 29, 2014, and the testing set comprising ca. 9.5M connections between July 29, 2014 and Dec 31, 2014. We select two edge server placement schemes for serving the user workload aggregated through the APs: 1) 20 edge servers, corresponding to MEC or edge cloud, with a sparse deployment of high capacity edge servers serving all users; and 2) 150 edge servers, corresponding to a dense deployment with a large number of low capacity servers.

We employ the PACK placement method, based on capacitated clustering, and an R-based `rpack` software tool implementing the method [11]. We use the offline training data set to find the placement and allocation for the edge servers in both deployment schemas. The employed placement method interprets the clustering task as an optimization problem, minimizing an objective function comprising AP allocation and their distances from edge servers, with constraints for e.g. an upper and a lower limit for edge server capacity. Each AP is assigned a weight in relation to its maximum workload within the training set time period. The resulting ES placement and allocation for 20 edge servers and a user task duration mean of 100s is shown as an example in Fig. 5. Further, the first weeks of resulting online edge server workloads are depicted in Fig. 6.

Simulation and analysis. For each combination of duration mean (10s, 100s, 1000s) and edge deployment schema (20 ESs, 150 ESs), totaling 6 scenarios, we run a number of simulations. Each simulation assumes the ESs all have identical capacity,

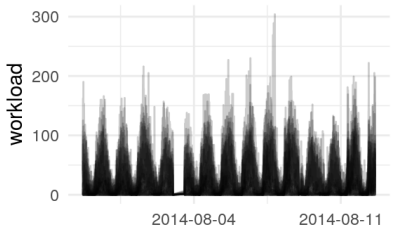

Fig. 6. **Example workloads on edge servers.** The example is derived with duration mean 100s and 20 edge servers. The circadian rhythm of workloads is clearly visible, as is a service outage around Aug-03.

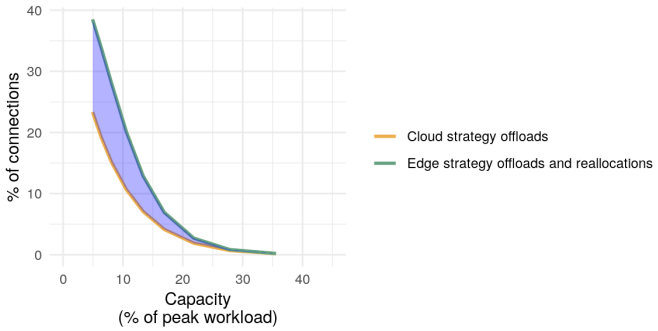

Fig. 7. **Superfluous reallocations**. The difference (blue area) between the number of offloads for cloud strategy (orange line) and the number of reallocations and offloads for an edge strategy (green line) is the number of superfluous reallocations, shown here as a proportion of all connections.

and that capacity is increased for each subsequent simulation. Further, for each scenario and each capacity, we simulate all of the strategies, namely, cloud, proximity, bottom-up and random. Further, we set the cloud strategy as the fallback for the edge strategies (proximity, random and bottom-up) in case all of the ESs are over capacity. For all of those strategies, we analyse the resulting workloads on the ESs, the number and workload of cloud offloads and reallocations, as well as the relative latency in the ES network.

The difference between the number of cloud offloads for the cloud strategy, and the sum of the number of cloud offloads and the reallocations for the edge strategies (proximity, bottom-up, random) gives us the number of superfluous reallocations:

$$P_E = R_E + O_E - O_C, \tag{1}$$

where P is the number of superfluous reallocations, R is the number of reallocations, and O is the number of cloud offloads, while the subscript E designates an edge strategy and C the cloud strategy.

Indeed, since the cloud offloads in the cloud strategy affect only the offloading ES, they give a baseline of the times when that ES was over capacity due to workload from its allocated APs. Since reallocation transfers workload from one ES to another, that extra workload on the target ES may cause there superfluous reallocations, which manifest as a number of reallocations exceeding the baseline. Further, the number of cloud allocations for the edge strategies, i.e. the times when an edge strategy had to revert to cloud offloading due to over capacity in all edge servers, are also included (Fig. 7). As proxies for latency in the ES network, we measure the distances between ESs along the great circles connecting them.

IV. Results

Results are depicted in Figs. 8 and 9. The superfluous reallocations for proximity strategy (in red) peak at ca. 12-17% of all connections for durations with mean 100 and 1000 seconds, and ca. 2% and 3% of connections for 20 and 150 edge servers for task durations with mean 10, respectively. Bottom-up (blue) and random (black) strategies consistently result in a lower number of superfluous reallocations.

Clearly, the proportion of superfluous reallocations is high, indicating a reallocation storm, when edge servers have low capacity. Indeed, a lower capacity is exceeded more quickly, resulting in more superfluous reallocations. However, at very low capacity values, we see the number of superfluous reallocations suddenly dropping in many scenarios. This is likely the result of the number of regular (i.e. not superfluous) reallocations growing so high that they start to dominate over the superfluous ones.

The higher the mean of the task duration, the wider the storm. For the proximity strategy, the capacity where the reallocation storm ends, that is, where the proportion of superfluous reallocations drops below 0.1% of all connections, peaks at ca. 35% of top ES workload for task duration means 100s and 1000s, with 20 ESs.

The effect of the number of edge servers is inverse. The more edge servers, that is, the denser the ES deployment, the narrower the storm, with the proportion of superfluous reallocations always lower with 150 ESs than with 20 ESs. In more detail, superfluous reallocations are caused by user workload that is reallocated to a server, which exceeds its capacity within the duration of the task. Such an event is less likely to happen when there are more ESs around.

On the other hand, average reallocation distances (Fig. 9), as a proxy to communication latency in the network connecting the ESs, show that the proximity strategy results in consistently shorter distances than the other two strategies. Interestingly, with a high number of ESs, random strategy and bottom-up strategy have nearly identical distances. Since they also all but agree on the number of superfluous reallocations, the results suggest employing the random strategy instead of the bottom-up strategy in dense ES deployments due to its lower communication overhead and simpler decentralization. In sparser deployments, and with short average task duration, the bottom-up strategy may however provide a slight edge over the random strategy in terms of the number of superfluous reallocations, at the cost of some additional latency.

V. Conclusion

We employed a real-world large-scale Wi-Fi connection dataset for the simulation of workload in a number of edge computing scenarios. While the study considered the Wi-Fi deployment of one geographical area, earlier studies [11], [12] have shown the deployment is representative of an edge deployment spanning urban areas with a high AP density as well

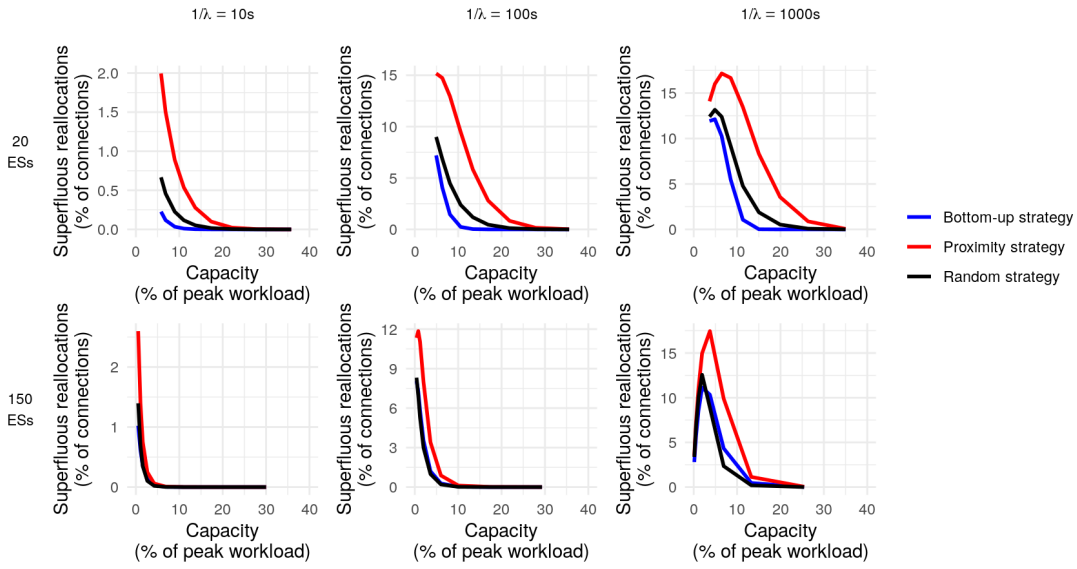

Fig. 8. **Superfluous reallocations in each scenario**. Proximity strategy (red) produces the severest reallocation storm in each scenario, while bottom-up (blue) and random (black) strategies result in much lower numbers of superfluous reallocations.

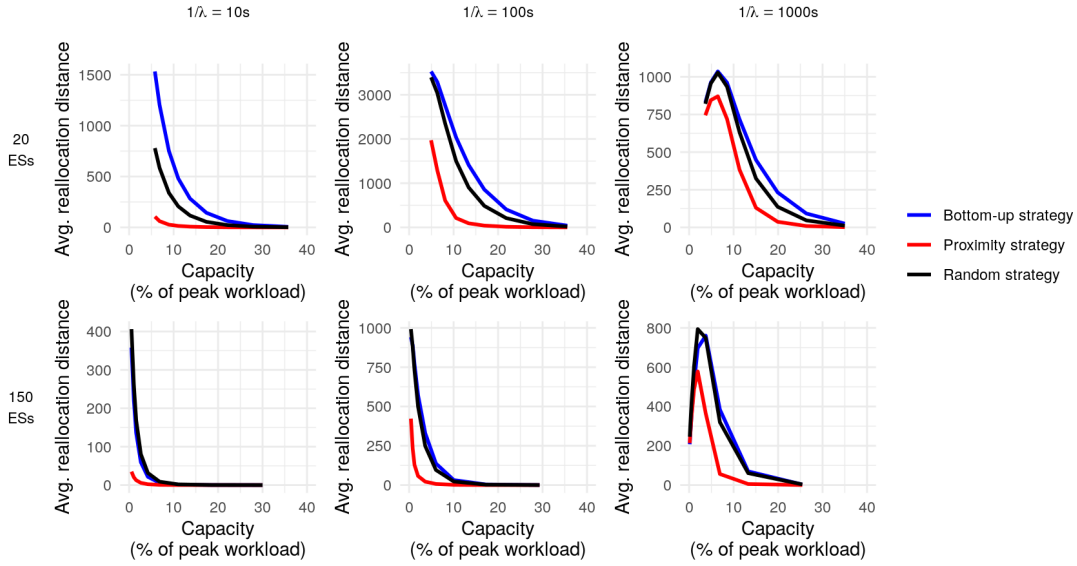

Fig. 9. **Average reallocation distances in each scenario**. Proximity strategy (red) results, as expected, in the shortest distances in each scenario, while bottom-up (blue) and random (black) strategies result in somewhat higher distances.

as sub-urban areas with a low AP density. Further, while the studied period was as early as 2013–2014, we augmented the data to accommodate potential future edge use cases with usage patterns ranging from predominantly short to predominantly long, sampling the actual task execution times and workloads from realistic distributions.

We studied four distinct strategies, namely, the cloud, proximity, bottom-up and random strategies, for reallocating workload when edge servers exceed their capacity. We discovered a reallocation storm with a large number of superfluous reallocations, triggered when a task is reallocated to an ES whose capacity is exceeded within the duration of the task.

The proximity strategy results in the highest number of superfluous reallocations in all conditions. Further, the superfluous reallocations vanish when capacity was increased above a certain threshold, unique for each strategy, peaking at ca. 35% of top ES workload for the proximity strategy in a sparse ES deployment with heavy workload. Further, with dense ES deployments, bottom-up and random strategies were roughly equal in terms of both superfluous reallocations and latency. Since random strategy requires much less coordination, we recommend it over bottom-up in dense ES deployments.

Limitations. The study simplified edge server capacity to one scalar, exceeding which lead to ES unavailability. In reality, capacity is a more complex concept, with execution slowing down with increasing workload. Especially for edge applications where reallocation is a lightweight operation (e.g., reallocation does not require the transfer of application data from one ES to another), reallocation could be triggered well before the actual capacity of the ES is reached. However, this can be reflected in the study by employing a lower capacity value.

Impact. In general, the value of these findings is threefold. First, avoiding the reallocation storm improves QoS/QoE and resource-efficiency, as superfluous reallocations causing network burden and increasing latency are minimized. Second, edge operators have more information for selecting the most suitable reallocation strategy. Finally, the results point towards future studies.

Future work. We plan to further study the conditions leading to reallocation storms, focusing on the spatio-temporal aspects of the phenomenon. In particular, we are interested in when and where do they appear, how long do they last, and how they propagate through the edge network. Further, we plan to study refined methods for avoiding the storm while optimizing ES capacity by means of novel reallocation strategies.

ACKNOWLEDGEMENTS

This research is supported by Academy of Finland 6Genesis Flagship and DigiHealth programs (grants 318927, 326291); the ECSEL JU FRACTAL (grant 877056), receiving support from the EU Horizon 2020 programme and Spain, Italy, Austria, Germany, France, Finland, Switzerland; Infotech Oulu research institute; the Future Makers program of Jane and Aatos Erkko and Technology Industries of Finland Centennial foundations; and the personal grant for Lauri Lovén on edge-native AI research by the Tauno Tönning foundation.

REFERENCES

[1] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[2] T. Braud, P. Zhou, J. Kangasharju, and P. Hui, "Multipath computation offloading for mobile augmented reality," *IEEE Int. Conf. on Pervasive Computing and Communications (PerCom)*, no. January, 2020.

[3] J. Lin, W. Yu, X. Yang, P. Zhao, H. Zhang, and W. Zhao, "An edge computing based public vehicle system for smart transportation," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, 2020.

[4] K. Zhang *et al.*, "Optimal delay constrained offloading for vehicular edge computing networks," in *IEEE Int. Conf. on Communications (ICC)*. IEEE, 2017, pp. 1–6.

[5] P. Pace *et al.*, "An edge-based architecture to support efficient applications for healthcare industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 481–489, 2019.

[6] Y. Zhang, C. Pang, and G. Yang, "A real-time computation task reconfiguration mechanism for industrial edge computing," in *The Annual Conf. of the IEEE Industrial Electronics Society (IECON)*. Singapore: IEEE, 2020, pp. 3799–3804.

[7] L. Lovén, T. Leppänen, E. Peltonen *et al.*, "EdgeAI: A vision for distributed, edge-native artificial intelligence in future 6G networks," in *The 1st 6G Wireless Summit*, Levi, Finland, 2019, pp. 1–2.

[8] E. Peltonen, L. Lovén *et al.*, *6G white paper on edge intelligence*. Oulu, Finland: 6G Flagship, University of Oulu, 2020.

[9] T. C. Chieu, A. Mohindra, A. A. Karve, and A. Segal, "Dynamic scaling of web applications in a virtualized cloud computing environment," in *2009 IEEE International Conference on e-Business Engineering*, 2009, pp. 281–286.

[10] A. Rahman, X. Liu, and F. Kong, "A survey on geographic load balancing based data center power management in the smart grid environment," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 214–233, 2014.

[11] T. Lähderanta, L. Lovén, T. Leppänen, L. Ruha, E. Harjula, M. Ylianttila, J. Riekki, and M. J. Sillanpää, "Edge computing server placement with capacitated location allocation," *Journal of Parallel and Distributed Computing*, 2021.

[12] L. Lovén, T. Lähderanta, L. Ruha, T. Leppänen, E. Peltonen, J. Riekki, and M. J. Sillanpää, "Scaling up an edge server deployment," in *IEEE Int. Conf. on Pervasive Computing and Communications Workshops (PerCom Workshops)*. Austin, TX, US: IEEE, 2020, pp. 1–7.

[13] L. Ruha, T. Lähderanta, L. Lovén, T. Leppänen, J. Riekki, and M. J. Sillanpää, "Capacitated spatial clustering with multiple constraints and attributes: A case study in edge server placement," *arXiv preprint arXiv:2010.0633v3*, 2021.

[14] W. Chang *et al.*, "An offloading scheme leveraging on neighboring node resources for edge computing over fiber-wireless (FiWi) access networks," *China Communications*, vol. 6, no. 11, pp. 107–119, 2019.

[15] X. Niu *et al.*, "Workload allocation mechanism for minimum service delay in edge computing-based power internet of things," *IEEE Access*, vol. 7, 2019.

[16] Q. Fan and N. Ansari, "Application aware workload allocation for edge computing-based IoT," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2146–2153, 2018.

[17] S. Wang *et al.*, "A machine learning approach for task and resource allocation in mobile edge computing based networks," *IEEE Internet of Things Journal*, 2020.

[18] Y. Miao *et al.*, "Intelligent task prediction and computation offloading based on mobile-edge cloud computing," *Future Generation Computer Systems*, vol. 102, pp. 925–931, 2020.

[19] C. Wu, Y. Zhang, and Y. Deng, "Toward fast and distributed computation migration system for edge computing in IoT," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 041–10 052, 2019.

[20] Y. Chen, J. P. Walters, and S. P. Crago, "Load balancing for minimizing deadline misses and total runtime for connected car systems in fog computing," in *IEEE Int. Symp. on Parallel and Distributed Processing with Applications and IEEE Int. Conf. on Ubiquitous Computing and Communications (ISPA/IUCC)*, 2017, pp. 683–690.

[21] J. Qadir *et al.*, "Towards mobile edge computing: Taxonomy, challenges, applications and future realms," *IEEE Access*, vol. 8, no. October, pp. 189 129–189 162, 2020.

[22] A. Yousefpour *et al.*, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *Journal of Systems Architecture*, vol. 98, no. Feb, pp. 289–330, 2019.

[23] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, 2018.

[24] T. Taleb *et al.*, "Mobile edge computing potential in making cities smarter," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 38–43, 2017.

[25] K. Bhardwaj *et al.*, "Fast, scalable and secure onloading of edge functions using Airbox," *IEEE/ACM Symposium on Edge Computing*, pp. 14–27, 2016.

[26] Y. Caniou, G. Charrier, and F. Desprez, "Analysis of tasks reallocation in a dedicated Grid environment," in *IEEE Int. Conf. on Cluster Computing (ICCC)*. Heraklion, Crete, Greece: IEEE, 2010, pp. 284–291.

[27] Group Report, "GR MEC 031 - V2.1.1 - Multi-access Edge Computing (MEC) MEC 5G Integration," ETSI, Tech. Rep., 2020.

[28] D. Puthal, M. S. Obaidat, P. Nanda, M. Prasad, S. P. Mohanty, and A. Y. Zomaya, "Secure and sustainable load balancing of edge data centers in fog computing," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 60–65, 2018.

[29] V. Kostakos, T. Ojala, and T. Juntunen, "Traffic in the Smart City," *Internet Computing, IEEE*, vol. 17, no. 6, pp. 22–29, 2013.

[30] D. Hintze, P. Hintze, R. D. Findling, and R. Mayrhofer, "A large-scale, long-term analysis of mobile device usage characteristics," *ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 2, pp. 1—-21, 2017.