

Latency-aware Privacy-preserving Service Migration in Federated Edges

Paulo Souza¹^a, Ângelo Crestani¹^b, Felipe Rubin¹^c, Tiago Ferreto¹^d and Fábio Rossi²^e

¹*Pontifical Catholic University of Rio Grande do Sul, Porto Alegre, Brazil*

²*Federal Institute Farroupilha, Alegrete, Brazil*

Keywords: Edge Computing, Microservices, Edge Federation, Infrastructure Providers, Service Migration.

Abstract: Edge computing has been in the spotlight for bringing data processing nearby data sources to support the requirements of latency-sensitive and bandwidth-hungry applications. Previous investigations have explored the coordination between multiple edge infrastructure providers to maximize profit while delivering enhanced applications' performance and availability. However, existing solutions overlook potential conflicts between data protection policies implemented by infrastructure providers and security requirements of privacy-sensitive applications (e.g., databases and payment gateways). Therefore, this paper presents Argos, a heuristic algorithm that migrates applications according to their latency and privacy requirements in federated edges. Experimental results demonstrate that Argos can reduce latency and privacy issues by 16.98% and 7.95%, respectively, compared to state-of-the-art approaches.

1 INTRODUCTION


Edge computing brings the idea of extending the cloud, decentralizing computing resources from large data centers to the network's edge (Satyanarayanan et al., 2009). In practice, servers and other devices with processing capabilities are placed near data sources, reducing the communication delay imposed by the physical distance between end devices and computing resources that process data (Satyanarayanan, 2017).


Unlike traditional cloud deployments, edge computing sites comprise computing resources geographically distributed across the environment, possibly managed by different providers in a federated manner (Mor et al., 2019). This heterogeneous nature allows distributed software architectures such as microservices to run ahead of traditional monoliths as they enable multiple edge servers to handle the demand of resource-intensive applications (Mahmud et al., 2020a). At the same time, resource wastage can be avoided by scaling up/down microservices individually based on their demand, which is pleasing given the resource scarcity of the edge infrastructure.


The heterogeneous nature of edge computing environments and the dynamics of moving users increase the complexity of deciding where to allocate microservices. From a performance perspective, microservices should remain close enough to their users to ensure low latency while avoiding network saturation, which implies that migrations should take place according to users' mobility (Ahmed et al., 2015). While deciding when and where to migrate microservices based on users' mobility is already challenging, these allocation decisions get even harder to perform as infrastructures with multiple providers raise concerns about the level of trustworthiness of edge servers.


There has been considerable prior work regarding resource allocation in federated edge computing scenarios considering applications' privacy (Wang et al., 2020; He et al., 2019; Xu et al., 2019). Despite their contributions, migrating microservices according to performance and privacy goals still represents an open research topic. By studying the state-of-the-art, we identified some significant shortcomings in the existing migration strategies focused on enforcing the privacy of applications running on edge computing infrastructures:


- Most strategies equalize the level of trust of entire regions while making allocation decisions, overlooking the existence of servers managed by different providers in the same region yielding pos-

^a  <https://orcid.org/0000-0003-4945-3329>

^b  <https://orcid.org/0000-0002-1806-7241>

^c  <https://orcid.org/0000-0003-1612-078X>

^d  <https://orcid.org/0000-0001-8485-529X>

^e  <https://orcid.org/0000-0002-2450-1024>

sibly different levels of trust.

- Existing solutions assume the same privacy requirements for all microservices from an application, neglecting that microservices with different responsibilities may have distinct privacy requirements (e.g., database microservices may demand more privacy than front-end microservices).

This work presents Argos, a heuristic algorithm that coordinates the privacy requirements of application microservices and the levels of trustworthiness of edge servers while performing migrations based on users' mobility. In summary, we make the following contributions:

- We propose a novel heuristic, called Argos, for migrating services in edge computing which strengthens the safety of applications considering the privacy requirements of each application microservice and the level of trust between the user that accesses the application and edge servers.
- We quantitatively demonstrate the effectiveness of Argos through a set of experiments which shows that our solution can reduce violations in maximum latency and privacy requirements by 16.98% and 7.95%, respectively, against a state-of-the-art approach.

The remainder of this paper is organized as follows. In Section 2, we present a theoretical background on resource allocation in edge computing environments. In Section 3, we list the related work. Section 4 details the system model. Section 5 presents Argos, our proposal for enhancing the privacy of applications in federated edges. Section 6 describes a set of experiments used to validate Argos. Finally, Section 7 presents our final remarks.

2 BACKGROUND

The emergence of mobile and IoT applications (e.g., augmented and virtual reality, online gaming, and media streaming) increased the demand for low-latency resources and locality-aware content distribution in recent years (Satyanarayanan, 2017). Since the limited computational power and battery capacity available at clients' devices (mobile and IoT) cannot meet these applications' requirements, offloading computational-intensive tasks to remote processing units such as cloud data centers becomes necessary. However, despite the resiliency and seemingly unlimited resources of cloud computing, the inherent latency of long-distance transmissions cannot meet the ever-increasing performance requirements of applications (Satyanarayanan, 2017).

Edge computing tackles the latency issues of mobile and IoT applications, bringing data processing to the network's edge in proximity to end-user devices (Satyanarayanan et al., 2009). In contrast to centralized cloud data centers, edge servers are dispersed in the environment, possibly connected to the available power grid, allowing nearby devices to offload resource-intensive tasks with reduced latency, which grants improved performance while reducing the battery consumption of end-user devices.

To deliver services and process users' requests, edge infrastructure providers rely on virtualization techniques such as virtual machines (VMs) and containers (He et al., 2018). The virtualization of physical resources supports the design and implementation of resource allocation strategies for infrastructure providers, tailored accordingly for energy efficiency, network performance, monetary costs, and performance contracts such as Service Level Agreements (SLAs). However, there is a limit to the resources available from a single provider, especially on edge scenarios, that have more limited resources than their cloud counterparts (Li et al., 2021). A viable approach to mitigate capacity limitations is to allocate resources from multiple providers. In this context, edge federations employ strategies that incorporate resources from many providers, which benefit both providers and their clients (Anglano et al., 2020).

Whereas clients benefit from increased availability and scalability, infrastructure providers can reduce operational costs by combining owned and leased resources to meet their clients' demands. The advantages of a federation of edge providers are even more evident considering industry trends such as microservice-based applications, which comprise several loosely-coupled modules (also known as microservices), each with different performance, security, and privacy requirements. As microservices work independently, they can be distributed across resources from different providers to enhance the use of the infrastructure (Faticanti et al., 2020).

While the emergence of decoupled software architectures brings several benefits, it also yields resource allocation challenges for infrastructure providers. In addition to allocation problems inherited from the cloud, edge computing has unique privacy and security issues. Organizations in the same federation have different policies that affect allocation decisions. For instance, proactive caching of content using users' locations and usage patterns can lead to performance improvements but might not be acceptable for privacy-sensitive users (Zhu et al., 2021). Accordingly, there is a need for resource management strategies that drive allocation decisions not only by per-

formance goals but also based on providers' data protection policies and microservices' privacy requirements to avoid potential security flaws on federated edge computing infrastructures.

3 RELATED WORK

This section discusses existing work in two areas that intersect the scope of our study. Section 3.1 reviews studies that consider privacy requirements of edge applications. Section 3.2 presents studies focused on provisioning resources in federated edges. Finally, Section 3.3 compares our study to the related work.

3.1 Privacy-aware Service Allocation

As edge environments are typically regarded as a group of highly distributed nodes interconnected with network infrastructures less robust than their cloud counterparts (Aral and Brandić, 2020), proper security mechanisms are needed to ensure that network transmissions remain reliable. In this context, Xu et al. (Xu et al., 2019) present allocation strategies to ensure the privacy of applications' data transmitted over the edge network infrastructure. The proposed method splits and shuffles the application's data before transferring it throughout the network. Consequently, data leakage is less likely if some data chunks are compromised.

In typical edge computing scenarios with mobile users, latency-sensitive applications are migrated as users move across the map to ensure that latency keeps as low as possible. In this case, if attackers manage to identify the target host of applications being migrated, they can infer the location of nearby users in the map. To tackle this problem, Wang et al. (Wang et al., 2020) present a migration strategy that avoids migrating applications to hosts either too far or too close to their users. Thus, applications' latency keeps low while users' location remains secure.

As edge servers usually present resource constraints, applications can be accommodated on cloud servers when users do not request them after a while. In such a scenario, Qian et al. (Qian et al., 2019) argue that whether to migrate services from the cloud to the edge and vice-versa depends on users' preference for the services, which is known by analyzing their historical usage data. However, the authors discuss the risks of data leakage from transferring historical usage data across the network to external processing. Therefore, the authors propose a federated learning scheme that enables users to collect and analyze their own data usage and send updated offload parameters

to edge servers to determine services' locations accordingly.

Li et al. (Li et al., 2020) demonstrate the security challenges of offloading and transmitting data across edge network infrastructures, where attackers can estimate users' location based on their communication pattern to a given server. Consequently, the authors argue that always offloading applications to the closest server may not be the optimal privacy-preserving decision to make. Accordingly, the authors present an offloading strategy that finds the best trade-off between users' privacy, communication delay between users and their applications, and the infrastructure's energy consumption cost.

3.2 Resource Allocation in Federated Edges

In federated edges, an infrastructure provider can lease resources from other providers if it lacks resources to host all the applications. In situations like that, one of the main goals is finding the provisioning scheme that incurs the least amount of external resources used so that the allocation cost is as low as possible. To address that scenario, Faticanti et al. (Faticanti et al., 2020) present a placement strategy that allocates microservice-based applications on federated edges based on topological sorting schemes that define the order in which microservices are allocated by their position in their application's workflow.

For infrastructure providers that own cloud and edge resources, choosing the right place to host applications is key to balancing allocation cost and application performance. Whereas allocating applications at the edge guarantees low latency at high costs as resources are limited, cloud resources are abundant but incur in higher latencies. In such a scenario, Mahmud et al. (Mahmud et al., 2020b) present an allocation strategy that defines whether applications should be hosted by edge or cloud resources to increase the profit of infrastructure providers while delivering satisfactory performance levels for end-users.

Whereas low occupation periods are concerning for infrastructure providers, as the allocation cost can exceed the price paid by users, demand peaks may overload the infrastructure leading to revenue losses. In such a scenario, Anglano et al. (Anglano et al., 2020) present an allocation strategy where infrastructure providers cooperate to maximize their profit by sharing resources to process time-varying workloads.

As the popularity of mobile and IoT applications grows significantly, resource-constrained edge infrastructures may lack resources to accommodate all applications simultaneously. In such scenarios, deter-

mining proper placement and scheduling for applications is critical to ensure users satisfaction and minimize allocation costs. With that goal, Li et al. (Li et al., 2021) present a placement and scheduling framework called JCPS, which leverages the cooperation between multiple edge clouds to host applications based on their deadline and allocation cost.

3.3 Our Contributions

As discussed in previous sections, considerable prior work has targeted privacy concerns of edge applications and the heterogeneity of federated edge infrastructures. However, existing solutions present some significant shortcomings, discussed as follows.

First, most strategies assume the same privacy requirement for entire applications, overlooking that applications can be composed of multiple independent components that yield distinct privacy requirements. Second, existing solutions equalize the level of trust of entire regions, overlooking that each region can have servers managed by different providers with possibly distinct levels of trust.

Our solution, called Argos, represents the first steps toward performing service migrations according to users' mobility while considering service privacy requirements and users' level of trust on infrastructure providers. Table 1 summarizes the main differences between Argos and the related work.

Table 1: Comparison between Argos and related studies.

Work	Edge Type	Privacy Awareness	Allocation Decision
(Xu et al., 2019)	Private	✓	Migration
(Wang et al., 2020)	Private	✓	Migration
(Qian et al., 2019)	Private	✓	Placement
(Li et al., 2020)	Private	✓	Migration
(Faticanti et al., 2020)	Federated	×	Placement
(Mahmud et al., 2020b)	Federated	×	Placement
(Anglano et al., 2020)	Federated	×	Migration
(Li et al., 2021)	Federated	×	Migration
Argos (This Work)	Federated	✓	Migration

4 SYSTEM MODEL

This section presents our system model. First, we describe the edge scenario considered in our modeling. Then, we formulate the application provisioning process. Table 2 summarizes the notations.

We represent the environment as in Figure 1, adopting the model by Aral et al. (Aral et al., 2021), which divides the map into hexagonal cells. In such a scenario, the edge infrastructure comprises a set of links \mathcal{L} interconnecting base stations \mathcal{B} equipped with edge servers \mathcal{E} , positioned at each map cell. Whereas

Table 2: List of notations used in the paper.

Symbol	Description
w_f	Wireless delay of base station \mathcal{B}_f
d_u	Delay of link \mathcal{L}_u
b_u	Bandwidth capacity of link \mathcal{L}_u
c_i	Capacity of edge server \mathcal{E}_i
o_i	Demand of edge server \mathcal{E}_i
p_i	Provider that manages edge server \mathcal{E}_i
r_i	Base station equipped with edge server \mathcal{E}_i
\wp_e	Infrastructure providers trusted by user \mathcal{U}_e
α_e	Application accessed by user \mathcal{U}_e
β_e	Base station to which user \mathcal{U}_e is connected
l_j	Latency of application \mathcal{A}_j
t_j	Latency threshold of application \mathcal{A}_j
s_j	List of services that compose application \mathcal{A}_j
ρ_k	Capacity demand of service \mathcal{S}_k
μ_k	Privacy requirement of service \mathcal{S}_k
$x_{o,i,k}$	Matrix that represents the service placement

base stations provide wireless connectivity to a set of users \mathcal{U} , edge servers accommodate the set of user applications \mathcal{A} . We model a base station as $\mathcal{B}_f = (w_f)$, where w_f denotes \mathcal{B}_f 's wireless latency, and a network link as $\mathcal{L}_u = (d_u, b_u)$ where attributes d_u and b_u represent \mathcal{L}_u 's latency and bandwidth, respectively.

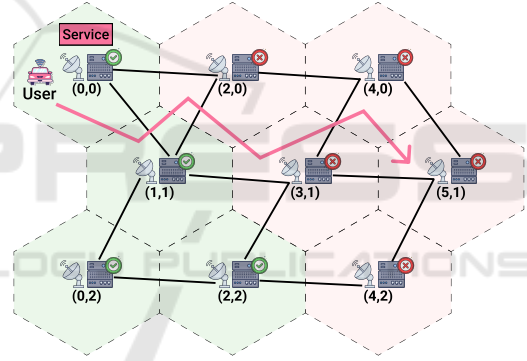


Figure 1: Sample scenario with a mobile user where migrations need to be performed based on the service latency and the user's level of trust in infrastructure providers.

We assume that the edge infrastructure is federated. Accordingly, the set of edge servers \mathcal{E} is managed by a group of infrastructure providers \mathcal{P} . We model an edge server as $\mathcal{E}_i = (c_i, o_i, p_i, r_i)$. While attributes c_i and o_i represent \mathcal{E}_i 's capacity and occupation, respectively, p_i references the infrastructure provider that manages \mathcal{E}_i , and r_i references the base station to which \mathcal{E}_i is connected.

As infrastructure providers may adopt different data protection policies, we assume users trust a specific group of providers to host their services. Although not mandatory, it is desirable to host the services accessed by a user \mathcal{U}_e on edge servers managed by infrastructure providers that \mathcal{U}_e trusts. We model a user as $\mathcal{U}_e = (\wp_e, \alpha_e, \beta_e)$, where \wp_e denotes the list of infrastructure providers \mathcal{U}_e trusts, α_e references the application accessed by \mathcal{U}_e , and β_e references the

base station to which \mathcal{U}_e is connected. In our scenario, we consider a one-to-one relationship between users and applications in the edge infrastructure.

We model applications as directed acyclic graphs composed of a set of services. An application is denoted as $\mathcal{A}_j = (l_j, t_j, s_j)$. While attributes l_j and t_j represent \mathcal{A}_j 's latency at a time step $\mathcal{T}_o \in \mathcal{T}$ and \mathcal{A}_j 's maximum latency threshold, respectively, s_j references the list of services that compose \mathcal{A}_j . We assume that application services have specific responsibilities (e.g., one microservice might be responsible for the database, another for the front end, and so on). Thus, different services within an application may have different capacity and privacy requirements depending on the tasks they perform.

A service is modeled as $\mathcal{S}_k = (\rho_k, \mu_k)$, where attributes ρ_k and μ_k denote the amount of resources and the degree of privacy required by \mathcal{S}_k , respectively. More specifically, $\mu_k = 1$ if \mathcal{S}_k has a special privacy requirement and 0 otherwise. In such a scenario, privacy violations occur when services with special privacy requirements (i.e., services with $\mu_k = 1$) are hosted by edge servers managed by infrastructure providers not trusted by the service users. The placement of services on edge servers at a given time step $\mathcal{T}_o \in \mathcal{T}$ is given by $x_{o,i,k}$, where:

$$x_{o,i,k} = \begin{cases} 1 & \text{if } \mathcal{E}_i \text{ hosts } \mathcal{S}_k \text{ at time step } \mathcal{T}_o \\ 0 & \text{otherwise.} \end{cases}$$

We assume the absence of shared storage in the edge infrastructure. Therefore, migrating a service \mathcal{S}_k to an edge server \mathcal{E}_i implies transferring \mathcal{S}_k 's capacity demand from \mathcal{S}_k 's current host to \mathcal{E}_i throughout a set of links $\Omega(\mathcal{S}_k, \mathcal{E}_i)$ that interconnect the edge servers' base stations. We define $\Omega(\mathcal{S}_k, \mathcal{E}_i)$ with the Dijkstra shortest path algorithm (Dijkstra et al., 1959) using link bandwidths as path weight. The migration time of \mathcal{S}_k to \mathcal{E}_i is given by $\phi(\mathcal{S}_k, \mathcal{E}_i)$, as denoted in Equation 1. Assuming that links in the network infrastructure may have heterogeneous configurations, the lowest bandwidth available between the links in $\Omega(\mathcal{S}_k, \mathcal{E}_i)$ is considered as the available bandwidth to perform the migration.

$$\phi(\mathcal{S}_k, \mathcal{E}_i) = \frac{\rho_k}{\min\{b_u \mid u \in \Omega(\mathcal{S}_k, \mathcal{E}_i)\}} \quad (1)$$

We model the latency of an application \mathcal{A}_j in Equation 2, considering the wireless latency of the base station of \mathcal{A}_j 's user (denoted as \mathcal{B}_f) and the aggregated latency of a set of network links κ , used to route the data from \mathcal{B}_f across each of \mathcal{A}_j 's services in the infrastructure. If all \mathcal{A}_j 's services are hosted by \mathcal{B}_f 's edge server, $|\kappa| = 0$, meaning that no links are needed to route \mathcal{A}_j 's data throughout the network.

We define the set of links κ through the Dijkstra shortest path algorithm (links' latency are used as path weight) (Dijkstra et al., 1959). We assume that SLA violations occur whenever \mathcal{A}_j 's latency l_j surpasses \mathcal{A}_j 's latency threshold t_j .

$$l_j = w_f + \sum_{u=1}^{|\kappa|} d_u \quad (2)$$

In such a scenario, our goal consists in migrating services in the edge infrastructure as users move across the map to minimize the number of latency and privacy violations. Latency violations are minimized by hosting application services on edge servers close enough to the application users. And for privacy violations the heuristic aims at putting as many application services with special privacy requirements as possible on edge servers managed by infrastructure providers trusted by the applications' users.

5 ARGOS DESIGN

This section presents Argos, our migration strategy that considers user mobility and service privacy requirements on federated edges. Algorithm 1 describes Argos, which is executed at each time step $\mathcal{T}_o \in \mathcal{T}$.

Algorithm 1: Argos migration algorithm.

```

1   $\eta = (\mathcal{A}_j \in \mathcal{A} \mid l_j > t_j)$  sorted by Eq. 3 (asc.)
2  foreach application  $\eta_j \in \eta$  do
3       $\lambda =$  Services that compose  $\eta_j$  sorted by privacy
        requirement (desc.) and demand (desc.)
4       $\xi =$  List of servers trusted by  $\eta_j$ 's user sorted by latency
5       $\chi =$  List of servers not trusted by  $\eta_j$ 's user sorted by latency
6       $\psi = \xi \cup \chi$ 
7      foreach service  $\lambda_k \in \lambda$  do
8          foreach edge server  $\psi_i \in \psi$  do
9              if  $x_{o,i,k} = 1$  then
10                 break
11             else
12                 if  $c_i - o_i \geq \rho_k$  then
13                     Migrate service  $\lambda_k$  to edge server  $\psi_i$ 
14                     break
15                 end
16             end
17         end
18     end
19 end

```

Argos initially gets the list of applications whose latency exceeds their latency thresholds (Algorithm 1, line 1), sorting these applications based on their latency and latency thresholds. More specifically, each application \mathcal{A}_j receives a weight ∂_j (denoted in Equation 3) according to the tightness of their latency threshold and on how much their actual latency exceeds their latency threshold. Accordingly, Argos

avoids unnecessary migrations by migrating only the applications presenting latency bottlenecks while prioritizing applications with the most critical latency issues (i.e., those whose latency most exceeds their latency threshold).

$$\partial_j = t_j - l_j \quad (3)$$

After selecting and sorting applications with latency issues, Argos sorts the services of each of these applications based on their privacy requirement and demand (Algorithm 1, line 4). The first sorting parameter (services' privacy requirement) prevents services without special requirements from saturating all edge servers managed by trusted infrastructure providers while services with special privacy requirements are hosted on edge servers from not trusted infrastructure providers. The second sorting parameter (demand) prioritizes services with a higher demand to ensure better use of edge server resources.

Once Argos arranges the services from an application, it sorts edge servers based on the trust of the application's user in their infrastructure providers and their delay (Algorithm 1, lines 5–7). Specifically on the edge servers' delay, we estimate what would be the application's delay if the edge servers were chosen to host one or more of the application's services (according to Equation 2). Then, Argos iterates over the sorted list of edge servers, choosing the first edge server with enough capacity as the new host of each application service (Algorithm 1, lines 9–18).

6 PERFORMANCE EVALUATION

6.1 Methodology

The dataset used during our experiments is described as follows. Unless stated otherwise, parameters in the dataset are distributed according to an uniform distribution. We consider a federated edge computing infrastructure comprising 100 base stations with wireless delay = 10 equipped with 60 edge servers (we randomly distribute the edge servers to 60 base stations). We assume that edge servers are managed by two infrastructure providers (each provider managing 30 servers). Edge servers have capacity = {100, 200}.

The network topology is defined according to the Barabási-Albert model (Barabási and Albert, 1999), where links have delays = {4, 8} and bandwidths = {2, 4}. In our scenario, we have a set of 120 users accessing an application of two services each. Users move across the map according to the Pathway mobility model (Bai and Helmy, 2004). Each user in the scenario trusts one of the infrastructure providers.

Applications have latency thresholds = {60, 120}. The set of 240 services have capacities = {10, 20, 30, 40, 50}. We assume that 120 out of the 240 services have special privacy requirements. The initial service placement is defined according to the First-Fit heuristic described in Algorithm 2.

Algorithm 2: Initial service placement scheme.

```

1   $S' \leftarrow$  List of services in  $S$  arranged randomly
2   $\mathcal{E}' \leftarrow$  List of edge servers in  $\mathcal{E}$ 
3  foreach  $S'_k \in S'$  do
4      foreach edge server  $\mathcal{E}'_i \in \mathcal{E}'$  do
5          if  $c_i - o_i \geq \rho_k$  then
6              Host service  $S'_k$  on edge server  $\mathcal{E}'_i$ 
7              break
8          end
9      end
10 end
    
```

We compare Argos against two naive strategies called Never Follow and Follow User, presented by Yao et al. (Yao et al., 2015), and the strategy proposed by Faticanti et al. (Faticanti et al., 2020), described in Section 3. Whereas Never Follow performs no migration, Follow User migrates services every time users move around the map, regardless of applications' latency. We chose to compare Argos against Faticanti's strategy as it allocates resources on federated edge computing scenarios while minimizing the number of services on specific infrastructure providers. As Faticanti's strategy was designed to decide the initial placement of services, we adapted it to migrate services whenever the latency of applications exceeds their latency threshold, as Argos does.

We evaluate the selected strategies regarding SLA violations, privacy violations, and number of migrations. Whereas SLA violation refers to the number of applications whose latency exceeds their latency thresholds, privacy violation refers to the number of services hosted by edge servers managed by infrastructure providers not trusted by services' users. The results presented next are the sum of latency and privacy violations during each simulation time step. The source code of our simulator and the dataset are available at our GitHub repository¹.

6.2 Results and Discussion

6.2.1 SLA Violations

Figure 2(a) shows the number of SLA violations during the execution of the evaluated strategies. As expected, the two naive strategies presented the worst results. Never Follow had the worst result for not per-

¹<https://github.com/paulosevero/argos>.

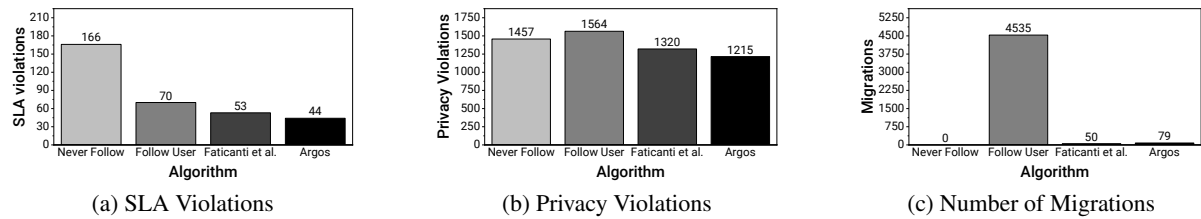


Figure 2: Simulation results regarding the number of SLA violations, privacy violations, and migrations.

forming migrations, and Follow User had the second-worst result for performing migrations excessively, benefiting some applications but harming others.

Argos and Faticanti's strategy obtained the best and second-best results, respectively. The main difference between them was the order of migrations. While Faticanti's strategy migrates services according to their position in their application's topology (migrating the first services of each application, then the second services of each application, etc.), Argos migrates services of each application at a time, prioritizing applications with more severe latency bottlenecks.

Accordingly, when users are close to each other, Faticanti's topological sort wastes the resources of nearby edge servers with the first services of each of these users' applications, placing the last services in more distant edge servers as the closer ones are filled. Consequently, it penalizes the applications with tighter SLAs instead of prioritizing them when occupying the nearby edge servers as Argos does.

6.2.2 Privacy Violations

Figure 2(b) presents the number of privacy violations of the evaluated strategies. Follow User and Never Follow obtained the worst results by overlooking the existence of different infrastructure providers in the environment. Compared to them, Faticanti's strategy reduced the number of privacy violations by 15.6% and 9.4%, respectively, by prioritizing migrating services to edge servers managed by infrastructure providers trusted by the services' users.

Argos obtained the best result among the strategies evaluated, reducing the number of privacy violations by 7.95% compared to Faticanti's strategy, which presented the second-best result. Unlike Faticanti's strategy, which migrates as many services as possible to edge servers trusted by its users, Argos prioritizes the available space on trusted edge servers with services that hold special privacy requirements. As a result, Argos uses the resources managed by reliable infrastructure providers in a better way, which makes a significant difference, especially when there are few trusted edge servers nearby users.

6.2.3 Service Migrations

Figure 2(c) shows the migration results. While Never Follow performed no migration, Follow User was the strategy with most migrations, relocating services 151 times per step on average. Compared to Follow User, Argos reduced the number of migrations by 98%, migrating only those services from applications suffering from latency issues. However, it performed 36.7% more migrations than Faticanti's strategy.

While Argos migrates all the services of an application after identifying it has an excessively high latency, Faticanti's strategy migrates each of the services of an application at a time according to its topological order. Thus, it avoids unnecessary migrations when only just some of the services of an application need to be migrated to mitigate latency issues.

7 CONCLUSIONS

As edge computing becomes more popular, it is expected the emergence of infrastructure providers renting edge resources on a pay-as-you-go basis as in the cloud, allowing multi-sized organizations to benefit from edge computing without the need of managing their own infrastructure (Anglano et al., 2020).

Previous studies aim at different goals when allocating resources on federated edges, such as maximizing profit by managing the cooperation among infrastructure providers. However, existing solutions overlook that providers may implement distinct data protection policies, limiting the number of resources suitable for hosting services with strict privacy requirements like databases and payment gateways.

To address existing limitations in the state-of-the-art, this paper presents Argos, a heuristic algorithm that migrates microservice-based applications according to users' mobility in federated edges while considering services' privacy requirements and users' trust levels on infrastructure providers. Experimental results demonstrate that Argos outperforms state-of-the-art approaches, reducing latency and privacy issues by 16.98% and 7.95%, respectively. As future work, we intend to extend our heuristic to minimize

the allocation cost of privacy-sensitive microservice-based applications on federated edges.

ACKNOWLEDGEMENTS

This work was supported by the PDTI Program, funded by Dell Computadores do Brasil Ltda (Law 8.248 / 91). The authors acknowledge the High-Performance Computing Laboratory of the Pontifical Catholic University of Rio Grande do Sul for providing resources for this project.

REFERENCES

- Ahmed, E., Gani, A., Khan, M. K., Buyya, R., and Khan, S. U. (2015). Seamless application execution in mobile cloud computing: Motivation, taxonomy, and open challenges. *Journal of Network and Computer Applications*, 52:154–172.
- Anglano, C., Canonico, M., Castagno, P., Guazzone, M., and Sereno, M. (2020). Profit-aware coalition formation in fog computing providers: A game-theoretic approach. *Concurrency and Computation: Practice and Experience*, 32(21):e5220.
- Aral, A. and Brandić, I. (2020). Learning spatiotemporal failure dependencies for resilient edge computing services. *IEEE Transactions on Parallel and Distributed Systems*, 32(7):1578–1590.
- Aral, A., Demaio, V., and Brandic, I. (2021). Ares: Reliable and sustainable edge provisioning for wireless sensor networks. *IEEE Transactions on Sustainable Computing*, pages 1–12.
- Bai, F. and Helmy, A. (2004). A survey of mobility models. *Wireless Adhoc Networks. University of Southern California, USA*, 206:147.
- Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *science*, 286(5439):509–512.
- Dijkstra, E. W. et al. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.
- Faticanti, F., Savi, M., De Pellegrini, F., Kochovski, P., Stankovski, V., and Siracusa, D. (2020). Deployment of application microservices in multi-domain federated fog environments. In *International Conference on Omni-layer Intelligent Systems*, pages 1–6. IEEE.
- He, T., Khamfroush, H., Wang, S., La Porta, T., and Stein, S. (2018). It’s hard to share: Joint service placement and request scheduling in edge clouds with sharable and non-sharable resources. In *International Conference on Distributed Computing Systems*, pages 365–375. IEEE.
- He, X., Jin, R., and Dai, H. (2019). Peace: Privacy-preserving and cost-efficient task offloading for mobile-edge computing. *IEEE Transactions on Wireless Communications*, 19(3):1814–1824.
- Li, T., Liu, H., Liang, J., Zhang, H., Geng, L., and Liu, Y. (2020). Privacy-aware online task offloading for mobile-edge computing. In *International Conference on Wireless Algorithms, Systems, and Applications*, pages 244–255. Springer.
- Li, Y., Dai, W., Gan, X., Jin, H., Fu, L., Ma, H., and Wang, X. (2021). Cooperative service placement and scheduling in edge clouds: A deadline-driven approach. *IEEE Transactions on Mobile Computing*.
- Mahmud, R., Ramamohanarao, K., and Buyya, R. (2020a). Application management in fog computing environments: A taxonomy, review and future directions. *ACM Computing Surveys*, 53(4):1–43.
- Mahmud, R., Srirama, S. N., Ramamohanarao, K., and Buyya, R. (2020b). Profit-aware application placement for integrated fog–cloud computing environments. *Journal of Parallel and Distributed Computing*, 135:177–190.
- Mor, N., Pratt, R., Allman, E., Lutz, K., and Kubiawicz, J. (2019). Global data plane: A federated vision for secure data in edge computing. In *International Conference on Distributed Computing Systems*, pages 1652–1663. IEEE.
- Qian, Y., Hu, L., Chen, J., Guan, X., Hassan, M. M., and Alelaiwi, A. (2019). Privacy-aware service placement for mobile edge computing via federated learning. *Information Sciences*, 505:562–570.
- Satyantarayanan, M. (2017). The emergence of edge computing. *Computer*, 50(1):30–39.
- Satyantarayanan, M., Bahl, P., Caceres, R., and Davies, N. (2009). The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, 8(4):14–23.
- Wang, W., Ge, S., and Zhou, X. (2020). Location-privacy-aware service migration in mobile edge computing. In *Wireless Communications and Networking Conference*, pages 1–6. IEEE.
- Xu, X., He, C., Xu, Z., Qi, L., Wan, S., and Bhuiyan, M. Z. A. (2019). Joint optimization of offloading utility and privacy for edge computing enabled iot. *IEEE Internet of Things Journal*, 7(4):2622–2629.
- Yao, H., Bai, C., Zeng, D., Liang, Q., and Fan, Y. (2015). Migrate or not? exploring virtual machine migration in roadside cloudlet-based vehicular cloud. *Concurrency and Computation: Practice and Experience*, 27(18):5780–5792.
- Zhu, D., Li, T., Liu, H., Sun, J., Geng, L., and Liu, Y. (2021). Privacy-aware online task offloading for mobile-edge computing. *Wireless Communications and Mobile Computing*, 2021.