Applying Fuzzy Logic to Efficiently Manage Shared Edge Infrastructure

Tidiane Sylla, Mohamed Aymen Chalouf, Leo Mendiboure, Francine Krief, Hasnaâ Aniss, Lylia Alouache

Applying Fuzzy Logic to Efficiently Manage Shared Edge Infrastructure

Tidiane Sylla[1], Mohamed Aymen Chalouf[2], Leo Mendiboure[3] *(IEEE Member)*, Francine Krief[4], Hasnaa Aniss[3], Lylia Alouache[5]

[1]*Dept. ISA-GEII,Univ. of Sciences, Techniques and Technologies of Bamako, Bamako, Mali (tidiane.sylla@usttb.edu.ml)*

[2]*IRISA Lab, University of Rennes 1, F-22300 Lannion, France*

[3]*COSYS-ERENA, Univ. Gustave Eiffel, F-77454 Marne-la-Vallée, France*

[4]*LaBRI Lab, Bordeaux INP, F-33000 Bordeaux, France*

[5]*ETIS UMR 8051 CNRS lab CY Cergy Paris University F-95000 Cergy, France*

***Abstract*—Edge Computing promise a bright future for Internet of Things (IoT). Edge Computing servers can be geographically distributed to be closer to users and ensure low latency communications and real-time data processing for the third-party applications using this infrastructure. However, user mobility and limited Edge servers capabilities (CPU, memory, bandwidth) may cause many services placement failures. That's why in this article we propose a new strategy aimed at enabling a fair sharing of available Edge resources through a dynamic and real-time IoT Service placement. Our strategy uses Fuzzy Logic to enable 1) quick and low-cost deployment of the solution and 2) real-time modification of the policies defined by the Edge operator. It takes advantage, among others, of a microservice decomposition to optimize the use of the Edge architecture. Experiments demonstrate the benefits of our approach in terms of placement reliability, execution time and resources utilization.**

***Index Terms*—IoT, Edge Computing, Infrastructure Sharing, Placement Strategy, Fuzzy logic, Utility function**

I. Introduction

Internet of Things (IoT) has contributed to the improvement of users' daily life, business and industrial processes. The connected devices collect and transfer large volumes of data which are then processed and analyzed to provide efficient services [1]. IoT services are classically deployed in Cloud infrastructures resulting in long response times and non-context-aware services [1]. That is why new data processing architectures emerged: Edge Computing architectures [2]. They enable IoT data to be processed by third-party services hosted at the network's Edge, close to data sources, reducing the applications delay as well as the bandwidth required to transfer data to the Cloud. Edge Computing infrastructure is composed of small data centers and includes heterogeneous nodes with often limited bandwidth and computing resources: Access Points, mobile phones, etc. The available resources depend on the geographical area, time of day and users density.

Many IoT services are designed following the as-a-service approach, where functional components are implemented by microservices [3]. These microservices can be deployed on different nodes of the Edge infrastructure depending on their available resources [4] and according to the demands of the various users of the infrastructure. The Edge Orchestrator automates and coordinates the management of Edge infrastructure resources and the placement of IoT services.

Placing IoT services with user mobility and quality of service (QoS) in mind is a challenging task [5]. That is why, several strategies for dynamic service placement in Edge infrastructures have been proposed in the literature [5]–[8]. However, the as-a-service approach is not considered even though it could address dynamicity and flexibility issues in Edge environments. In addition, these strategies need long processing time and are not suitable for low-latency services. These important limitations reduce the reliability of placement strategies and could disrupt the operation of IoT services [4].

Thus, we propose, in this paper, a new Fuzzy Logic-based strategy for dynamic and real-time IoT service placement in Edge infrastructures. Based on our strategy, the orchestrator will be able to obtain a placement for an IoT service after a quick evaluation considering user mobility, nodes resources availability, node end-to-end delay and data migration cost. The proposed strategy considers every service as a set of microservices to be placed. To the best of our knowledge, such a resilient placement strategy, considering user's mobility and applications QoS, has not yet been proposed. This could be an efficient tool for managing shared Edge infrastructure.

The rest of this paper is organized as follows. Section II compares state-of-the-art approaches. Section III presents our placement strategy. Section IV details the proposed algorithms. Section V discusses the performance evaluation results while comparing our solution to some relevant existing ones.

II. Related work

A. State-of-the-art strategies

Service placement approaches can be primarily categorized into three classes: centralized, decentralized, and hierarchical approach. Each class uses classical mathematics and meta-heuristic optimization, fuzzy logic and ML approaches [9].

- Studies based on classical mathematics optimization [5]–[8]: they mainly characterize the problem of dynamic service placement at Edge nodes considering the constraints

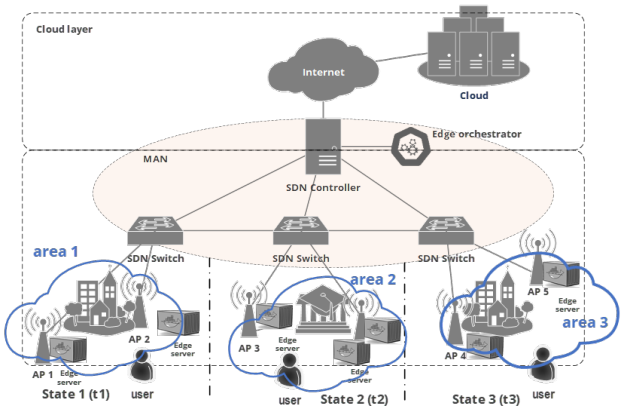


Fig. 1: Proposed Edge Infrastructure architecture and scenario

of these nodes as a complex optimization problem. The proposed strategies use the Lyapunov optimization method [6], [8] or Integer Linear Programming method [7] to optimize the dynamic placement of IoT services on the Edge. They show significant results in terms of placement reliability, QoS guarantee and energy consumption [9]. However, they did not consider resource availability and user mobility resulting in placement failure;

- Studies based on AI techniques [10]–[13]: these papers aimed to propose a dynamic placement strategy that considers the flexibility of Edge infrastructure resources. They differentiated IoT services according to application requirements (CPU, memory, bandwidth) and targeted an efficient lightweight placement strategy. Several AI-based techniques are used in this context: 1) fuzzy logic [11], [14], 2) reinforcement learning [10] and 3) deep reinforcement learning [12], [13]. However, they have limitations, in particular, they 1) propose workload distribution methods and do not consider the as-a-service approach (micro-service granularity), and 2) do not provide a reconfigurable system that considers the random mobility of nodes.Therefore, the strategies based on distributed approaches [15] may not be suitable for the placement of delay and failure sensitive IoT services.

B. Positioning

Existing solutions do not consider: as-a-service property (IoT service decomposition), post-placement IoT service and the use of lightweight solutions. Thus, this paper proposes a new Fuzzy Logic-based placement strategy to overcome these limitations by: 1) considering user mobility, 2) estimating candidate nodes resources availability, 3) estimating data migration cost and latency perceived by users, and 4) defining a new algorithm for fast evaluation of instantaneous conditions of Edge nodes to obtain a placement on one or many nodes.

III. Fuzzy Logic-based placement strategy

A. Overview

The first idea is to predict the user's destination area based on its mobility and evaluate, in advance, the available resources in that area to orchestrate an efficient service placement. To achieve that, our strategy requires information from: Edge infrastructure (available resources), user devices (perceived QoS) and Service Level Agreement (SLA) (required latency and bandwidth). Figure 1 illustrates the Edge architecture in a Smart City use case: several Access Points (AP) are deployed and connected to a set of Edge servers.

Our strategy is implemented through two algorithms: 1) global placement strategy (Fig. 2.1) and 2) service blocks' placement strategy (Algorithm 1). Global algorithm uses utility functions to evaluate and rank potential destination areas based on the SLA satisfaction degree. Service Block Algorithm uses utility functions to rank the Edge nodes in an area according to their ability to receive service blocks to be placed. Subsequently, it uses fuzzy logic to determine whether service blocks can be placed on one or several nodes while satisfying the service constraints (see Section IV).

B. Mobility prediction

The smart city scenario considers users as pedestrians following the nomadic mobility model [16] that can cause frequent placement failures due to frequent needs to place services when conditions are degraded [17]. To reduce this frequent need for placement, we propose to form groups of Edge nodes that are located within the same geographical area and have close average delays. For example, access points covering 3-4 city blocks with 40 ms average delay can form a group. The problem of service placement considering user mobility is a stochastic process, composed of random variables with discrete evolution. The process is divided into time intervals (t_i) and states (s_i). In a state s_i, the service is placed in a block on a single node, or in several blocks on several nodes. Figure 1 illustrates the mobility of one user in a smart city. A user typically follows a regular route (e.g. going to work) and may exceptionally take other routes. The mobility model enables determining from a mobile connected to a base station controller the next station controller in the area to which the mobile will be connected. To predict user routes, Hidden Markov Model (HMM) is used.

C. Resources availability, delay and migration cost

The second step consists in selecting candidate Edge nodes that can run the service. The nodes in the predicted area are classified in terms of resources availability (CPU, RAM, bandwidth) to identify those whose the available resources exceed the threshold required to keep the entire service or specific microservices running without failure. When candidate nodes are identified, the latency is evaluated to select those having the lowest latency. Placement in the predicted area is not relevant if none of the candidate nodes has a latency that meets the service threshold. The orchestrator evaluates whether to maintain the service on the nodes in the user's current area. The available bandwidth of an Edge node is used to estimate the cost of data migration to determine candidate nodes allowing optimal placement.

IV. Proposed Strategy algorithms

A. Algorithms description

The Edge orchestrator performs the main algorithm (Fig. 2.1) whenever there is a placement request or if the conditions of a running service degrade (e.g., latency). When user is moving, the prediction of the user's probable destination areas is performed. Multiple probable destination areas can be detected in addition to the current one. The proposed utility function evaluates (compute a score) these areas according to their conditions and ranks them in descending order. Areas are then successively selected based on theirs scores and the blocks placement algorithm is executed to place/migrate the service. When no node in the selected area is able to receive all the blocks of the service, these blocks will be placed on several nodes according to theirs characteristics and the service's needs. Simultaneous placements on several nodes is studied to find a suitable placement solution.

Algorithm 1 Fuzzy-based placement of a service's blocks

Require: NA,SBw,SBlocks
Ensure: Result : The service is placed or not (True or False)
Compute the score of nodes N and rank them;
$RankedN \leftarrow UtilityFunction(NA)$
$i \leftarrow 1$;
while $Card(SBlocks) > 0$ and $i \leq Card(RankedN)$ **do**
  $N \leftarrow RankedN(i)$;
  $NCPULoad(N) \leftarrow N$ *CPU load*;
  $NMemLoad(N) \leftarrow N$ *Memory usage* ;
  $FLSRes \leftarrow FLS(NCPULoad, NMemLoad, NBw\ SBlocksLoad, SBw)$;
  if $FLSRes = Verygood$ or $FLSRes = Good$ **then**
    $placedBlocks \leftarrow PlaceSBlocksMax(FLSRes, SBlocks, SBw, N)$;
  else
    if $FLSRes = Medium$ **then**
      $placedBlocks \leftarrow PlaceSBlocksMin(FLSRes, SBlocks, SBw, N)$;
    end if
  end if
  Remove placedBlocks from $SBlocks$;
  $i \leftarrow i + 1$;
end while
if $Card(SBlocks = 0)$ **then**
  $Result \leftarrow True$
else
  $Result \leftarrow False$
end if
return Result

Service Block's Algorithm (Algorithm 1) takes as input information about nodes of the selected area (NA), service throughput (SBw), and service's blocks ($SBlocks$). A second utility function is then used to compute scores of all the nodes of the NA area and rank them. For each node, starting with the one with the highest score, available resources are evaluated using a proposed Fuzzy Logic System (FLS) (see Section IV-B). According to the FLS result, service blocks are either all placed on a same node or successively placed on the best nodes of the NA area. Based on FLS output and service's blocks information (memory: NMemLoad, processor: NCPULoad, bandwidth: NBw, load per block: SBlocksLoad), placements are performed using two functions: PlaceSBlocksMax and PlaceSBlocksMin. PlaceSBlocksMax function is called when the placement of all the remaining service's blocks on one node is possible (FLS result=Very Good or Good). PlaceSBlocksMin function is called, if needed (FLS result=Medium), to place some service's blocks according to the nodes available resources (Fig. 2.2).

At the critical time, there may be no nodes that satisfy the conditions for optimal placement in either case. When placement fails in the Edge, the service is placed in the Cloud.

B. Proposed utility functions and Fuzzy Logic System

Utility functions are used to determine the satisfaction level related to multi-criteria decision. Since the satisfaction degree of a service placement in an Edge infrastructure depends on the service constraints, we use utility functions to evaluate: the utility of potential user destination areas as well as that of one area's nodes for service placement based on SLA constraints.

a) Utility functions: These functions are used to rank potential destination areas and theirs nodes capabilities.

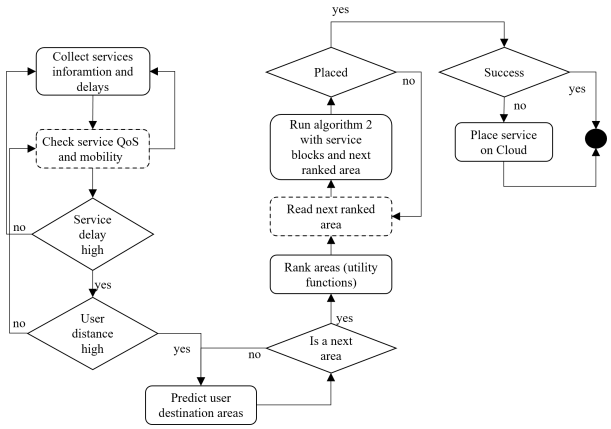
Ranking of the user's predicted destination areas: Depending on the considered attributes nature, various utility functions are used before computing the aggregate utility value [18]. We use the sigmoidal utility function to evaluate the utility of latency and bandwidth, and the linear utility function to evaluate the number of users. For a sigmoidal utility function, parameter *a* represents the threshold, and *b* is used to adjust the slope of the function. Area utility is defined by Equ. (1).

$$\mathrm{S_{NA}} = \left(\frac{(l/l_{min})^4}{1+(l/l_{min})^4}\right) \times (0.5) \times \left(\frac{(bw/bw_{min})^{10}}{1+(bw/bw_{min})^{10}}\right) \times (0.3) \times 1 + \frac{1}{70} * n \times (0.2) \quad (1)$$

Ranking of nodes in a selected destination area: Sigmoidal utility functions are also used to evaluate the utility of a service placement on a set of nodes in a selected area: CPU load, memory usage, and available bandwidth. The aggregate utility value of a node is defined by equation (2).

$$\mathrm{S_N} = \left(\frac{(c/c_{min})^4}{1+(c/c_{min})^4}\right) \times (0.4) \times \left(\frac{(m/m_{min})^4}{1+(m/m_{min})^4}\right) \times (0.4) \times \left(\frac{(b/b_{min})^4}{1+(b/b_{min})^4}\right) \times (0.3) \quad (2)$$

b) Fuzzy Logic Inference System: Our approach uses Fuzzy Logic, a well-known AI technique for decision making in intelligent IoT control systems [19]. The proposed FLS takes as input service blocks and Edge nodes resources parameters to determine the performance level of a node (a set of nodes) after placing the service's blocks on it (them). This allows our algorithm to make a placement decision: all microservices on one node or blocks distribution on a set of nodes. The considered parameters are: node's CPU load (CPULoad), memory usage (MemLoad), available node bandwidth (Bw), load required for running service's blocks (SBlocksLoad) and service's required bandwidth (Sbw). They allow evaluation of node's instant state and thus whether this last is able to receive all (or a set of) service's blocks. These inputs are converted into linguistic terms (fuzzified). For



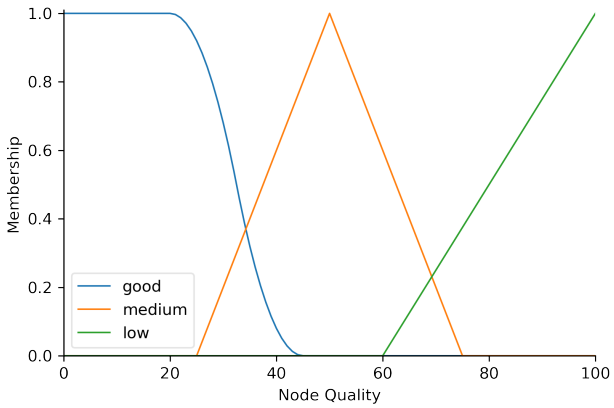


Fig. 2: 1) Fuzzy Logic-based placement strategy main algorithm; 2) Node quality membership function

example, *Low*, *Medium* and *High* (denoted respectively μ_{Lo}, μ_M and μHi in the provided equations) correspond to the considered parameter's ratio like available memory, whereas *Saturated* and *Not Saturated* correspond to the use level of a resource like bandwidth. The equations (3)-(18) define the proposed fuzzy variables membership functions of our FLS. The membership functions intervals values are defined for each parameter and intervals boundaries, depending on the system's operational goals, are fixed on the basis of the Edge infrastructure performance requirements [2], [14].

1) CPU Load (c):

$$\mu_{Lo}(c) = \begin{cases} 1, & c \leq 30\% \\ \frac{30\%-c}{30\%-10\%}, & 10\% < c \leq 30\% \\ 0, & c \geq 30\% \end{cases} \quad (3)$$

$$\mu_M(c) = \begin{cases} 0, & c \leq 30\% \\ \frac{c-55\%}{55\%-30\%}, & 30\% < c \leq 55\% \\ \frac{75\%-c}{75\%-55\%}, & 55\% < c \leq 75\% \\ 0, & c \geq 75\% \end{cases} \quad (4)$$

$$\mu_{Hi}(c) = \begin{cases} 0, & c \leq 70\% \\ \frac{c-80\%}{95\%-85\%}, & 85\% < c \leq 95\% \\ 1, & c > 95\% \end{cases} \quad (5)$$

2) Memory usage (m):

$$\mu_{Lo}(m) = \begin{cases} 1, & m \leq 30\% \\ \frac{30\%-m}{30\%-10\%}, & 10\% < m \leq 30\% \\ 0, & m \geq 30\% \end{cases} \quad (6)$$

$$\mu_M(m) = \begin{cases} 0, & m \leq 30\% \\ \frac{m-55\%}{55\%-30\%}, & 30\% < m \leq 55\% \\ \frac{75\%-m}{75\%-55\%}, & 55\% < m \leq 75\% \\ 0, & m \geq 75\% \end{cases} \quad (7)$$

$$\mu_{Hi}(m) = \begin{cases} 0, & m \leq 70\% \\ \frac{m-80\%}{95\%-85\%}, & 85\% < m \leq 95\% \\ 1, & m > 95\% \end{cases} \quad (8)$$

3) Bandwidth (ab in Mbps) :

$$\mu_S(ab) = \begin{cases} 1, & ab \leq 30 \\ \frac{50-ab}{50-30}, & 30 < ab \leq 50 \\ 0, & ab \geq 50 \end{cases} \quad (9)$$

$$\mu_{Lo}(ab) = \begin{cases} 0, & ab \leq 40 \\ \frac{ab-80}{80-40}, & 40 < ab \leq 80 \\ \frac{120-ab}{120-80}, & 80 < ab \leq 120 \\ 0, & ab \geq 120 \end{cases} \quad (10)$$

$$\mu_M(ab) = \begin{cases} 0, & ab \leq 100 \\ \frac{ab-240}{240-100}, & 100 < ab \leq 240 \\ \frac{440-ab}{440-240}, & 240 < ab \leq 440 \\ 0, & ab \geq 440 \end{cases} \quad (11)$$

$$\mu_{Hi}(ab) = \begin{cases} 0, & ab \leq 420 \\ \frac{ab-540}{540-420}, & 420 < ab \leq 540 \\ 1, & ab \geq 540 \end{cases} \quad (12)$$

4) Service's blocks load (bl) :

$$\mu_{Lo}(bl) = \begin{cases} 1, & bl \leq 30\% \\ \frac{30\%-bl}{30\%-10\%}, & 10\% < bl \leq 30\% \\ 0, & bl \geq 30\% \end{cases} \quad (13)$$

$$\mu_M(bl) = \begin{cases} 0, & bl \leq 30\% \\ \frac{bl-55\%}{55\%-30\%}, & 30\% < bl \leq 55\% \\ \frac{75\%-bl}{75\%-55\%}, & 55\% < bl \leq 75\% \\ 0, & bl \geq 75\% \end{cases} \quad (14)$$

$$\mu_{Hi}(bl) = \begin{cases} 0, & bl \leq 70\% \\ \frac{bl-80\%}{95\%-85\%}, & 85\% < bl \leq 95\% \\ 1, & bl > 95\% \end{cases} \quad (15)$$

5) Service bandwidth (sb in Mbps):

$$\mu_{Lo}(sb) = \begin{cases} 0, & sb \leq 40 \\ \frac{sb-80}{80-40}, & 40 < sb \leq 80 \\ \frac{120-sb}{120-80}, & 80 < sb \leq 120 \\ 0, & sb \geq 120 \end{cases} \quad (16)$$

$$\mu_M(sb) = \begin{cases} 0, & sb \leq 100 \\ \frac{sb-240}{240-100}, & 100 < sb \leq 240 \\ \frac{440-sb}{440-240}, & 240 < sb \leq 440 \\ 0, & sb \geq 440 \end{cases} \quad (17)$$

$$\mu_{Hi}(sb) = \begin{cases} 0, & sb \leq 420 \\ \frac{sb-540}{540-420}, & 420 < sb \leq 540 \\ 1, & sb \geq 540 \end{cases} \quad (18)$$

Our fuzzy inference system uses the Mamdani model. Fuzzy rules are used to determine nodes capabilities levels (instantaneous conditions). For each rule (from Rule Base), an associated implication, composed of an antecedent and a consequent [19], is applied. The output of the fuzzy inference system is a fuzzy variable (Low, Medium, Good) having also a membership function (Fig. 2.2). The Rule Base proposed for our FLS is composed of 243 rules. Table II extracts some of the proposed rules (R1-R12).

The FLS evaluates rules, aggregates results before defuzzifying these results to obtain the node quality (capabilities level). Defuzzification consists in converting fuzzy output into a crispy value using a defuzzification method (e.g. CoS: Center of Sum, CoG: Center of Gravity and maxima methods). Our proposed FLS uses the most suitable defuzzification method; the CoG method implemented with Mamdani's Min-Max inference that has no interpolation effect. Equation (19) defines the defuzzication process to obtain the node quality.

$$z* = \frac{\sum_{x=a}^{b} \mu_A(x) \cdot x}{\sum_{x=a}^{b} \mu_A(x)} \quad (19)$$

The scenario detailed below illustrates the effectiveness of the proposed FLS. Input parameters have the following values: CPU load c=60%, RAM utilization m=45%, available bandwidth ab=355 Mbps, required block load bl=45% and

required service throughput *sb*=150 Mbps. According to inference table, the inference system output, shown in Figure 4.1, gives the following fuzzy output variable value: *Medium* that corresponds to the crispy z* value.

V. Performance Evaluation

This section presents performance evaluation of the proposed strategy compared to state-of-the-art strategies: i) resource utilization-based strategy and ii) Fuzzy Competitor-based strategy [16]. The Fuzzy Competitor method employs fuzzy logic to determine whether tasks should be performed on an Edge or Cloud server, using bandwidth, data transfer size, CPU speed, and delay as inputs for the FLS [14]. The fuzzy utilization-based method prioritizes Edge server offloading if VM CPU utilization is low. It aims at using Edge servers as long as they are not overwhelmed by CPU utilization [14].

A. Experiment setup

Experiments were performed on EdgeCloudSim [16] that enables building realistic Edge models since it supports: dynamic WLAN or WAN communication models, realistic load generation, mobility models, etc. Simulations were performed on a server configured with Ubuntu 20.04.5 LTS, Intel (R) Xeon (R) Gold 5218R 2.10 GHz and having 80 cores, and 64GB RAM. Simulations parameters are listed in Table I. In the considered scenario, smart city's Edge servers were uniformly distributed. Each area is managed by a nearby datacenter and composed of at least one Edge server. Each Edge server is composed of up to eight VM (Virtual Machine). Each VM is able to perform several dozen tasks.

B. Results

TABLE I: Simulation parameters

Parameter	Value
Datacenters	14
Edge servers per datacenter	4
VM per Edge/Cloud server	8/4
VM CPU core	2/4
VM RAM	2 Gbytes
Simulation time	33 minutes
WAN/WLAN bandwidth	Empirical [16]
MAN bandwidth	MMPP/M/1 [16]
Maximum number of mobile devices	2000
Probability of selecting a location type	Equiprobable

Simulation results includes performance indicators of Edge Computing systems as described in [8], [16], [17]:

a) Placement reliability rate: Since IoT applications are highly dynamic (due to, among others, users mobility), the placement failure ratio is a critical performance indicator. Figure 3.2 shows the failure rate of the compared approaches. When infrastructure is less loaded (under 600 devices), the compared strategies have a fairly close placement failure rate with a slightly lower one for resource utilization-based strategy. This is due to the implementation of the Least Load Algorithm when placing services. When infrastructure is moderately or high loaded (respectively between 600 and 1400 or more than 1400 devices), our Fuzzy Logic-based placement strategy has the most interesting failure rate (under 2%). This can be explained by the fact that network connections' state and Edge servers resources availability are considered.

b) Placement failures distribution: Analysis of placement failure rate distribution between Edge and Cloud servers (Figures 3.3 and 4.1) indicates how the strategy manages services placement requests depending on infrastructure load. Placement failure rate on Edge servers for our strategy at very high load (more than 1600 devices) is above 80%, whereas the compared strategies have less than 50%. Regarding placement failure rate on Cloud servers, utilization-based strategy has the highest one (above 80%) against the lowest one for our strategy (under 20%) in case of very high load.

TABLE II: Some inference rules for area best node selection

Rule	CPU	Memory	B.W.	B.L.	S.Thr.	N.Q.
R1	Low	Low	Low	Low	Low	Good
R2	Low	Low	Medium	Low	Low	Good
R3	Low	Low	High	Medium	High	Good
R4	Medium	Medium	Saturated	Medium	Low	Medium
R5	Medium	Medium	Low	High	Medium	Low
R6	Medium	Medium	Medium	Medium	Medium	Medium
R7	High	High	Low	High	High	Low
R8	High	High	Saturated	High	High	Low
R9	High	High	Medium	Medium	Medium	Medium
R10	High	Medium	Low	Medium	Medium	Low
R11	Medium	High	High	Low	Low	Medium
R12	Low	Low	High	High	Medium	Good

Legends : B.W.: Bandwidth, B.L: Service Blocks Load, S. Thr: Service Throughput, N. Q:Node Quality

c) Average resource usage of Edge nodes: Edge servers resources utilization rate is a key indicator of the efficient use of Edge infrastructure. Figure 4.2 shows the average utilization rate of Edge resources. When the system load is low, studied strategies have similar Edge and Cloud utilization rates. For higher load, utilization-based strategy has an Edge resources utilization rate slightly lower than others. This is due to the systematic placement of new services on Cloud servers (Fig. 4.1) when Edge ones are loaded. It can then cause more failures when WAN connections are congested (Fig. 3.1). In contrast, our strategy has a better Edge resources utilization rate even in case of high system load. The compared strategies have fairly similar Edge nodes resources. This can be explained by the ability of these strategies to take advantage of the high availability of the MAN to place new services. Thus, they can adapt to dynamic changes. Unlike the two other strategies, our approach has a low Cloud utilization rate for low/medium system loads (under 30% for up to 900 devices), maintaining a high QoS level, as a high Cloud resources utilization can result in a significant QoS degradation.

VI. Conclusion

Edge Computing enables to fully leverage ubiquitous systems such as IoT. In this paper, we proposed a novel Fuzzy-based strategy for dynamic IoT service placement in shared Edge infrastructure. It takes into account, among others, users

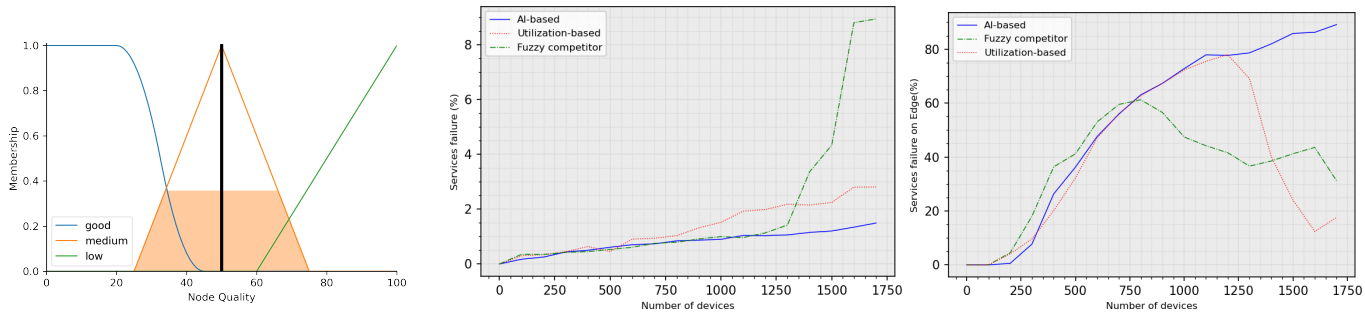


Fig. 3: 1) Selected node quality output; 2) Placements failure rate; 3) Services placement failure on Edge

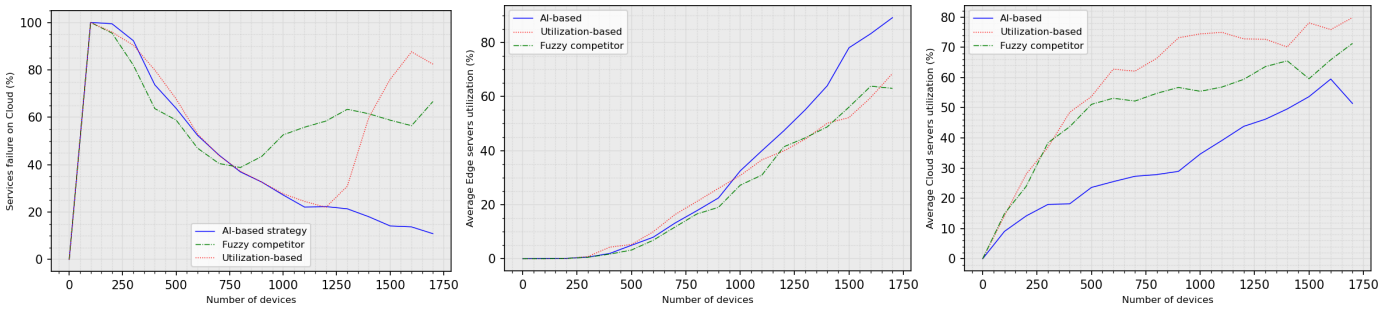


Fig. 4: 1) Services placement failure in Cloud; 2) Edge servers resources usage; 3) Cloud servers resources usage

mobility and nodes available resources in order to fasten placement process and minimize placement failure rate while optimizing the use of available Edge resources. Performance evaluation confirmed benefits of the proposed placement strategy in terms of placement failure rate as well as Edge and Cloud resources utilization rates. In future work, we plan to extend the proposed solution by considering more complex architectures integrating heterogeneous Edge domains.

References

[1] T. Sylla, M. A. Chalouf, F. Krief, and K. Samake, "Context-Aware Security in the Internet of Things: A survey," *International Journal of Autonomous and Adaptive Communications Systems*, no. 3, 2021.

[2] F. Liu, G. Tang, Y. Li, Z. Cai, X. Zhang, and T. Zhou, "A Survey on Edge Computing Systems and Tools," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1537–1562, Aug. 2019.

[3] T. Sylla, M. A. Chalouf, F. Krief, and K. Samaké, "Towards a context-aware security and privacy as a service in the internet of things," in *Information Security Theory and Practice*. Springer, 2020.

[4] H. Tabatabaee Malazi, S. R. Chaudhry, A. Kazmi, A. Palade, C. Cabrera, G. White, and S. Clarke, "Dynamic service placement in multi-access edge computing: A systematic literature review," *IEEE Access*, vol. 10, pp. 32 639–32 688, 2022.

[5] I. Petri, O. Rana, A. R. Zamani, and Y. Rezgui, "Edge-cloud orchestration: Strategies for service placement and enactment," in *2019 IEEE International Conference on Cloud Engineering (IC2E)*, 2019.

[6] 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, pp. 2333–2345, Oct. 2018.

[7] E. F. Maleki and L. Mashayekhy, "Mobility-aware computation offloading in edge computing using prediction," in *2020 IEEE 4th International Conference on Fog and Edge Computing (ICFEC)*. Melbourne, Australia: IEEE, May 2020, pp. 69–74.

[8] K. Lu, J. Song, L. Yang, G. Xu, and M. Li, "Dynamic Service Placement Algorithm for Partitionable Applications in Mobile Edge Computing," in *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. Taormina, Italy: IEEE, May 2022.

[9] L. Heng, G. Yin, and X. Zhao, "Energy aware cloud-edge service placement approaches in the internet of things communications," *International Journal of Communication Systems*, vol. 35, no. 1, p. e4899, 2022.

[10] H. Gao, W. Huang, T. Liu, Y. Yin, and Y. Li, "Ppo2: location privacy-oriented task offloading to edge computing using reinforcement learning for intelligent autonomous transport systems," *IEEE transactions on intelligent transportation systems*, 2022.

[11] F. Tavousi, S. Azizi, and A. Ghaderzadeh, "A fuzzy approach for optimal placement of IoT applications in fog-cloud computing," *Cluster Computing*, vol. 25, no. 1, pp. 303–320, Feb. 2022.

[12] S. Lu, J. Wu, J. Shi, P. Lu, J. Fang, and H. Liu, "A dynamic service placement based on deep reinforcement learning in mobile edge computing," *Network*, vol. 2, no. 1, pp. 106–122, 2022.

[13] A. Talpur and M. Gurusamy, "Drld-sp: A deep-reinforcement-learning-based dynamic service placement in edge-enabled internet of vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 8, pp. 6239–6251, 2022.

[14] C. Sonmez, A. Ozgovde, and C. Ersoy, "Fuzzy Workload Orchestration for Edge Computing," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 769–782, Jun. 2019.

[15] Z. Nezami, K. Zamanifar, K. Djemame, and E. Pournaras, "Decentralized edge-to-cloud load balancing: Service placement for the internet of things," *IEEE Access*, vol. 9, pp. 64 983–65 000, 2021.

[16] C. Sonmez, A. Ozgovde, and C. Ersoy, "EdgeCloudSim: An environment for performance evaluation of edge computing systems: EdgeCloudSim," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 11, p. e3493, Nov. 2018.

[17] K. Goel, A. Bhaumick, D. Kaushal, and S. Bagchi, "Reliability Analysis of Edge Scenarios Using Pedestrian Mobility," in *2020 50th Annual IEEE-IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*. Valencia, Spain: IEEE, Jun. 2020, pp. 61–62.

[18] G. Liang and H. Yu, "Network selection algorithm for heterogeneous wireless networks based on service characteristics and user preferences," *EURASIP Journal on Wireless Communications and Networking*, vol. 2018, no. 1, p. 241, Dec. 2018.

[19] T. Sylla, M. A. Chalouf, F. Krief, and K. Samaké, "Setucom: Secure and trustworthy context management for context-aware security and privacy in the internet of things," *Security and communication networks*, 2021.