Dynamic Migration Strategy for Mobile Multi-Access Edge Computing Services

Ibtissam Labriji[1], Emilio Calvanese Strinati[2], Eric Perraud[3], Frederic Joly[1]
e-mail: [1]{name.surname}@renault.fr, [2]{name.surname}@cea.fr, [3]{name.surname}@qorvo.fr

2022 IEEE Wireless Communications and Networking Conference (WCNC) | 978-1-6654-4266-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/WCNC51071.2022.9771612

***Abstract*—The concept of Internet-of-Vehicles (IoV) has emerged to support the future Intelligent Transportation System (ITS). As an enabling technology for the IoV, Multi-access Edge Computing (MEC) provides cloud computing capabilities at the edge of the radio access network and supports vehicular task offloading of low-latency services. However, vehicles move across radio cells and, computing services offloaded to the Mobile Edge Hosts (MEH)s might be interrupted. The key to ensuring service continuity is to preemptively anticipate the service migration process to a well-identified subset of available MEHs, at the optimal time. In this paper, we develop a new algorithm that combines the exponential-weight algorithm for exploration and exploitation (EXP3) and the Lyapunov optimization to jointly and proactively decide *when* to trigger migration and *where* to migrate the service before the handover. Our algorithm leads to significant cost saving while meeting any given *risk* and *availability* targets which are new metrics introduced to assess service continuity.**

***Index Terms*—Multi-access Edge Computing (MEC), internet of vehicles (IoV), service continuity, multi-armed bandit (MAB), Lyapunov optimization.**

I. Introduction

The automobile industry has been a major economic sector for over a century and is currently experiencing one of its biggest shifts in decades: the so-called IoV. In parallel, the MEC is regarded as an important building block in the future IoV ecosystem. Introduced by the European Telecommunications Standards Institute (ETSI) [1], the MEC is a framework that brings computation and storage resources to the edge of a mobile network, enabling it to run computation-intensive applications while avoiding the excessive latency experienced with cloud computing. One of the main issues to explore in vehicular cloud computing networks is minimizing service interruption during handovers. For instance, User Equipments (UE)s can change their serving base stations without service interruption, thanks to the radio handover (HO) mechanisms. We should be able to apply the same principle to offloaded MEC services. Also, many of the new use cases of edge computing are strongly stateful [2], which makes it necessary to move the states of running applications as vehicles move within a MEC network. Consequently, when a vehicle is about to leave its current MEH, it expects its running service to be directly resumed in its next serving MEH. Previously in [3], we addressed the topic of service continuity across MEHs, with a focus on where to migrate (only) the running service instance to avoid discontinuity. Accordingly, we introduced a new metric to evaluate the risk of service discontinuity in case the service instance is migrated only to MEHs which will not be visited after the HO. In the present paper, we tackle the same problem as before ("where to migrate"), but this time we investigate the "when to migrate" issue in addition, which is as fundamental as the first one, in the sense that, it is critical to trigger the migration process at the right time to avoid any delays in resuming the service after the HO. Note that when migration is triggered very early, it leads to higher energy consumption, since the next serving MEH cannot be predicted with certainty. Therefore, the system is forced to perform more migrations in the neighboring MEHs to minimize the risk of service interruption. We propose a novel lightweight service migration algorithm that decides the optimal instant to trigger migration (when) depending on how fast the vehicle is moving and the MEHs where these instances are to be migrated. To the best of our knowledge, this is the first time the question "when to trigger migration" is addressed and the strength of our solution lies in jointly deciding when and where to migrate the service instance. Also, our solution achieves relevant energy savings while guaranteeing a predefined level of service continuity depending on the requested type of service. Our paper is organized as follows: Section II gives a brief overview of the related works. Section III describes the system model and details the problem formulation. The designed algorithm along with the online closed-form solution is presented in section IV. In section V we present our numerical results. Our conclusions are drawn in the final section VI.

II. Related Works

In the literature, several works addressed the issue of guaranteeing service continuity across MEHs. As a result, *migration* has gained more importance in recent years, specifically *proactive* migration [4] (i.e. triggered in advance, before the UE hands over to its next destination) which is more adapted to ultra-short latency applications. Much is known about 1) whether the ongoing service should be migrated out of the current edge server [5] or not 2) which edge server the service instance should be migrated to [6] 3) how the service migration process should be carried out, considering the Quality of Service (QoS) requirements [7]; but little is known about the optimal time to trigger migration to make services readily available and allow vehicles to have immediate access in their next serving MEH. Moreover, the management of the MEC handover must be anticipated, unlike the HO. The data exchanged during the radio HO is small and only requires milliseconds to tens of milliseconds to be transferred. While for service migration in MEC, the data that should be transferred (memory state data, application image data, etc.) is always very large and requires up to a few seconds to be fully transferred when migration is container-based [8]. Consequently, service migration triggering cannot wait until the radio HO is detected, otherwise, the UE will wait additional time before resuming its service in the next

MEH. This is a critical issue for delay-sensitive applications. The ETSI standards [9] took an interest in this issue and proposed to (i) wait until the MEH-UE connection is lost to decide what's the optimal next MEH, based on the UE location, (ii) notify the neighboring MEHs when the HO starts through Radio Network Information Service (RNIS) and then trigger service migration. Both options might result in service discontinuity due to the unavailability of the migrated service when the UE arrives under the new MEH. In this paper, we propose to jointly and proactively determine when and where to migrate containers that host offloaded services.

III. System Model

A. System Description

We consider a MEC network consisting of a set $\mathcal{S}$ of MEH servers and a set $\mathcal{V}$ of vehicles. A MEH server can either be co-located with one or several Next Generation NodeBs (gNB)s and is equipped with multi-core CPUs that process the different requests offloaded by vehicles to access services hosted on the MEC. Neither the vehicles nor the MEC has prior information about which MEH will be optimal to resume the vehicle's service when the HO happens. Also, in the proximity of a MEH handover area, there is a set of neighboring MEHs $\mathcal{N}_{MEH} = \{1, 2, \ldots, N_{MEH}\}$ available to host the transferred processing. A mobility prediction unit is installed in the MEH and returns, in the form of a vector, a probability distribution over $\mathcal{N}_{MEH}$. Additionally, an admission control unit is set up to only accept requests that are likely to be migrated i.e. submitted by vehicles exiting the serving MEH soon (at a distance smaller than δ meters from the edge of the coverage area). The set of these requests, ranging from 0 to N_{Max}, is referred to as $r \in \mathcal{A}(t)$ and $N(t) = |\mathcal{A}(t)|$. Finally, the designed algorithm that assesses when and where the computing services are migrated, is hosted in another unit at the MEH and is designed in a time slot manner: the timeline of our algorithm is slotted and slots are indexed by $t \in \mathbb{N}$. Every time slot t, the algorithm collects requests of vehicles that are existing the MEH coverage soon, selects the optimal instant to trigger migration (within a predefined time window) to avoid early/late migration, and decides where to migrate the running services. We focus on proactive live migration where stateful application processes are migrated across MEHs using containers, a lightweight technology.

B. Problem Formulation

In this paper, we extend the work presented in [3]: We identify the optimal instant to trigger migration, in addition to the number of instances to migrate before the HO event. For every $r \in \mathcal{A}(t)$, the optimal instant to trigger migration is selected from a set of K possible instants $\mathcal{T}_K = \{T_1, T_2, ..., T_K\}$. $\mathcal{T}_K$ is within a time window, lower-bounded by $T^{Notification}$, the instant when the vehicle is notified that its request must be migrated and upper-bounded by T^{HO}, the instant when the HO is detected. Note that, as time goes on, the initial set $\mathcal{N}_{MEH}$ starts to contain fewer candidate MEHs (MEHs from which the vehicle moves away are not considered). We refer to the set of neighboring MEHs when migration is triggered at T_k as $\mathcal{N}^k_{MEH} = \{1, 2, \ldots, N^k_{MEH}\}$, where $N^k_{MEH} \leq N_{MEH}$ represents the number of remaining available MEH candidates at T_k. The mobility prediction vector associated with $\mathcal{N}_{MEH}$ is referred to as: $p^k_r = \left(p^k_{r,1}, p^k_{r,2}, \ldots, p^k_{r,N_{MEH}}\right)$ and evolves according to T_k. $p^k_{r,i}$ represents the probability that the vehicle requesting r, hands over to MEH $i \in \mathcal{N}_{MEH}$ considering that migration is triggered at T_k. The entries corresponding to the most likely MEHs to be visited at T_k get closer to 1 while others get closer to 0. Finally, the entries of p^k_r are arranged in a decreasing order, $p^k_{r,1} \geq p^k_{r,2} \geq \cdots \geq p^k_{r,N_{MEH}}$ and $\sum_{i=1}^{N_{MEH}} p^k_{r,i} = 1$. We introduce a new variable to control when migration is triggered: $\theta_k(t) = 1$ and $\sum_{k=1}^{K} \theta_k(t) = 1$ means that the instant chosen to trigger migration is T_k. We associate this set of indicator functions $\mathcal{I}_K = \{\theta_1(t), \theta_2(t), ..., \theta_K(t)\}$ to $\mathcal{T}_K$, to specify which of these instants is chosen. It is possible to consider more instants for migration triggering i.e. with a finer timeline granularity or rather focus on instants in which we observe a significant change in p^k_r i.e. when some MEHs are dropped. This way the set of considered MEH candidates is reduced and less replications need to be performed since $N^1_{MEH} \leq N^2_{MEH} \leq \cdots \leq N^K_{MEH}$. At time slot t, when migration is triggered at T_k there are N^k_{MEH} MEH candidates for migration and $M^k_r(t) \leq N^k_{MEH}$ of these are selected. We define the control action vector as $\mathbf{\Omega}(t) = \left(\mathbf{\Omega}_1(t), \ldots, \mathbf{\Omega}_{N(t)}(t)\right)$, where $\mathbf{\Omega}_r(t) = \left(\theta_k(t), M^k_r(t)\right)$ is the control action taken for r which identifies the optimal instant to trigger migration as well as the number of replicas to perform. Let ψ_r be the energy required to perform a single migration of the container hosting the service requested by r, we define the energy consumed to migrate at T_k the container instance to $M^k_r(t)$ different MEH sites as:

$$E_r(t) = \sum_{k=1}^{K} M^k_r(t)\ \psi_r\ \theta_k(t), \tag{1}$$

The total energy consumed for container migrations is $E(t) = \sum_{r \in \mathcal{A}(t)} E_r(t)$ and the objective function we minimize is the long-term average energy associated with the migration of containers in multiple neighboring MEHs:

$$\lim_{T \to +\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[E(t)]. \tag{2}$$

Hereafter, we present the constraints of our problem: The risk (the same definition as in [3]) and the availability. The risk represents the probability to lose the service in case the container is migrated only to MEHs that will not be visited after the handover:

$$\zeta_r(t) = \sum_{k=1}^{K} \left(1 - \sum_{i=1}^{M^k_r(t)} p^k_{r,i}(t)\right) \theta_k(t), \tag{3}$$

Therefore, the average risk across all requests in t is:

$$\overline{\zeta(t)} = \frac{1}{N(t)} \sum_{r \in \mathcal{A}(t)} \sum_{k=1}^{K} \left(1 - \sum_{i=1}^{M^k_r(t)} p^k_{r,i}(t)\right) \theta_k(t). \tag{4}$$

As a first constraint of the system, the **C1: average long-term risk** $\overline{\zeta(t)}$ has to be close to a predefined target parameter

$\xi > 0$. This is why, we introduce the virtual queue $Z(t)$ which is incremented each time $\overline{\zeta(t)}$ exceeds ξ by $\overline{\zeta(t)} - \xi$:

$$Z(t+1) = \max\left\{Z(t) + \overline{\zeta(t)} - \xi, 0\right\}, \tag{5}$$

$Z(t)$ must be maintained mean rate stable i.e. not diverging to infinity to ensure that the **C1** constraint is satisfied. This is mathematically expressed by:

$$\lim_{t\to+\infty} \frac{\mathbb{E}[Z(t)]}{t} = 0. \tag{6}$$

Migrating enough containers to land on the right MEH is not enough to guarantee service continuity. It is necessary to make sure that the container is available at the arrival of the vehicle at its next MEH. We want to ensure that the migration is completed when the vehicle attaches with its new serving MEH, to avoid any additional waiting time. This is referred to as the availability and is expressed by $T_r^{Mig} \leq T_{r,k}^{Rem}(t)\ \forall\ r$. Where, $T_{r,k}^{Rem}(t)$ is the time remaining for the vehicle to reach the next MEH when migration is triggered at T_k, which depends on the vehicle's speed, and T_r^{Mig} is the migration time required to migrate the container handling r. Our aim is to control, every time slot t, the rate of requests for which $T_r^{Mig} \leq T_{r,k}^{Rem}(t)$ is met. ($\mathbb{1}\{.\}$ is the indicator function)

$$\overline{\Gamma(t)} = \frac{1}{N(t)} \sum_{r\in\mathcal{A}(t)} \sum_{k=1}^{K} \mathbb{1}\{T_r^{Mig} \leq T_{r,k}^{Rem}(t)\}\ \theta_k(t). \tag{7}$$

To do so, we force $\overline{\Gamma(t)}$ to be close to a predefined target percentage γ. This represents the second constraint of our system: The **C2: average long-term availability**. Similarly, to enforce constraint **C2**, we define another virtual queue $Y(t)$ with the following update equation:

$$Y(t+1) = \max\left\{Y(t) + \gamma - \overline{\Gamma(t)}, 0\right\}, \tag{8}$$

When $\overline{\Gamma(t)}$ goes below γ, $Y(t)$ is incremented by $\gamma - \overline{\Gamma(t)}$. The mean rate stability of $Y(t)$ will allow us to control the availability of the migrated containers as vehicles hand over to new MEHs. It is mathematically expressed by:

$$\lim_{t\to+\infty} \frac{\mathbb{E}[Y(t)]}{t} = 0. \tag{9}$$

Both the risk and availability are very much influenced by the choice of T_k. Triggering a migration late leads to a higher risk of unavailability. Conversely, triggering it early is not very optimal since p_k^r is not yet significant (all possible MEHs are equiprobable) and more migrations should be performed. The solution we propose, efficiently choose the actions $\Omega(t)$ to control the backlog of $Z(t)$ and $Y(t)$. The overall optimization problem is written as:

$$\lim_{T\to+\infty} \min_{\substack{\mathbf{\Omega}(t),\\ t\in\{0,\dots,T-1\}}} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[E(t)]$$

Subject to:

$$\begin{aligned}
&(a)\ \lim_{T\to\infty} \frac{\mathbb{E}\left[Z(T)\right]}{T} = 0, (b)\ \lim_{T\to\infty} \frac{\mathbb{E}\left[Y(T)\right]}{T} = 0,\\
&(c)\ \theta_k(t) \in \{0,1\}, \sum_{k=1}^{K} \theta_k(t) = 1, \forall\, k, t,\\
&(d)\ M_r^k(t) \in \{0,\dots,N_{\mathrm{MEH}}^k\}, \forall\ k, t.
\end{aligned} \tag{10}$$

IV. Proposed proactive migration algorithm

To tackle the problem in Eq. (10) we use mathematical techniques from *Lyapunov* optimization, see e.g. [10]. First, we define the Lyapunov function as:

$$L(\mathbf{\Theta}(t)) \triangleq \frac{1}{2}\left[Z(t)^2 + Y(t)^2\right], \tag{11}$$

Where, $\mathbf{\Theta}(t) = (Z(t), Y(t))$. Next, we define the *one-slot conditional Lyapunov drift* which represents the expected change in the Lyapunov function over one slot, given that the current state in slot t is $\mathbf{\Theta}(t)$:

$$\Delta(\mathbf{\Theta}(t)) \triangleq \mathbb{E}[L(\mathbf{\Theta}(t+1)) - L(\mathbf{\Theta}(t))|\mathbf{\Theta}(t)], \tag{12}$$

Minimizing $\Delta(\mathbf{\Theta}(t))$ provides queues stability. At this point, the problem is only half solved as virtual queues only help to ensure the desired time average constraints are met. We incorporate the energy cost through a control parameter $V > 0$. Therefore, a control action is taken every slot t to greedily minimize the following *drift-plus-penalty* expression:

$$\Delta(t) = \Delta(\mathbf{\Theta}(t)) + V\mathbb{E}\Big[E(t)|\mathbf{\Theta}(t)\Big], \tag{13}$$

Instead, we minimize the following upper-bound of Eq. (13) (the proof of the upper-bound computation is referred to in the appendix):

$$\begin{aligned}
\Delta(t) \leq 1 + \mathbb{E}\Bigg[&Z(t)\left(\overline{\zeta(t)} - \xi\right) + Y(t)\left(\gamma - \overline{\Gamma(t)}\right)\\
&+ VE(t)|\mathbf{\Theta}(t)\Bigg]
\end{aligned} \tag{14}$$

To minimize the upper-bound computed in Eq. (14), we use the concept of *opportunistically minimizing* an expectation [10, $\mathcal{S}1.8$]. Each time slot t, we identify the control action $\Omega(t)$, solution to the online optimization problem defined as: (The objective function $\phi(t)$ is none other than the upper-bound in (14) rearranged, where $\xi Z(t)$ and $\gamma Y(t)$ are are omitted, since they are not depending on the optimization variables $M_r^k(t)$ and $\theta_k(t)$)

$$\begin{aligned}
\min_{\Omega(t)}\ \phi(t) = \sum_{k=1}^{K} \theta_k(t) \sum_{r\in\mathcal{A}(t)} \Bigg[&VM_r^k(t)\ \psi_r\\
&+ \frac{Z(t)}{N(t)}\left(1 - \sum_{i=1}^{M_r^k(t)} p_{r,i}^k(t)\right)\\
&- \frac{Y(t)}{N(t)}\ \mathbb{1}\{T_r^{Mig} \leq T_{r,k}^{Rem}(t)\}\Bigg]
\end{aligned} \tag{15}$$

Subject to:

$$\begin{aligned}
&(a)\ \theta_k(t) \in \{0,1\}, \sum_{k=1}^{K} \theta_k(t) = 1,\ \forall k, t,\\
&(b)\ M_r^k(t) \in \mathcal{N}_{MEH}^k\ \forall k, r.
\end{aligned}$$

To simplify the algorithm design, we decide the k for which $\theta_k(t) = 1$ first and determine $M_r^k(t)$ for every r next. To identify the optimal k, each time slot t, we use EXP3, a powerful tool to address the exploration-exploitation trade-off dilemma: in fact, we must select one option k in advance (at the decision phase), while the payoffs of the options are not known in advance. The objective is to balance the need

to explore the options to find profitable ones and the desire to choose an option that has paid off well in the past. EXP3 is the most well-known algorithm to handle non-stochastic Multi-Armed Bandit (MAB) problems. Suggested by Auer et al. [11], the algorithm maintains a list of weights for the K options $\{\omega_1(t), \ldots, \omega_K(t)\}$ and selects, each time slot t, an option k^* according to a Gibbs distribution, a mixture of a uniform distribution and an empirical performance-related distribution:

$$p_k(t) = (1-\rho)\frac{\omega_k(t)}{\sum_{k=1}^{K}\omega_k(t)} + \frac{\rho}{K}. \tag{16}$$

After selecting k^*, the algorithm earns a payoff which in this case corresponds to the $\overline{\Gamma(t)}$ achieved at the end of time slot t having chosen the instant T_{k^*}. On its own, EXP3 is not able to guarantee the mean rate stability of the queue $Z(t)$ nor the minimization of the energy $E(t)$. Our objective is to propose a *context-aware* (CA) algorithm. Therefore, we combine EXP3 with the Lyapunov optimization through the payoff of the options: The payoff observed in **Step 3** in Algorithm 1, is none other than the Lyapunov objective function outcome $\phi(t)$ that correspond to the choice k^*, which will be noted:

$$x_{k^*}(t) = \sum_{r\in\mathcal{A}(t)} VM_r^{k^*}(t)\ \psi_r + \frac{Z(t)}{N(t)}\left(1 - \sum_{i=1}^{M_r^{k^*}(t)} p_{r,i}^{k^*}(t)\right) - \frac{Y(t)}{N(t)}\ \mathbb{1}\{T_r^{Mig} \le T_{r,k}^{Rem}(t)\} = \sum_{r\in\mathcal{A}(t)} x_{k^*,r}(t). \tag{17}$$

Combining EXP3 and the Lyapunov optimization allows the control actions $\Omega(t)$ to be dynamically adjusted to make choices that (i) maintain both queues of problem Eq. (15) mean rate stable (not only $Y(t)$) and (ii) minimize the energy cost incurred by migrations. After k^* is selected the next step is to find the optimal value of $M_r^{k^*}(t)$ given that the set of possible neighboring MEHs is $\mathcal{N}_{MEH}^{k^*}$. $x_{k^*}(t)$ naturally decomposes across r, therefore we minimize each term $x_{k^*,r}(t)$ individually, to find $M_r^{k^*}(t)^*$:

$$M_r^{k^*}(t)^* = \underset{M_r^{k^*}(t)\in\mathcal{N}_{MEH}^{k*}}{\operatorname{argmin}} x_{k^*,r}(t). \tag{18}$$

$M_r^{k^*}(t)^*$ is found by performing an exhaustive search. After identifying $M_r^{k^*}(t)^*$ we calculate $x_{k^*}(t)$ and inject it **Step 3** of Algorithm 1 so that the algorithm improves its choice of options in the future. Options yielding high payoffs quickly gain a high probability of being chosen through the weights since the system updates the weights of the selected option as:

$$\omega_{k^*}(t+1) = \omega_{k^*}(t)e^{\frac{\rho x_{k^*}(t)}{K p_{k^*}(t)}}. \tag{19}$$

To compensate the payoff of options with infrequent selection $\frac{x_{k^*}(t)}{p_{k^*}(t)}$ is considered instead of $x_{k^*}(t)$ in Eq. (19). For the unselected options $k \neq k^*$, the corresponding weights remain unchanged: $\omega_k(t+1) = \omega_k(t)$. $Z(t+1)$ and $Y(t+1)$ are also updated Eq. (5) and (8), according to the option that was previously selected.

Algorithm 1 CA-EXP3

Initialization: Let $\rho \in [0,1]$.
Initialize the weights of the options: $\omega_k(t) = 1,\ \forall k$,
repeat
  Step 1: Calculate $p_k(t)$ for every option, Eq. (16),
  Step 2: Select an option k^* through the probability distribution computed previously,
  Step 3: Observe the payoff of the selected option k^* Eq. (17),
  Step 4: Update $\omega_{k^*}(t+1)$ as in Eq. (19),
  Step 5: Update $Z(t+1)$ and $Y(t+1)$, according to Eq. (5) and (8) respectively,
until $t > T$;

The performance of EXP3 depends on tuning the parameter ρ. In [12], it was shown that for ρ to be appropriately set, an upper bound on the total expected payoff of the best option $EG_{max} = \max_{1\le k\le K}\mathbb{E}\left[\sum_t x_k(t)\right]$ should be known in advance. We design our algorithm so that it does not require any prior knowledge. Therefore, we use another version of EXP3 referred to as *EXP3.1* in [13] which does not need EG_{max} and proceeds in epochs.

Algorithm 2 CA-EXP3.1

Initialization: Let $c = \frac{K\ln(K)}{e-1}$, $G_k(t=0) = 0$, $epoch = 0$, $g_{epoch} = c4^{epoch}$, $\rho = 2^{-epoch}$.
repeat
  Step 1, **Step 2**, **Step 3**, **Step 4**, **Step 5** of Algorithm 1,
  Step 6: Update the total payoff for every option k, Eq. (20)
  if $\max_k G_k(t) > g_{epoch} - \frac{K}{\rho}$ **then**
    $epoch = epoch + 1$, $g_{epoch} = c4^{epoch}$, $\rho = 2^{-epoch}$.
  end
until $t > T$;

On each epoch, Algorithm 2 makes a guess on the bound for the total payoff of the best action g_{epoch}, tunes the parameter ρ, and runs Algorithm 1. The system maintains an estimate of the total payoff of each option up to time slot t $G_k(t) = \sum_{t'=0}^{t} x_k(t')$. Using these estimates, the algorithm detects when the actual total payoff of some option $G_k(t)$ has advanced beyond g_{epoch}. When this happens, the algorithm goes on to the next epoch and and use the new tuned ρ in Algorithm 1 with a larger bound on g_{epoch}. As long as the condition in the **if** is not valid, steps of Algorithm 1 run and the total payoff of the options are updated according to:

$$\begin{cases} G_{k^*}(t+1) = G_{k^*}(t) + x_{k^*}(t) & \text{If } k = k^*, \\ G_k(t+1) = G_k(t) & k \neq k^*. \end{cases} \tag{20}$$

V. Numerical Results

In this section, we evaluate the performance of our proposed algorithm CA-EXP3.1. We mimic an outdoor urban area in which gNBs are deployed at the street crosses or along the streets and where each gNB is colocated with a MEH server. CA-EXP3.1 decides when and where to migrate containers,

every time slot t and for every $r \in \mathcal{A}(t)$. This is done while minimizing the migration cost and meeting the requirements in terms of the risk and availability constraints. The parameters ξ and γ depend on the type of service requested and the associated QoS. The speed of vehicles issuing requests varies between v_{min} and v_{max}. We consider *Object detection* as an example of an offloaded service application hosted in a MEH container. We denote the size of the container hosting this service as W and the time required to migrate it across MEHs as T^{Mig} [14]. We use the model in [15], to estimate ψ_r which is a function of W. The mobility predictor remains the same as in [3]. Table I reports the numerical values of the parameters described previously. We run our simulations

TABLE I: Simulation parameters

Param	Value	Param	Value	Param	Value
K	3	N_{MEH}	4	T^{Mig}	70.1 s
ξ	0.20	γ	0.80	W	365 MB
v_{min}	20 km/h	v_{max}	60 km/h	ψ_r	0.84 MJ
N_{Max}	100	δ	500 m	V	1

for $T = 10000$ slots. However, for better visualization of the instantaneous metrics, the plot interval is restricted to the last 1000 slots towards the end of the simulation. Our simulation results show that queues $Z(t)$ and $Y(t)$ are stabilized and the energy cost related to migrations is minimized when $V = 1$. This value is found through exhaustive search. We compare our approach referred to as *CA-EXP3.1 approach* with two states of the art approaches: (i) triggering the migrations late at T^{HO}: *Migration at* T^{HO} (ii) triggering it early at $T^{Notification}$: *Migration at* $T^{Notification}$. In Fig. 1, we illustrate the average long-term availability for the 3 approaches. As expected, the availability metric $\overline{\Gamma(t)}$ is always zero for (i): with a migration that lasts T^{Mig}, the container will never be completely transferred when the vehicle arrives at the next MEH. Consequently, the service will be lost. Now, when migration is triggered at $T^{Notification}$ (ii), the container instances are always available and immediately accessible when the vehicle hands over since the system has enough time to transfer the saved in-memory states of the container, hence $\overline{\Gamma(t)} = 1$ always. Our approach curve is very close to (ii) with very small variations (see the zoomed part). In fact, we achieve an average long-term availability of 0.99 i.e. the container will not be immediately accessible only once every 100 requests, on average. This way, we are able to keep $\overline{\Gamma(t)}$ above the target availability $\gamma = 0.8$, thanks to EXP3 and its careful selection of the migration instants $T_k \in \mathcal{T}_K$. Moreover, our approach is more energy efficient compared to (ii), and controls, in parallel, the risk metric as required. This is illustrated in the following. Fig. 2, display the performance of the average long-term risk which should remain bounded around the target risk $\xi = 0.2$. This is achieved thanks to the Lyapunov framework, which controls $\overline{\zeta(t)}$ through the performed number of replicas $M_r^k(t)$. The lowest average risk value is achieved, as predicted with (i): when the container migration is triggered at the last moment, fewer neighboring MEHs are left to possibly host migration instances. Consequently, the possibility of migrating the container to MEHs that will never be visited by the vehicle is very small by design. Although the probability of

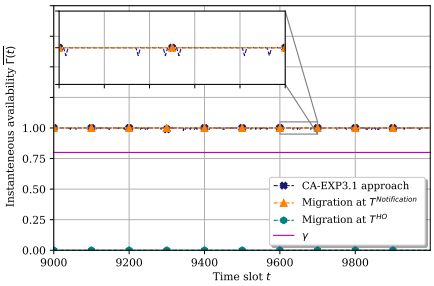

Fig. 1: Temporal evolution of the availability which is controlled and bounded, in an average sense (long-term), by the target availability $\gamma = 0.8$.

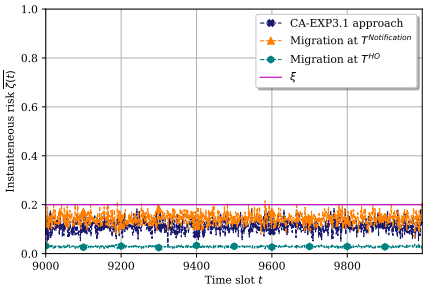

Fig. 2: Temporal evolution of the risk. Our algorithm is constrained, in an average sense (long-term), to stay bounded around the chosen target $\xi = 0.2$.

migrating to the wrong MEH is greater when migration is triggered earlier $< T^{HO}$, our solution can maintain this risk below ξ. Unlike (ii) which tries with great difficulty not to exceed the threshold ξ (the peaks of the curve constantly exceed the the threshold). In Fig. 3 we investigate the time evolution of the cumulative energy consumption per request for the considered approaches. From the graph, we can see that *Migration at* $T^{Notification}$ leads to the highest power consumption as it considers replicating the processing to the initial set $\mathcal{N}_{MEH}$. In this case, the system is forced to perform many more replications to respect the risk constraint, compared with Migration at T^{HO} where a reduced set of neighboring MEHs is considered for container migrations. Our approach lies between the two and effectively reduces (by more than 26%) the total energy consumed per request as compared to migration at $T^{Notification}$ approach. Our algorithm makes decisions without requiring prior knowledge of the vehicles arrival process and dynamically adapts to the changes of the speeds of the vehicles. This is illustrated in Fig. 4 through a part of the temporal evolution of the probability vector of the $K = 3$ options Eq. (16), where vehicles move according to 3 types of speed mode: Mode 1: high $(50 - 60$ km/h), Mode 2: Medium $(30 - 40$ km/h) and Mode 3: Low $(20 - 30$ km/h). Our algorithm tends to choose instant T_1 which is closer to $T_{Notification}$ (early migration triggering) when vehicles are driving very fast (Mode 1), T_3 which is closer to T_{HO} (late migration triggering) when they are driving slower (Mode 3) and T_2 which is $T_1 \leq T_2 \leq T_3$ in medium speed (Mode 2).

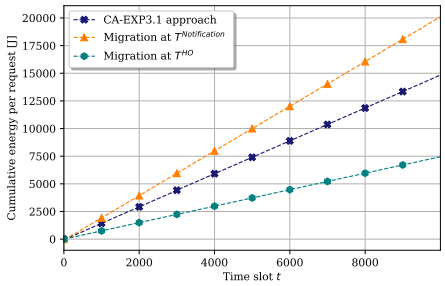

Fig. 3: Energy consumption per request *vs* time.

Fig. 4: Temporal evolution of the probability vector of the options Eq. (16) according to the speed modes.

VI. Conclusion

To guarantee the continuity of service across MEHs, it is essential to identify in advance when and where to migrate the running service. In this paper, we present a new algorithm that combines the EXP3 and Lyapunov techniques to jointly decide, before the HO, the optimal time to trigger service migration and select which MEHs are better placed to host the migration of the running service. Our numerical results show that the closed-form solution we propose outperforms two state of the art approaches by achieving the best compromise between minimizing the migration cost and meeting the service continuity constraints. The latter are configurable according to the desired QoS.

Appendix

Proof. In our problem formulation, we introduce an additional queue due to constraint (b) (Eq. (10)). The availability is controlled by queue $Y(t)$ which is upper-bounded as follows: (The proof for $Z(t)$ remains the same as in [3])
Applying inequality $\max(x,0)^2 \leq x^2, \forall x \in \mathbb{R}$, to Eq. (5) and (8) we obtain:

$$Z(t+1)^2 \leq Z(t)^2 + \left(\overline{\zeta(t)} - \xi\right)^2 + 2Z(t)\left(\overline{\zeta(t)} - \xi\right), \quad (21)$$

$$Y(t+1)^2 \leq Y(t)^2 + \left(\gamma - \overline{\Gamma(t)}\right)^2 + 2Y(t)\left(\gamma - \overline{\Gamma(t)}\right), \quad (22)$$

By construction, since $0 \leq \overline{\zeta(t)} \leq 1$ and $0 \leq \xi \leq 1$ it follows that $\left(\overline{\zeta(t)} - \xi\right)^2 \leq 1$ and:

$$Z(t+1)^2 - Z(t)^2 \leq 1 + 2Z(t)\left(\overline{\zeta(t)} - \xi\right), \quad (23)$$

Similarly, as $0 \leq \overline{\Gamma(t)} \leq 1$ and $0 \leq \gamma \leq 1$:

$$Y(t+1)^2 - Y(t)^2 \leq 1 + 2Y(t)\left(\gamma - \overline{\Gamma(t)}\right), \quad (24)$$

Subsequently:

$$\begin{aligned}\Delta(t) &= \mathbb{E}\Bigg[\frac{1}{2}\left(Z(t+1)^2 - Z(t)^2 + Y(t+1)^2 - Y(t)^2\right) \\ &\quad + VE(t)|\boldsymbol{\Theta}(t)\Bigg] \\ &\leq 1 + \mathbb{E}\Bigg[Z(t)\left(\overline{\zeta(t)} - \xi\right) + Y(t)\left(\gamma - \overline{\Gamma(t)}\right) \\ &\quad + VE(t)|\boldsymbol{\Theta}(t)\Bigg].\end{aligned} \quad (25)$$

□

References

[1] "Multi-access edge computing (MEC); framework and reference architecture," Dec. 2020, GS MEC 003 V2.2.1.

[2] Koichiro Amemiya and Akihiro Nakao, "Layer-integrated edge distributed data store for real-time and stateful services," in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, 2020, pp. 1–9.

[3] Ibtissam Labriji, Francesca Meneghello, Davide Cecchinato, Stefania Sesia, Eric Perraud, Emilio Calvanese Strinati, and Michele Rossi, "Mobility aware and dynamic migration of mec services for the internet of vehicles," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 570–584, 2021.

[4] Tung V Doan, Zhongyi Fan, Giang T Nguyen, Dongho You, Alexander Kropp, Hani Salah, and Frank HP Fitzek, "Seamless service migration framework for autonomous driving in mobile edge cloud," in *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2020, pp. 1–2.

[5] Jan Plachy, Zdenek Becvar, and Emilio Calvanese Strinati, "Dynamic resource allocation exploiting mobility prediction in mobile edge computing," in *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. IEEE, 2016, pp. 1–6.

[6] Fei Zhang, Guangming Liu, Bo Zhao, Xiaoming Fu, and Ramin Yahyapour, "Reducing the network overhead of user mobility–induced virtual machine migration in mobile edge computing," *Software: Practice and Experience*, vol. 49, no. 4, pp. 673–693, 2019.

[7] Ivan Farris, Tarik Taleb, Hannu Flinck, and Antonio Iera, "Providing ultra-short latency to user-centric 5G applications at the mobile network edge," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 4, pp. e3169, 2018.

[8] Andrew Machen, Shiqiang Wang, Kin K Leung, Bong Jun Ko, and Theodoros Salonidis, "Migrating running applications across mobile edge clouds: poster," in *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, 2016, pp. 435–436.

[9] "Mobile edge computing (MEC); end to end mobility aspects," Oct. 2017, GR MEC 018 V1.1.1.

[10] Michael J Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[11] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2, pp. 235–256, 2002.

[12] Omar Besbes, Yonatan Gur, and Assaf Zeevi, "Stochastic multi-armed-bandit problem with non-stationary rewards," *Advances in neural information processing systems*, vol. 27, pp. 199–207, 2014.

[13] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire, "Gambling in a rigged casino: The adversarial multi-armed bandit problem," in *Proceedings of IEEE 36th Annual Foundations of Computer Science*. IEEE, 1995, pp. 322–331.

[14] Andrew Machen, Shiqiang Wang, Kin K Leung, Bong Jun Ko, and Theodoros Salonidis, "Live service migration in mobile edge clouds," *IEEE Wireless Communications*, vol. 25, no. 1, pp. 140–147, 2017.

[15] Muhammad Zakarya, Lee Gillam, Ayaz Ali Khan, and Izaz Ur Rahman, "Perficientcloudsim: a tool to simulate large-scale computation in heterogeneous clouds," *The Journal of Supercomputing*, vol. 77, no. 4, pp. 3959–4013, 2021.