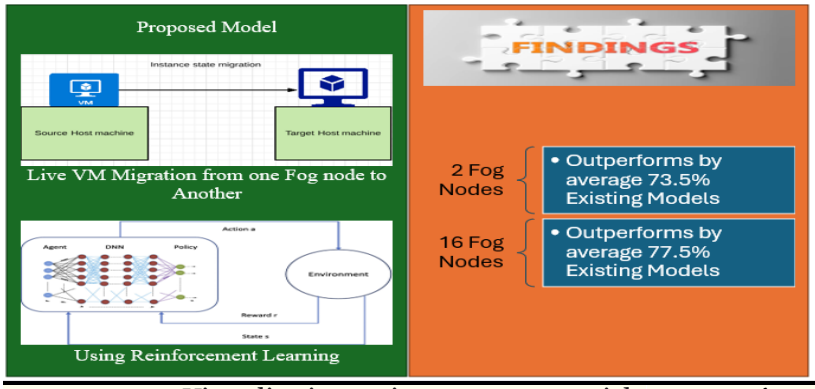
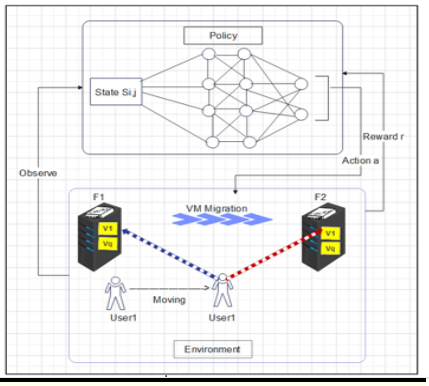
# Reinforcement Learning-Driven Decision-Making for Live Virtual Machine Migration in Fog Computing

Shahd Alqam*, Nasser Al- Zidi, Abderrezak Touzene, and Khaled Day
College of Science, Sultan Qaboos University, Muscat, Oman




**ABSTRACT:** Virtualization is an essential mechanism in fog computing that enables elasticity and isolation, which in turn helps achieve resource efficiency. To bring high flexibility in a fog environment, migration of virtual machines from one node to another is required. This can be achieved by live virtual machine migration to reduce downtime and delays. Multiple existing studies have discussed live virtual machine migration in a fog environment. However, these studies have some limitations, such as pre-migrating the virtual machines based on mobility prediction only or based on the load only, which causes an issue of late and early handover. Due to the dynamic nature of fog environments, VM migration decisions require consideration of multiple factors. Hence, there is a need to develop a system that considers multiple factors to decide to migrate a virtual machine or not to solve the issue of early and late handover. This study proposes a novel approach to live virtual machine migration that applies reinforcement learning for decision-making. Experiments show that the proposed approach significantly reduces the latency of time-critical applications. The proposed system, outperforms the existing systems in terms of total average reward. The system outperformed the mobility-only-based system by 97% when tested with two fog nodes and by 80% when tested with sixteen fog nodes in terms of average reward. Further, the proposed system outperforms the load-based system by 50% and 75% when the environment consists of two fog nodes and sixteen fog nodes, respectively. This proved that considering multiple factors in deciding virtual machine migration in a fog environment can be effectively applied in time-critical applications to reduce latency.

**الملخص:** إن المحاكاة الافتراضية هي آلية أساسية في الحوسبة الضبابية، حيث تتيح المرونة التي تساعد في كفاءة الموارد لتحقيق مرونة عالية في بيئة الحوسبة الضبابية. ولذا يلزم نقل بعض الأجهزة الافتراضية (VMs) من جهاز إلى آخر. ويمكن تحقيق ذلك عن طريق نقل الجهاز الافتراضي إذ يعمل على تقليل وقت التوقف عن العمل والتأخير في إنجازه. وفي هذا الصدد ناقشت العديد من الدراسات الحالية نقل الأجهزة الافتراضية أثناء عملها في بيئة الحوسبة الضبابية، إلا أن هذه الدراسات يتخللها بعض نقاط الضعف مثل: النقل المسبق لـ الأجهزة الافتراضية بناءً على تنبؤات حركة المستخدمين، أو بناءً على حمولة الأجهزة فقط، مما يسبب مشكلة النقل المبكر أو المتأخر. كما يُفضل النظر في بيئة الحوسبة الضبابية إلى عدة عوامل لتقرير نقل الأجهزة الافتراضية؛ لأنها بيئة ديناميكية وتحيط بها العديد من العوامل. ومن هذا المنطلق تظهر الحاجة لتطوير نظام يأخذ في الاعتبار عوامل متعددة لتقرير ما إذا كان سيتم نقل الجهاز الافتراضي أم لا. وللتمكن من حل مشكلة النقل المبكر أو المتأخر تطرح هذه الدراسة نهجًا جديدًا لنقل VM في بيئة الحوسبة الضبابية عن طريق تطبيق التعلم المعزز لاتخاذ القرار. ولقد أظهرت التجارب أن هذا النهج يقلل بشكل كبير من زمن انتقال التطبيقات ذات الأهمية الزمنية. بالإضافة إلى أن نظامنا المقترح هذا، المسمى VM_MIG ، يفوق النماذج الأخرى من حيث زمن الوصول بحوالي 77.5٪ ، وقد أثبت هذا أنه من الأفضل أخذ عوامل متعددة في عين الاعتبار لتحديد نقل الأجهزة الافتراضية في بيئة الحوسبة الضبابية بشكل فعال في التطبيقات ذات الأهمية الزمنية لتقليل زمن الوصول.

***Keywords:*** Latency reduction, live virtual machine migration, load balancing, reinforcement learning, Reinforcement Learning algorithms, reward mechanism

***الكلمات المفتاحية:*** تقليل وقت الاستجابة، نقل الجهاز الافتراضي المباشر، موازنة الحمل، تعزيز التعلم، خوارزميات تعزيز التعلم، آلية التحفيز.

Corresponding author's e-mail: s125248@student.squ.edu.om

DOI:10.53540/tjer.vol.20iss2pp113-122

## INTRODUCTION

According to (Giri et al., 2017), fog computing is an architecture that uses one or more near-edge devices to carry out some amount of storage, communication, control, configuration, and management of the cloud. It does not substitute the cloud but extends its functionality near the edge of users. Fog means "a cloud closer to the ground" and is a new technology that extends - but doesn't substitute - the cloud computing services to the end users (Mouradian et al., 2018). Fog provides different facilities and services to the users, i.e., latency-aware, geo-distributed, and mobility-aware services.

Fog computing was initially designed and introduced to support latency-sensitive applications. This includes Smart City applications, Intelligent Transportation Systems, Augmented Reality, Healthcare, Tele-Surveillance, Smart Grid, Smart Agriculture, Smart Waste Management, Smart Water Management, Smart Retail stores, etc. (Haouari et al., 2018) (Naha et al., 2018),(Yi et al., 2016),(Hu et al., 2017).

Virtualization plays a crucial role in efficient resource allocations to end users in fog computing scenarios. However, improper placement of virtual machines on physical machines can lead to performance degradation(Akintoye & Bagula, 2019). In the context of a fog scenario, the importance of considering performance and minimizing downtime becomes even more critical(Yi et al., 2016). Hence, it could be important to move the virtual machines from one fog node to another to ensure a smooth transition and no disturbance to the real-time applications(Zhou et al., 2019; Bittencourt et al., 2015; Govindaraj & Artemenko, 2018; Puliafito et al., 2019; Rodrigues et al., 2017; Roig et al., 2019; Rosário et al., 2018; J. Wang et al., 2020; F. Zhang et al., 2018).

In the area of fog computing, determining whether or not to migrate virtual machines/services from one node to another is a critical decision(Osanaiye et al., 2017), and the research community and industry have developed various mechanisms to make this decision effective based on different factors and parameters(Agarwal et al., 2016; Filiposka et al., 2018; Govindaraj & Artemenko, 2018; Habibi et al., 2020; Live Virtual Machine Migration to Support Real-Time IoT Applications in Heterogeneous Fog Environment V3, n.d.; Machen et al., 2018; Rahbari & Nickray, 2019; Zhu et al., 2017). Several studies have addressed this issue by developing different migration strategies based on user mobility, load balancing, energy saving, etc. In this context, reinforcement learning can be applied to make decisions more effective.

In previous studies, researchers explored the live virtual machine migration in the fog computing environment. Some studies (Goncalves et al., 2018) and (C. Zhang & Zheng, 2019) only migrated virtual machines in advance based on mobility predictions. As a result, it is difficult to make an accurate decision on whether or not to migrate the VM based on mobility prediction only, as the fog environment is dynamic and the users' movements cannot be accurately predicted, which could lead to the issue of early handover and wrong decision-making. Other studies consider the load as a factor in deciding the migration of the VMs and only migrate the VMs when the machines become overloaded (Tang et al., 2018). As a result, this might lead to the issue of late handover and may cause a disturbance to the real-time application and hence increase the delay.

Load and mobility are critical factors when deciding on VM migration because they are among the most influential factors that impact the performance, resource utilization, and latency requirements of real-time applications, as mentioned before.

In a fog environment, the joint consideration of both mobility and load factors with threshold values significantly impacts the decision-making process of whether or not to migrate the VMs.

However, previous studies have applied conventional algorithms as well as artificial intelligence-based algorithms to design their model. The study (C. Zhang & Zheng, 2019) proposed deep Reinforcement Learning in a Mobile Edge Computing system to decide the migration of VM. In the study (Basu et al., 2019), the authors applied RL to utilize the load details as a parameter to determine whether or not to migrate the VMs. RL agents monitor the machines' loads in the environment and decide on the migration. (Goncalves et al., 2018) I proposed using Integer Linear Programming (ILP) to find out the movement of the users and decide the migration accordingly.

For delay-sensitive applications in a fog environment, it is important to consider the high dynamic level of the environment and the complexity as a fog environment is surrounded by many factors. Delay is critical to the real-time application, and hence, the algorithms used to decide the VM migration should be carefully chosen and designed to achieve the possible reduction in latency and response time. It is worth noting that the fog environment is dynamic, consists of different types of devices, is highly affected by the users' mobility, and the load factor is critical in making decisions(Hammoudeh Rlr, 2018). Therefore, it is suitable and advised to design a model to decide the VM migration while applying RL algorithms. In Reinforcement learning, the system doesn't have a set of data to train, but rather, it learns by trial and error (Han, 2018). The system learns from the interaction with the environment to achieve the goal, which is usually maximizing the reward in the long term. RL is suitable for solving complex but narrow problems where the dynamic and large environment is not a problem and the input of the surrounding area is not expected, but there is a goal to be achieved. Hence, in this paper, we propose a reinforcement learning-based approach to improve the decision-making about whether or not to migrate virtual machines. This proposal aims to solve the issue of early and late handover of virtual machines. This is done by combining mobility and load factors into a single comprehensive model to reduce disruptions to time-critical applications. By combining both factors in one model, the RL can make a more informed and context-aware decision.

**Table 1**. Comparison between some existing studies and our proposed research.

| Reference study | Timing of the designed solution | | Decision to be taken | |
|---|---|---|---|---|
| | Reactive | Proactive | When | Where |
| (Bao et al., 2017) | | √ | √ | |
| (Goncalves et al., 2018) | | √ | √ | √ |
| (C. Zhang & Zheng, 2019) | | √ | √ | |
| (Basu et al., 2019) | √ | | √ | |
| Proposed Research | | √ | √ | |

This comprehensive approach can lead to improved performance optimization and reduction of latency for time-critical applications. Further, the model can become more adaptive to the changing conditions in the fog environment and can adjust the VM migration decision accordingly. This model enhances the performance, adaptability, and synergy between the incorporated factors. The rest of this paper is organized as follows: Section 2 provides an overview of the related work. Section 3 presents the system architecture and model. In section 4, the problem formulation is outlined. The RL algorithm is detailed in section 5. Section 6 illustrates the evaluation of the proposed model. The paper concludes in section 7, followed by a discussion on limitations and future work in section 8.

## RELATED WORK

A study (Bao et al., 2017) proposed a framework to support smooth handover between the fog nodes in a timely manner. In this study, the authors are following the principle of "Follow me", similar to follow me cloud, which aimed at a smooth migration from one data centre to another (S. Wang et al., 2018). This proposal results in a reduction in the service

Interruption time and downtime. The study guarantees service continuity and reduces the latency during handover. It proactively makes the decision to migrate to the virtual machine. However, the shortcoming of this research is that this method is good in scenarios in which the users' movement and the movement are fixed. There is no specific algorithm used in this proposal, only a framework to measure the strength of the signal in the access point and then migrate it. Further, the study considers only one factor in deciding following the user movement and not the load of the source and destination machines, which might lead to wrong decision-making.

A different approach is used in the study (Goncalves et al., 2018), where the proposal used Integer Linear Programming (ILP) to decide when to migrate the VMs. It takes the mobility prediction and starts the process 5 minutes before the user's movement. They used ILP to optimize decision-making. The objective functions are 1) Maximizing the accepted requests and 2) Minimizing the latency. However, the limitation of this study is that it proactively migrates the VM based on mobility prediction. This might lead to early handover and lead to a wrong decision.

The study doesn't consider another factor in deciding when to migrate, which might also lead to inappropriate decisions as the fog environment is surrounded by multiple factors.

A study (C. Zhang & Zheng, 2019) proposed deep Q-network for task migration in Mobile Edge Computing to decide whether to migrate the virtual machine. This research applies the reinforcement learning algorithm to learn the environment and then make the decision. In this research, mobility is the main trigger for migration, where a fog node master learns the status of the environment, takes action accordingly, and then gets a minimum cost value. This study shows better improvement compared to conventional algorithms, however, but it only considers one factor, which might lead to inappropriate decisions in the fog context.

In the study (Basu et al., 2019), the authors considered the load factor in making migration decisions without considering a threshold, and they migrated virtual machines after the fog node was already overloaded. RL can utilize the load details as a parameter to determine whether or not to migrate the VMs. RL agents can monitor the machines' loads in the environment and decide on the migration. Considering the load factor while adopting RL, facilitating the load balancing, and
distributing the workload. It also helps in improving efficiency. However, waiting for the system to be overloaded and then deciding to migrate may cause an issue of late handover and hence incur a delay.
The following table summarizes the existing studies in this context in comparison with the proposed research.

## SYSTEM MODEL

The proposed system is designed for a square area. The area size is assumed to be 50 m × 50 m. This site has been chosen because the coverage of the network in the fog nodes could reach a maximum of 60 meters. So, to ensure that all fog nodes are connected, this area size is chosen.
Before discussing the model, it is important to define some of the notations and variables used in the system model and problem formula. Table 2 defines the important notations and their abbreviations:

**Table 2.** List of notations applied in designing the system model.

| Symbol | The meaning |
|---|---|
| $\mathcal{F}$ | fog node set |
| $J$ | number of the fog nodes |
| $F^{j}_{\left(La^{j}_{k},Lo^{j}_{l}\right)}$ | represents a $j^{th}$ fog node with latitude and longitude positions of k and l, respectively, ranging from 1 to $I$. |
| $\mathcal{V}\left(F^{j}_{\left(La^{j}_{k},Lo^{j}_{l}\right)}\right)$ | The virtual machines set in fog node $j$ |
| $Q$ | number of virtual machines in one fog node |
| $v^{j}_{\left(La^{j}_{k},Lo^{j}_{l}\right),\mathfrak{q}}$ | Represents the is the $\mathfrak{q}^{th}$ VM running on the fog node $F^{j}_{\left(La^{j}_{k},Lo^{j}_{l}\right)}$ |
| $\mathcal{U}$ | Set of users |
| $M$ | number of users |
| $UI$ | $i^{th}$ user ($i \in [1\ m]$) |
| $La_c, Lo_c$ | Represents the latitude and longitude coordinates of the user's current location, respectively. |
| $La_n, Lo_n$ | Represents the latitude and longitude coordinates of the user's new location, respectively. |
| $D((La_c, Lo_c), (La_n, Lo_n))$ | The distance that the user will move from its current location $(La_c, Lo_c)$ to its new location $(La_n, Lo_n)$. |
| $\mathcal{C}_{F_i}$ | the capacity of the ith fog node |
| $\mathcal{V}_{\mathcal{P}_i}$ | the total number of virtual CPUs in fog node $i$ |
| $\mathcal{P}^{i}$ | the number of Physical CPUs in the fog node $i$ |
| $\mathcal{C}^{i}$ | the total cores in each physical CPU of fog node $i$ |
| $v_{CPU}$ | the number of assigned virtual CPUs per VM |
| $\mathcal{L}_{F_i}$ | The load of the i$^{th}$ fog node |
| $\mathfrak{r}^{i}_{CPU}$ | the number of virtual CPUs running on fog node $i$ |
| $(\mathcal{T}_c)$ | total cost |
| $(\mathcal{M}_c)$ | migration Cost |
| $(\mathcal{C}_c)$ | computation Cost |
| $S_t$ | state-space at a time $t$ |
| $\mathcal{R}$ | the reward of the system |
| $TH_{\mathcal{L}_{F_i}}$ | the threshold of the load |

In this model, the set of $J$ fog nodes is assumed with locations given by their longitudes and latitudes in a square grid with grid cells k$^{th}$ row and $l^{th}$ column ranging from 1 to I: $\left\{\left(La^{j}_{k}, Lo^{j}_{l}\right): k, l = 1, \cdots, I\right\}$ for the $j^{th}$ fog node. It is represented in equation 1.

Each fog node is equipped with GPS to define its location in correspondence to the latitude and longitude values. Thence, the set of fog nodes is represented as

$$\mathcal{F} = \left\{F^{j}_{\left(La^{j}_{k},Lo^{j}_{l}\right)} : j = 1, \cdots, J; k, l = 1, \cdots, I\right\}. \quad (1)$$

In other words, $\mathcal{F}$ is a collection of $J$ tuples, where each tuple consists of a pair of latitude and longitude values indexed by $k\ and\ l$, where $F^{j}_{\left(La^{j}_{k},Lo^{j}_{l}\right)}$ represents a $j^{th}$ fog node with latitude and longitude positions of k and $l$, respectively, ranging from 1 to I.

For example, $F^{5}_{(6,7)}$ represents the fif$^{th}$ fog node, which has a latitude value of 6 and a longitude value of 7.

Each fog node hosts a set $Q$ virtual machines of all have the same size and configuration, denoted by equation 2.

$$\mathcal{V}\left(F^{j}_{\left(La^{j}_{k},Lo^{j}_{l}\right)}\right) = \left\{ v^{j}_{\left(La^{j}_{k},Lo^{j}_{l}\right)_{\mathfrak{q}}} : \mathfrak{q} = 1, \ldots, Q \right\}. \quad (2)$$

where $v^{j}_{\left(La^{j}_{k},Lo^{j}_{l}\right),\mathfrak{q}}$ Represents the is the $\mathfrak{q}^{th}$ VM running on the fog node $F^{j}_{\left(La^{j}_{k},Lo^{j}_{l}\right)}$ And $Q$ is the total number of the VMs running on that fog node. Hence, the above representation represents all VMs running on the specific fog node.

Each fog node has a controller which handles the details of fog nodes in a certain coverage area. Let's assume each fog node covers 60 meters only in all directions. This assumption is based on the fact that wireless coverage can vary between fog nodes, but an average cover range of 60 meters is typical for fog nodes, which have a range of around 100 meters. However, the wireless signal could be weakened as they travel through the air and would become too weak beyond a certain distance. Hence, by setting the coverage area of each fog node to 60 meters, it can be ensured that the fog nodes are close enough to communicate effectively to provide enough coverage to serve the needs of the users.

The controller in each fog node senses the environment and maintains details about the environment, including the load, users' information, and location details. There are $M \leq Q$ users. Each user may be connected to at most one of the virtual machines at any point in time, as shown in equation 3.

$$\mathcal{U} = \left\{u^{(j,k,l,\mathfrak{q}_i)}_{(La_s,Lo_r)} : \mathfrak{q}_i = 1, \cdots, Q; i = 1, \ldots, M\right\} \quad (3)$$

In this notation, U represents the set of $M$ users. $La_s, Lo_r$ Represent the latitude and longitude coordinates of the user's location, respectively. $j, k\ and\ l$ represent the fog node where the user is connected, and $j$ is the index of the fog node, $k\ and\ l$ are the indices of the virtual machine within the fog node. $\mathfrak{q}_i$ Represents the index of the virtual machine that the user is connected to, which ranges from 1 to $Q$. In other words, $\mathcal{U}$ is a set of $M$ tuples where each tuple contains information about the user's location and which virtual machine they are connected to (if any).

Each mobile user is equipped with a GPS system to define its location. It is assumed that all mobile users are connected to fog nodes using wireless connections. It is also assumed that all fog nodes are connected using wireless connections. The user will move a distance $((La_c, Lo_c), (La_n, Lo_n))$ from its current location $(La_c, Lo_c)$ to its new location $(La_n, Lo_n)$.

**Algorithm 2: Exchange the information between the fog nodes.**

1. Begin the procedure
2. For each fog node F in the system:
3. Calculate the Euclidean distance between F and every other fog node in the system
4. Store the distance in M
5. If M is less than or equal to 60 m
6. Add the distance value to F's memory
7. End If
8. End For
9. End the procedure

### 3.1 Fog Node Structure

The following defines the structure of the fog nodes:

$\mathcal{C}_{F_i}$ **The capacity of the fog node:** In our proposed scenario, the number of virtual machines as a parameter to find the capacity of the fog node is taken by ignoring other factors and processes that need to be run on the fog node.

Hence, the capacity is defined as the total number of VMs that can be run on the machine. It is calculated as the number of processors and cores available in the fog node assigned to the virtual machines. It is assumed that all VMs have the same CPU allocation.

$$\mathcal{V}_{\mathcal{P}_i} = \mathcal{P}^i \times C^i \quad (4)$$

Where $\mathcal{P}^i$ Is the number of Physical CPUs in the fog node, $i$ and $C^i$ Is the total number of cores in each physical CPU of fog node $i$. Hence, the capacity of a fog node is calculated as follows:

$$\mathcal{C}_{F_i} = \mathcal{V}_{\mathcal{P}_i} / v_{CPU} \quad (5)$$

Where $\mathcal{V}_{\mathcal{P}_i}$ represented the total number of virtual CPUs in fog node, $i$ and $v_{CPU}$ Is the total number of assigned virtual CPUs per VM.

$\mathcal{L}_{F_i}$ **The load of the fog node:** It is the number of currently running VMs. However, for simplicity, this calculation is done by ignoring the other processes running in the fog node and assuming that the only load of the fog node is the virtual machine's load.

$$\mathcal{L}_{F_i} = \mathrm{r}^i_{CPU}/v_{CPU} \quad (6)$$

Where, $\mathrm{r}^i_{\mathrm{CPU}}$ represents the number of virtual CPUs running on fog node $i$ and $v_{CPU}$ Represents the assigned virtual CPUs per VM.

Initially, when the environment is initiated, each fog node exchanges the information messages regarding its location index values ($La$ and $Lo$) with all nodes in the area. At startup, the fog nodes will acquire the values of their location positions in terms of latitude and longitude. It is assumed that all fog nodes are fixed in their locations. Then, the fog nodes will exchange information. The following section explains how the nodes exchange information.

**Algorithm 1: Calculate the location index of each fog node (Environment Initialization).**

1. Initialize ( )
2. Begin
3. For j = 1 to J
4. Initialize La and Lo location values for fog node Fj
5. Store location in each node controller
6. End For
7. End

## VM MIGRATION PROBLEM FORMULATION

The migration of virtual machines has the benefit of reducing the latency of real-time applications. However, if a wrong/late decision is taken and an unsuitable migration strategy is applied, this may result in a longer time to accomplish the real-time applications. Therefore, with the objective of finding the best strategy, Reinforcement Learning is applied to reach the optimum decision. Taking the load, the location of the fog nodes and the movement of the users into account, the problem can be formulated as follows:

**Total Cost** ($\mathcal{T}_c$)**:** The total cost consists of two parts: The Migration Cost $\mathcal{M}_c$ and the Computation cost $\mathcal{C}_c$ .

**Migration Cost** ($\mathcal{M}_c$)**:** It is the delay incurred by the network to migrate the virtual machine. This includes the transmission delay. $\mathcal{T}_{delay}$ and the signal delay $\mathcal{S}_{delay}$ . The processing delay of the VM transfer is ignored as the main concern is to calculate the delay incurred by the media of transfer and the VM size and as the processing delay is minimal due to the use of high-speed processors. It can be calculated as:

$$\mathcal{T}_{delay} = \frac{v_{\mathcal{S}}}{\mathcal{B}_w} \quad (7)$$

Where, $v_{\mathcal{S}}$ is the size of the virtual machine (in bits) and $\mathcal{B}_w$ It is the bandwidth of the network (in bits per second). The signal delay can be calculated as:

$$\mathcal{S}_{delay} = \frac{\mathcal{D}_{U_i}}{v} \quad (8)$$

Where, $\mathcal{D}_{U_i}$ Is the distance between the user's new location to the source fog node or to the destination fog node based on user movement, which is measured in meters, and $v$ is the media speed, which is in this case measured in meters/second.

Hence, the migration cost is

$$\mathcal{M}_c = \mathcal{T}_{delay} + \mathcal{S}_{delay} \quad (9)$$

**Computation Cost** ($\mathcal{C}_c$)**:** It represents the cost of the computational resources required to perform the migration process. This also has a positive relationship with the load. If the load is high, the computational cost inside a particular fog node will be high and vice versa. It is calculated as:

$$\mathcal{C}_c = \mathcal{L}_{F_i} \times P_s \quad (10)$$

where $\mathcal{L}_{F_i}$ Is the load of the fog node and $P_s$ It

is the processing speed of the node in terms of VMs/sec, which is the inverse of the processing time of each VM. Hence,

$$\mathcal{T}_c = \mathcal{M}_c + \mathcal{C}_c \quad (11)$$

In this case, the computational cost must be compared to whether the VM is not migrated and if it is migrated to a destination fog node.

The system evolves over time into a set of infinite episodes $\mathcal{T} = \{1, 2, \ldots, \infty\}$. At every episode (i.e., time t), the user $u_i$ will be connected to one fog node (say $F^j$). Let us assume that the fog node where the VM will be migrated is $F^{j'}$. The system state is defined using two parameters: The load difference state $(LD)$ between a pair of fog nodes $(F^j, F^{j'})$, and the distance state for the user $u_i$ $(LC_i)$ to the source fog node and to the destination fog node. LD (i.e., the load difference state between $F^j$ and $F^{j'}$ This can be defined as follows:

$$LD\left(F^j_{\left(La^j_k, Lo^j_l\right)}, F^{j'}_{\left(La^{j'}_{k'}, Lo^{j'}_{l'}\right)}\right) = \begin{cases} 0, & if \quad LF^j_{\left(La^j_k, Lo^j_l\right)} > LF^{j'}_{\left(La^{j'}_{k'}, Lo^{j'}_{l'}\right)}, \\ 1, & if\ LF^j_{\left(La^j_k, Lo^j_l\right)} = LF^{j'}_{\left(La^{j'}_{k'}, Lo^{j'}_{l'}\right)}, \\ 2, & if\ LF^j_{\left(La^j_k, Lo^j_l\right)} < LF^{j'}_{\left(La^{j'}_{k'}, Lo^{j'}_{l'}\right)}, \end{cases} \quad (12)$$

Further, $LC_i$ (i.e., the distance state for the user $u_i$) will be defined based on the distance between the user's new location and the fog nodes $F^j$ and $F^{j'}$. Therefore, a new parameter is introduced to represent the user's new location, denoted as $u'$, which consists of the latitude and longitude values of $(La_{s'}, Lo_{r'})$. Hence, $LC_i$ This can be expressed as follows:

$$LC_i = \begin{cases} 0, & if\ d(F^j, u') > d\left(F^{j'}, u'\right), \\ 1, & if\ d(F^j, u') = d\left(F^{j'}, u'\right), \\ 2, & if\ d(F^j, u') < d\left(F^{j'}, u'\right) \end{cases} \quad (13)$$

The overall state of the system at time $t$ can be written as:

$and S_t = \{(LD, LC_i)\}$, where $S_t$ is state space at time $t$ and,

- $LD = 0, one\ or\ 2$, are the possible states of the load difference between fog node pairs $(F^j, F^{j'})$
- $LC_i = 0, 1\ or\ 2$ are the possible states of the distance for the user $u_i$. It is defined based on the distance between the user's new location and the fog nodes. $F^j$ and $F^{j'}$.
  Hence, the possible state space could be re-written as:

$$S_t = \begin{Bmatrix} (0,0), (0,1), (0,2) \\ (1,0), (1,1), (1,2) \\ (2,0), (2,1), (2,2) \end{Bmatrix}$$

At any time $t$, the fog node controller, which in the proposed model represents the reinforcement-learning
an agent will take action to migrate or not. The action is denoted. $A_t \in \{0, 1\}$. A value of **one** means migrate, and a value of **zero** means do not migrate. Therefore, the system will generate a state-action pair. $(S_t, A_t)$.

**Reward:** This research aims to reduce the total cost of accomplishing tasks for mobile users and maximize the reward. The reward can be defined as the minimal cost of accomplishing the tasks, which is defined in this scenario as the total cost. $(\mathcal{T}_c)$. The reward is represented as follows:

$$\mathcal{R} = {}^1/_{\mathcal{T}_c} \quad (14)$$

There is an inverse relationship between the reward and the total cost. If the total cost is high, the reward is less and vice versa.

## ALGORITHM AND AGENT TRAINING

To start with, the system proposed in this paper will be referred to as ***VM_MIG*** from now on. Next, the load threshold will be defined. The load threshold is the percentage of the fog node utilization that should not be exceeded. If the load threshold is exceeded, virtual machines should be migrated. A load threshold is necessary to prevent fog nodes from becoming overloaded and negatively impacting the system's performance. If a fog node's load exceeds a certain threshold, it may begin to delay responses, resulting in degraded user experience and decreased system performance. By setting a threshold for the load on a fog node, the system can be designed to migrate virtual machines to other nodes before the load becomes too high. This can help balance the load across the fog nodes, preventing any node from becoming overloaded and avoiding performance degradation. Additionally, setting a threshold can help optimize resource utilization and reduce energy consumption by ensuring that nodes are not operating at higher than necessary loads. The values range from 0% to 100%. If the threshold is set too low, it can lead to degraded system performance and increased overhead due to VM migration. On the other hand, setting the threshold too high can cause the fog node to delay the migration, potentially affecting the experience of users running real-time applications. In this proposed scenario, the threshold of the load for any fog node is set as follows:

$$\mathrm{TH}_{\mathcal{L}_{F_i}} = \frac{1}{2}\mathcal{C}_{F_i} \quad (15)$$

The following algorithm describes how the system will learn whether to migrate or not based on the Reinforcement Learning setup:

Before looking at the algorithm, some set of terminologies are defined in table 3.

## EVALUATION RESULTS

In this section, the results of a simulation-based performance evaluation to assess the effectiveness of the proposed algorithm are presented and compared with existing algorithms. The simulation was implemented in the MATLAB R2021b environment, which includes Reinforcement Learning Applications that facilitate the simulation of reinforcement learning-based research.

### 6.1 Experiments Settings

In this evaluation, a comparison of the proposed algorithm (*VM_MIG*) with the following algorithms is done:

1. When no migration decision is made, this is done because the VM_MIG system must be tested against the systems that do not make any migration decision based on system efficiency.
2. The proposed model is compared with the study (C. Zhang & Zheng, 2019), where the authors use only the mobility factor to decide about migration. This is done because this study considers VM and task migration in an environment similar to ours, which is edge computing, by adopting RL.
3. The proposed system is also compared to the study (Basu et al., 2019), where the authors use the load factor to make migration decisions without considering a threshold. They migrate virtual machines after the fog node is already overloaded. The proposed study decides the migration based on multiple factors, including mobility and load, while considering a threshold value of the load before deciding on the migration.

**Table 3.**List of notations used in Reinforcement Learning Algorithm.

| Notation | Definition |
|---|---|
| Capacity (N) | A number |
| replay memory (M) | Replay Memory in the RL setting allows the agent to learn from past experiences instead of only learning from its most recent experiences. |
| $\mathbb{Q}$ value with $\theta$ weight | Q-value function with its parameters initialized to the value of θ. The Q-value function is updated during the learning process to better approximate the true Q-values for different states and actions. |
| $\bar{\mathbb{Q}}$ value with $\bar{\theta}$ | Target Q-value function. This is a copy of the Q-value function that is updated less frequently, typically after a certain number of iterations. |
| γ | A discounted rate is a factor used to balance the importance of immediate rewards versus future rewards. It is represented by the symbol γ (gamma) and is a value between 0 and 1. |
| learning rate $\alpha$ | It controls the rate at which the agent learns from new experiences and determines how much weight should be given to new information. |
| exploration rate $\epsilon$ | The probability that an agent will choose a random action instead of the action that it believes to be the best based on its current knowledge or policy. |

**Algorithm 3: Virtual Machine Migration in Fog Environment Using Reinforcement Learning**

1. Initialize a replay memory (M) to capacity $\mathbb{N}$
2. Initialize $\mathbb{Q}$ value with $\theta$ weight,
3. Initialize the target $\bar{\mathbb{Q}}$ value to $\bar{\theta}$
4. Initialize ☋ value randomly.
5. Initialize γ
6. Initialize learning rate $\alpha$
7. Set update steps U,
8. Set Batch size
9. Set exploration rate $\epsilon$
10. The set decay rate of $\epsilon$
11. While t ≤ ∞ :
1. Generate random number ☋ from [0,1]
2. Calculate $\mathrm{TH}_{\mathcal{L}_{F_i}}$ According to eq. **(10)**
3. If $\mathcal{L}_{F_j}$ reaches, $\mathrm{TH}_{\mathcal{L}_{F_j}}$ Then:
4. Calculate $LC_i$
5. If $d(\mathrm{F}^j, u') = \mathrm{d}(\mathrm{F}^{j'}, u')$ Then:
6. Calculate $\mathcal{L}_{F_{j'}}$ According to eq. **(3)**
7. If $\mathcal{L}_{F_j} > \mathcal{L}_{F_{j'}}$ Then:
8. Migrate the VM
9. Else:
10. Calculate the total cost of migration.
11. If the Total Cost of migration is higher
12. than the total cost of no migration, then:
13. Don't Migrate VM
14. Else:
15. Migrate VM
16. End If
17. End If
18. End If
19. End While

Table 4 shows the configuration parameters of the simulation process.

**Table 4.** The configuration parameters for the simulation

| Parameter | Value |
|---|---|
| Sample Time | 5 seconds |
| Maximum Episodes | 200, 500, 1000 |
| Maximum Episodes Length | 100 time-steps |
| α – Learning Rate | 0.01 |
| γ – Discounted Factor | 0.99 |
| Batch Size | 64 |
| ε- start | 1 |
| ε- min | 0.01 |
| ε- decay | 0.005 |
| Memory Size (experiences) | 10000 experiences |

These values have been selected based on different factors: The learning rate α is an important factor as it approximates a function to best map the inputs to outputs in Neural Networks. However, a large learning rate will make the system faster. However, this is not our target, as the proposed model is concerned with future rewards and not immediate rewards. Hence, the system is tested with a small learning

rate, so the system will provide more optimal values but will take longer, which is expected. Regarding the discounted factor γ, the value is set based on the immediate or future rewards. If γ = 0, this means that we care more about immediate and current rewards and if γ = 1, this means we care more about the future and long-term rewards. So, in this test, the discounted factor γ is set to 0.99, which means we are looking for the future reward.

### 6.2 Performance Metrics

The performance of the proposed migration algorithm is studied and compared with the performance of existing algorithms. The following performance metric is compared:

**Total Average Reward:** The goal of migration is to maximize the total or cumulative reward. The reward is represented as

$$\mathcal{R} = {}^{1}/_{\mathcal{T}_c} \qquad (16)$$

### 6.3 Results Analysis

The results obtained through the simulation measured the effectiveness of ***VM_MIG*** in terms of total average reward. The following sub-sections discuss the results.

Figure 1 shows that the reward varies between different stages of the system's learning process. The number of fog nodes used here is two, four, and sixteen.

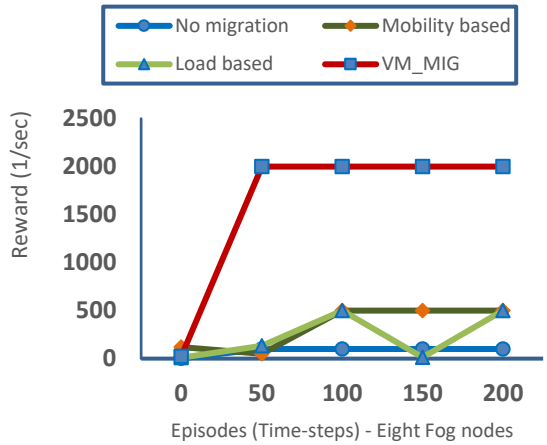



**Figure 1.** Reward per episode vs the number of fog nodes.

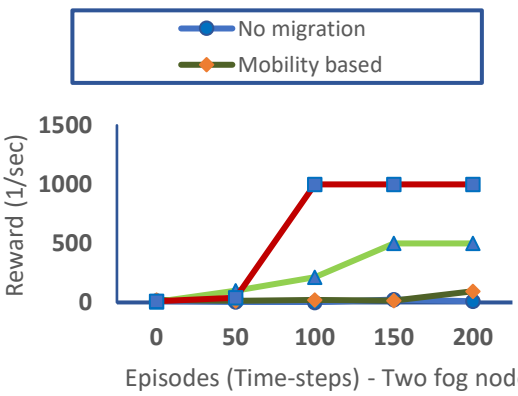



**Figure 2.** Reward per episode vs the number of fog nodes.

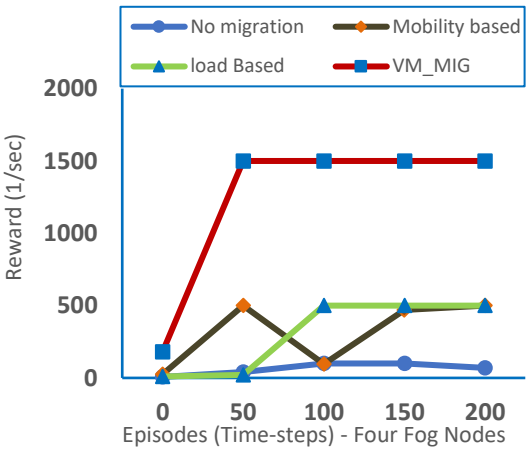



**Figure 3.** Reward per episode vs the number of fog nodes.

In this scenario, the effect of the number of fog nodes in making the decision to migrate is tested. It runs the experiment through 2,4,8, and 16 fog nodes while fixing the episodes.

Looking at Figure 2, it is realized that almost all the systems started with less reward. It is noticed that when there is no migration, the system starts with normal performance, and then it decreases from 14 at the beginning to 10 at the end of the episodes. Further, it is noticed that when only mobility is considered, the reward will slowly increase. However, by looking closely, it is noticed that all four systems have almost similar values at the beginning, and then ***VM_MIG*** recorded a sharp increase after episode 50 of the learning lifecycle. This is because, in the ***VM_MIG*** system, the immediate result is not the main concern; rather, the goal is to target future rewards and results.

It can be concluded that the "mobility-based" and the "no migration" algorithms behave similarly because mobility, in this case, does not have a significant impact on the system. Furthermore, the "load-based" algorithm demonstrated good performance compared to the "mobility-based" and the "no migration" systems, as the load keeps changing dynamically. However, as ***VM_MIG*** also considers the load but with a threshold value before deciding, it outperforms the other three systems. Figure 2 shows that the ***VM_MIG*** system outperforms the mobility-only-based system by 97% in episode 200.

Figure 3 shows that when there is no migration decision taken, the reward will be at extremely low values. However, mobility-based systems show better performance when applied to four nodes because the chance of mobility is increased, so the system can learn better and make better decisions. Mobility-based and load-based recorded nearly the same values in the final stage of learning, which is almost 500. Looking at ***VM_MIG***, it records better performance in terms of reward, starting from 180 at the beginning to 1500 at episode 200. It outperforms mobility-based and load-based systems by 66.6%.

By applying all the systems to eight fog nodes, all systems perform better by achieving lower latency as the options are larger, and the VM can

be migrated to any available fog node. However, looking closely at the figure, it is realized that ***VM_MIG*** also outperforms the other systems. In episode 50, the ***VM_MIG*** system recorded a sharp increase in reward to reach 2000, which it maintained for the remainder of the simulation. This represents a 75% improvement over the other systems.

Further, even when the number of fog nodes is increased to 16, all systems perform better by achieving lower latency as the options are larger and the VM can be migrated to any available fog node. It is realized that ***VM_MIG*** outperforms the other systems, although it recorded a sharp decrease in episode 50 as it is still in the learning phases. Later, it recorded a sharp increase after episode 50 to reach 2500 in episode 100 and continues to have the same reward value. This shows an 80% improvement rate compared to other systems. (Figure 4)

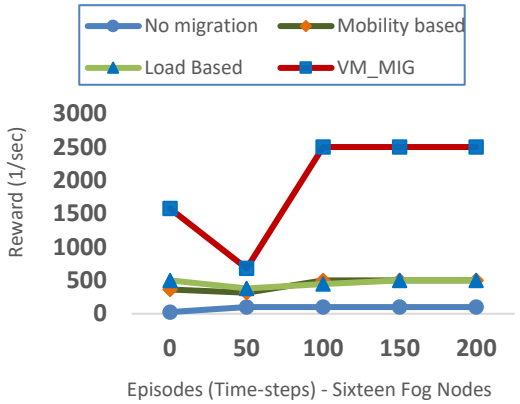



**Figure 4.** Reward per episode vs the number of fog nodes

## CONCLUSION

In this paper, a reinforcement learning-based live Virtual Machine Migration system is presented. This system considers multiple factors when deciding whether to migrate the virtual machine from one node to another, as opposed to relying on a single factor. VM_MIG considers the load of the fog nodes with a pre-defined threshold value and the mobility of the end-users. By applying RL algorithms, the VM_MIG aims to achieve low latency.

The system was evaluated against two existing systems, each of which considers only one factor in deciding about migration. One of the two systems considers only the mobility factor, and the second one considers only the load factor. VM_MIG outperforms these existing systems in terms of total average reward. The system VM_MIG outperformed the mobility-only-based system by 97% when tested with two fog nodes and by 80% when tested with sixteen fog nodes in terms of average reward. Further, the proposed system outperforms the load-based system. 50% and 75% when the environment consists of two fog nodes and sixteen fog nodes, respectively. According to these results, it is concluded that considering multiple factors enhances the overall system performance in terms of the average reward in the long term.

## LIMITATIONS AND FUTURE WORK

Although the proposed system solves the issue of late and early handover and improves the decision-making about whether or not to migrate virtual machines by combining mobility and load into a single comprehensive model to reduce disruptions to time-critical applications, it has some limitations.

1. Fixed fog node locations: The fog nodes are assumed to have fixed locations in a square grid. This might cause limited scalability in a fog environment. As the number of fog nodes increases and the network expands, the fixed grid might limit the growing demands.
2. Homogenous fog node configurations: The fog nodes are assumed to have the same size and configuration and host a fixed number of virtual machines. Homogenous fog nodes may not easily adapt to the changing demands in the dynamic environment.

Based on the above limitations, possible future work directions are:

1. Explore adaptive and dynamic placement strategies for fog nodes to optimize resource allocation. By focusing on developing more flexible and adaptive approaches. This can better adapt to any changes in the network and fog environment.
2. Investigate techniques to accommodate diverse fog node configurations and efficiently allocate resources on their capabilities, allowing adaptive resource allocation and, hence, better decisions on migration.

## CONFLICT OF INTEREST

The authors declare that there are no conflicts of interest regarding this publication.


## FUNDING

No Funding was received for this system.


## REFERENCES


Agarwal, S., Yadav, S., & Yadav, A. K. (2016). An Efficient Architecture and Algorithm for Resource Provisioning in Fog Computing. *International Journal of Information Engineering and Electronic Business*, 8, 1, 48–61.

Akintoye, S. B., & Bagula, A. (2019). Improving quality-of-service in cloud/fog computing through efficient resource allocation. *Sensors (Switzerland)*, 19, 6.

Bao, W., Yuan, D., Yang, Z., Wang, S., Li, W., Zhou, B. B., & Zomaya, A. Y. (2017). Follow Me Fog: Toward Seamless Handover Timing Schemes in a Fog Computing Environment. *IEEE Communications Magazine*, 55, 11, 72–78.

Basu, D., Wang, X., Hong, Y., Chen, H., & Bressan, S. (2019). Learn-as-you-go with Megh:

Efficient Live Migration of Virtual Machines. *IEEE Transactions on Parallel and Distributed Systems*, 30, 8, 1786–1801.

Bittencourt, L. F., Lopes, M. M., Petri, I., & Rana, O. F. (2015). Towards Virtual Machine Migration in Fog Computing. *Proceedings - 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC 2015*, 1–8.

Filiposka, S., Mishev, A., & Gilly, K. (2018). Community-based allocation and migration strategies for fog computing. *IEEE Wireless Communications and Networking Conference, WCNC, 2018-April*, 1–6.

Goncalves, D., Velasquez, K., Curado, M., Bittencourt, L., & Madeira, E. (2018). Proactive Virtual Machine Migration in Fog Environments. *Proceedings - IEEE Symposium on Computers and Communications, 2018-June*, 742–745.

Govindaraj, K., & Artemenko, A. (2018). Container Live Migration for Latency Critical Industrial Applications on Edge Computing. *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA, 2018-Sept, iii*, 83–90.

Habibi, P., Farhoudi, M., Kazemian, S., Khorsandi, S., & Leon-Garcia, A. (2020). Fog Computing: A Comprehensive Architectural Survey. *IEEE Access*, 8, 69105–69133.

Hammoudeh, R. A. (2018). A Concise Introduction to Reinforcement Learning. Journal of Reinforcement Learning 1(1), 1-11.

Han, M. (2018). Reinforcement Learning Approaches in Dynamic Environments [Doctoral dissertation, ParisTech]. HAL Archives Ouvertes. HAL Id: tel-01891805.

Machen, A., Wang, S., Leung, K. K., Ko, B. J., & Salonidis, T. (2018). Live Service Migration in Mobile Edge Clouds. *IEEE Wireless Communications*, 25, 1, 140–147.

Osanaiye, O., Chen, S., Yan, Z., Lu, R., Choo, K. K. R., & Dlodlo, M. (2017). From Cloud to Fog Computing: A Review and a Conceptual Live VM Migration Framework. *IEEE Access*, 5, 8284–8300.

Puliafito, C., Vallati, C., Mingozzi, E., Merlino, G., Longo, F., & Puliafito, A. (2019). Container migration in the fog: A performance evaluation. *Sensors (Switzerland)*, 19, 7, 1–22.

Rahbari, D., & Nickray, M. (2019). Computation Offloading and Scheduling in Edge-Fog Cloud Computing. *Journal of Electronic & Information Systems*, 1, 1, 24–34.

Rodrigues, T. G., Suto, K., Nishiyama, H., & Kato, N. (2017). Hybrid Method for Minimizing Service Delay in Edge Cloud Computing Through VM Migration and Transmission Power Control. *IEEE Transactions on Computers*, 66, 5, 810–819.

Roig, P. J., Alcaraz, S., Gilly, K., & Juiz, C. (2019). Modelling VM migration in a fog computing environment. *Elektronika Ir Elektrotechnika*, 25(5), 75–81.

Rosário, D., Schimuneck, M., Camargo, J., Nobre, J., Both, C., Rochol, J., & Gerla, M. (2018). Service migration from cloud to multi-tier fog nodes for multimedia dissemination with QoE support. *Sensors (Switzerland)*, 18, 2, 1–17.

Tang, Z., Zhou, X., Zhang, F., Jia, W., & Zhao, W. (2018). Migration Modeling and Learning Algorithms for Containers in Fog Computing. *IEEE Transactions on Services Computing*, 14, 8.

Wang, J., Hu, J., & Min, G. (2020). Online Service Migration in Edge Computing with Incomplete Information: A Deep Recurrent Actor-Critic Method. 1–12.

Wang, S., Xu, J., Zhang, N., & Liu, Y. (2018). A Survey on Service Migration in Mobile Edge Computing. *IEEE Access*, 6, 23511–23528.

Yi, S., Hao, Z., Qin, Z., & Li, Q. (2016). Fog computing: Platform and applications. *Proceedings - 3rd Workshop on Hot Topics in Web Systems and Technologies, HotWeb 2015*, 73–78.

Zhang, C., & Zheng, Z. (2019). Task migration for mobile edge computing using deep reinforcement learning. *Future Generation Computer Systems*, 96, 111–118.

Zhang, F., Liu, G., Fu, X., & Yahyapour, R. (2018). A Survey on Virtual Machine Migration: Challenges, Techniques, and Open Issues. *IEEE Communications Surveys and Tutorials*, 20, 2, 1206–1243.

Zhou, Z., Liao, H., Zhao, X., Ai, B., & Guizani, M. (2019). Reliable Task Offloading for Vehicular Fog Computing under Information Asymmetry and Information Uncertainty. *IEEE Transactions on Vehicular Technology*, 1–1.

Zhu, Q., Si, B., Yang, F., & Ma, Y. (2017). Task offloading decision in the fog computing system. *China Communications*, 14, 11, 59–68.