Migration-aware Network Services with Edge Computing

Mukhopadhyay, Atri; Iosifidis, George; Ruffini, Marco

DOI
10.1109/TNSM.2021.3139857

Publication date
2022

Document Version
Accepted author manuscript

Published in
IEEE Transactions on Network and Service Management

Citation (APA)
Mukhopadhyay, A., Iosifidis, G., & Ruffini, M. (2022). Migration-aware Network Services with Edge Computing. *IEEE Transactions on Network and Service Management*, *19*(2), 1458-1471. https://doi.org/10.1109/TNSM.2021.3139857

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3139857, IEEE Transactions on Network and Service Management

Migration-aware Network Services with Edge Computing

Atri Mukhopadhyay, George Iosifidis, and Marco Ruffini

***Abstract*—The development of Multi-access edge computing (MEC) has resulted from the requirement for supporting next generation mobile services, which need high capacity, high reliability and low latency. The key issue in such MEC architectures is to decide which edge nodes will be employed for serving the needs of the different end users. Here, we take a fresh look into this problem by focusing on the minimization of migration events rather than focusing on maximizing usage of resources. This is important because service migrations can create significant service downtime to applications that need low latency and high reliability, in addition to increasing traffic congestion in the underlying network. This paper introduces a priority induced service migration minimization (PrISMM) algorithm, which aims at minimizing service migration for both high and low priority services, through the use of Markov decision process, learning automata and combinatorial optimization. We carry out extensive simulations and produce results showing its effectiveness in reducing the mean service downtime of lower priority services and the mean admission time of the higher priority services.**

***Index Terms*—generalized assignment problem, learning automata, markov decision process, multi-access edge computing, service migration.**

I. Introduction

In the past few years, the emergence of revolutionary applications has resulted in an explosive increase in the usage of hand-held mobile devices like smartphones and tablets. On the other hand, the unprecedented deployment of resource constrained nodes in the form of Internet-of-Things (IoT), have been made in order to penetrate fields like health, transportation, industry, smart home, smart city, agriculture and education [1]. Both of these scenarios involve enormous volumes of raw data transfer with stringent quality of service (QoS) requirements [2], [3].

Unfortunately, both hand-held and IoT devices have limited processing, memory and energy resources. Therefore, the idea of multi-access edge computing (MEC) servers situated close to the end-users was materialised in order to process computations and store cached content on behalf of mobile devices [4]–[6]. MEC allows a user to offload processing tasks to a computing facility situated at the edge of the network [7]. The proximity between the user and the server helps in lowering the delay of communication. Moreover, keeping the edge servers close to the user also avoids consumption of networking resources for transmitting the offloaded tasks over the network [8]. As a result, MEC alleviates network congestion. Note that MEC is an improvement over cloud computing. Cloud computing could provide high performance in converged fixed/mobile/computing architectures but at the cost of increased latency as it hosted the server far from the user [9]. In MEC, a task is forwarded to the remote cloud server only if the resources available in the MEC server is not sufficient to process the task. Fig. 1 illustrates a sample MEC architecture, where the MEC Server is connected to a macro base-station (BS).

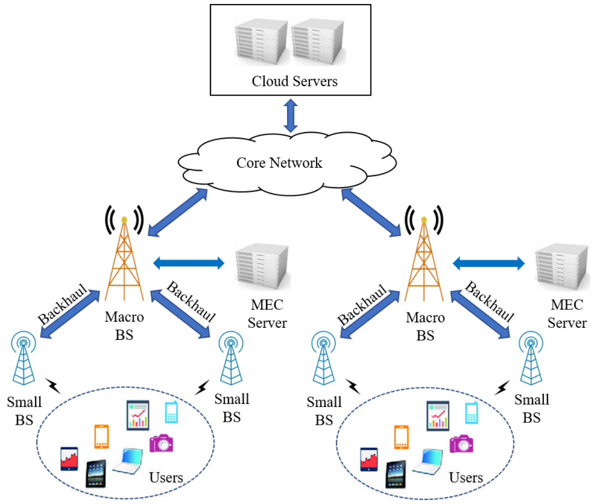


Fig. 1: MEC Architecture with localized service area.

A. Motivation

One of the primary challenges towards implementing an MEC enabled network is user mobility. Since the MEC paradigm revolves around the idea of maintaining proximity between the MEC computing resource and the mobile user, providing a single MEC server is not sufficient when the users are mobile. As a result, multiple MEC servers are placed at different locations. Therefore, when a mobile user moves from the vicinity of one MEC server to another, the user's service instance is *"handed-over"* to the second MEC server. This hand-over of the MEC service instance from one MEC server to another is termed as service migration [10], [11].

In addition to mobility, service migration may also be triggered when users with different access priority are served by the same set of MEC servers. For example, if a MEC server is already operating at full capacity when a high priority user requests a service, then a lower priority user already in service will be migrated to another MEC server in order to accommodate the high priority user.

Atri Mukhopadhyay and Marco Ruffini are with the CONNECT Center, School of Computer Science and Statistics, Trinity College Dublin, Dublin, Ireland. E-mail: amukhopa@tcd.ie and marco.ruffini@tcd.ie.

George Iosifidis is with Delft University of Technology, Delft, Netherlands. E-mail: g.iosifidis@tudelft.nl.

Unfortunately, service migration is an expensive process. Firstly, during service migration, the service must be temporarily stopped, which results in sudden increase in latency or even temporary suspension of the service [11]. A temporary downtime heavily degrades the performance of reliable applications. Moreover, when the service instance is migrated to the target MEC server, the network links are swamped with huge volumes of information, thereby hindering other transmissions over the network. Hence, the number of service migrations should be minimized for efficient operation of a network of MEC servers. However, minimizing the number of service migrations of mobile users is a complex task because of the randomness in user movements. Incorporating learning based techniques helps in proactively predicting user mobility. Nevertheless, a centralized learning approach might have a long convergence time because of the large state-space. Hence, distributed learning is an ideal approach in this scenario.

B. Contributions

The focal point of this study is to devise an efficient resource allocation algorithm such that the frequency and cost of service migrations can be minimized given an expected traffic arrival pattern.

To that end, we propose a priority induced service migration minimization (PrISMM) algorithm where the service migration of both the higher and lower priority services can be minimized. In our study, the higher priority service users are mobile and therefore, need to migrate when they wander too far from their serving MEC server. Unfortunately, the migration events of the higher priority services might trigger migration of the lower priority services as well; even when the lower priority service users are static.

Therefore, for the evaluation of PrISMM, we consider two kinds of users; the high priority mobile users and low priority static users. Such a combination of users/services allows us to gauge the efficiency of PrISMM as a mobile user with high priority services requires careful resource provisioning. On the other hand, a low priority static user provides us with comparative relaxation to design the system optimally. Further, the scenario also encompasses two very important modern day use cases; the *mobile ambulance* (MA) and the static *opera house* (OH) users. These two use cases have different characteristics. The MA users need to run critical life saving services, whose functional chains run partly in the ambulance and partly at MEC nodes. On the other hand, the OH service is based on the support for multi-media services running in an Opera House, where, for example, users might avail instant and synchronised audio translation, subtitle or other experience enhancing applications. We consider that both user types have real-time service requirements and therefore, minimizing service delay is essential.

Please note that, one might consider a dedicated system for the critical MA users in order to guarantee resource availability to the MA users. However, as in any operational network system, the MA users' service load will exhibit temporal variation. Therefore, a dedicated system exclusively designed for the MA users will have low utilisation. Hence, in order to increase system utilisation, the system resources might be allotted to multiple classes of users subject to the condition that the service quality of the MA users is not hampered. In such a scenario, slicing can be employed. However, even with slicing, the system eventually has to distribute resources among slices. Thus, the trade-off of service guarantee and system utilisation is a major concern that needs to be solved. Hence, we carefully design the PrISMM algorithm by keeping this delicate balance in mind.

The contributions of the paper are summarised as follows:

- We have employed a next generation passive optical network (PON) backhaul to provide connectivity among the MEC servers. Thereafter, we have included the impact of PON imposed delay in the allocation criteria of PrISMM.
- We have proposed a *Markov decision process* (MDP) [12] based resource allocation procedure for the high priority MA users with the target of minimizing service migration throughout the journey of the ambulance from its point of origination to the destination hospital. We utilise the already available knowledge about the destination hospital of the MA for designing the MDP parameters.
- We have engineered a *learning automata* (LA) [13] module in each of the MEC servers that identifies the frequency and volume of resource requests coming towards it from the high priority MA users. After identifying the resource requirements for the MA users, the LA module optimally reserves sufficient resources for the MA users.
- We have formulated a *generalized assignment program* (GAP) [14] based allocation methodology for the low priority OH users in the available server resources that are left after proactively reserving enough resources for the incoming MA users by the LA module.

Thus, PrISMM avoids unnecessary migration of the lower priority services. It may be noted that reserving resources for high priority services reduces the utilisation of the system. However, the LA based determination of reservation strives to enhance the utilisation of the system by optimally reserving the resources. Finally, in order to optimize service migration time, we assume that the system utilises the three-layer framework augmented service migration flow [10].

The remaining part of the paper is organized as follows. Section II presents the literature survey. In Section III, the system model is discussed. Section IV elaborates the proposed priority induced service migration minimization algorithm. We provide the simulation setup and the results in Section V, followed by the conclusion.

II. Related Work

In this section, we briefly report on the existing literature on service migration in MEC. The authors of [10] provide an excellent survey on service migration. The main target of service migration research, according to the majority of the literature, focuses on the balance between the cost of service migration and the QoS enhancement achieved after carrying out the migration procedure. Such a problem can be solved by formulating a Markov decision process (MDP) [15]–[17].

In [15] and [16], the movement of the user has been modeled as a one-dimensional MDP where the states of the MDP corresponds to the distance of the agents from the serving MEC servers. The available set of actions in a given state are to either continue with the same MEC server or else migrate to one that can offer better performance. However, modeling the mobility of the user as a one-dimensional MDP requires making a large amount of simplifying abstractions, making the model not useful in practical situations.

The problem associated with abstraction in one-dimensional MDP was addressed by formulation of the user movement as two-dimensional MDP [18], [19]. However, two-dimensional MDP suffers from the state explosion problem. This was addressed through state aggregation by the authors in [20] and [21]. Unfortunately, a two-dimensional MDP is also computationally more complex compared to one-dimensional MDP. However, the authors of [17] provide an analytical model for one dimensional user mobility and a close approximation for two dimensional user mobility.

Another approach towards service migration is based on time windowing [22], where the objective is to minimize the average cost over a look-ahead time window. Unfortunately, the predictive look-ahead time-window based approach is prone to prediction errors if the look-ahead time-window is large. Conversely, a very small look-ahead time-window makes the approach ineffective. Therefore, the time-window approach is sensitive to the determination of the window-size, which in turn depends on several other factors. Thus, the time-window based approach can lead to low accuracy. "Follow me edge" is yet another idea that can initiate service migrations [6], [23], [24]. It employs a MDP based process for performing service migrations [15]. On the other hand, we find a Lyapunov optimization based service migration proposal in [24]. The authors of [25] propose a learning and Lyapunov optimization based service migration algorithm with the target of developing an energy-aware service migration algorithm. Similarly, [26], discusses a multi-attribute decision making algorithm that aids mobility based service migration.

Our work differentiates from the literature, where service migration is caused mostly by user mobility. We target to minimize service migration induced by both user mobility and the arrival of higher priority services. The service migration of lower priority services induced by the arrival of higher priority services can only be minimized with careful and proactive reservation of resources for the yet to be arriving higher priority services. To the best of our knowledge, the literature deals with mobility based migration of a single type of service and does not delve into a situation that incorporates two services of varying priority. Hence, a comprehensive protocol for performing service migration in a situation where multiple services with varying priority co-exists is absent in the literature. We believe that our PrISMM proposal can fill these gaps.

III. System Model

In this section, we describe the system model which forms the premise for our proposal.

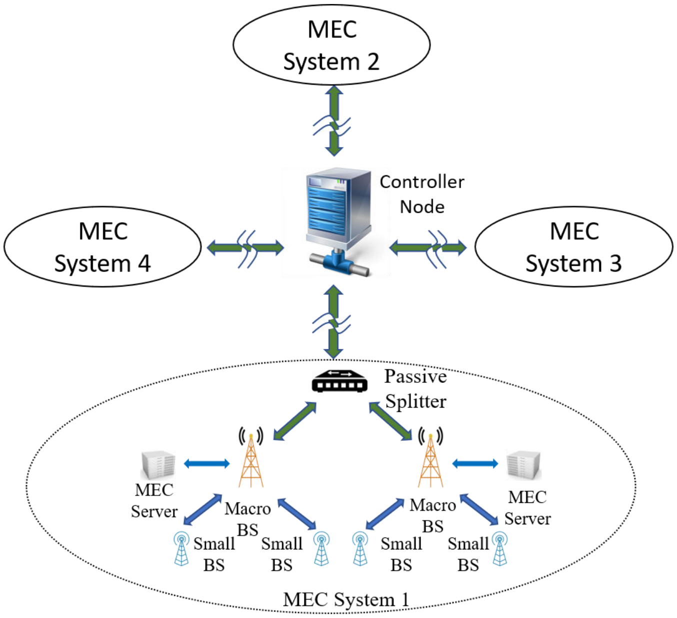


Fig. 2: System Model showing multiple MEC systems connected to a controller node with the help of a PON

A. Backhaul Network

The MEC servers are placed in the macro BSs. We consider multiple passive optical networks (PONs) [27] emanating from a controller node for providing connection to a city-wide network of MEC servers, illustrated in Fig. 2. A group of MEC servers form a MEC system. In Fig. 2, we have shown four MEC systems. We assume that a controller node is situated at the center of the city. To be more specific, we use the 10-Gigabit-capable symmetric passive optical network (XGS-PON) version of PON [28]. XGS-PON provides a data rate of 10 Gbps (with a nominal line rate of 9.95328 Gbps) in both downlink and uplink. The controller node is assumed to be co-located with the optical line terminal (OLT) of the PON. On the other hand, the MEC servers are directly connected with the optical network units (ONUs) of the PON. The controller node is responsible for providing allocation information to all the MEC servers. It also supervises the service migrations. The MEC servers are interconnected to each other via the controller node.

It is worth mentioning that a single controller node might become a bottle-neck. Hence, a second-tier controller node might be installed in each of the MEC systems. The second-tier controller would be responsible for the allocations in its respective MEC system, whereas inter-MEC system allocations will be taken care by the central controller. Unfortunately, such a design might reduce the optimality of the solution as each second-tier controller will have the information about its MEC system only. Further, as we assume that the central controller node is co-located with the OLT, the controller node has very high operational capacity. Therefore, for the sake of optimality, we go ahead with a single central controller node controlling all the MEC server allocation in a city.

We opt for PONs as they provide a low cost architecture that can supply ubiquitous connectivity while enabling resilient operations [29], [30]. Further, a PON can serve residential broadband as well as mobile BSs at the same time [31]. Recently, for example, PON virtualisation techniques have been proposed to support multi-service and multi-tenant operations [32]. Such advantages make PONs an economically sustainable choice for building the backhaul network for the MEC servers. As per normal practice, we have connected 16 ONUs per PON [27] and the ONUs support both residential broadband and mobile BSs. The controller node has multiple line cards. Every single line card can serve as a OLT.

However, the background traffic present in the PON significantly affects the placement and network latencies of the requested services. The uplink protocol of the PON and buffer capacities of the ONUs also influence the operational delays. We need to consider these factors while allocating the requesting services in the MEC servers.

B. User Traffic Model and User Location

We introduce the MA users into the system with an average arrival rate of λ_{MA} users/second and an average holding time of H_{MA} seconds. Similarly for the OH users, we assume an average arrival rate of λ_{OH} users/second and an average call holding time of H_{OH} seconds[1]. Since, the OH users are associated with an on-going Opera show, we assume OH users arrive and depart in groups. Therefore, for determining the group size, we use a Poisson mean of Λ_{OH}.

The locations of both the MA and the OH users are chosen randomly in the map by using a uniform distribution. The destination hospital of an incoming MA user is chosen uniformly from the set of available hospitals. The chosen destination hospital sets the path taken by the ambulance over the map. We assume that a straight path is taken from the starting point to the destination hospital and the time of journey is specified by the average holding time of the ambulance.

C. Servers

We presume that the servers have equal processing capabilities in all their processing units, i.e., no service unit is preferred over another service unit for their processing capacity. However, we assume the OH user application prefers the use of servers in its vicinity and such preference decreases linearly with the increase in communication delay between the OH user and the MEC server.

D. Transmission Latencies

We derive the average delay suffered by an incoming service from the delay vs load curve of a XGS-PON (see Fig. 3). When the new service request arrives, we identify the average background traffic load on the PON and based on that, we determine the average traffic transmission delay in the PON uplink. Thereafter, we calculate the end-to-end delays after considering the connections from our system topology. However, if the serving MEC server is directly connected to the connecting macro BS, then the service connectivity does not traverse through the PON and hence, the PON load does not influence the delay for this kind of connections. On the other hand, if the serving MEC server is connected to another macro BS, then the connection path must traverse through the controller node (which houses the OLTs of the PONs). As a result, the connection suffers from the delay induced by the background traffic on the PONs.

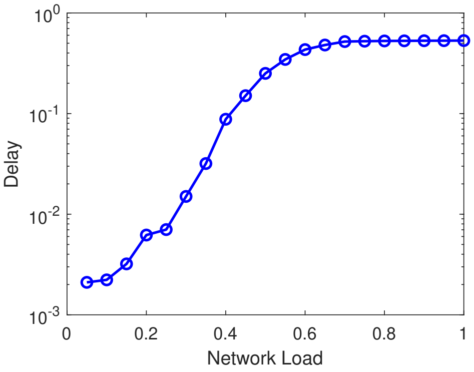


Fig. 3: GPON Delay

- Transmission latency to server $s \in S$ (d_{ts})
- Transmission latency to server $s \in S$ at journey start (d_{ts}^{start})
- Transmission latency to server $s \in S$ at journey end (d_{ts}^{end})

Please note that we have used the average delay for the end-to-end delay calculations as the arriving service is expected to stay in the system for a substantially long interval of time (tens of seconds to minutes) and therefore, the instantaneous delay values may become outdated after a short interval (a few seconds).

E. Service Migration Messages

We adopt the message exchange procedure for service migration from [33]. Service migration finishes with the actual exchange of the service instance from the serving MEC server to the destination MEC server. However, prior to that, control messages like decision request, decision response, measurement request, measurement response, service hand over command are exchanged [33].

Further, in our model, the MEC servers are connected to the controller node via a PON. Therefore, the MEC servers are essentially ONUs and the controller node is an OLT. Therefore, the PON control messages like GRANT and REPORT are also exchanged between the MEC servers and the controller node while performing service migration. These messages are used for dynamic bandwidth allocation in the PON uplink [27].

IV. Priority Induced Service Migration Minimization

In this section, we describe our proposal, the PrISMM algorithm. For the MA application, we assume the final destinations of the users to be the hospitals. So, the destinations of the MA users are precisely known at the time of generation of their service requests. Hence, it becomes apparent that if

[1] In both the cases, the arrival rate is derived from Poisson distribution and the holding time is derived from the exponential distribution.

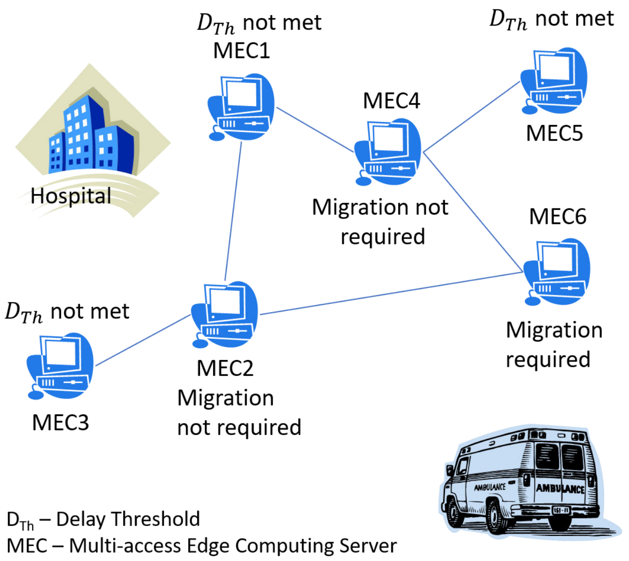


Fig. 4: A Random MA User MEC Scenario in an Arbitrary Network Topology where the MA user is allocated to the optimal MEC server situated in the MA user's path while meeting the delay requirements of the services.

we try to serve the MA users by the nearest situated MEC server, we will finally be migrating to the MEC servers situated close to the hospitals, when the MA users near their journey completion. For example, in Fig. 4, the final serving MEC server for the ambulance will be MEC server 1, near the hospital. Hence, if we initially allocate the ambulance service to MEC server 6, there can be a mobility based migration as the ambulance moves towards the hospital. On the other hand, a MEC server resource near the destination hospital may be too far from the ambulance's current location for meeting the MA user's delay requirements; thus making it unsuitable (MEC server 1 in Fig. 4). Therefore, the best candidate MEC server will be a server which is located somewhere in the route of the ambulance so that mobility based service migration can be minimized (and if possible eliminated) (MEC server 4 in Fig. 4). The allocation must consider the network link and traffic dynamics in order to be optimal.

Therefore, once we identify the candidate MEC servers that are suitable for executing the service requests from the MA users, we can proceed with the allocation of the lower priority OH users service requests in such a way that enough MEC server resources are reserved for the MA users. Thus, the problem of allocating the OH users is equivalent to a GAP, where we want to allocate the service requests from OH users to the MEC servers with varying resource availability. However, we need to keep in mind that we are dealing with variable service request arrivals from the MA users, both in terms of arriving instant and origination position. Hence, the number of server units that need to be reserved for the MA users is dynamic. Hence, we employ LA to determine the precise quantity of server resource units that need to be reserved for the MA users depending upon the load imposed on the system by the MA user service requests. We proceed to elaborate on the main components of PrISMM in the following sub-sections. Finally, we summarize PrISMM in Section IV-D.

A. *Markov Decision Process based Allocation of the MA users*

Allocating the MA users is of primary importance as it serves the life critical applications. Therefore, utmost care is taken to allocate the MA services in such a way that the network provides optimal service to the MA users and at the same time provides acceptable services to the other users. Therefore, we need an optimization technique to come up with the allocation of the MA users. However, the network introduces variable delays due to background traffic and therefore a stochastic optimization technique is required for the purpose. MDP is a stochastic optimization technique that perfectly suits our purpose [12].

Hence, in this section, we explain the MDP based approach that is being employed to allocate resources to the MA users. When an MA user requests a service, we already have the knowledge of its destination as it will be moving towards a specific hospital as illustrated in Fig. 4. Once the start and destination of the MA user is known, an MDP can be designed after looking into the current link and MEC server occupancies throughout the network. Please note that a separate MDP is designed exclusively for a MA user, which means allocation of one MA user is independent of another MA user. Since our primary target is to minimize the occurrence of service migrations, we design the rewards for the MDP by including the service migration penalty. Thus, we can choose the MEC servers that are located in places that may reduce the possibility of mobility-based service migration of the MA users. This approach indirectly reduces the priority induced service migration of the lower priority services as relocation of MA users are carried out less frequently.

In the example network, depicted in Fig. 4, the MDP will choose MEC server 2 or MEC server 4 over the other MEC servers as the other MEC servers will either fail to meet the delay requirements or will not be able to support the ambulance throughout its journey. We next describe the different components of the MDP. Fig. 5 illustrates an example MDP with two states (servers in the MEC system). Once, a MA user arrives into the system, the MDP is executed to identify the optimum MEC server that can host the MA function. In case that no suitable MEC server is found, the termination state (s_t) is reached and the MA user exits the system. Whenever a radio handover between the BSs takes place, the MDP moves to a new time-step. For greater precision, MDP can be re-evaluated when a radio hand over is encountered.

1) Inputs: We list the known inputs along with the symbol used to define them while designing the MDP in Table I.

2) States: In our MDP design, we map the servers to the states. For example, MEC server 0 is mapped to state 0, MEC server 1 is mapped to state 1 and so on. Thereafter, since each MA user has a distinct MDP, we define the current serving server of the MA user as the current state of the MDP. For example, if server $s \in S$ is currently serving the MA user, the process corresponding to MA user i is said to be in State s. We define one initialisation (s_o) state and a termination

TABLE I: Summary of Notations used in MDP

Symbol	Input Description
D_{Th}	Delay Threshold for service
S	Set of states
d_{ts}	Transmission latency to server $s \in S$
d_{ts}^{start}	Transmission latency to server $s \in S$ at journey start
d_{ts}^{end}	Transmission latency to server $s \in S$ at journey end
d_{ps}	Processing latency in server $s \in S$
$S_{cap,s}$	Capacity of Server $(s \in S)$
$S_{cap,s}^{rem}$	Remaining Capacity in Server $(s \in S)$
M_c	Migration Cost for service
a_s	The action taken with the intention of reaching state $s \in S$
$T(s, a_{s'}, s')$	Transition probability of reaching state $s' \in S$ from state $s \in S$ by taking action $a_{s'}$
$P(s'\|s, a_{s'})$	Same as $T(s, a_{s'}, s')$
$R(s, a_{s'}, s')$	Reward obtained on reaching State $s' \in S$ from state $s \in S$ by taking action $a_{s'}$

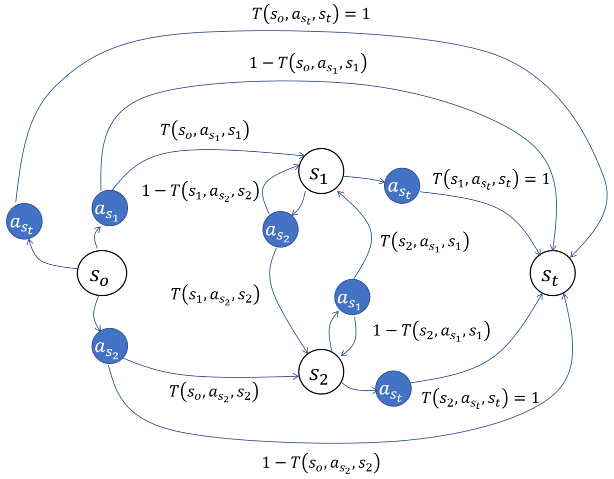

$T(s_o, a_{s_t}, s_t) = 1$
$1 - T(s_o, a_{s_1}, s_1)$
$T(s_o, a_{s_1}, s_1)$
$1 - T(s_1, a_{s_2}, s_2)$
s_1
a_{s_t}
$T(s_1, a_{s_t}, s_t) = 1$
a_{s_t}
a_{s_1}
a_{s_2}
$T(s_2, a_{s_1}, s_1)$
s_o
s_t
$T(s_1, a_{s_2}, s_2)$
a_{s_1}
$1 - T(s_2, a_{s_1}, s_1)$
a_{s_2}
s_2
a_{s_t}
$T(s_o, a_{s_2}, s_2)$
$T(s_2, a_{s_t}, s_t) = 1$
$1 - T(s_o, a_{s_2}, s_2)$

Fig. 5: An Example MDP with two States (Servers).

state (s_t) to complete the MDP. The process is in initialisation state when the MA user initially requests for service from the MEC system. Similarly, when the MA user leaves the system, the process corresponding to that specific user reaches the termination state and the MDP terminates. The termination state may be reached for the following reasons:

1) The MA user cannot be accommodated in the system. In this case, the process moves directly from the initialisation state (s_o) to the termination state (s_t).
2) The migrated MA user may not be accommodated and the original servicing MEC server can no longer satisfy the delay criteria.
3) The MA user has reached its destination.

3) Actions: The actions define the decisions that the system must take w.r.t a MA user whenever a decision taking instant (i.e., a radio handover) is encountered. The actions that can be taken are defined as follows:

1) *From the initialisation state* – The possible decisions are allocation of the MA user to a specific MEC server $(s \in S)$ or dropping the user. The allocation of the user to a certain server (s) corresponds to the process reaching a state $(s \in S)$ and the action that is responsible for the allocation is denoted as a_s. On the other hand, if the system decides to drop the user, it takes the action a_{s_t}. Thus, reaching the termination state (s_t) means that the MA user was denied service by the system.
2) *From the termination state* – The process reaches its end and is removed.
3) *From any other state* $(s \in S)$ – The possible actions involve performing a service migration or not. If a service migration is performed from server $(s \in S)$ to server $(s' \in S)$ by taking the action $a_{s'}$, then the process moves to state $(s' \in S)$ from state $(s \in S)$. However, if service migration is not performed then the MDP remains in the current state (s). Another instance may occur when the MA user cannot be migrated to any server (s') or the user reaches the destination. In such a situation, the process moves to the termination state.

In Fig.5, we have illustrated the how MDP moves from one state to another as and when the appropriate actions are taken. For example, the MDP proceeds to move from state s_1 to state s_2 when the action a_{s_2} is taken.

4) Rewards: The rewards (and/or penalties) are always dependent upon the destination state $(s' \in S)$. In such a way, it also corresponds to the action as a certain action $(a_{s'} \in A_s)$ at state $(s \in S)$ is mapped to a certain destination state $(s' \in S)$. The three different cases of reward calculation are:

1) *The delay requirement of a certain MA user can be satisfied by the MEC server $(s' \in S)$ both at the MA user's current and final position* - Only the costs pertaining to network delay and server capacity is considered [see (1)]. The delay metric is evaluated in order to minimize the data transmission delay. On the other hand the service capacity metric employs load balancing as the lightly loaded MEC servers are given more weightage. Since, the user can be served by the MEC server s' throughout the journey, migration cost is not considered for MEC server s'. Both these metrics fall in the interval $[0, 1]$. Therefore their sum lies in the interval $[0, 2]$.
2) *The MEC server $(s' \in S)$ can satisfy the delay requirement of the MA user at the MA user's current location but not at the final destination* - In addition to the metrics given in the previous case, the cost incurred due to service migration is also included in the reward calculation. Here, we include the migration cost in the reward because if the user is allocated to MEC server s', a service migration would become inevitable at some point of the journey. We have included a fixed Migration cost $(M_c = 1)$ as all the MEC servers are connected via the same OLT/controller node. Therefore, the cost of migration remains same irrespective of the current and destination MEC server. Since, the other two metrics normalised to lie within a fixed interval $[0, 1]$, setting a fixed value of $M_c = 1$ gives an equal weightage to the migration cost as well.
3) *The MEC server $(s' \in S)$ cannot meet the delay requirement of the MA user at the MA user's current location* - In this case, the given MEC server s' is not a suitable choice for allocating an user. Therefore, a large penalty is introduced so that the chances of allocation to that MEC server can be minimized. Please

note that we have used -10 in (1); however, any large value for indicating a large penalty, which is larger than migration cost, will be sufficient for the algorithm to operate efficiently.

We present the rewards mathematically in (1). Note that all the values are normalized.

$$R(s, a_{s'}, s') = \begin{cases} \left(1 - \frac{d_{ts'}^{start}}{D_{Th}}\right) + \frac{S_{cap,s}^{rem}}{S_{cap,s}}, & \text{if } d_{ts'}^{start} < D_{Th} \\ & \& \; d_{ts'}^{end} < D_{Th} \\ \left(1 - \frac{d_{ts'}^{start}}{D_{Th}}\right) + \frac{S_{cap,s}^{rem}}{S_{cap,s}} - M_c, & \text{if } d_{ts'}^{start} < D_{Th} \\ & \& \; d_{ts'}^{end} > D_{Th} \\ -10, & \text{otherwise} \end{cases} \tag{1}$$

It is worth mentioning that another alternative approach for minimizing overall delay could be reducing the processing delay by increasing the service capacity of the server in question. However, even though this can be a temporary solution, as the MA users moves further and further away from the current server, the routing delay increases and eventually, the service migration becomes essential. Further, a service may not allow multiple threads and as a result, parallel processing might not be possible. In such a situation, increasing the service capacity is of no effect. Hence, we design the rewards in such a way that network delay is considered while assuming that the service capacity is fixed. However, we also strive to perform load balancing and therefore, include the remaining service capacity term in the reward calculations.

5) Transition Probability: The transition probability $T(s, a_{s'}, s')$ [also written as $P(s'|s, a_{s'})$] defines the chances of reaching state $s' \in S$ from state $s \in S$ after taking action $a_{s'} \in A_s$. The transition probabilities are modelled according to (2) and a pictorial representation of the state transitions along with the transition probabilities can be seen in Fig. 5, which models a two MEC server system.

$$T(s, a_s, s) = 1, s \in S \tag{2a}$$

$$T(s, a_{s'}, s') = p, s \notin \{s_t\}, s' \notin \{s_o, s_t\} \tag{2b}$$

$$T(s, a_{s'}, s) = (1 - p), s \notin \{s_o\}, s' \notin \{s_o, s_t\} \tag{2c}$$

$$T(s, a_{s'}, s_t) = (1 - p), s \in \{s_o\}, s' \notin \{s_o, s_t\} \tag{2d}$$

$$T(s, a_{s'}, s') = 1, s \in S, s' \in \{s_t\} \tag{2e}$$

The rules have been designed such that an action taken with the intention of self transition has a probability equal to one ($p = 1$) for all states as shown in (2a). Equation (2b) shows that there is a probability p that a transition from state s to state s' may be successful. Otherwise, the transition does not take place with probability $(1-p)$ and the agent remains in the same state s as illustrated in (2c). However, if the transition to state s' fails from the initialisation state (s_o), the MDP goes to the termination state (s_t) as depicted by (2d). Finally, action taken with the desire of going to the termination state is always successful as per (2e). Thus, $T(s, a_{s'}, s')$ models the physical phenomenon of uncertainty in service migration.

TABLE II: Summary of Notations used in Learning Automata

Symbol	Input Description
C_{act}	Actual capacity of the individual servers
C_j	Virtual capacity of server j
A_s	Set of available actions
$p_{s,a}(t)$	Probability of choosing action a while in state s at time instant t
β	Variable to capture outcome of an action
R	Reward
P	Penalty

6) Solving the MDP – Value Iteration: The optimum solution of the MDP is evaluated using the value iteration approach. We employ the Bellman Equation shown in (3) for evaluating the optimum policy π [34].

$$V_{i+1}(s) = \max_{a} \sum_{s'} T(s, a, s')[R(s, a, s') + V_i(s')] \tag{3}$$

where, $s \in S$ is the current state (MEC), $s' \in S$ is the target state, $a \in A_s$ is the action that is to be chosen when at state s, $V_i(s)$ is the value obtained for state $s \in S$ at iteration i, $T(s, a, s')$ is the transition probability from state $s \in S$ to state $s' \in S$ by taking action a, and $R(s, a, s')$ denotes the reward obtained by moving from state $s \in S$ to state $s' \in S$ by taking action a. During initialisation of the algorithm, the values are set to zero [$V_0(s) = 0, \forall \; s \in S$]. *Here "Iteration" denotes the steps of optimum policy calculation with the available information.*

B. Learning Automata (LA) based Virtual Capacity Determination

In this sub-section, we explain the learning algorithm employed to reserve service units for the MA users. The actions taken by the learning algorithm is equated to the *"virtual capacities"* (C_j) for MEC server j that will be described in Section IV-C. In our scenario, the arrivals rates of both the MA and the OH users are derived from the Poisson distribution as we have described in Section III-B. Similarly, the holding times of both the MA and the OH users are derived from the exponential distribution. Therefore, a stochastic learning technique is desirable in such a situation. LA, which learns the optimal action through repeated interactions with its environment, is a perfect candidate for our algorithm design [13], [35]. Further, LA being a distributed learning algorithm, it can be independently executed in each MEC server. Thus, the work-load on the central node can also be reduced in such a manner.

The objective of LA is to find the optimum action that needs to be taken when the learning entity (MEC server in our case) is in a particular state. We have defined a single state because the current server occupancy does not alter the requirement of the number of server units that need to be reserved for the MA services. The algorithm is initiated by giving equal probabilities for choosing all the available actions in a particular state as shown in (4).

$$p_{s,a} = \frac{1}{|A_s|}, \forall a \in A_s, s \in S \tag{4}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3139857, IEEE Transactions on Network and Service Management

where, $p_{s,a}$ is the probability of choosing action a while being in state s. $|A_s|$ is the number of available actions in the state s.

During the learning procedure, the learning agent observes the outcome of the chosen action and updates $p_{s,a}$ accordingly. If the outcome (β) of the action (a) chosen at time instant (t) is beneficial (i.e., $\beta = 0$) to the learning agent, the probability of choosing action (a) at time instant (t+1) is increased, while the probabilities of choosing all other actions are reduced (since, we want to repeat this beneficial action in the future). If the outcome is detrimental (i.e., $\beta = 1$), then the probability of choosing action (a) is reduced while the probabilities of choosing all other actions are enhanced (since, we want to avoid this detrimental action in the future). The probability updates are summarized in (5).

$$\begin{aligned}
&\text{If } \beta = 0: \\
&\qquad p_{s,a}(t+1) = p_{s,a}(t) + R[1 - p_{s,a}(t)], \\
&\qquad p_{s,j}(t+1) = (1 - R)p_{s,j}(t), \forall j \neq a \\
&\text{If } \beta = 1: \\
&\qquad p_{s,a}(t+1) = (1 - P)p_{s,a}(t), \\
&\qquad p_{s,j}(t+1) = \frac{P}{1 - |A_s|} + (1 - P)p_{s,j}(t), \forall j \neq a
\end{aligned} \tag{5}$$

where, $p_{s,a}(t)$ is the probability of choosing action (a) at time instant (t) while being in state (s), R is the reward and P is the penalty. Note that the reward mentioned for the MDP in Section IV-A is *not related* to the reward mentioned in this section.

The set of available actions is given by the set $\{0, 1, 2, ..., C_{act}\}$, where C_{act} is the actual service capacity of the MEC server. The chosen action is the *"virtual capacity"* value that is fed to the GAP module described in Section IV-C.

1) Calculation of Rewards: In PrISMM, the chosen action (a) is considered beneficial if all the services that are requesting server resources are successfully allocated (both MA and OH services). Further, PrISMM also strives for the maximum utilisation of resources. Therefore, whenever a certain action (a), i.e., the chosen virtual capacity, provides beneficial results, the reward R is made equal to the virtual capacity chosen. As an example, if the virtual capacity (action) was chosen to be 20 and all the services were allocated, then the reward is also set to be 20. Setting such a reward criteria makes PrISMM inclined towards choosing the highest virtual capacity values that provide successful allocation. However, as we are dealing with probability values, for the reward to be usable in (5), we need to normalize the value of R. The minimum and maximum values that R can achieve are 0 and C_{act} respectively. Therefore, we normalize R by (6).

$$R_{norm} = \frac{R - R_{min}}{R_{max} - R_{min}} \tag{6}$$

where, R_{norm} is the normalized value of R, R_{min} and R_{max} are the minimum and maximum values that R can attain respectively.

2) Calculation of Penalties: Similar to the method used for calculation of rewards for favoring higher virtual capacity values, PrISMM generates a penalty P when the chosen virtual capacity forces dropping of service requests or priority induced service migrations. Again, higher virtual capacity selection is always favored in order to maximize the utilisation of the system. We design the penalty by setting ($P = a - C_{act}$), where a was the last virtual capacity (action) chosen and C_{act} is the service capacity of the server. For example, if the virtual capacity (action) was chosen to be 20 and the service capacity is equal to 50, then the penalty will be set to ($P = 20 - 50$). Thus, the lower the value of the chosen action, the higher will be the penalty in case of a failure. Similar to the normalization of the reward R, the penalty P also needs to be normalized. The minimum and maximum values that P can achieve are $-C_{act}$ and 0 respectively.

C. Generalized Assignment based Allocation of the Lower Priority Services

In this sub-section, we formulate an integer linear programming (ILP) based approach for allocating the service requests arriving from the OH users. If there are M servers and N services are to be allocated at a time instant, the ILP can be formulated as -

$$\text{maximize} \sum_{i=1}^{N} \sum_{j=1}^{M} \alpha_{i,j} \rho_{i,j} \tag{7a}$$

$$\text{subject to} \sum_{i=1}^{N} \alpha_{i,j} w_i \leqslant C_j, j = 1, ..., M \tag{7b}$$

$$\sum_{j=1}^{M} \alpha_{i,j} \leqslant 1, i = 1, ..., N \tag{7c}$$

$$\alpha_{i,j} \in \{0, 1\}, i = 1, ..., N, j = 1, ..., M \tag{7d}$$

where, $\alpha_{i,j}$ is a binary variable which is equal to 1 if service i is allocated to server j. Otherwise $\alpha_{i,j} = 0$. $\rho_{i,j}$ is the profit associated with allocation of service i to server j. $w_{i,j}$ is the number of service units consumed by service i. C_j is the virtual capacity of server j. (7b) denotes that a server cannot accommodate services more than its processing capacity. (7c) denotes that a service cannot be allocated to more than one server. Finally, (7d) indicates that a service may be either fully allocated to a server or may not be allocated at all.

Please note, we have introduced the term *"virtual capacity"*, which is obtained from the learning algorithm that has been described in Section IV-B. The virtual capacity is always less than or equal to the current actual capacity of the server. The difference between the virtual and actual capacities determines the service units that are reserved for the MA services so that OH services can avoid the migration caused by the service requests generated by high priority services.

Finally, GAP is *np-hard* [36]. Therefore, solving the ILP in (7) is not practical for real-time allocations. Hence, *we employ an approximation algorithm introduced in [14] to solve the GAP* [2].

[2] Since, GAP is a combination of the Assignment Problem and the Knapsack problem, GAP does not have the Total Unimodularity property. As a result, GAP cannot be solved by an equivalent linear program.

TABLE III: Summary of PrISMM

Markov Decision Process based Allocation of the high priority MA users
Inputs:
System information and ambulance path
Objective:
Choosing the optimal MEC server for MA users with the target of minimizing mobility-based service migration.
Reason:
Priority services should be given unhindered service while meeting QoS requirements.
Time of Execution:
On arrival of a MA user. Also on MA user migration.
Learning Automata (LA) based Virtual Capacity Determination
Inputs:
Allocation details of MA users from MDP framework described Section IV-A.
Objective:
Identifying the MEC servers that are more frequently used by MA users and reserving resources for them.
Reason:
Minimizing priority induced service migration of the lower priority OH users and lowering admission time of the higher priority MA users.
Time of Execution:
On MA and OH user allocation (both arrival and migration induced allocation).
Generalized Assignment Programming (GAP) based Allocation of the Lower Priority Services
Inputs:
System information and Virtual capacity from LA based framework described in Section IV-B.
Objective:
Optimally mapping the lower priority OH users to the best candidate MEC Server.
Reason:
Finding the best combination (MEC-OH user) for meeting SLA of the OH users and maximizing network performance.
Time of Execution:
On arrival of a OH user. Also on OH user migration.

1) Profits: In the allocation procedure of the OH users, we assume they have a preference for the MEC server that has lowest communication delay. In order to integrate that into the optimization, we sort the MEC servers by increasing communication delays. Allocation to the nearest MEC server fetches the highest reward and the allocation to the most distant MEC server produces the lowest reward. The reward can be designed in any manner as long as the ranking order is maintained. For simplicity, we use $(M-rank)$ as the profit associated to a certain server. Therefore, the MEC server with least communication delay is ranked 0 and hence, the profit associated to it is equal to M, while the MEC server having highest communication delay is ranked $(M-1)$ and as a result, the profit associated to it is 1.

2) Weights: We assume that each OH service consumes a single server processing unit. However, the formulation is perfectly valid for a situation where a single OH service can consume multiple processing units.

D. Summary of PrISMM

In this sub-section, we summarize the operation of PrISMM and provide the relations between the three components of PrISMM, i.e., the MDP Framework, the LA framework and the GAP framework in Table III.

TABLE IV: Simulation Parameters

Parameter	**Values**
No. of MEC Servers	10
No. of Hospitals	10
No. of Base Stations	300
No. of Service units in each MEC Server (M)	10
OH Average Arrival Rate (λ_{OH})	$0.01s^{-1}$
OH Average Holding Time (H_{OH})	$1000s$
OH Average Bulk Size (Λ_{OH})	variable
MA Average Arrival Rate (λ_{MA})	variable
MA Average Holding Time (H_{MA})	$500s$
OH Service Migration Time [37]	3.3 s
OH Service Migration File Size [37]	184.6 MB
MA Data Rate	5 Mbps
OH Data Rate	2 Mbps
ONU Buffer-Size [27]	9.95328 Mb
GPON Upstream Link Rate [27]	9.95328 Gbps
GPON Downstream Link Rate [27]	9.95328 Gbps
No. of PONs	5
No. of ONUs per PON	64

V. Evaluation

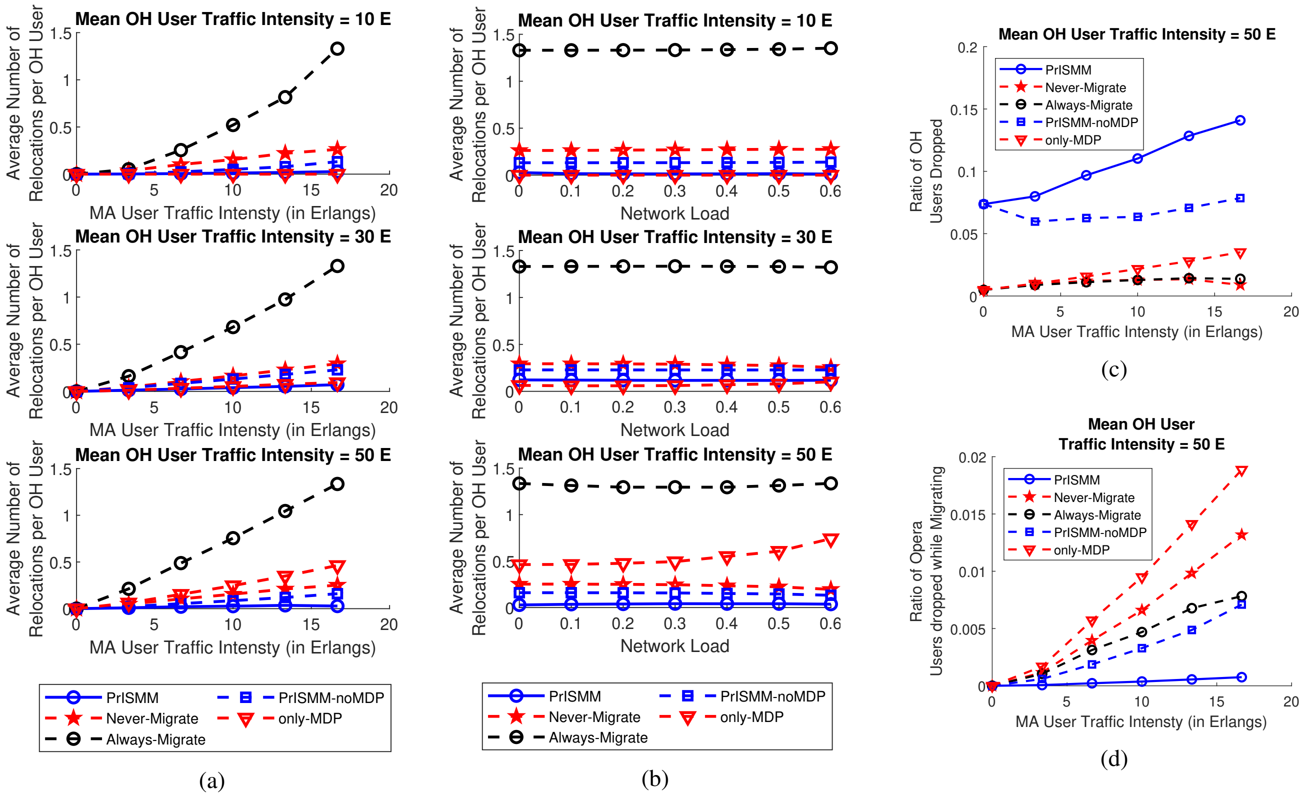
In this section, we provide the simulation set up, the metrics, the baseline cases that were used to evaluate the performance of PrISMM and the comparative performance results of PrISMM against the four baseline cases (please see Section V-A2).

A. Simulation Setup

The simulation environment was developed using the OMNeT++ network simulator. The average arrival rate and the average holding time for the OH users were fixed. We varied the average bulk size of the OH users to alter the system load. On the other hand, average holding time of the MA users was fixed. The average arrival rate was varied in order to change the system load. The scenario was developed by using the geographical details of the city of Milan, with real location of mobile BSs and hospitals. Thereafter, we randomly placed the MEC servers. We maintained the same positions of the MEC servers for the entire simulation study. The OH and the MA users were also placed at random locations. The central node was placed approximately at the centre of the city. The parameters used in the simulations are summarized in Table IV.

1) Metrics: We have evaluated the performance of PrISMM against the baseline approaches using five metrics:

1) *Average number of relocations per OH User* – This metric represents the average number of relocations experienced by the OH users that are already in service. In other words, the metric measures the average number of service migrations experienced by OH users. These relocation take place in order to accommodate the MA user requesting for service.
2) *Overall drop ratio of the OH users* – This metric gives a measurement of the overall percentage of OH users that are denied service by the system.



(a)

(b)

(c)

(d)

Fig. 6: Average Number of Relocations per OH User and OH Drop Ratio. (a) Average Number of Relocations per OH User with no Background Traffic. (b) Average Number of Relocations per OH User in the presence of Background Traffic (MA User Traffic Intensity = 16.67 E). (c) Overall Drop Ratio of the OH Users (d) Overall Drop Ratio of the OH Users resulting from Service Migrations.

3) *Mean admission delay for the MA users* – This metric captures the time for which the MA user must wait before it is granted service by the MEC system.
4) *Overall drop ratio of the MA users* - This metric provides a measurement of the overall percentage of MA users that are dropped by the system.
5) *Average Number of Relocations per MA user Radio Handover* - This metric represents a measure of the mobility based service migrations of the MA users.

2) Baseline Cases: In this sub-section, we describe the baseline cases against which the performance of PrISMM was evaluated.

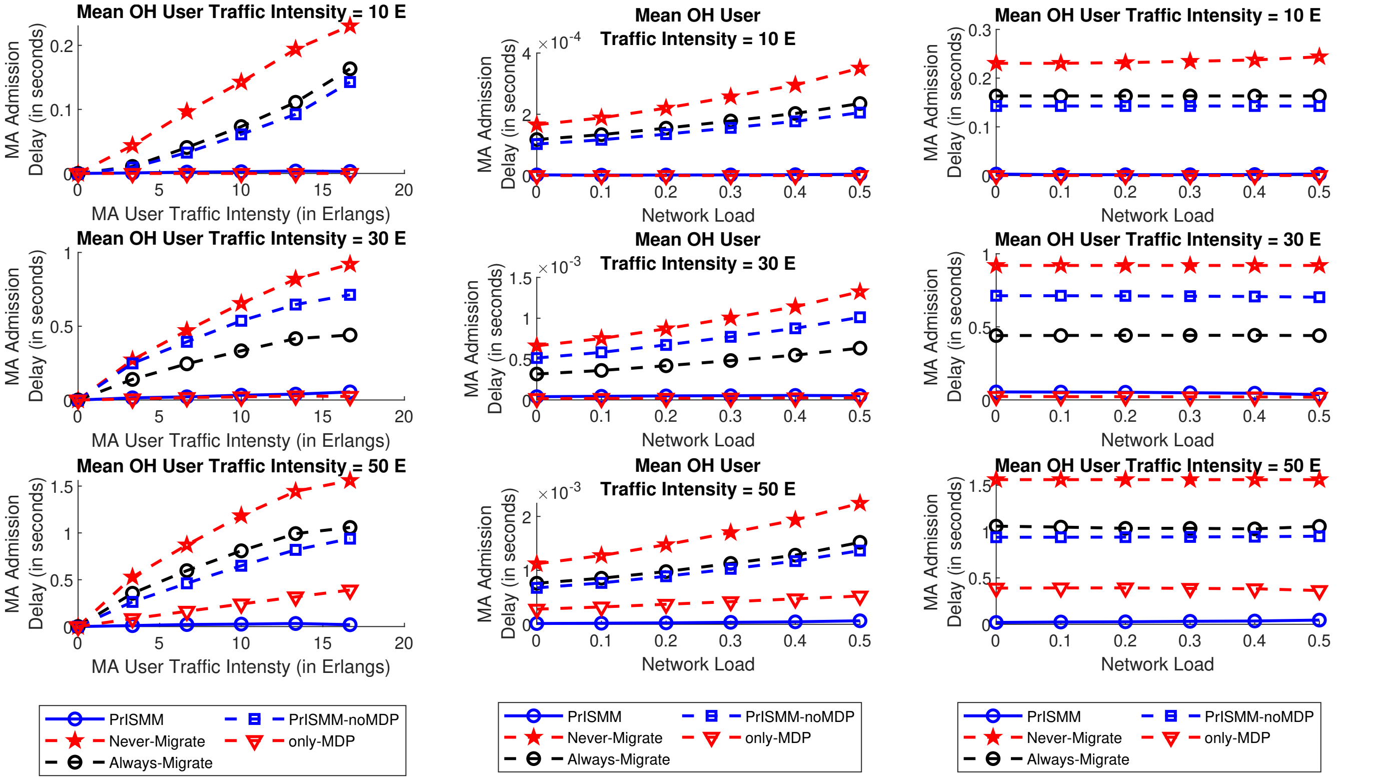
1) In the first baseline case, the OH user services are allocated with preference to the nearest geographically located MEC server, whenever they arrive, and the MA users are allocated in the MEC server that is located nearest to the destination hospital. We call this baseline method as *Never-Migrate* in the comparative graphs.
2) The second baseline case is similar to the first baseline case, but the MA users are allocated to their nearest MEC server. Service migration of the MEC servers are also performed whenever the delay of communication between the MA user and the MEC server exceeds the delay threshold. We call this method as *Always-Migrate* in the comparative graphs.
3) The third baseline case utilizes Learning Automata (mentioned in Section IV-B) along with GAP (mentioned in Section IV-C) based assignment for the OH user allocation. However, the MA users are always allocated to the MEC server that are located nearest to the hospital [38]. We call this method as *PrISMM-noMDP* in the comparative graphs.
4) The fourth baseline case uses MDP to allocate and perform service migration of the MA users as discussed in [17]. However, for the OH users, the nearest geographically located MEC server is chosen. We have named this baseline as *only-MDP*.

B. Results

In this sub-section, we discuss the results of the evaluation of PrISMM.

1) Average number of Relocations per OH User: In Fig. 6a, we can see that PrISMM reduces the average number of service migrations of the lower priority OH users, induced by high priority users (MA). We observe such a marked fall in the average number of relocation because PrISMM proactively reserves MEC server units for the higher priority MA users on one hand while optimally allocating the MA users by exploiting a MDP. Therefore, whenever a MA user requests service from the MEC servers, it is more likely to get access without triggering a lower priority service migration. Further, the service migration of the MA user is performed using MDP

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3139857, IEEE Transactions on Network and Service Management



(a)

(b)

(c)

Fig. 7: MA Admission delay. (a) Mean Admission Delay for MA Users with no Background Traffic. (b) Mean Admission Delay (only Network Delay) for MA Users in the presence of Background Traffic (MA User Traffic Intensity = 16.67 E). (c) Mean Admission Delay for MA Users in the presence of Background Traffic (MA User Traffic Intensity = 16.67 E).

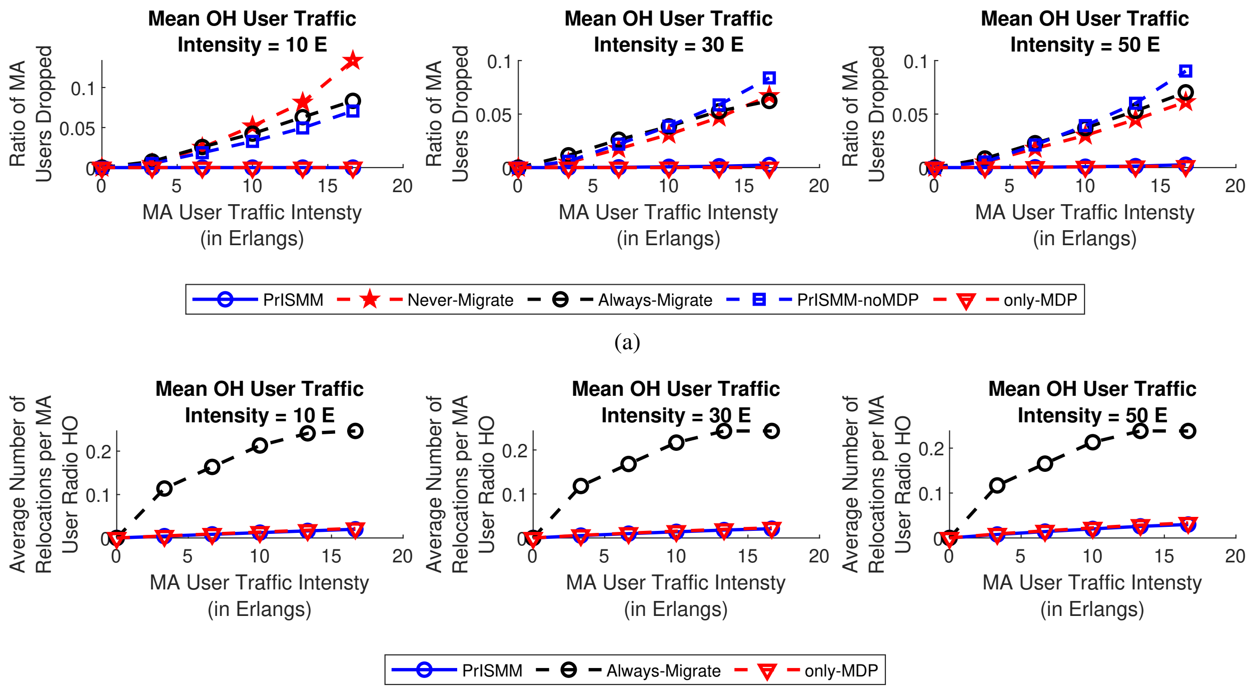
and as a result, the mobility induced service migration of the MA users are minimized. Consequently, the priority induced service migration of the OH users are also minimized in this case.

Always-Migrate results in the highest number of OH user relocation as the MA users are always allocated to the nearest MEC server to the MA user. Therefore, with the movement of the MA users, service migrations are performed. As a result, further priority (MA user) induced OH service migrations take place. *Never-Migrate* performs slightly better than *Always-Migrate* because *Never-Migrate* eliminates MA user service migration. *Only-MDP*, on the other hand, performs better at low OH user load as the lower OH load leaves sufficient resources for the MA users. As a result, MA users can be accommodated without OH service migration. However, the non-proactive allocation of OH services increases the number of priority induced service migrations with the increase in OH load. Finally, *PrISMM-noMDP* does not support MA user service migration. At the same time, it proactively reserves resources for the MA users while allocating the OH user. As a result, it performs better than the other baselines, especially at higher OH user load. However, the overall optimal and proactive allocation procedure of PrISMM allows it to stand out from the rest of the allocation mechanisms.

Thereafter, the effect of background traffic on the average number of OH user service migration is studied in Fig. 6b. For this purpose, we have fixed the MA user traffic intensity at a high value of 16.67 Erlangs and observed the effect of background traffic on different OH user load. We can observe that the average number of OH service migrations remains more or less unaffected by the change in network background traffic load. Slight increase in the OH service migration can be observed in case of *Only-MDP* at higher OH user load as the background traffic induces more latency in the MA services, thereby inducing more frequent MA service migration, which in turn forces priority induced service migration of the OH users. However, the optimal allocation of PrISMM, still performs better than the other baseline algorithms in variable network loads.

2) Overall drop ratio of the OH users: Fig. 6c shows that PrISMM slightly increases the overall drop ratio (blocking probability) of the lower priority OH users. The observation is in line with the blocking probability increase in the Erlang-B (M/M/S/S) queuing model [39]. Since, we are effectively decreasing the number of service units available to the OH users, their blocking probability increases and hence we get the higher blocking characteristic observed in Fig. 6c. Even though PrISMM introduces a small increase in service blocking, it provides a much higher quality of service by reducing the service interruptions for the active users.

On the other hand, Fig. 6d suggests that the OH drop resulting from OH service migrations is minimized by PrISMM and *PrISMM-noMDP*. This happens because both PrISMM and *PriSMM-noMDP* reduces the overall probability of service



(a)

(b)

Fig. 8: (a) Overall Drop Ratio of the MA Users. (b) Average Number of Relocations per MA User Radio Handover.

migrations and therefore, the chances of a failed service migration is also reduced. We have not shown results with lower OH user traffic intensity as there are negligible OH drops during service migrations at lower OH traffic intensity.

3) Mean admission delay for the MA users: PrISMM lowers the admission time of the higher priority service into the MEC server as the high priority service does not have to wait for the migration of the lower priority service before getting access to the server. Similarly, the lower priority users already in service have a much lower chance of facing service downtime that arises due to service migration. Hence, PrISMM reduces the chances of facing a service down time that can be as high as 3.3s for a simple video streaming application [37]. Fig. 7a, illustrates the lowering of average admission time (and also lower priority service downtime) by PrISMM. It can be seen that PrISMM provides the lowest MA admission delay as the MDP along with the GAP and LA based placement of the OH users ultimately provide the best possible allocation for the MA users. *Never-Migrate* provides the highest average waiting time for MA admission as it allocates the MA users in a limited set of MEC servers while allocating the OH users in a sub-optimal manner. Consequently, huge number of OH service migrations take place during the operation time.

When looked into the admission delay with the changing network load, PrISMM performs better than the baselines (especially at higher OH loads) [see Fig. 7b and Fig. 7c]. The primary reason for this better performance results from lower number of service migrations of the OH users due to the overall optimal allocation methodology used in PrISMM.

4) Overall drop ratio of the MA users: Thereafter, we compare the performance of the algorithms in terms of the ratio of MA users that are dropped. From Fig. 8a, we can clearly see that *Never-Migrate* and *PrISMM-noMDP*, which do not provision any mobility based service migration of the MA users, lead to high MA user drop. The reason for such a behaviour can be attributed to the fact that when a MEC server is situated far away from the user, the server can no longer return the processed information back the user within the prescribed delay deadlines. As the load on the network increases, the drop ratio shoots up due to the increase in network delay. On the other hand, *Always-Migrate* performs frequent service migrations and may not find vacant service units in nearby servers and hence, leads to higher MA user drops. This problem aggravates with increase in MA user load. Finally, the algorithms which support migration but with optimal MA function placement (*only-MDP* and PrISMM) lead to negligible MA user drop owing to their tendency to allocate the servers optimally.

5) Average Number of Relocations per MA user Radio Handover: Finally, we compare the average number of mobility based service migrations experienced by the MA users. In Fig. 8b, we observe that *Always-Migrate* suffers from the highest number of mobility based relocations of the MA users as it always allocates the user to the MEC server situated nearest to the serving BS. Hence, whenever a radio handover takes place, the proximity of the serving MEC server is checked against the other candidate MEC servers. Service migration is performed if one of the candidate MEC server is closer to the BS than the serving MEC server. On the other hand, due to the MDP based optimal allocation of the MA users, *only-MDP* and PrISMM influences much lesser mobility based service migration of the MA users. We do not include *Never-Migrate* and *PrISMM-noMDP* in Fig. 8b as they do not support any mobility based service migration of the MA users. Please note that due to the absence of mobility based service migration of the MA users, *Never-Migrate* and *PrISMM-noMDP* suffer from higher drop of MA users (see Fig. 8a).

VI. Conclusion

MEC is a recent paradigm that was developed to enable offload of processing tasks to nodes located close to end users, in order to support high reliability, high capacity and low latency applications. In this paper, we presented a novel approach to the service allocation problem, by focusing on the minimization of migrations, when services with different priorities, but with similar low latency and high capacity constraints, share the same network. The simulation results showed that our priority induced service migration minimization algorithm (PrISMM), considerably reduces the cost associated with service migrations. Consequently, PrISMM reduces the service downtime for both the higher and lower priority users resulting from service migrations and also reduces the expected admission time and drop percentage for the new higher priority users. PrISMM also reduces the overall drop of OH users that result from service migrations. This can be of great advantage in MEC systems aiming to support reliable and low-latency services.

Acknowledgment

Financial support from Science Foundation Ireland grants [16/SP/3804 (ENABLE), 17/CDA/4760 (SoftEdge) and 13/RC/2077 (CONNECT)] and European Commission grant [101017109 (DAEMON)] is gratefully acknowledged.

References

[1] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, “Mobile edge computing—a key technology towards 5g,” *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[2] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, “Deep learning for iot big data and streaming analytics: A survey,” *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 2923–2960, Fourth quarter 2018.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Oct 2016.

[4] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, “A survey on mobile edge computing: The communication perspective,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[5] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, “Mobile edge computing: A survey,” *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, Feb 2018.

[6] T. Taleb, S. Dutta, A. Ksentini, M. Iqbal, and H. Flinck, “Mobile edge computing potential in making cities smarter,” *IEEE Communications Magazine*, vol. 55, no. 3, pp. 38–43, 2017.

[7] P. Mach and Z. Becvar, “Mobile edge computing: A survey on architecture and computation offloading,” *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1628–1656, Third quarter 2017.

[8] M. Ruffini and F. Slyne, “Moving the network to the cloud: The cloud central office revolution and its implications for the optical layer,” *Journal of Lightwave Technology*, vol. 37, no. 7, pp. 1706–1716, April 2019.

[9] M. Ruffini, “Multidimensional convergence in future 5g networks,” *Journal of Lightwave Technology*, vol. 35, no. 3, pp. 535–549, Feb 2017.

[10] S. Wang, J. Xu, N. Zhang, and Y. Liu, “A survey on service migration in mobile edge computing,” *IEEE Access*, vol. 6, pp. 23 511–23 528, 2018.

[11] A. Machen, S. Wang, K. K. Leung, B. J. Ko, and T. Salonidis, “Live service migration in mobile edge clouds,” *Wireless Commun.*, vol. 25, no. 1, pp. 140–147, Feb. 2018.

[12] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[13] K. S. Narendra and M. A. L. Thathachar, “Learning automata - a survey,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-4, no. 4, pp. 323–334, July 1974.

[14] R. Cohen, L. Katzir, and D. Raz, “An efficient approximation for the generalized assignment problem,” *Information Processing Letters*, vol. 100, no. 4, pp. 162–166, November 2006.

[15] A. Ksentini, T. Taleb, and M. Chen, “A markov decision process-based service migration procedure for follow me cloud,” in *2014 IEEE International Conference on Communications (ICC)*, June 2014, pp. 1350–1354.

[16] S. Wang, R. Urgaonkar, T. He, M. Zafer, K. Chan, and K. K. Leung, “Mobility-induced service migration in mobile micro-clouds,” in *2014 IEEE Military Communications Conference*, Oct 2014, pp. 835–840.

[17] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, “Dynamic service migration in mobile edge computing based on markov decision process,” *IEEE/ACM Transactions on Networking*, vol. 27, no. 3, pp. 1272–1288, 2019.

[18] T. Taleb, A. Ksentini, and P. A. Frangoudis, “Follow-me cloud: When cloud services follow mobile users,” *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 369–382, April 2019.

[19] T. Taleb and A. Ksentini, “An analytical model for follow me cloud,” in *2013 IEEE Global Communications Conference (GLOBECOM)*, Dec 2013, pp. 1291–1296.

[20] Kuo-Hsing Chiang and N. Shenoy, “A 2-d random-walk mobility model for location-management studies in wireless networks,” *IEEE Transactions on Vehicular Technology*, vol. 53, no. 2, pp. 413–424, March 2004.

[21] R. Langar, N. Bouabdallah, and R. Boutaba, “A comprehensive analysis of mobility management in mpls-based wireless access networks,” *IEEE/ACM Trans. Netw.*, vol. 16, no. 4, pp. 918–931, Aug. 2008.

[22] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, “Dynamic service placement for mobile micro-clouds with predicted future costs,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1002–1016, April 2017.

[23] A. Ksentini, T. Taleb, and F. Messaoudi, “A lisp-based implementation of follow me cloud,” *IEEE Access*, vol. 2, pp. 1340–1347, 2014.

[24] 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, 2018.

[25] Y. Sun, S. Zhou, and J. Xu, “Emm: Energy-aware mobility management for mobile edge computing in ultra dense networks,” *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2637–2646, 2017.

[26] D. Zhao, T. Yang, Y. Jin, and Y. Xu, “A service migration strategy based on multiple attribute decision in mobile edge computing,” in *2017 IEEE 17th International Conference on Communication Technology (ICCT)*. IEEE, 2017, pp. 986–990.

[27] B. Skubic, J. Chen, J. Ahmed, L. Wosinska, and B. Mukherjee, “A comparison of dynamic bandwidth allocation for epon, gpon, and next-generation tdm pon,” *IEEE Communications Magazine*, vol. 47, no. 3, pp. S40–S48, 2009.

[28] I. T. S. Sector, “10-gigabit-capable symmetric passive optical network (xgs-pon),” *ITU-T Recommendations, G Series*, 2016.

[29] S. McGettrick, F. Slyne, N. Kitsuwan, D. B. Payne, and M. Ruffini, “Experimental end-to-end demonstration of shared n:m dual-homed protection in sdn-controlled long-reach pon and pan-european core,” *Journal of Lightwave Technology*, vol. 34, no. 18, pp. 4205–4213, 2016.

[30] A. Mukhopadhyay and G. Das, “A ring-based wireless optical network to reduce the handover latency,” *Journal of Lightwave Technology*, vol. 33, no. 17, pp. 3687–3697, 2015.

[31] P. Alvarez, N. Marchetti, D. Payne, and M. Ruffini, “Backhauling mobile systems with xg-pon using grouped assured bandwidth,” in *2014 19th European Conference on Networks and Optical Communications - (NOC)*, 2014, pp. 91–96.

[32] A. Elrasad, N. Afraz, and M. Ruffini, “Virtual dynamic bandwidth allocation enabling true pon multi-tenancy,” in *2017 Optical Fiber Communications Conference and Exhibition (OFC)*, 2017, pp. 1–3.

[33] H. Lin, X. Xu, J. Zhao, and X. Wang, “Dynamic service migration in ultra-dense multi-access edge computing network for high-mobility scenarios,” *EURASIP Journal on Wireless Communications and Networking*, vol. 2020, no. 1, pp. 1–18, 2020.

[34] R. Bellman, “A markovian decision process,” *Journal of mathematics and mechanics*, pp. 679–684, 1957.

[35] K. S. Narendra and M. A. Thathachar, *Learning Automata: An Introduction*. Courier Corporation, 2012.

[36] D. G. Cattrysse, M. Salomon, and L. N. Van Wassenhove, "A set partitioning heuristic for the generalized assignment problem," *European Journal of Operational Research*, vol. 72, no. 1, pp. 167–174, 1994.

[37] A. Machen, S. Wang, K. K. Leung, B. J. Ko, and T. Salonidis, "Migrating running applications across mobile edge clouds: Poster," in *Proceedings of the 22Nd Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '16, 2016, pp. 435–436.

[38] A. Mukhopadhyay and M. Ruffini, "Learning automata for multi-access edge computing server allocation with minimal service migration," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[39] L. Kleinrock, *Queueing Systems. Volume I: Theory*. Wiley New York, 1975.

Atri Mukhopadhyay received his M.Tech degree from Jadavpur University, Kolkata, India in 2011 and Ph.D. degree from the Indian Institute of Technology, Kharagpur, India in 2019. He is presently working as a Postdoctoral researcher in CONNECT Telecommunications Research Centre, Trinity College Dublin, Ireland. He has been a part of the ENABLE, CODYSUN and Enterprise Ireland projects. His research interests include wireless-optical integrated networks, optical access networks, wireless access networks, edge computing, dynamic spectrum sharing and free space optics.

George Iosifidis is an Assistant Professor at Delft University of Technology, Delft, Netherlands. He received a Diploma in Electronics and Telecommunications Engineering from the Greek Air Force Academy (Athens, 2000), and a PhD degree from the ECE Department, University of Thessaly in 2012. He was a Postdoctoral researcher ('12-'14) at CERTH, and at Yale University ('14-'17). His research interests lie in the broad area of network optimization and economics. He is a co-recipient of the best paper awards in IEEE WiOPT 2013 and IEEE INFOCOM 2017, has served as a guest editor for the IEEE Journal on Selected Areas in Communications, and is currently an editor for IEEE Transactions on Communications and IEEE/ACM Transactions on Networking.

Marco Ruffini received the M.Eng. degree in telecommunications from the Polytechnic University of Marche, Italy, in 2002, and the Ph.D. degree Trinity College Dublin (TCD) in 2007, where he joined Trinity College Dublin in 2005, after working as a Research Scientist with Philips, Germany.

He is Associate Professor and Fellow of Trinity College and he is Principal Investigator of both the IPIC Photonics Integration Centre and the CONNECT Telecommunications Research Centre. He is currently involved in several Science Foundation Ireland and H2020 projects, including a new research infrastructure to build a beyond 5G testbed in Dublin. Prof. Ruffini leads the Optical Network Architecture Group, TCD and has authored over 150 international publications, over ten patents and contributed to standards at the broadband forum. He has raised research funding in excess of €7M. His main research is in the area of 5G optical networks, where he carries out pioneering work on the convergence of fixed-mobile and access-metro networks, and on the virtualization of next generation networks, and has been invited to share his vision through several keynote and talks at major international conferences across the world. He leads the new SFI funded Ireland's Open Networking testbed infrastructure.