Future Generation Computer Systems 102 (2020) 723–737

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Osmotic computing-based service migration and resource scheduling in Mobile Augmented Reality Networks (MARN)☆

Vishal Sharma [a], Dushantha Nalin K. Jayakody [b,*], Marwa Qaraqe [c]

[a] Department of Information Security Engineering, Soonchunhyang University, Asan-si 31538, South Korea
[b] School of Computer Science and Robotics, National Research Tomsk Polytechnic University, Russia
[c] Hamad Bin Khalifa University, Qatar

ARTICLE INFO

Article history:
Received 20 April 2019
Received in revised form 19 August 2019
Accepted 6 September 2019
Available online 18 September 2019

Keywords:
Osmotic computing
Edge-computing
Augmented reality
Resource scheduling
Service migrations

ABSTRACT

Resources and services between the servers in Mobile Augmented Reality Networks (MARN) are tedious to manage. These networks comprise users possessing Augmented Reality (AR)-Virtual Reality (VR) applications. Low latency, robustness, and tolerance are the key requirements of these networks, which can be attained by using near-user solutions such as edge computing. However, management of services and scheduling them to near-user servers in an integrated environment of edge and public/private infrastructure are complex tasks. These require an optimal solution, which can be obtained by using "Osmotic Computing", that has been recently proposed as a paradigm for the integration of edge and public/private cloud. This paper uses osmotic computing for effectively migrating and scheduling the services between the servers of the different layers. The paper also presents the details on various components that are used for applying osmotic computing to a network followed by core applications, types, service classification, migration, and scheduling through the rules of osmotic game formulated for its operations. The evaluations are conducted on 100,000 requests and the proposed approach shows significant performance with the probability of the error being 0.1 at 55.72% conservation of the energy and memory resources for the entire network despite the increasing number of users. The proposed approach also satisfies the conditions of the joint optimization functions presented in the system model and demonstrates that the system holds true even with varying users, thus, proving its robustness and tolerance against the number of users.

1. Introduction

Networks are playing a crucial role in connecting the real world applications through logical and algorithmic solutions ranging from traffic management to processing. Realistic services and applications demand a tremendous amount of resources in the next generation of wireless communication [1]. The advent of higher spectrum allows better resources for the users and also supports the applications that lay an enormous burden on the network [2,3].

Network services operating through real-time traffic have formed the background for the development of Mobile Augmented Reality Networks (MARN) that focus on Augmented Reality (AR) and Virtual Reality (VR) applications [4,5]. The applications developed by these industries are heavy on resources and require effectual management for efficient services and control with low-complex solutions for the transmission of multimedia traffic [6,7]. Most of the solutions rely on the network layout and vary depending on the configurations of the packet and routing strategies. The current standards and solutions are incapable of providing a generic solution for most of the real-time applications [8]. This raises the requirement of an efficient solution which operates irrespective of the configurations and the type of the network.

These applications are also seen as potential seekers for the formation of edge networks [9,10]. Edge networks allow near user-site evaluations by reducing the burden on the core of the network [11–13]. This protects the centralized server from the burden of simultaneous requests generated by a large number of applications [14–17]. Edge networks rely on various scheduling algorithms and resource allocation strategies for effective communication with users involving AR-VR applications. Inter-hop distance, multi-path facilities, routing protocols, distributed control, and management are the other key challenges associated with the networks serving AR-VR applications. Network accessibility and portability of resources play a key role in supporting

☆ This work was funded, in part, by the framework of Competitiveness Enhancement Program of the National Research Tomsk Polytechnic University, Russia.

* Corresponding author.
E-mail addresses: vishal_sharma2012@hotmail.com (V. Sharma), nalin@tpu.ru (D.N.K. Jayakody), mqaraqe@hbku.edu.qa (M. Qaraqe).

https://doi.org/10.1016/j.future.2019.09.008
0167-739X/

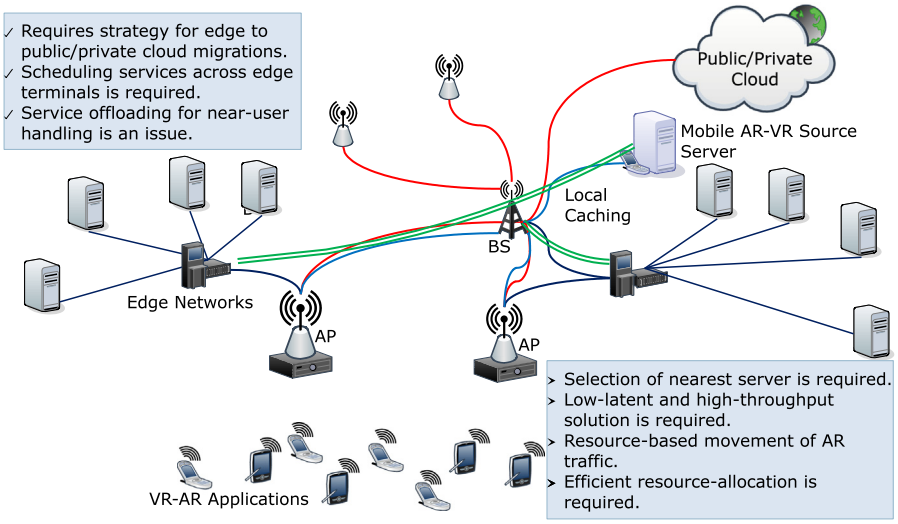


Fig. 1. An exemplary illustration of MARN with challenges and requirements.

Table 1
Requirements of MARN w.r.t. devices [17].

Parameters	Handheld	Wearable	Desktop
Connection-Type	Wireless	Wireless	Wired/Wireless
Storage Available	Low	Low	High
Transmission Range	Moderate	Low	High/Moderate
Latency Issue	Moderate	High	Low
Visibility	3D	3D	3D
Reconfigurability	Required	Required	Required
Local Caching	N/W dependent	N/W dependent	N/W dependent

users in AR networks. Usually, applications for AR-VR are dedicated and uses their own set of protocols for communications which are sometimes greedy and result in overpowering other crucial applications on the same device [18].

The next generation of mobile Apps industry is focused on the combination of AR and VR with mobile networks [19]. In general, AR uses a series of computing algorithms which illustrates the physical world entities into visually displayable sensory information. This information is interpreted by the displaying devices and displayed on the screen. With a combination of AR with VR, the sensory information is presented as a logical real world to its users. Earlier, majority of the applications were developed as a stand-alone solution such as simulation-based pilot training [20], evaluation of Supervisory Control and Data Acquisition (SCADA) systems [21], or inter-terrestrial image transmissions [22]. But now, the scale of such applications has improved and the majority of the applications uses a centralized server for managing information from one application and mixes it with the other for a new VR interface for the end users. Such applications involve online-gaming, tactical planners for pilots, or even drone-maneuvering. The current deployment depends on the communication network for the transmission of real-time feed necessary for the successful operations of AR-VR Apps. Most recently, these systems are studied as MARN, as shown in Fig. 1. These networks are subject to high throughput and low-latency communications [17]. These requirements, as shown in Table 1, can be handled by the use of high-frequency networks as planned under 5G-PPP. However, because of high-frequency, these networks involve using more intermediate hops leading to more computations and raise the issue of computational offloading. Even with the cached information, the processing needs to be faster with fewer delays. The processing issues in MARN are handled by the extremely reliable and fast computing algorithms. However, the support for communication between these far-operated algorithms is still an issue. There are several algorithms and protocols available focusing on the transmission of multimedia traffic with less latency, yet these are subject to investigation considering the new deployment of networks as their incapability in managing near-user evaluations [6–8].

The network architecture plays a crucial role in MARN. With the advent of the edge-enabled network, it becomes relatively easy to manage user requests near to them without many traversals. Not all the requests can be managed near the user; some require dependencies on additional traffic while some are bound by the constraints of network resources. Thus, it becomes necessary to identify a solution that can support these requirements and can efficiently manage the requests between the core and the user-site servers. All these issues can be handled by the efficient deployment of osmotic computing, which is a novel paradigm for integrating edge-cloud and public/private infrastructure [23]. But, the current implementation of osmotic computing is in its early stage and requires further extensions for various problems discussed in the next section.

2. Problem statement and motivation

With a large number of devices demanding an equal amount of time and resources for similar or different types of applications in MARN, it is important to manage their operations efficiently. This management is dependent on regulating the demands and traffic for users with efficient utilization of resources and congestion-free transmission. The key problems with the existing architectures are:

- There is no facility for near-user decision system which can regulate the traffic between the cloud and edge infrastructures especially focusing the MARN applications.
- The existing solutions are unable to deal with the counterfeit mobility issues of users in edge-enabled networks for AR. Maintaining consistent flow across the network is still an open issue.
- Reduction in latency while migrating the services across the layers is a key challenge.
- Optimized allocation of resources and classification of services into macro- and micro-units is a major requirement for efficient migration and scheduling.
- The existing osmotic solutions do not use optimal theory for handling users in edge-enabled MARN.

Osmotic computing is one of the promising concepts for integration of edge and public/private cloud infrastructure. Initially proposed by Villari et al. [24] and extended for an application (management of service heterogeneity) by Sharma et al. [25,26], osmotic computing is a new research paradigm to look forward to, especially in resolving the research and engineering aspects associated with the fully-fledged implementation of edge computing [23,27,28]. The recent study shows that the lack of inferential control over the AR-VR-based devices makes it difficult to use edge solutions [19]. This issue is one of the reasons for enterprises to avoid investments in edge-solutions. These challenges motivated us to developing a framework that can handle the classification of services as well as efficient resource scheduling by utilizing the osmotic paradigm.

3. Our contribution

In this paper, the concept of osmotic computing is further extended by providing detailed definitions of the key components required for its operations. A novel MARN architecture is considered for the deployment of osmotic servers in combination with the public/private infrastructure. Various types and applications of the demonstrated architecture are also presented. The proposed approach resolves the issues related to the migration and scheduling of services in MARN by following the novel classification of services on the basis of size and bounds. A novel osmotic game helps in scheduling through user-initiated osmosis or edge-initiated osmosis. Further, the system model is formulated into algorithms to provide an overall idea of the proposed solution. Next, the theoretical analyses are presented along with performance evaluation for a large set of users. The proposed approach provides a significant contribution in managing services across the edge and the public/private infrastructures with robust decision making and tolerance while conserving available resources (energy and memory) against the increasing number of incoming requests.

4. Background: Osmotic computing

Osmotic computing is the paradigm for migrating components across different processing layers to facilitate the end users. Here, components refer to any tangible computing entity or resource. Fig. 2 illustrates the chemical osmosis, which inspired the formation of osmotic computing. The details of osmotic computing are as follows:

4.1. Definitions

Let $\mathcal{U}$ be the set of all the users and services that migrate between different layers of servers such that for an ideal state, $\mathcal{S} = \phi$, where $\mathcal{S}$ is the set of unhandled services or pending user requests.

- Solute— Like osmosis, solute represents the dissolvable part of the chemical solution. In computing, solute refers to the network properties and entities, such as memory, energy, processing time, overheads, load, connections, etc. A computing paradigm may have a varying amount of solute which leads to the formation of an equilibrium state. The ratio of solvent present in the network for consuming the solute drives the equilibrium in osmotic computing.
- Solvent— These are the components of solution which absorbs the solute to maintain the fluid state. In osmotic computing, solvent refers to the applications, layered interfaces, users and services that undergo migration in a network. This is the movable part of the network which requires scheduling and allocation depending on the state of the network as well as equilibrium conditions.
- Concentration— It is the ratio of the solute to the solvent. In [25], it is the ratio of services to computational resources, whereas, in this paper, it is the cumulative ratio of every component treated as a solute to every computational resource treated as a solvent.
- Semipermeable Membrane— This is the decision-making system of the osmotic computing. It is the most crucial entity which decides on the maintenance of equilibrium and balance of services and applications in the network. This paper proposes an Osmotic Decision System (ODS), which handles migrations and scheduling of the resources and services across the network. For an optimized network, the semipermeable membrane should operate with low-complexity and fewer overheads.
- Solution— This is the combination of $\mathcal{U}$, $\mathcal{S}$ and osmotic-decision support system. Usually, solution refers to the entire network, but, depending on the scale of implementation and applicable solutions, a solution can also refer to subgroups which combine together to form the entire network.
- Properties-Following are the general decisions which form the key properties of the proposed osmotic computing model:
  - If, during any time in the network, $|\mathcal{S}| = \max$, then the existing servers cannot handle extra users. Such situations are manageable by load balancing, the addition of solvent, or removal of unwanted solute.
  - Centralized servers should not affect the decision for load balancing. Rather, these should be user initiated, but, such scenarios raise need for security solutions to confirm a user before migrations. In the proposed model, both users as well as infrastructure (Edge)-initiated approaches manages the services and the scheduling of resources.

4.2. Application scenarios

This paper presents two major applications of edge-enabled networks for handling a large number of users requiring AR-VR traffic. These users are the crucial part of the proposed model and an efficient formation of the osmotic network can ease their operations in edge-setup. The descriptions of these are as follows:

- Service Migration— It is one of the major applications of osmotic computing for integration of edge-public/private cloud infrastructure. As stated in [25], public/private cloud or the osmotic layer handles the services by dividing them into multiple sets.
- Resource Scheduling— After classification of services and clustering of potential servers with a label for handling a particular service, the network uses an osmotic game for optimized scheduling. This helps in the allocation of resources without much latency as well as overheads. Resource scheduling is applicable to a much lower level and classified services are divisible into tasks or jobs for migration. Thus, resource scheduling is applicable by performing service migration, job to job, or process to process migration.

4.3. Types

There are two ways of implementing osmotic computing:

- Intra-Osmosis: Consider a network with many servers with variation in computational resources on the same tier. Now, the applications are migrated and scheduled on these servers of same tier allowing applicability of the concept

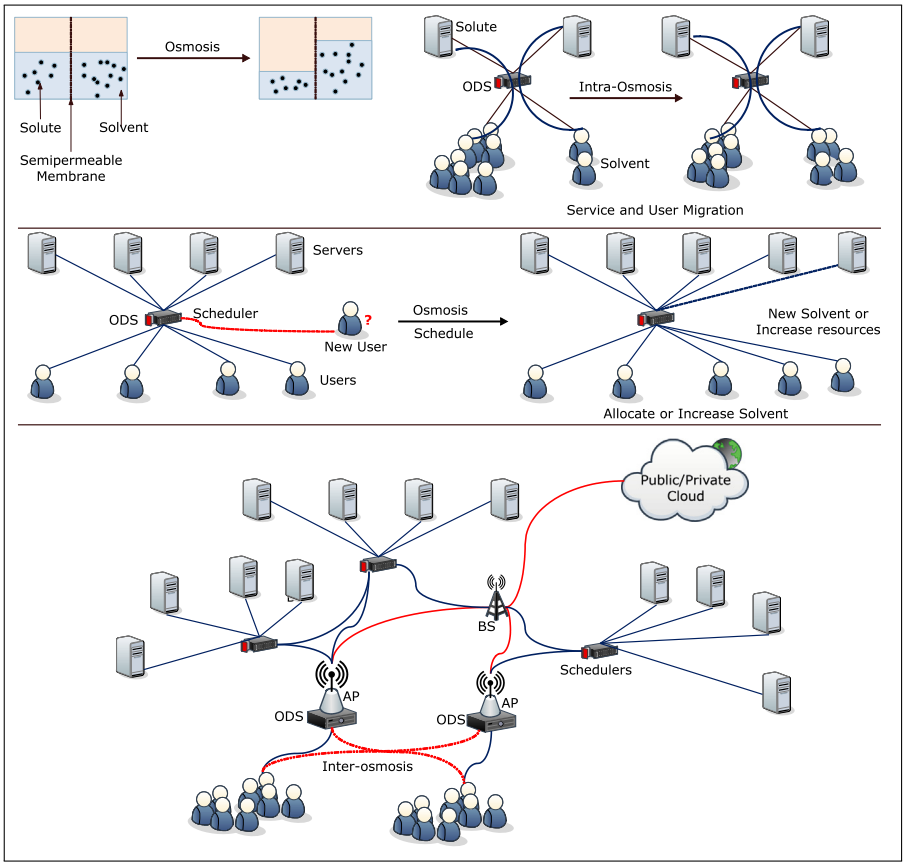


Fig. 2. An illustration of osmotic computing for service migration and resource scheduling.

of intra-osmosis. Intra-osmosis is applicable when decisions are made in a single network and there is no hierarchy in the servers. The upcoming networks with flattened architectures can take advantage of this approach for efficiently handling its resources.

- Inter-Osmosis: Inter-osmosis applies to a network when two different hierarchical servers are involved in service provisioning for users. Usually, inter-osmosis operates between different type of technologies, e.g. between the edge cloud and private/public cloud. Inter-osmosis holds the true concept of osmosis and balances the load in the network. The existing hierarchical networks that especially focus on the applications of augmented reality are good candidates for this approach.

5. Network model

The augmented reality network comprises an edge layer near to the users, the osmotic layer, and the core (public/private cloud) for handling the users denoted by a set $\mathcal{U}$ as shown in Fig. 3. The osmotic layer is the integration layer as suggested by its initial definition. The ODS for identifying the services are installed on the edge layer as well as on the user devices. The initial task is to formalize the components of the system which helps in allocating the services generated from the users to appropriate servers. This is followed by the load balancing and scheduling depending on the type and arrival rate of services.

In general, a network can have multiple osmotic layers between the edge layer and the public/private cloud. Every layer can operate with a set of servers with similar or different configurations. To support common applications these servers must be made available as generic support for the services which are configured to be operating over the entire network. However, depending on the deployment, osmotic computing can also be considered in a private mode, but such discussions are beyond the scope of this article. Let $\mathcal{K}$ be the set of osmotic layers, such that each layer has set $\mathcal{N}$ of servers and every server can handle a maximum of m users. In the system model, it is assumed that each user generates some services denoted by j. Such that total services for a server are represented as a set $\mathcal{M}$ with elements $j_1, j_2, \ldots, j_m$ denoting services from m users. Now, with $j = 1$ for every user, total load, which can be handled by the osmotic network, is given as[1]:

$$\mathcal{L}_{osmotic} = \sum_{i=1}^{|\mathcal{K}|} \left(\sum_{r=1}^{|\mathcal{N}|} (|\mathcal{M}|)_r \right)_i, \quad \text{for } j_{1\ldots m} = 1. \tag{1}$$

Now, the model operates towards the formation of memory, energy, time and capacity model to facilitate a migration network between the edge layer, osmotic layer and the public/private cloud. Considering that each server has memory represented by β, the total memory β_T is given as:

$$\beta_T = \sum_{i=1}^{|\mathcal{K}|} \left(\sum_{r=1}^{|\mathcal{N}|} (\beta)_r \right)_i, \tag{2}$$

which can be distributed equally on the basis of available memory across the servers in each osmotic layer. Let b be the memory consumed by services from each user (assuming a single service

[1] Following notations have same meaning in the article (A, $\mathcal{B}$, C are used as simple variables without any physical meaning):- $\sum_{i=1}^{\mathcal{B}}(A)_i = \sum_i^{\mathcal{B}}(A) = \sum_i^{\mathcal{B}} A$ and $\sum_{i=1}^{\mathcal{B}}(\sum_{j=1}^{C}(A)_j)_i = \sum_i^{\mathcal{B}} \sum_j^{C}(A)$

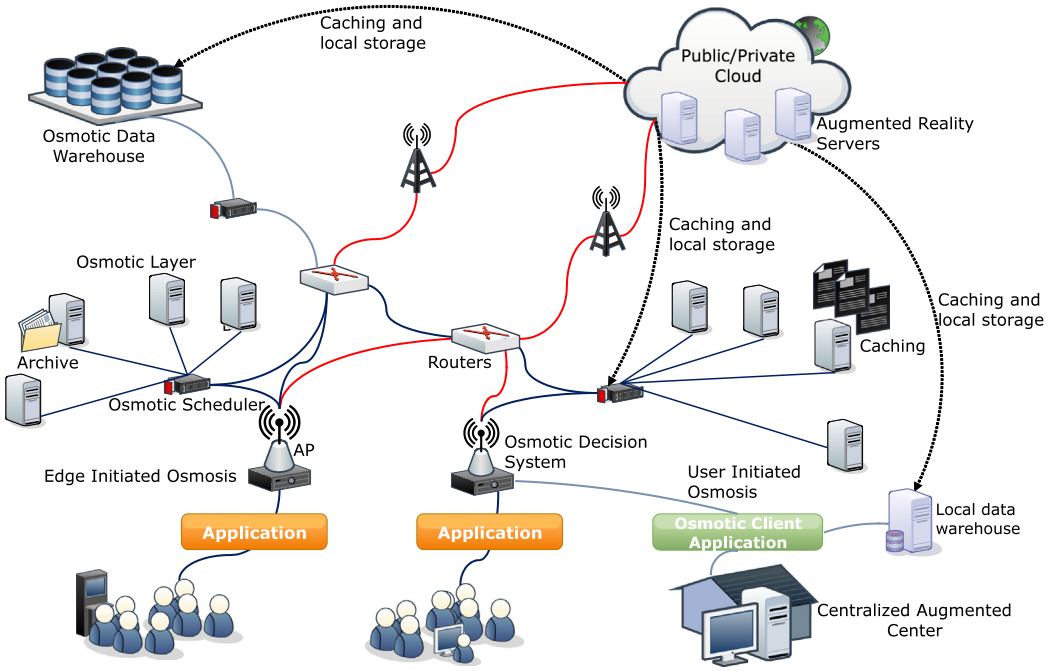


Fig. 3. An illustration of osmotic computing for edge-initiated and user-initiated service-migrations in MARN.

per user), then the available memory will be calculated as:

$$\beta_A = \beta_T - \sum_{i=1}^{|\mathcal{U}|} (b)_i \,. \tag{3}$$

Now, with $\mathcal{F}$ as a fresh set of services each having same memory needs, the total services that can be accommodated depending on the memory requirements can be calculated as $\frac{\beta_A}{|\mathcal{F}|}$. This model can be extended to each osmotic layer as well as to a single server, such that

$$\beta_A^{(\mathcal{K})} = \frac{\beta_T}{|\mathcal{K}|} - \sum_{i=1}^{|\mathcal{Z}|} (b)_i \,, \text{ s.t. } \sum_{i=1}^{|\mathcal{Z}|} (b)_i \leq \sum_{j=1}^{|\mathcal{U}|} (b)_j \,, \mathcal{Z} \subseteq \mathcal{U}, \tag{4}$$

where $\mathcal{Z}$ and $\beta_A^{(\mathcal{K})}$ represent the set of users and the available memory, respectively, of a single osmotic layer.

Let $\mathcal{E}_p$ be the per service energy consumption with $\mathcal{E}_T$ being the total energy of the network across all the servers, such that available energy in the network is given as:

$$\mathcal{E}_A = \mathcal{E}_T - \sum_{i=1}^{|\mathcal{U}|} \left(\mathcal{E}_p\right)_i \,. \tag{5}$$

Similar to memory model, energy model can also be applied for a single osmotic layer such that $\mathcal{E}_A^{(\mathcal{K})} = \left(\frac{\mathcal{E}_T}{|\mathcal{K}|} - \sum_{j=1}^{|\mathcal{Z}|} \left(\mathcal{E}_p\right)_j\right)$. Each service in the network requires some time for processing given by τ_p. The processing time is affected by the parallel runs, i.e. number of services which can be operated simultaneously. Thus, considering this, the processing time per server can be calculated as:

$$\tau_T^{(\mathcal{K})} = \left(\frac{|\mathcal{M}|}{\gamma}\right) \tau_p, \tag{6}$$

where γ is the number of parallel units of services for each server that can run simultaneously.

6. Proposed approach

The proposed approach aims at efficient service migrations and resource scheduling between the edge servers and the public/private cloud setup. As stated in the system model, osmotic layers serve the edge layer of the network comprising near-user servers, which are controlled by local Access Points (APs). On the contrary, a network can only have APs which themselves act as the osmotic servers and manage the purpose of edge layer near to a user. In such a case, the service migrations and scheduling occur only between the APs and the public/private cloud. Despite the type of architecture, the proposed approach supports low-latency solution for service migration. The proposed approach can serve as one of the key solutions for the low-complex and low-latent deployment of augmented reality networks. The details of the proposed approach are provided below:

6.1. Service classification

"Service classification" refers to the difference in the characteristics of operations and requirements of resources associated with a particular task. The existing solutions rely on the difference in computational characteristics of the entire application and schedule them according to the availability of serving servers. However, such situations can lead to mixed allocation of services to a single server. This means a single server has to perform computations for both heavy and light applications, which lead to complex optimization issues. These issues can overburden a single server and waste the resources available with the other servers. Of course, load balancing is available for such situations, but load balancing is applied as a failure-resilient solution when a particular server is unable to handle the requests. However, load balancing does not account for the difference in the types of services, and this difference in services arises when a server supports different kinds of application or even acts as a lookup server for the demanded services. Thus, it is important to understand the heterogeneity of the services in a network and classify them for efficient migration and scheduling. The proposed approach classifies the services on the basis of:

- Size: The heterogeneous nature of services can be understood by their divisibility on the basis of size. Such divisibility classifies the services into macro-and micro-services by identifying the resources demanded by each of them. The division can be accounted for the batch size of each service, which takes into account the number of resources required by each service. The one with requirements of high computational resources for execution of its batch are the macro-services, while with relatively lesser requirements are the micro-services. The division of services into macro and micro parts is manageable by defining a threshold limit on each of the configuration-resources, such as energy, memory and processing time as stated in [26], such that

$$\beta_{\mathcal{R}}^{(macro)} > \beta_{\mathcal{TH}} \geq \beta_{\mathcal{R}}^{(micro)}, \tag{7}$$

$$\mathcal{E}_{\mathcal{R}}^{(macro)} > \mathcal{E}_{\mathcal{TH}} \geq \mathcal{E}_{\mathcal{R}}^{(micro)}, \tag{8}$$

and

$$\tau_p^{(macro)} > \tau_{\mathcal{TH}} \geq \tau_p^{(micro)}, \tag{9}$$

where $\beta_{\mathcal{R}}^{(macro)}$, $\beta_{\mathcal{R}}^{(micro)}$, $\mathcal{E}_{\mathcal{R}}^{(macro)}$, $\beta_{\mathcal{R}}^{(micro)}$, $\tau_p^{(macro)}$, and $\tau_p^{(micro)}$ are the memory requirements, energy requirements, and processing time for macro and micro classification of services, respectively. The subscript $\mathcal{TH}$ denotes the thresholds for memory, energy and processing time. The details on the thresholds and their values are provided during performance evaluation (Section 8). The initial approach begins with analyzing whether the load can be handled by the osmotic network or not, which is allowed if $\sum macroservices + \sum microservices \leq \mathcal{L}$. The load helps to check the sustainability of osmotic layer for incoming requests. $\tau_{\mathcal{TH}}$ is set by ODS on the basis of minimum response time. This controls the flow of macro- and micro-services across the network. $\mathcal{E}_{\mathcal{TH}}$ is set by considering the available energy in (5). It is to be ensured that $\max(\mathcal{E}_{\mathcal{R}}^{(macro)}) \leq \mathcal{E}_{\mathcal{A}}^{(\mathcal{K})}$, provided that there are free servers for handling such large requests. In the simplest of form, ODS shifts macro-services to the public/private cloud. But for augmented reality networks, it should first decide on handling these at the near-user layer and perform shifting when no other options are available. The above formulation is subject to linear modeling in the defined system model. The three equations for the energy, memory, and processing time in (3), (5), and (6) can be given as $y_3 = x_3 - c_3$, $y_2 = x_2 - c_2$, and $y_1 = x_1$, respectively. Now, x_3 can be given as $\tau_p\left(\frac{|\mathcal{M}|}{\gamma}\right)x_1$, which gives $y_3 = \left(\frac{\tau_p|\mathcal{M}|}{\gamma}\right)x_1 - c_3$, where τ_p is the processing time of services. By ignoring the constraints of intercepts, $y_2 = x_2 y_1$, which can be rewritten as $y_2 = x_2\left(\frac{|\mathcal{M}|}{\gamma}\right)x_1$. From these, $y_3 = \tau_p\left(\frac{y_2}{x_2}\right)$, which shows that the energy equations can be defined only as the function of memory while fixing the thresholds for size-based osmosis. Thus, size-based osmosis is presentable as a joint optimization problem of the servers with constraints of processing time. Now, if $f(\mathcal{E}_R)$ and $f(\beta_R)$ are the functions for energy and memory, respectively, the common optimization problem for osmotic involvement will be given by:

$$\min\left(\mathcal{G}\left(f(\mathcal{E}_R), f(\beta_R)\right)\right), \tag{10}$$

s.t.

$$\tau_p(j) \leq \tau_{\mathcal{TH}}, \forall j \in \mathcal{M}, \forall \mathcal{K}, \forall \mathcal{N},$$
$$\mathcal{E}_R(j) \leq \mathcal{E}_{\mathcal{TH}}, \forall j \in \mathcal{M}, \forall \mathcal{K}, \forall \mathcal{N},$$
$$\beta_R(j) \leq \beta_{\mathcal{TH}}, \forall j \in \mathcal{M}, \forall \mathcal{K}, \forall \mathcal{N},$$
$$\mathcal{S} \rightarrow \phi \ or \ \max(|\mathcal{S}'|),$$
$$\max(\mathcal{L}_{osmotic}), \forall \mathcal{M}, \forall \mathcal{K}, \forall \mathcal{N}$$
$$\mathcal{L}_{public} = \min(\mathcal{L}). \tag{11}$$

The formal values of variables in (10) are discussed in the theoretical analysis (Section 7) of the proposed approach.

- Bounds: The arrival rate and density affects the service migration policies. Arrival rate is the incoming service requests from the users and density is defined as the grouping of services on a single serving gateway or AP. Both these factors allow classification of services into hardbound and softbound services. The softbound services are handled by the osmotic layers, whereas the hardbounds are transferred to public/private infrastructure. The proposed approach needs to minimize the dependency over a singled osmotic server or layer on the basis of arrival rate of services. Thus, apart from migration, the classification on the basis of bounds should manage the affects of density and arrival rate of services. For high availability of the proposed approach in support of augmented reality, QoS plays a critical role. Thus, if α is the arrival rate, and ω is the density, the model proceeds towards $\min(\omega)\ \forall\mathcal{K}$ and $\min(\alpha)\ \forall\mathcal{K}, \forall\mathcal{N}$. This allows appropriate handling of services by the osmotic servers. However, with these two factors, the possibilities for QoS and their corresponding effects on time, energy and memory are shown in Table 2.
Now, for ith ODS,

$$\omega_i = \frac{\eta_i \gamma_i}{\mathcal{O}_d} \leq \omega_{\mathcal{TH}}, \tag{12}$$

where η_i and γ_i represent the number of applications and number of services per application, respectively, associated with the ith decision system. $\mathcal{O}_d$ represents the total ODSs and $\omega_{\mathcal{TH}}$ represents the common threshold for all ODSs. A network can have all or some of its APs as ODS. For α, there are two possibilities at ODS, one to send towards the osmotic manager and the other towards the public/private infrastructure. This is also subject to availability of resources. In the proposed approach, it is assumed that the local servers maintain a cache for every application which is used frequently. The service requests may arrive at same time, but these are independent, thus, exponential distribution is used to mark the arrival of each application, such that each AP's probability density function is given as:

$$\mathcal{P}_a = \alpha e^{-\alpha\eta_x}, \eta_x = \sum \eta. \tag{13}$$

Now, for applications equaling the number of users $|\mathcal{U}|$, the system can be represented as a 3-tuple, such that $\mathcal{X}_1(\omega_1, \tau_{p,1}, \alpha_1), \mathcal{X}_2(\omega_2, \tau_{p,2}, \alpha_2), \ldots, \mathcal{X}_j(\omega_j, \tau_{p,j}, \alpha_j)$, where $j = |\mathcal{U}|$. The probability of handling the applications by the osmotic layer is driven by negative binomial distribution (referring to handling by public/private as a success), such that

$$\mathcal{P}(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_j) = \sum \binom{j-1}{j'} p^{j'}(1-p)^{j''}, \tag{14}$$

j' is the number of applications allotted to public/private infrastructure, j'' is the remaining applications such that p is the probability of each application having $\omega > \omega_{\mathcal{TH}}$ and $\tau_p > \tau_{\mathcal{TH}}$. If $\mathcal{T}_d$ is the decision time for selecting a particular layer for the incoming applications, then it should be minimized. The decision time is affected by ω. This can be controlled by understanding the upper bounds on the ω for its threshold value. Generally, the threshold changes w.r.t.

classification and configuration of the network. However, by fixing these limits, the approach can continue irrespective of the network variations.

In general, the density of services for all the ODSs follows a normal distribution such that their probability distribution is given as:

$$\mathcal{P}_d(\overline{\omega}, \sigma, \omega) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\omega-\overline{\omega})^2}{2\sigma^2}}, \tag{15}$$

where $\overline{\omega}$ is the mean density of services over ODSs, and σ being the standard deviation. Now, from the definition of likelihood on normal distribution, the maximum likelihood estimators of mean and variance are give as $\hat{\omega} = \frac{1}{\mathcal{O}_d}\sum_{i=1}^{\mathcal{O}_d} \omega_i$ and $\hat{\sigma}^2 = \frac{1}{\mathcal{O}_d}\sum_{i=1}^{\mathcal{O}_d} \left(\omega_i - \hat{\omega}\right)^2$, respectively. Assuming that the confidence level of a server is denoted by ϑ, which is obtained from the probability of number of services with osmotic layers in (14), the approximate confidence interval [29] for ω is given as:

$$-\frac{1}{\sqrt{2\pi}} \int_{\mathcal{H}_{1-\vartheta}}^{\infty} e^{-\frac{(\omega)^2}{2}} d\omega < \frac{\hat{\omega} - \omega_{\mathcal{TH}}}{\hat{\sigma}} < \frac{1}{\sqrt{2\pi}} \int_{\mathcal{H}_{1-\vartheta}}^{\infty} e^{-\frac{(\omega)^2}{2}} d\omega. \tag{16}$$

Thus, the upper bound on ω_{TH} will be given as:

$$\omega_{TH}^{U} \leq \hat{\omega} + \mathcal{H}_{1-\vartheta}\ \hat{\sigma}, \tag{17}$$

where

$$\mathcal{H}_{1-\vartheta} = 1 - \frac{1}{\sqrt{2\pi}} \int_{0}^{1-\vartheta} e^{-\frac{(\omega)^2}{2}} d\omega. \tag{18}$$

Now, considering the first two conditions in Table 2, a joint cost function by using Bayesian inference is required which can be minimized to maintain the flow across osmotic layers. This joint function can be given as:

$$\mathcal{C}_{\alpha,\omega} = \frac{\epsilon_1}{\epsilon_1+\epsilon_2}\mathcal{P}_a + \frac{\epsilon_2}{\epsilon_1+\epsilon_2}\mathcal{P}_d, \tag{19}$$

which is formulated as:

$$\min\left(\mathcal{C}_{\alpha,\omega}\right), \forall\mathcal{K}, \forall\mathcal{N}, \tag{20}$$

s.t.

$$\epsilon_1, \epsilon_2 \geq 0,$$
$$\epsilon_1 + \epsilon_2 \geq \varkappa,$$
$$\omega_{TH} \leq \varkappa \leq \omega_{TH}^{U}. \tag{21}$$

Here, ϵ_1 and ϵ_2 are the Bayesian weights associated with arrival rate and density of services across ODSs, respectively. Note that, $\varkappa$ is s.t. ω, as from Table 2 it is evident that its value affects the system when α is maximum. The above problem is similar to the barrier-penalty method [30]. Thus, the barrier function for the given cost can be given as:

$$\mathbb{C}_{\alpha,\omega}(\zeta, \epsilon) = \frac{\epsilon_1\ \mathcal{P}_a}{\epsilon_1+\epsilon_2} + \frac{\epsilon_2\ \mathcal{P}_d}{\epsilon_1+\epsilon_2} + \zeta\sum_{i=1}^{\mathcal{O}_d}\left(\frac{-1}{-\epsilon_1-\epsilon_2+\varkappa}\right)_i, \tag{22}$$

where ζ is the barrier parameter which drives the cost function barriers. At each moment in the network, $\mathcal{P}_a$ and $\mathcal{P}_d$ attains a constant value. Thus, by taking gradient $\nabla_\epsilon \mathbb{C}_{\alpha,\omega}(\zeta, \epsilon)$ equal to zero for a single system, either $\zeta = 0$ with $\varkappa - \epsilon > 0$, which is achievable at $\varkappa = \omega_{TH}^{U}$ and $\epsilon = \omega_{TH}$, or $\zeta \neq 0$ with $\varkappa = \epsilon$. Thus, considering these, the objective function can be simplified as:

$$\min\left(\frac{1}{2}\left(\mathcal{P}_a + \mathcal{P}_d\right)\right), \varkappa = \epsilon_1 = \epsilon_2, \forall\mathcal{K}, \forall\mathcal{N}, \tag{23}$$

Table 2
Effects on QoS in terms of availability (✓: Affect, ✗: No affect).

Conditions	τ	$\mathcal{E}$	β
max(α), max(ω)	✓	✓	✓
max(α), min(ω)	✓	✗	✗
min(α), min(ω)	✗	✗	✗
min(α), max(ω)	✗	✓	✓

which is further deducible, such that

$$\min\left(\frac{1}{\alpha} + \overline{\omega}\right), \forall\mathcal{K}, \forall\mathcal{N}. \tag{24}$$

6.2. Formation of osmotic game

Once the network is finalized for its components and the servers are installed with the desired capabilities for osmosis, the next step uses the osmotic game. The osmotic game helps to decide how the services will be migrated across the networks and who will initiate the migrations. The osmotic game has two major participants, the first involves the user and the second involves ODS or osmotic server. The main focus of the osmotic game is to migrate services either by following the requirements of a user or burden of the network. The former can be greedy and the later can be optimal. Both these differ in implementation and have their own pros and cons associated with their implementation. Considering these, the osmotic game can be divided into two parts, namely, user-initiated osmosis or edge-initiated osmosis. Both these operate towards the integration of user to edge to the public/private cloud allowing efficient implementation of service migrations. Since both have to be optimally driven and must obey each other rules, Semi-Markov Decision Processes (SMDP) [31] can be one of the optimal strategies for them. SMDP helps to identify the type of osmosis better suited in the network at a given instance and also helps in maintaining a balance of resources in the network.

According to this model, the service requests, from the users, form the system space each having three possible actions: user-initiated, edge-initiated, or hybrid. The system space is formulated into policies which are based on the optimization criteria and helps to decide which the best possible outcome for a particular service is. Let total time for the service being split into τ_p and $\mathcal{T}_d$, and the sum of both denotes the sojourn time τ_{p+d}. Now, by definition [31], this expected waiting time can be given as:

$$\tau_{p+d}(\mathcal{U}, \mathcal{X}) = \lim_{\tau_s\to\infty}\int_{o}^{\tau_s}\left(1 - \mathcal{P}(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_j)\,\mathcal{C}_{\alpha,\omega}(t)\right)dt, \tag{25}$$

where τ_s is the time slots for which the decision is to be taken such that $\tau_{p+d} \leq \tau_s$.

Usually, SMDP is operated by the service vendors and helps in supporting the decision of going for user-initiated or edge-initiated osmosis. Also, SMDP helps the decision metrics on the user and osmotic servers to check the present state for moving the services. This also forms the background for osmotic resource scheduler. Now, the expected osmotic burden $\mathcal{O}_{\mathcal{B}}^{(\mathcal{E})}$ can be derived by using the similar concept of SMDP. At any instance, the number of services on the osmotic layer is given as

$$\Re(t) = |\mathcal{M}|_c(t) + |\mathcal{M}|_i(t) - |\mathcal{M}|_o(t), \tag{26}$$

and service per server load can be calculated as

$$\frac{\Re(t)}{|\mathcal{K}|_u|\mathcal{N}|_u} = \frac{|\mathcal{M}|_c(t) + |\mathcal{M}|_i(t) - |\mathcal{M}|_o(t)}{|\mathcal{K}|_u|\mathcal{N}|_u}, \tag{27}$$

where $|\mathcal{M}|_c$ is the current services, $|\mathcal{M}|_i$ is the incoming services, $|\mathcal{M}|_o$ is the outgoing services. The subscript u denotes the utilized

servers on the osmotic layer. Now, the considering continuous operations of ODS, the osmotic burden can be obtained by integral, i.e.,

$$\int_0^{\tau_s} \frac{\Re(t)}{|\mathcal{K}|_u|\mathcal{N}|_u} dt = \int_0^{\tau_s} \frac{|\mathcal{M}|_c(t) + |\mathcal{M}|_i - |\mathcal{M}|_o(t)}{|\mathcal{K}|_u|\mathcal{N}|_u} dt, \tag{28}$$

such that

$$\mathcal{O}_{\mathcal{B}} = \int_0^{\tau_s} \frac{\Re(t)}{|\mathcal{K}|_u|\mathcal{N}|_u} dt. \tag{29}$$

Now, by definition of SMDP [31],

$$\mathcal{O}_{\mathcal{B}}^{(\mathcal{E})}(\mathcal{M}, \mathcal{K}, \mathcal{N}) = \lim_{\tau_s \to \infty} \int_o^{\tau_s} \left(1 - \mathcal{P}_{\mathcal{B}}^{(\mathcal{E})} \frac{\Re(t)}{|\mathcal{K}|_u|\mathcal{N}|_u}\right) dt, \tag{30}$$

where

$$\mathcal{P}_{\mathcal{B}}^{(\mathcal{E})} = \mathcal{P}(\mathcal{X}_1 \parallel \mathcal{X}_2 \parallel \ldots \parallel \mathcal{X}_j). \tag{31}$$

If the osmotic network handles the maximum load of the network at any given time, then,

$$\max\left(\lim_{\tau_s \to \infty} \int_o^{\tau_s} \left(1 - \mathcal{P}_{\mathcal{B}}^{(\mathcal{E})} \frac{\Re(t)}{|\mathcal{K}|_u|\mathcal{N}|_u}\right) dt\right), \tag{32}$$

which is achievable at minimizing of the negative binomial probability ($\mathcal{P}_{\mathcal{B}}^{(\mathcal{E})}$). The number of osmotic transitions per unit time by following (13) and definition of SMDP, is given as:

$$\mathcal{O}_{\mathcal{S}}^{(t)} = \left(1 - e^{-\alpha\eta_x}\right) - \frac{(1 - \mathcal{P}(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_j))\mathcal{C}_{\alpha,\omega}}{\sum_{\eta \le \eta_x} \tau_{p+d}^{(p)}(1 - \mathcal{P}(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_j))\mathcal{C}_{\alpha,\omega}}, \tag{33}$$

where

$$\tau_{p+d}^{(p)} = \lim_{\tau_s \to \infty} \int_o^{\tau_s} \mathcal{P}(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_j)\, \mathcal{C}_{\alpha,\omega}(t) dt. \tag{34}$$

Now, the probability of error [32] that some services are transferred to public/private cloud even on the satisfaction of osmotic criteria is given as:

$$\begin{aligned} \mathcal{P}_{\mathcal{P}}^{(\mathcal{E})} &= \frac{1}{|\mathcal{K}||\mathcal{N}|} \sum_i \sum_j \frac{(1 - \mathcal{P}(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_j))\mathcal{C}_{\alpha,\omega}}{\sum_i \tau_{p+d}^{(p)}(1 - \mathcal{P}(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_j))\mathcal{C}_{\alpha,\omega}}, \\ &= \frac{1}{|\mathcal{K}||\mathcal{N}|} \sum_i \sum_j \left(1 - \left(e^{-\alpha\eta} + \mathcal{O}_{\mathcal{S}}^{(t)}\right)\right), \\ & j \le \eta_x, i = |\mathcal{M}|. \end{aligned} \tag{35}$$

The network needs to reconfigure itself if the associativity between the osmotic and the public/private cloud servers decreases. This associativity is expressed as the difference between the observed value and expected value of the osmotic burden w.r.t decision time, such that:

$$\mathcal{O}_{\mathcal{B}}^{(\mathcal{A})} = \frac{\Omega}{\mathcal{T}_d} |\mathcal{O}_{\mathcal{B}} - \mathcal{O}_{\mathcal{B}}^{(\mathcal{E})}|, \tag{36}$$

which is governed by the principle that $\mathcal{O}_{\mathcal{B}}^{(\mathcal{A})}(t)$ should be minimum, where Ω is the per service transitions. This can be attained by controlling $\mathcal{P}_{\mathcal{P}}^{(\mathcal{E})}$ irrespective of the number of osmotic shifts. However, in real scenarios, the osmotic shifts will affect the performance; thus, following conditions should be satisfied for the given network:

$$\begin{aligned} &\max\left(\mathcal{P}_{\mathcal{P}}^{(\mathcal{E})}\right), \\ &s.t. \quad \mathcal{O}_{\mathcal{S}}^{(t)} \le \mathcal{O}_{\mathcal{S}}^{(A)}, \end{aligned} \tag{37}$$

where $\mathcal{O}_{\mathcal{S}}^{(A)}$ is the allowed number of osmotic transitions per unit time for each osmotic layer. These are governed by the initially derived conditions of energy, memory and processing time. By using the SMDP approach, the proposed approach uses movement reward functions $\mathfrak{F}_{\mathcal{R}}^{(t)}$ for identifying the state of the network. Whenever, a service request is handled by the osmotic server, the burden of the network is reduced. The load over osmotic server as derived previously can be obtained through queries to every server. However, this will be time as well as resource consuming. Alternatively, $\mathfrak{F}_{\mathcal{R}}^{(t)}$ can lead to exact state, which will help monitoring of servers at any instance. Now, by definition of SMDP [31],

$$\mathfrak{F}_{\mathcal{R}}^{(t+1)} = \mathcal{O}_{\mathcal{B}} + \tau_{p+d} \left(-\mathcal{O}_{\mathcal{B}}^{(\mathcal{A})}\right), t \le \tau_s, \tag{38}$$

where $\mathfrak{F}_{\mathcal{R}}^{(t+1)}$ has to be maximized s.t.

$$\max\left(\frac{1}{|\mathcal{K}||\mathcal{N}|} \sum_i \sum_j \frac{\mathcal{P}(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_j)\mathcal{C}_{\alpha,\omega}}{\sum_i \tau_{p+d}^{(p)} \mathcal{P}(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_j)\mathcal{C}_{\alpha,\omega}}\right), \tag{39}$$

and

$$\max\left(\mathcal{O}_{\mathcal{S}}^{(t)}\right), \tag{40}$$

which has to balance the tradeoff between the rate of incoming requests and the decision time. Thus, average τ_{p+d} should be less to satisfy this condition and it will be driven by the performance of the configured ODSs.

6.2.1. User-initiated osmosis

Irrespective of configurations, user-initiated osmosis are suitable for both intra- as well as inter-osmosis. However, user-initiated osmosis requires additional infrastructure, which can identify in-place requirements of the requests generated by the users. In real applications, this is supported by an Osmotic Client Application (OCA) which is installed on the gateway and connects the users to the nearest ODS. The OCA is responsible for dividing the decision operations of actual decision system. However, the services which are transferred on the osmotic servers by the OCA also require scheduling. OCA, in some cases, can be greedy as it is unaware of the actual conditions of the network. However, such implementations are left on the capacity and capability of osmotic resource scheduler. User-initiated osmosis is supported by a local data warehouse, which caches the frequently accessed information to reduce the routing-gap between the user and the core servers. One of the examples of such an implementation is centralized augmented centers that are used for gaming or virtual reality applications. User-initiated osmosis operates only with computational issues, which are identified by the OCA. This is one of the trivial approaches for implementing osmotic computing as also discussed in Ref. [26]. These steps are illustrated in Algorithm 1. This algorithm checks the initial condition of $\mathcal{C}_{\alpha,\omega}$ and $\mathfrak{F}_{\mathcal{R}}^{(t+1)}$ from (19) and (38). Once satisfied, the algorithm iterates for incoming service requests and check their computational burden and migrates them to the osmotic schedule for allocation to appropriate osmotic layer and osmotic server.

6.2.2. Edge-initiated osmosis

Contrary to user-initiated osmosis, edge-initiated osmosis is less computationally expensive and are most reliable for inter-osmosis. Although, user-initiated osmosis reduces the decisions to be taken for every service, yet the ODSs have to check the flags for identifying whether the incoming requests are directly from the users or OCA. Apart from this, ODSs can directly own the entire process and can know about the exact state of the network. However, latency can be expected for the applications which do not require any intermediate decisions. Edge-initiated osmosis is best suited for stand-alone user applications with multiple

Algorithm 1 User-initiated Osmosis via OCA

Input: Service_Request(β, $\mathcal{E}$, τ_p), $\mathcal{M}$, $\mathcal{S}$
Output: Invoke Scheduler⟵Service_Request(β, $\mathcal{E}$, τ_p), Flag,
Evaluate Eqs. (1)–(5)
Fetch and store β_{TH}, $\mathcal{E}_{TH}$, τ_{TH} from ODS
Set counter=count(service_Requests())
Set user_osmotic_counter=0, i=1
Set thresholds for α and $\overline{\omega}$
Calculate $\mathcal{C}_{\alpha,\omega} = \frac{\epsilon_1}{\epsilon_1+\epsilon_2}\mathcal{P}_a + \frac{\epsilon_2}{\epsilon_1+\epsilon_2}\mathcal{P}_d$
Calculate $\mathfrak{F}_{\mathcal{R}}^{(t+1)}$ and set threshold as $\mathfrak{F}_{\mathcal{R}}^{(t+1)}(TH)$
if (min($\mathcal{C}_{\alpha,\omega}$) = *true* && max($\mathfrak{F}_{\mathcal{R}}^{(t+1)}$) $\geq$ $\mathfrak{F}_{\mathcal{R}}^{(t+1)}(TH)$) **then**
  while ($i \leq counter$) **do**
    Select Service_Request(β, $\mathcal{E}$, τ_p)[i]
    if ($\beta \leq \beta_{TH}$, $\mathcal{E} \leq \mathcal{E}_{TH}$, $\tau_p \leq \tau_{TH}$) **then**
      shift service to ODS with Flag=1
      user_osmotic_counter=user_osmotic_counter+1
    else
      shift service to ODS with Flag=0
    end if
    i=i+1
  end while
else if (Thresholds_variablility=False) **then**
  Reset OCA to edge-initiated mode
else
  Alter thresholds for $\mathfrak{F}_{\mathcal{R}}^{(t+1)}$, α and $\overline{\omega}$
  Reevaluate $\mathcal{C}_{\alpha,\omega}$, $\mathfrak{F}_{\mathcal{R}}^{(t+1)}$
end if
Track logs and continue

requests. Also, this does not require any additional infrastructure and application for osmosis. Edge-initiated osmosis can be applied as a general network layout where a user is unaware of the exact source of information. One of the key examples of edge-initiated osmosis can be the formation of a hybrid sensor beds with on-site data analysis. Unlike the user-initiated osmosis, the edge-initiated osmosis uses more set of constraints for analyzing the incoming requests and deciding on shifting services to osmotic layer or public/private cloud. Algorithm 2 illustrates the steps for edge-initiated osmosis. The algorithm analyzes the constraints on osmotic transitions and probability of error along with resource-constrains similar to user-initiated osmosis.

6.3. Osmotic resource scheduling

Once a decision is taken for the user- or edge-initiated osmosis, the next step involves the scheduling of incoming requests to appropriate servers while maintaining the optimization conditions of the network. The ODSs are themselves responsible for scheduling tasks. However, in much sub-network level, there can be another scheduler that manages the osmotic servers. In such cases, the decision on the osmotic layer is taken by the ODS and the decision on the server is taken by the separate osmotic scheduler. This is the base scenario considered in the proposed approach, and the scheduling is performed as a collaborative task between the ODS and osmotic schedulers. Once the service requests arrive from the end users or OCA, the ODS differentiates between the ones to be sent to public/private cloud and the ones meant for osmotic layers. It then evaluates the load of the each layer and analyzes the excessive service requests it can handle. Once decided, it sends the service requests to schedulers which takes into account the values of β, $\mathcal{E}$, τ_p of the involved servers and calculate the β_A, $\mathcal{E}_A$, τ_{p+d}. If there is no conflict and the availability of servers satisfies the requirements, it migrates

Algorithm 2 Edge-initiated Osmosis

Input: Service_Request(β, $\mathcal{E}$, τ_p), $\mathcal{M}$, $\mathcal{S}$
Output: Invoke Scheduler⟵Service_Request(β, $\mathcal{E}$, τ_p)
Evaluate entire system model
Fetch and store β_{TH}, $\mathcal{E}_{TH}$, τ_{TH} from ODS
Set counter=count(service_Requests())
Set edge_osmotic_counter=0, i=1
Set thresholds for α and $\overline{\omega}$
Calculate $\mathcal{C}_{\alpha,\omega} = \frac{\epsilon_1}{\epsilon_1+\epsilon_2}\mathcal{P}_a + \frac{\epsilon_2}{\epsilon_1+\epsilon_2}\mathcal{P}_d$
Calculate $\mathfrak{F}_{\mathcal{R}}^{(t+1)}$ and set threshold as $\mathfrak{F}_{\mathcal{R}}^{(t+1)}(TH)$
while ($i \leq counter$ && $\mathcal{O}_{\mathcal{S}}^{(t)} \leq \mathcal{O}_{\mathcal{S}}^{(A)}$) **do**
  Select Service_Request(β, $\mathcal{E}$, τ_p)[i]
  if (min($\mathcal{C}_{\alpha,\omega}$) = *True* && max($\mathfrak{F}_{\mathcal{R}}^{(t+1)}$) $\geq$ $\mathfrak{F}_{\mathcal{R}}^{(t+1)}(TH)$) **then**
    if ($\beta \leq \beta_{TH}$, $\mathcal{E} \leq \mathcal{E}_{TH}$, $\tau_p \leq \tau_{TH}$) **then**
      shift service to Osmotic Scheduler
      edge_osmotic_counter=edge_osmotic_counter+1
    else
      shift service to nearest public/private cloud terminal/server
      Trace non-osmotic shifts and evaluate $\mathcal{P}_{\mathcal{P}}^{(\mathcal{E})}$
      if ($\mathcal{P}_{\mathcal{P}}^{(\mathcal{E})}(t) > \mathcal{P}_{\mathcal{P}}^{(\mathcal{E})}(t-1)$) && min($\mathcal{O}_{\mathcal{B}}^{(\mathcal{A})}$) = *True* **then**
        Adjust $\Re(t)$ till $\mathcal{O}_{\mathcal{B}}$== $\mathcal{O}_{\mathcal{B}}^{(\mathcal{E})}$
      else
        Increase $\mathcal{O}_{\mathcal{S}}^{(t)}$ by adjusting α and $\overline{\omega}$ or RESET
      end if
    end if
  else
    Check osmotic feedback and adjust osmotic thresholds
    Reevaluate $\mathcal{C}_{\alpha,\omega}$, $\mathfrak{F}_{\mathcal{R}}^{(t+1)}$, $\mathcal{T}_d$ and continue
  end if
  i=i+1
end while
Track logs and wait for input

the services in the order of highest precedence of computational resources.

Algorithm 3 Osmotic Resource Scheduling-Part A

Input: Service_Request(metric_list), flag, scheduler_info, $\mathcal{K}$, Output (Algorithms 1, 2)
Output: Service allocation and migration
Evaluate entire system model and fetch scheduler count⟶ ctr
Algorithm 2=True
Set osmotic_scheduler_counter=0
iteration_counter=0
Thresholds_ready=True
while (Network_Active=True) **do**
  if (Request_OCA(flag)==1 || Request_User=True) **then**
    Select layer ($\mathcal{K}$) with $\beta \leq \beta_A^{(\mathcal{K})}$ and $\mathcal{E} \leq \mathcal{E}_A^{(\mathcal{K})}$
    [feedback,State]=Scheduler(Service_Request(metric_list)): Algorithm 4
    if (!feedback && State==True) **then**
      Continue
      osmotic_scheduler_counter=osmotic_scheduler_counter+1
      ctr=ctr+1
    else
      Reevaluate request and analyze system for re-osmosis
    end if
  else
    shift service to nearest cloud terminal/server
  end if
  iteration_counter=iteration_counter+1
end while
Maintain logs

Algorithm 4 Osmotic Resource Scheduling-Part B

1: **Input**: Service_Request(metric_list), $\mathcal{N}$, $\mathcal{K}$, Output (Algorithms 1–3)
2: **Output**: Service scheduling and migration
3: counter_s=count(Service_Requests()), j=1
4: Fetch service request details
5: **while** (j $\leq$ counter_s) **do**
6: Identify requirements of β, $\mathcal{E}$, τ_p
7: Select Server($\mathcal{N}$) that satisfies the requirements
8: **if** (Server_Available==True && Resources==Available) **then**
9: Shift the traffic and return [NULL, True]
10: **else if** (Server_Available==True && Resources==Distributed) **then**
11: Wait until a server is available
12: **if** (Resources==Available && $\tau_{p+d} \leq \max(\tau_{p+d})$) **then**
13: Shift the traffic and return [NULL, True]
14: **else if** (Resources==Distributed && $\tau_{p+d} \leq \max(\tau_{p+d})$) **then**
15: Perform intra-osmosis
16: Shuffle services to ensure $\beta \leq \beta_A^{(\mathcal{K})}$ and $\mathcal{E} \leq \mathcal{E}_A^{(\mathcal{K})}$
17: operate until $\mathcal{S}! = \phi$
18: **else**
19: return [exit(-1), False]
20: **end if**
21: **else**
22: return [exit(-1), False]
23: **end if**
24: j=j+1
25: **end while**
26: Maintain logs

The issue arises when the overall evaluations of the osmotic layer show the availability at ODS, but the resources are distributed between the servers such that at any given instance, there is no single server on the osmotic layer that can handle the incoming requests. In such scenario, the schedulers first decide to carry-forward the request or not. This decision is taken on the basis of τ_{p+d}. If the sojourn time will not affect the expected value too much extent, it will use a waiting flag for the service and allocate it to the server once a free slot is available. Further, if sojourn time is high and there are no other layers to handle the requests, the scheduler performs intra-osmosis between its servers for accommodating the incoming requests. This is done by identifying the services with $\min(\tau_p)$ and $\beta \leq \beta_A^{(\mathcal{K})}$ and $\mathcal{E} \leq \mathcal{E}_A^{(\mathcal{K})}$ for any of the existing servers.

On the contrary, the schedulers can opt for back-out strategy and immediately communicate with the ODS about the distributed issue. The ODS then selects the next scheduler and continue until the requests are not handled. At the ODS, it is always crucial to maintaining the condition of $\min(\tau_{p+d})$ for the entire osmotic network visible to it. At any instance, if the optimization criteria are not fulfilled, the ODS can migrate the services to the public/private cloud and then again proceed for osmosis once the conditions are satisfied. For efficient operations, the schedulers should be programmable for operating with all possibilities as explained in the previous paragraph, or only two possibilities of either handling the service or reporting it to the ODS for selecting another scheduler. These steps of selecting the osmotic layer and operations at a scheduler are illustrated in Algorithms 3 and 4. Remember that this algorithm illustrates a single thread between one selected scheduler and its corresponding ODS. The proposed approach is directly applicable to the MARN-service requests coming either from the user application or OCA. The proposed strategy with osmotic scheduling efficiently manages the offloading issues and helps to realize the communication demands of AR-VR applications.

7. Theoretical analysis

This section presents the equilibrium and state analysis w.r.t. the theoretical observation over the defined system model.

Lemma 1. *Processing time τ is application specific and itself gets affected by the memory and energy constraints of the system, and thus, modeling for optimization in (10) is only formulated using memory and energy requirements.*

Proof. Considering the osmotic load, $\mathcal{L}_{osmotic}$, the constraints of maximization are observed either at $\beta_A^{(\mathcal{K})} = \beta_T$ or at $\mathcal{E}_A^{(\mathcal{K})} = \mathcal{E}_T$. Further, from (32), the maximum conditions are dependent only on $\Re$, $\mathcal{K}$ and $\mathcal{N}$, thus, satisfying the Lemma 1. □

Lemma 2. *The total latency observed by an application can only be given as the sojourn time of SMDP formulation on the defined system and it increases with the density of the services while arrival rate influences only in terms of availability.*

Proof. The latency can be calculated by taking a single slot for τ_s. Now, from (25) and (34), the sojourn time τ_{p+d} is dependent on the joint function, which is dependent on the mean values for arrival rate and density of services. Further, τ_{p+d} is the sum of processing and decision time. The decision time depends on $\mathcal{O}_d$, whereas processing time depends on the availability of required memory and energy resources, thus, satisfying the Lemma 2. □

Lemma 3. *For the problem in (10), the joint optimization is attainable as a ratio of energy to the memory, which should be minimized for a defined constant.*

Proof. The joint optimization problem in (10) depends on the definition of the function $\mathcal{G}(.)$. Now, considering the linear model, this function can be given as:

$$\mathcal{G}(f(\mathcal{E}_R), f(\beta_R)) = f(\mathcal{E}_R) - (\ell_1 f(\beta_R) + \ell_2)\,, \tag{41}$$

where ℓ_1 and ℓ_2 defines the energy consumed per memory operation and excessive energy consumed other than intended operations by the servers, respectively. Now, minimizing of $\mathcal{G}(.)$, as in (10) and (41), can be expressed as:

$$f(\mathcal{E}_R) \leq \ell_1 f(\beta_R) + \ell_2, \tag{42}$$

such that if excessive energy is neglected or reduced to 0, the joint optimization problem is solvable at

$$\frac{f(\mathcal{E}_R)}{f(\beta_R)} \leq \ell_1 \text{ for } \ell_2 = 0, \tag{43}$$

which satisfies the Lemma 3 and can be controlled by $\min(\ell_1)$. □

8. Performance evaluation

The proposed approach is evaluated by numerically generating 100,000 requests, equivalent to the number of users, which are transferred either to the osmotic servers or the public/private cloud depending on the resource requirements.[2] A function for web server acted as the ODS with 10 ($|\mathcal{K}|$) instances for large-scale evaluation with 20 servers ($|\mathcal{N}|$) in each osmotic layer. For simplification of evaluations, the number of applications and services are synthetically mapped and set to 1 for every active user and evaluated using Matlab™. Memory (β) of each server is 1TB and memory requirements (b) of each service is greater than

[2] Due to the lack of proper infrastructure for osmotic-based evaluations, only numerical simulations are presented at the moment.

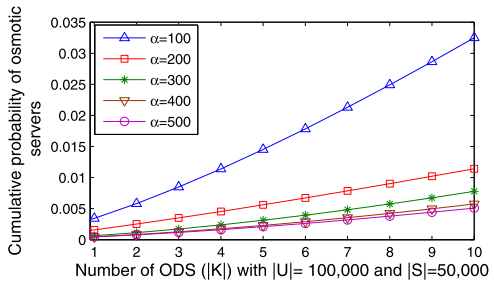


Fig. 4. Cumulative probability of servers in handling incoming services vs. $|\mathcal{K}|$.

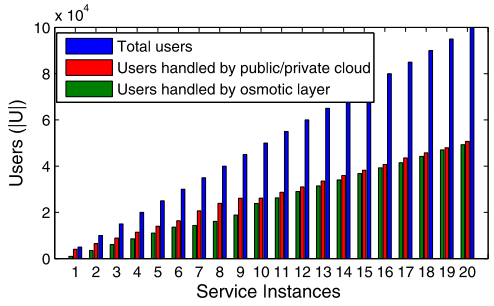


Fig. 5. $|\mathcal{U}|$ handled vs. service instances.

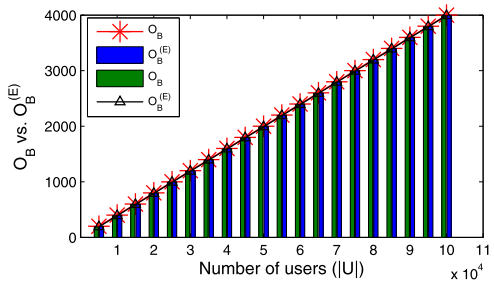


Fig. 6. $\mathcal{O}_{\mathcal{B}}$ and $\mathcal{O}_{\mathcal{B}}^{(\mathcal{E})}$ vs. $|\mathcal{U}|$.

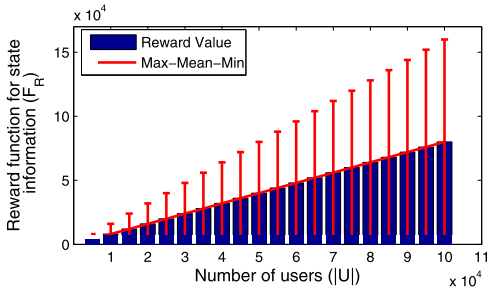


Fig. 7. $\mathfrak{F}_{\mathcal{R}}$ vs. $|\mathcal{U}|$.

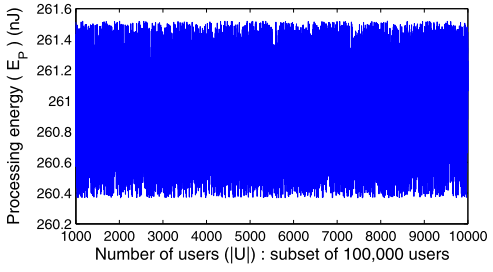


Fig. 8. $\mathcal{E}_p$ vs. $|\mathcal{U}|$.

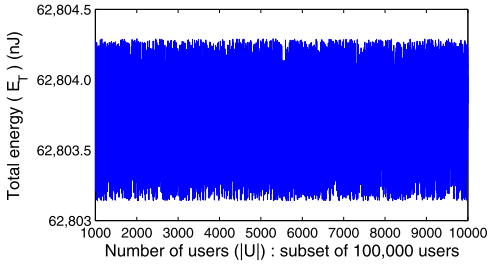


Fig. 9. $\mathcal{E}_T$ vs. $|\mathcal{U}|$.

4096 bytes. $\mathcal{E}_p$ is between 3.14b and 6.18b nJ following per byte energy consumption from [33], τ_p for each service is between 5 and 40 ms per computation, γ at 4 with thresholds $\mathcal{E}_{\mathcal{TH}}$, $\beta_{\mathcal{TH}}$, and $\tau_{\mathcal{TH}}$ at 3389.07 nJ, 1284.9 bytes per request, and 50^5 ms, respectively with number of total computations equaling 10^5. The thresholds are set by following the mean of the previous runs of the proposed approach until time t at which next calculations are performed. $\mathcal{P}_\alpha$ and $\mathcal{P}_d$ of AR-VR services in MARN are of the order 9.643749E−22 and 0.000840726, respectively. $\mathcal{O}_{\mathcal{S}}$ for all the users despite variation in the number of ODS is 0.9 with $\mathcal{P}_{\mathcal{P}}^{(\mathcal{E})}$ attaining a minimum value of 0.1 and τ_{p+d} attaining a constant value of 10 ms.

The arrival rate impacts the performance of MARN as the excessive services arriving at ODS require much processing. With an increasing value of α, the cumulative probability of osmotic servers in handling these services decreases, however, this decrease can be lowered by the increasing number of osmotic layers and servers as shown in Fig. 4. At present, the observation is done by varying the number of ODS up to 10, which surely at higher values will give better cumulative probability but at the expense of high cost. The plot illustrates results for 100,000 requests out of which 50,000 are actively running applications at the same instance. The proposed approach migrates incoming services and schedules them across the osmotic layer by following the size and bounds on service classification. The proposed approach shifts a maximum of 81.02% and a minimum of 50.50% users to osmotic servers, thus, supporting near user evaluation with low latency. Fig. 5 presents the difference in the number of users handled by the osmotic layer, public layer and the total users demanding services by a single application for 20 service instances. The number of users handled by osmotic layer is bounded by the threshold limits and the probability of error while allocating the services. These are controlled by predicting the osmotic burden of the entire network. The results in Fig. 6 suggest that the proposed approach provided efficient estimation of the expected osmotic burden ($\mathcal{O}_{\mathcal{B}}^{(\mathcal{E})}$) with a difference of ± 10 w.r.t. ($\mathcal{O}_{\mathcal{B}}$).

Further, these predictions are subjected to the calculation of reward function ($\mathfrak{F}_{\mathcal{R}}$), which helps in identifying the state of the network. $\mathfrak{F}_{\mathcal{R}}$ should be maximized for the incoming requests from the users as it allows optimal allocation and time-constraint allocation by the schedulers. The results in Fig. 7 confirm this increase and demonstrates the effective output of the proposed solution. As it is difficult to present a common graph for all the users, a subset of 10,000 user-requests is taken to highlight the amount of energy consumed, total energy available, and memory utilized by these users during operations as shown in Figs. 8–10, respectively. These results suggest that for 10,000 users, the proposed approach balances load by conserving 65.13% of the actual resources which otherwise would have been wasted if these users are handled by public/private cloud layers. Fig. 11 presents the details of the number of active ODS for different instances configured in simulations. It demonstrates that the

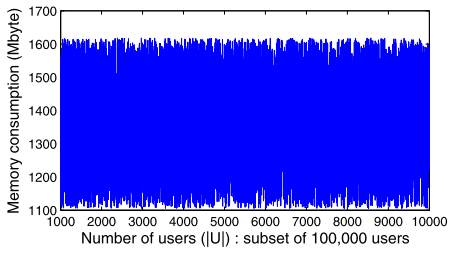


Fig. 10. Memory consumption vs. $|\mathcal{U}|$.

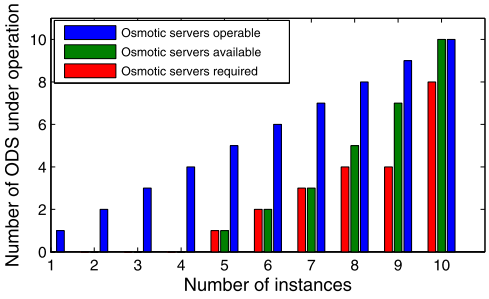


Fig. 11. ODS active vs. instances of ODS.

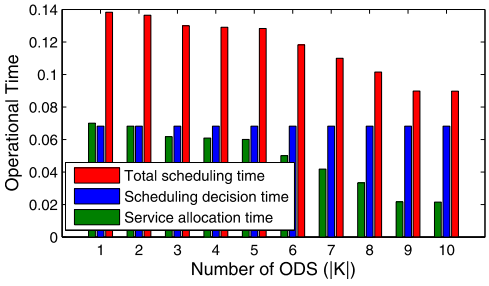


Fig. 12. Scheduling time vs. $|\mathcal{K}|$. Total scheduling time is evaluated as the sum of the scheduling decision time and the service allocation time.

proposed approach ensures that the load is divided such that the required number of ODS is always available during operations. The proposed scheduling procedures operate with a decision time of 0.682 ms irrespective of the number of applications and the users. The decision time is dependent on the number of ODS, however, at the considered number of ODS, this time remains constant as shown in Fig. 12. Further, this figure shows that the service allocation time and total scheduling time decrease with an increase in the number of ODS as alternative servers are available for shifting services across the osmotic layers. All these results suggest the high significance of the proposed osmotic computing-based migration and scheduling in managing services for AR-VR applications in MARN.

Further, the optimization results of the proposed approach are illustrated in Figs. 13–16. The results in Fig. 13 and 14 show the values of $\mathcal{C}_{\alpha,\omega}$ for varying number of ODS and its minimum value required to sustain the operations during migration and scheduling, respectively. It is noticeable that the values for the joint function are always lower than the limits fixed during the operations. The associativity ($\mathcal{O}_{\mathcal{B}}^{(\mathcal{A})}$) of the proposed approach increases with an increase in the number of uses, but its value remains constant for any number of ODS at a particular set of incoming requests. Further, Fig. 16 shows the density of services handled by the osmotic servers with an increasing number of ODS for an increasing number of users. The density is affected by the number of transitions made to the osmotic server by the

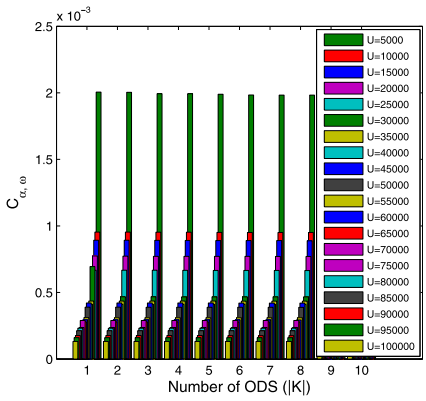


Fig. 13. $\mathcal{C}_{\alpha,\omega}$ vs. $|\mathcal{K}|$.

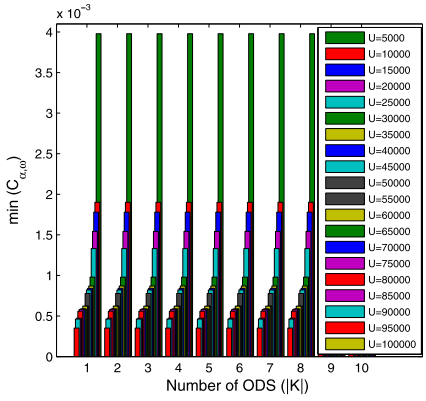


Fig. 14. $\min(\mathcal{C}_{\alpha,\omega})$ vs. $|\mathcal{K}|$.

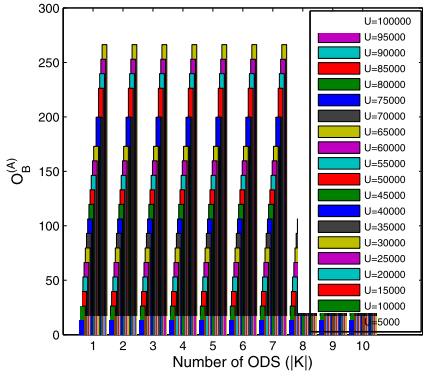


Fig. 15. $\mathcal{O}_{\mathcal{B}}^{(\mathcal{A})}$ vs. $|\mathcal{K}|$.

migration and scheduling algorithms, and it shows an increasing trend for an increasing number of servers and user requests. The final aim of the proposed approach is to conserve resources in terms of energy and memory while balancing the load according to the requirements of an application. These results are presented in Fig. 17 and show that the proposed approach is able to conserve 55.72% of resources in the considered MARN with an average load of 44.27% on the public/private infrastructure. However, the resources conserved follow a decreasing trend as the available resources go on decreasing with the more incoming services. This issue can be resolved by finding an optimal solution to the number of osmotic layers required in a network along with the number of servers in each layer. This optimization problem

Table 3
An overview of the existing solutions that can be enhanced to attain the application-specific requirements of osmotic computing as discussed in this article (✓: supports, ✗: no support, *: partially).

Approach	Ideology	Osmotic Transitions	Service Classifications	Service Migrations in osmotic setup	Resource Scheduling in osmotic setup	Service Aggregation and Distribution	General load/service migrations	General Scheduling
[25]	Computational offloading in pervasive online social networks	✓	✓	✓	✗	✗	✓	✗
[34]	Energy-aware virtual machine migration in cloud setup	✗	*	✗	✗	*	✓	✓
[35]	Service migration from cloud to fog nodes	✗	*	✗	✗	*	✓	✗
[36]	Cloud-IoT hybrid architecture	✗	*	✗	✗	*	✓	✓
[37]	Fault tolerance aware scheduling in cloud setup	✗	✗	✗	✗	✗	✓	✓
[38]	Layered virtual machine migration in cloud setup	✗	✗	✗	✗	✗	✓	✗
Proposed	Service migration and resource scheduling in MARN	✓	✓	✓	✓	*	✓	✓

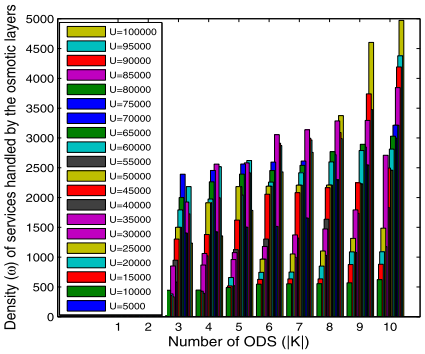


Fig. 16. ω vs. $|\mathcal{K}|$.

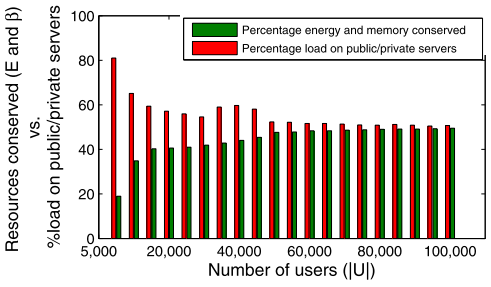


Fig. 17. Resources conserved and load transferred to public/private infrastructure vs. $|\mathcal{U}|$.

is beyond the scope of this paper and will form the part of our future reports.

Although, there are not so tightly related works available, which looked on the services from the macro and micro point of view and used a similar level of modeling for transitions. However, considering the efficiency and adaptability of existing methods, certain solutions are summarized in Table 3, which can be applied for performing service migrations using osmotic configurations. These approaches use a cloud or cloud-like setup and can be considered to show their behavior on the same level and system as discussed in this paper. However, comparisons from a simulation or experimentation point of view are skipped at this point of the study.

From the analysis presented in this paper, it is evident that the proposed approach efficiently handles the service requests in MARN and performs the efficient migration and scheduling for users with diversified requests. The results show that the number of users affects the outputs on the minute scale making the proposed solution robust despite the increase in the number of incoming requests. The major affecting factors include the deployment strategy for osmotic layers and osmotic servers, available resources and location of decision making (user or edge). Thus, the proposed approach can be employed for the applications that can be cached on to the local servers that otherwise consume much of the network resources and cause excessive network burden with avoidable computations.

9. Conclusion

Future networks will focus on applications which demand high time availability and maximum support from the request handling servers at zero-latency and high tolerance. One of these networks is MARN that primarily operates AR-VR applications. These applications are tedious to manage as these may require support from edge-infrastructure as well as the public/private cloud. In this paper, an efficient solution was proposed to handle this migration between the servers of the different layers by following the principles of osmotic computing. The proposed approach supported the scheduling of services to the appropriate osmotic servers that help in conserving a lot of available resources. The results demonstrate significant improvements in term of load balanced across the osmotic and public/private servers. The proposed approach showed remarkable performance with the probability of the error being 0.1 at 55.72% conservation of the energy and memory resources for the entire network despite the increasing number of users. The proposed approach also satisfied the conditions of the joint optimization functions presented in the system model and showed that the system holds true even with varying users, thus, proving its robustness and tolerance against the number of users. The percentage load handled by the osmotic servers varied between 50.50% and 81.02%, whereas for the public/private servers the load to be handled was between 18.98% and 49.50%.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Z. Tang, X. Zhou, F. Zhang, W. Jia, W. Zhao, Migration modeling and learning algorithms for containers in fog computing, IEEE Trans. Serv. Comput. (2018) 1.

[2] A. Prasad, A. Benjebbour, O. Bulakci, K.I. Pedersen, N.K. Pratas, M. Mezzavilla, Agile radio resource management techniques for 5G new radio, IEEE Commun. Mag. 55 (6) (2017) 62–63.

[3] Y. Desmouceaux, P. Pfister, J. Tollet, M. Townsley, T. Clausen, SRLB: The power of choices in load balancing with segment routing, in: ICDCS, IEEE, 2017, pp. 2011–2016.

[4] E. Bastug, M. Bennis, M. Médard, M. Debbah, Toward interconnected virtual reality: Opportunities, challenges, and enablers, IEEE Commun. Mag. 55 (6) (2017) 110–117.

[5] J. Orlosky, K. Kiyokawa, H. Takemura, Virtual and augmented reality on the 5G highway, J. Inf. Process. 25 (2017) 133–141.

[6] G. Paschos, E. Bastug, I. Land, G. Caire, M. Debbah, Wireless caching: Technical misconceptions and business barriers, IEEE Commun. Mag. 54 (8) (2016) 16–22.

[7] B. Tang, Z. Chen, G. Hefferman, S. Pei, T. Wei, H. He, Q. Yang, Incorporating intelligence in fog computing for big data analysis in smart cities, IEEE Trans. Ind. Inf. 13 (5) (2017) 2140–2150.

[8] Y. Wang, P. Li, L. Jiao, Z. Su, N. Cheng, X.S. Shen, P. Zhang, A data-driven architecture for personalized QoE management in 5G wireless networks, IEEE Wirel. Commun. 24 (1) (2017) 102–110.

[9] M. Satyanarayanan, P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, W. Hu, B. Amos, Edge analytics in the internet of things, IEEE Pervasive Comput. 14 (2) (2015) 24–31.

[10] W. Chen, D. Wang, K. Li, Multi-user multi-task computation offloading in green mobile edge cloud computing, IEEE Trans. Serv. Comput. (2018) 1.

[11] O. Skarlat, S. Schulte, M. Borkowski, P. Leitner, Resource provisioning for IoT services in the fog, in: 9th International Conference on Service-Oriented Computing and Applications (SOCA), IEEE, 2016, pp. 32–39.

[12] Z. Sanaei, S. Abolfazli, A. Gani, R. Buyya, Heterogeneity in mobile cloud computing: taxonomy and open challenges, IEEE Commun. Surv. Tutor. 16 (1) (2014) 369–392.

[13] S. Tang, B. Lee, B. He, Fair resource allocation for data-intensive computing in the cloud, IEEE Trans. Serv. Comput. 11 (1) (2018) 20–33.

[14] F. Bonomi, R. Milito, P. Natarajan, J. Zhu, Fog computing: A platform for internet of things and analytics, in: Big Data and Internet of Things: A Roadmap for Smart Environments, Springer, 2014, pp. 169–186.

[15] S.-Y. Lien, S.-C. Hung, H. Hsu, K.-C. Chen, Collaborative radio access of heterogeneous cloud radio access networks and edge computing networks, in: International Conference on Communications Workshops (ICC), IEEE, 2016, pp. 193–199.

[16] P. Corcoran, S.K. Datta, Mobile-edge computing and the internet of things for consumers: Extending cloud computing and services to the edge of the network, IEEE Consum. Electron. Mag. 5 (4) (2016) 73–74.

[17] T. Braud, F.H. Bijarbooneh, D. Chatzopoulos, P. Hui, Future networking challenges: The case of mobile augmented reality, in: 37th International Conference on Distributed Computing Systems (ICDCS), IEEE, 2017, pp. 1796–1807.

[18] M. Chen, Y. Qian, Y. Hao, Y. Li, J. Song, Data-driven computing and caching in 5G networks: Architecture and delay analysis, IEEE Wirel. Commun. 25 (1) (2018) 70–75.

[19] M. Erol-Kantarci, S. Sukhmani, Caching and computing at the edge for mobile augmented reality and virtual reality (AR/vr) in 5G, in: Ad Hoc Networks, Springer, 2018, pp. 169–177.

[20] S. Xue, L. Hu, W. Guanghui, Z. Yaoming, Training effectiveness evaluation of helicopter emergency relief based on virtual simulation, Chin. J. Aeronaut. 31 (10) (2018) 2000–2012.

[21] C. Wang, L. Fang, Y. Dai, A simulation environment for SCADA security analysis and assessment, in: 2010 International Conference on Measuring Technology and Mechatronics Automation, vol. 1, IEEE, 2010, pp. 342–347.

[22] M. Carvalho Marques Neto, R. Kulesza, T. Rodrigues, F.A. Machado, C.A. Santos, A tool to simulate the transmission, reception, and execution of interactive TV applications, Int. Sch. Res. Not. 2017 (2017).

[23] M. Villari, A. Celesti, M. Fazio, Towards osmotic computing: Looking at basic principles and technologies, in: Conference on Complex, Intelligent, and Software Intensive Systems, Springer, 2017, pp. 906–915.

[24] M. Villari, M. Fazio, S. Dustdar, O. Rana, R. Ranjan, Osmotic computing: A new paradigm for edge/cloud integration, IEEE Cloud Comput. 3 (6) (2016) 76–83.

[25] V. Sharma, I. You, R. Kumar, P. Kim, Computational offloading for efficient trust management in pervasive online social networks using osmotic computing, IEEE Access 5 (2017) 5084–5103.

[26] V. Sharma, K. Srinivasan, D.N.K. Jayakody, O. Rana, R. Kumar, Managing service-heterogeneity using osmotic computing, 2017, arXiv preprint arXiv:1704.04213.

[27] M. Nardelli, S. Nastic, S. Dustdar, M. Villari, R. Ranjan, Osmotic flow: Osmotic computing+ IoT workflow, IEEE Cloud Comput. 4 (2) (2017) 68–75.

[28] M. Villari, M. Fazio, S. Dustdar, O. Rana, L. Chen, R. Ranjan, Software defined membrane: Policy-driven edge and internet of things security, IEEE Cloud Comput. 4 (4) (2017) 92–99.

[29] W.B. Nelson, Applied Life Data Analysis, vol. 577, John Wiley & Sons, 2005.

[30] A. Auslender, R. Cominetti, M. Haddou, Asymptotic analysis for penalty and barrier methods in convex and linear programming, Math. Oper. Res. 22 (1) (1997) 43–62.

[31] S.A. Lippman, Semi-Markov decision processes with unbounded rewards, Manage. Sci. 19 (7) (1973) 717–731.

[32] G.L. Stüber, Principles of Mobile Communication, vol. 2, Springer, 2001.

[33] V. Sivaraman, A. Vishwanath, Z. Zhao, C. Russell, Profiling per-packet and per-byte energy consumption in the netfpga gigabit router, in: Conference on Computer Communications Workshops, IEEE, 2011, pp. 331–336.

[34] N.J. Kansal, I. Chana, Energy-aware virtual machine migration for cloud computing-a firefly optimization approach, J. Grid Comput. 14 (2) (2016) 327–345.

[35] D. Rosário, M. Schimuneck, J. Camargo, J. Nobre, C. Both, J. Rochol, M. Gerla, Service migration from cloud to multi-tier fog nodes for multimedia dissemination with QoE support, Sensors 18 (2) (2018) 329.

[36] M. Elhoseny, A. Abdelaziz, A.S. Salama, A.M. Riad, K. Muhammad, A.K. Sangaiah, A hybrid model of internet of things and cloud computing to manage big data in health services applications, Future Gener. Comput. Syst. 86 (2018) 1383–1394.

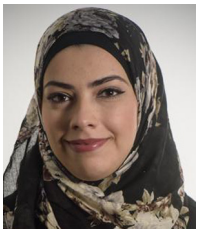
[37] S.M. Abdulhamid, M.S.A. Latiff, S.H.H. Madni, M. Abdullahi, Fault tolerance aware scheduling technique for cloud computing environment using dynamic clustering algorithm, Neural Comput. Appl. 29 (1) (2018) 279–293.

[38] X. Fu, J. Chen, S. Deng, J. Wang, L. Zhang, Layered virtual machine migration algorithm for network resource balancing in cloud computing, Front. Comput. Sci. 12 (1) (2018) 75–85.

Vishal Sharma received the Ph.D. and B.Tech. degrees in computer science and engineering from Thapar University (2016) and Punjab Technical University (2012), respectively. He worked at Thapar University as a Lecturer from Apr'16–Oct'16. From Nov. 2016 to Sept. 2017, he was a joint post-doctoral researcher in MobiSec Lab. at Department of Information Security Engineering, Soonchunhyang University, and Soongsil University, Republic of Korea. Dr. Sharma is now a Research Assistant Professor in the Department of Information Security Engineering, Soonchunhyang University, The Republic of Korea. Dr. Sharma received three best paper awards from the IEEE International Conference on Communication, Management and Information Technology (ICCMIT), Warsaw, Poland in April 2017; from CISC-S'17 South Korea in June 2017; and from IoTaas Taiwan in September 2017. He is the member of IEEE, a professional member of ACM and past Chair for ACM Student Chapter-Patiala. He has authored/coauthored more than 90 journal/conference articles and bookchapters. He served as the program committee member for the Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA). He was the track chair of MobiSec'16 and AIMS-FSS'16, and PC member and reviewer of MIST'16 and MIST'17, respectivelyHe served as a TPC member of ITNAC-'17, ICCMIT'18, CoCoNet'18, and ITNAC'18. He is serving as a TPC member of WiMo'19, ETIC'19, and ITNAC'19. He is the Associate Technical Editor of IEEE Communications Magazine, and Associate Editor of IET-CAAI Transactions on Intelligence Technology. Also, he serves as a reviewer for various IEEE Transactions and other journals. His areas of research and interests are 5G networks, UAVs, blockchain systems, estimation theory, and artificial intelligence.

Dushantha Nalin K. Jayakody received the Ph. D. degree in Electronics, Electrical, and Communications Engineering in 2014, from the University College Dublin, Ireland. He received his MSc degree in Electronics and Communications Engineering from the Department of Electrical and Electronics Engineering, Eastern Mediterranean University, Turkey (under the University full graduate scholarship) and ranked as the first merit position holder of the department, and B. E. electronics engineering degree (with first-class honors) from Pakistan and was ranked as the merit position holder of the University (under SAARC Scholarship.). From 2014–2016, he was a Postdoc Research Fellow at the Institute of computer science, University of Tartu, Estonia and Department of Informatics, University of Bergen, Norway. From summer 2016, he is a Professor at the School of Computer Science & Robotics, National Research Tomsk Polytechnic University, Russia, where he also serves as the Director of Tomsk Infocomm Lab. Dr Jayakody has received the best paper award from the IEEE International Conference on Communication,

Management and Information Technology (ICCMIT) in 2017. Dr. Jayakody has published over 80 international peer reviewed journal and conference papers. His research interests include PHY layer prospective of 5G communications, Cooperative wireless communications, device to device communications, LDPC codes, Unmanned Ariel Vehicle etc. Dr. Jayakody is a Member of IEEE and he has served as workshop chair, session chair or technical program committee member for various international conferences, such as IEEE PIMRC 2013/2014, IEEE WCNC 2014–2018, IEEE VTC 2015–2018 etc. He currently serves as a Area Editor the Elsevier Physical Communications Journal, MDPI Information journal and Wiley Internet of Technology Letters. Also, he serves as a reviewer for various IEEE Transactions and other journals.



Marwa Qaraqe received her Ph.D. and Master of Science degree in Electrical Engineering from Texas A&M University in College Station, Texas, USA in May of 2016 and August of 2012, respectively. She graduated Summa Cum Laude from Texas A&M University at Qatar in May of 2010. Currently, Dr. Qaraqe is an Assistant Professor at Hamad bin Khalifa University in Doha, Qatar. Dr. Qaraqe's current research interests lie in the area of early seizure onset detection using EEG and ECG signals. In particular, she exploits various signal processing and machine learning algorithms to detect the earliest signs of electrographic epileptic seizures.