Software-defined Internet of Multimedia Things: Energy-efficient and Load-balanced Resource Management

Ahmadreza Montazerolghaem

Software-defined Internet of Multimedia Things: Energy-efficient and Load-balanced Resource Management

Ahmadreza Montazerolghaem

Abstract—Internet of Multimedia Things (IoMT) is becoming attractive day by day and provides more services to the Internet users. The ever-increasing multimedia applications and services led to an outburst in IoMT. The multimedia things connected to IoMT are also increasing; millions of devices and high volume of traffic. This large traffic is directed toward the servers of the service provider in the IoMT cloud through the network switches. As a result, the IoMT infrastructure is facing the crisis of the resource management of switches and servers from two aspects: load imbalance and energy loss. This paper proves that the problem of optimal resource management of IoMT networks with the energy and load constraints simultaneously is an NP-hard problem that has a high time complexity. The problem has decomposed. Then, we propose a modular system of energy and load control in IoMT using the concepts of network softwarization and virtual resources. The proposed controller first dynamically adjusts the resources through accurate determination of the IoMT network size. It then distributes the load between the IoMT servers as well as routes the traffic between switches to the desired server. The Open vSwitch, Floodlight Controller, and Kaa Servers are respectively used for implementing the switches, controller, and servers of IoMT in the test platform. The results show that the proposed system both minimizes the number of IoMT active servers and switches and distributes the load between them. As a result, the parameters for evaluating the quality of service and quality of experience, such as throughput, multimedia delay, R factor, and MOS improved.

Index Terms—Green resource allocation, Load balancing, Software-defined networking (SDN), Internet of multimedia things, Network function virtualization (NFV), Two-tier cloud computing.

1 Introduction

Multimedia is rising sharply in IoT. Thus, extensive research has focused on the use and development of multimedia-based services and applications in IoT. IoMT is an extension to IoT, which aims to provide multimedia streaming as a part of the IoT realization [1].

Recent developments in the low-cost and low-scale design of equipment led to a significant increase in the number of things that can connect to the Internet. More than 10 billion things are connected to the Internet, which connect more than 3 billion individuals across the world. In addition, the last researches on predicting global Internet traffic predict a significant rise in multimedia traffic within the next few years. The real-time multimedia services, solutions, and applications such as videoconferencing, telecommunications, real-time content delivery, security/surveillance systems in smart homes, remote monitoring in smart hospitals, smart multimedia healthcare systems in smart cities, online games and many others resulted in a remarkable increase in the multimedia traffic in IoMT [2], [3].

The multimedia data, such as video, audio, which is obtained from processing of the physical environment, has distinctive characteristics compared to the scalar data achieved by conventional IoT equipment. For this reason, multimedia servers require huge storage and processing resources to process multimedia information. In pure IoT, physical environment information is obtained using wireless sensors and can include statistics about brightness, pressure, temperature, among others. The nature of this sensed data is cyclic and requires low computational and storage resources. Therefore, these operations can also be performed by low processing servers. In contrast, multimedia data in IoMT requires servers with massive storage and processing resources for real-time communication [4]–[7].

Also, in IoMT, multimedia must be sent with an acceptable quality of service so that it can provide good user experiences. It means that both network Quality of Service (QoS) and Quality of Experience (QoE) of the end-user must be met. This requires efficient network-aware communication mechanisms [8].

In this regard, multimedia cloud servers were developed to process the IoMT traffic and provide various multimedia services (Fig. 1). The goal of multimedia cloud is to provide a platform for storing data and strong distributed processing services for the end-user of IoMT in order to offer comprehensive access to the multimedia content.

The multimedia communication in wireless networks has been examined in many previous articles [1]–[3], [8], but they only focus on some particular applications, concepts, and equipment. However, none of these studies focused on IoMT cloud architecture. The IoMT cloud architecture is facing challenges, such as poor quality of service due to non-distribution of heavy multimedia traffic as well as the lack of integrated resource management of multimedia servers, which needs further investigation. Hence, there is a need for an approach to develop a new systematic architecture for the IoMT cloud, which focuses on the aforementioned problems.

- *A. Montazerolghaem is with the Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran. E-mail: a.montazerolghaem@comp.ui.ac.ir*

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/JIOT.2021.3095237, IEEE Internet of Things Journal

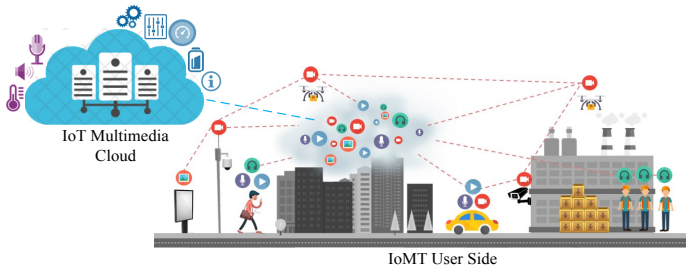


Figure 1. Schematic of IoMT in which a large volume of multimedia traffic of IoMT applications is collected and sent to the IoMT cloud servers through the network switches for storage, processing, and service.

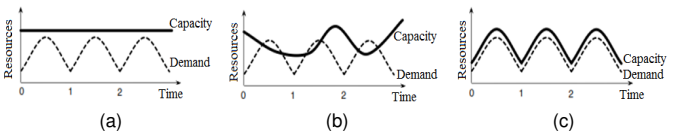


Figure 2. a) Fixed IoMT network capacity b) Variable IoMT network capacity but unaware of demand (load) c) Variable IoMT network capacity and aware of demand (NFV-based approach).

The main purpose of this article is to analyze, design, and implement a novel IoMT cloud architecture, which can manage resources (servers and switches of the IoT multimedia cloud) through determining the proper size of the network and load distribution in it. This architecture provides high-quality multimedia-based services for users. On the other hand, it reduces energy consumption for the providers of IoMT cloud service because a smart network sizing prevents energy loss.

In this respect, we employ the Software-defined Networking (SDN), which can centrally provide a global view of all the network resources, which results in integrated management and a balanced load. We also apply the Network Function Virtualization (NFV) to virtualize the network resources. These two concepts, which have recently become very popular, can create an abstract view of the IoMT network resources that can be controlled using software, APIs (like OpenFlow protocol) as well as a logically centralized controller. As a result, the network resources are allocated based on demand. It also brings improved QoS and optimized energy usage.

To be more precise, the SDN allows advanced management by separating the data plane from the control plane. The data plane includes simple forwarding devices such as OpenFlow switches, and the control plane includes controller and various network applications like routing, firewall, load balancing, monitoring, etc. The OpenFlow protocol connects these two planes using a set of messages. The collection of statistics or installation of rules is performed using messages such as `Features-request/reply`, `Packet-In`, and `Flow-mod`. Statistics can be related to the network topology, bandwidth, link delays, flow types, etc.

The NFV modifies the network size by the optimal allocation of IoMT virtual functions on demand. For example, at peak times when overload is most likely to occur, one can activate the inactive resources of the IoMT network, including servers, and allocate to them the virtual functions required by the network. This increases the capacity and size of the network and expands its boundary. In contrast, at the times of load reduction, idle network equipment can be taken out of the circuit for energy savings. This reduces the network size and limits its boundary. Thus, the capacity and, consequently, the boundary of the network can be consciously changed according to demand (Fig. 2).

A Virtualized Network Function (VNF) may consist of one or more Virtual Machines (VMs) that execute the network-specific function (like IoMT servers) on the Physical Machines (PMs). Therefore, there is no need for special-purpose hardware equipment for having the IoMT network functions. The instances of VNFs can be dynamically created and used or can be migrated. They can also be removed based on the network conditions. It should be noted that the VM live migration provides the ability to switch the mapping between VNFs and PMs without interruption while services are running. The main contributions of this paper can be summarized:

- Proving that the resource allocation problem of the IoMT network with energy and load constraints is NP-hard,
- Breaking down the above problem and designing a novel Software-defined Internet of Multimedia Things framework for the IoMT energy and load aware resource management,
- Designing of Energy-efficient and Load-balanced Resource Controller (ELRC),
- Proposing predictive heuristic algorithms based on the time-series analysis and the fuzzy logic to determine the boundary of the IoMT network and load balance,
- Implementation and design of various scenarios to assess the performance of the proposed framework in a real test platform with both QoS and QoE evaluation criteria.

2 Related Works

[1] starts by providing a horizontal overview of the IoMT. The authors discuss the issues considering the characteristics of multimedia and provide a summary of related IoMT architectures. However, the current research and development activities in the IoMT do not mandate the features of multimedia objects, thus leaving a gap to benefit from multimedia content based services and applications. [2] analyzes this issue by contemplating the concept of IoT and drawing an inspiration towards the perspective vision of IoMT. In [3], the authors propose a seamless and authorized multimedia streaming framework (SAMS) for a cluster-based hierarchical WMSN. In [4], the authors propose a new low-overhead High Efficiency Video Coding (HEVC) encryption scheme for energy constrained IoMT. A multilayer framework based on a multilevel edge computing architecture to manage end and edge devices is proposed in [5]. In [6], a low-cost and confidentiality-preserving multi-image compressed acquisition model in IoMT is proposed and separate image reconstruction is provided. A massive real time data distributed cluster storage for IoMT is proposed in [7]. In [8], the authors survey QoS/QoE provisioning schemes in IoMT. They discuss the basic concepts of multimedia communication, QoS and QoE.

While much prior investigation has proposed the potential gains of applying SDN in IoT communication networks in

order to facilitate network management [9]–[14], there have only been few papers about the practical mechanisms of applying SDN in IoMT. Therefore, the exploration of such an approach is crucial and timely, especially considering the fast growth of IoMT applications and the advent of SDN.

The rest of this article is organized as follows. Sect. 3 introduces the IoMT system model and shows that simultaneous satisfaction of the resource, energy, and load constraints has the NP-hard complexity. The proposed framework, controller, and algorithms are detailed in Sect. 4. Sect. 5 explains the implementation process, provides the results and evaluates and analyzes the performance of the proposed approach. Finally, Sect. 6 is the conclusion and future works.

3 System model and problem formulation

In general, the IoMT cloud is an n-element set of multimedia servers (set R) along with an m-element set of switches (set S) with limited processing, storage, and hard disc resources. Each server or switch consumes its resources to provide services to the IoMT users. The square matrix $\mathbb{C}^{ij}$ indicates the number of requests from server/switch i to server j. We assume that the optimal number of admitted requests is C^{ij}, in such a way that $C^{ij} \leq \mathbb{C}^{ij}$. Also, $\mathbb{D}^{ij}$ indicates the number of requests passing from switch i to switch j, and its optimal value is D^{ij}. Furthermore, A^{ij}_{kv} represents the number of admitted requests from origin i to destination j passing from two adjacent switches k and v ($A^{ij}_{kv} \leq D^{ij}$). Thus, the total number of admitted requests from i to j is specified by C^{ij} so that A^{ij}_{kv} indicates how it is distributed among the existing paths between i and j. Each server or switch depends on its remaining resources, namely CPU, memory, and hard disc, to perform its operation. The optimal CPU, memory, and hard disc usage are P_l, M_l, and H_l, respectively and E_l indicates power usage (it is specified with index ϕ for servers and θ for switches).

In the following, we present the mathematical model for the optimal management of the IoMT network resource management problem with constraints of resource, energy, and load.

$$\textbf{\textit{maximize}} \quad \xi\left(\frac{\sum_{i=1}^{n}\sum_{j=1}^{n} C^{ij}}{\sum_{i=1}^{n}\sum_{j=1}^{n} \mathbb{C}^{ij}}\right) + \zeta\left(\frac{\sum_{i=1}^{m}\sum_{j=1}^{m} D^{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{m} \mathbb{D}^{ij}}\right) - \varphi\left(\sum_{g=1}^{n} P_g^{\phi} + \sum_{g=1}^{n} M_g^{\phi} + \sum_{g=1}^{n} H_g^{\phi}\right) - \delta\left(\sum_{f=1}^{m} P_f^{\theta} + \sum_{f=1}^{m} M_f^{\theta} + \sum_{f=1}^{m} H_f^{\theta}\right) - \lambda \sum_{g=1}^{n} E_g^{\phi} - \vartheta \sum_{f=1}^{m} E_f^{\theta} \tag{1}$$

subject to:

$$C^{ij} \leq \mathbb{C}^{ij}, \forall i, j \in R \tag{2}$$

$$\sum_{k=1}^{n} B^{iv}_{kv} A^{iv}_{kv} = C^{iv}, \forall i, v \in R, i \neq v \tag{3}$$

$$\alpha_1 \sum_{i=1}^{n} C^{iv} \leq P_v^{\phi}, \forall v \in R \tag{4}$$

$$\alpha_2 \sum_{i=1}^{n} C^{iv} \leq M_v^{\phi}, \forall v \in R \tag{5}$$

$$\alpha_3 \sum_{i=1}^{n} C^{iv} \leq H_v^{\phi}, \forall v \in R \tag{6}$$

$$\alpha_4 \sum_{i=1}^{n} C^{il} \leq E_l^{\phi}, \forall l \in R \tag{7}$$

$$D^{ij} \leq \mathbb{D}^{ij}, \forall i, j \in S \tag{8}$$

$$\sum_{k=1}^{m} B^{ij}_{kv} A^{ij}_{kv} = \sum_{u=1}^{m} B^{ij}_{vu} A^{ij}_{vu}, \forall i, j, v \in S, i \neq v, j \neq v \tag{9}$$

$$\sum_{k=1}^{m} B^{iv}_{kv} A^{iv}_{kv} = D^{iv}, \forall i, v \in S, i \neq v \tag{10}$$

$$\sum_{u=1}^{m} B^{vj}_{vu} A^{vj}_{vu} = D^{vj}, \forall v, j \in S, j \neq v \tag{11}$$

$$A^{ii}_{kv} = 0, \forall i, k, v \in S \tag{12}$$

$$A^{ij}_{ki} = 0, \forall i, j, k \in S \tag{13}$$

$$\beta_1\left(\sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} B^{ij}_{vk} A^{ij}_{vk} + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} B^{ij}_{kv} A^{ij}_{kv}\right) \leq P_v^{\theta}, \forall v \in S \tag{14}$$

$$\beta_2\left(\sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} B^{ij}_{vk} A^{ij}_{vk} + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} B^{ij}_{kv} A^{ij}_{kv}\right) \leq M_v^{\theta}, \forall v \in S \tag{15}$$

$$\beta_3\left(\sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} B^{ij}_{vk} A^{ij}_{vk} + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} B^{ij}_{kv} A^{ij}_{kv}\right) \leq H_v^{\theta}, \forall v \in S \tag{16}$$

$$\beta_4\left(\sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} B^{ij}_{lk} A^{ij}_{lk} + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} B^{ij}_{kl} A^{ij}_{kl}\right) \leq E_l^{\theta}, \forall l \in S \tag{17}$$

Variables: $B^{ij}_{kl} \in \{0,1\}, C^{ij}, D^{ij}, A^{ij}_{kl} \geq 0,$
$0 \leq P_v^{\phi}, M_v^{\phi}, H_v^{\phi}, E_v^{\phi}, P_v^{\theta}, M_v^{\theta}, H_v^{\theta}, E_v^{\theta} \leq 1, \forall i, j, k, v, l$

In the proposed model, the objective function is to maximize the throughput of servers and switches and to minimize resource consumption and, as a result, their energy consumption. $\sum_{i=1}^{n}\sum_{j=1}^{n}(\frac{C^{ij}}{\mathbb{C}^{ij}})$ indicates the number of admitted requests to total requests of servers. $\sum_{i=1}^{m}\sum_{j=1}^{m}(\frac{D^{ij}}{\mathbb{D}^{ij}})$ represents the number of admitted requests to total requests of switches. The third and fourth terms in the objective function respectively represent the resource consumption of servers and switches, which we seek to minimize. E_g^{ϕ} and E_f^{θ} represent power consumption in servers and switches, respectively. The coefficients of the objective function indicate the degree of influence of throughput, resource, and energy efficiencies. With these coefficients, a trade-off can be made between throughput, resources, and energy. The constraints are divided into two general categories of servers constraints and switches constraints. Constraint (2) shows that the server can admit at max the number of requests ($C^{ij} \leq \mathbb{C}^{ij}$). Constraint (3) requires that the sum of incoming flows to server v from origin i that pass through the switches in path k be equal to C^{iv}. Constraints (4) to (7) are constraints on the resource allocation and power of servers. Constraints (4), (5), and (6) respectively indicate that the CPU, memory, and hard disc of the servers are used to handle the requests. Constraint (7) allocates the power consumption of servers to execute the requests. Constraint (8) specifies the maximum

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/JIOT.2021.3095237, IEEE Internet of Things Journal

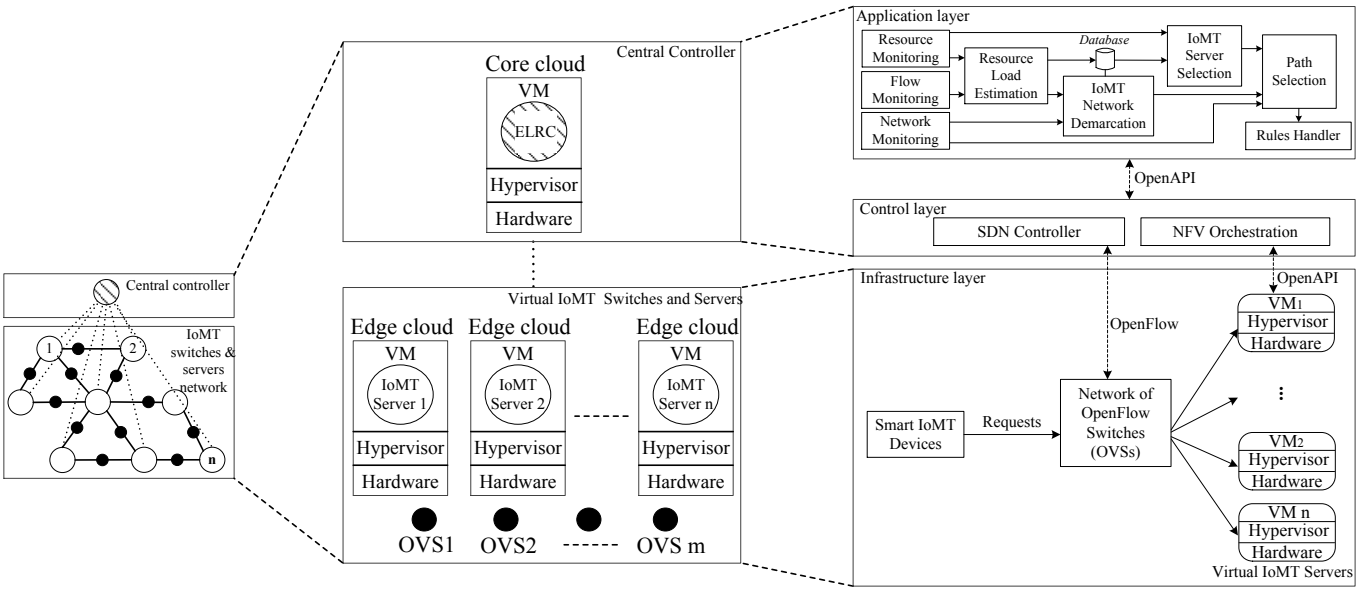


Figure 3. Proposed Software-defined Internet of Multimedia Things framework.

capacity of switches ($D^{ij} \leq \mathbb{D}^{ij}$). Constraints (9) to (13) are used for routing traffic in switches. Constraint (9) creates a balance between incoming and outgoing flows. That is, the sum of incoming and outgoing flows to switch v passing from origin i to destination j must be equal. Constraint (10) requires that the sum of incoming flows to switch v from origin i that pass through the switch k be equal to D^{iv}. The next constraint distributes the sum of outgoing flows from switch v to destination j between switches adjacent to switch v. Constraint (12) will not allow creating a flow with the same origin and destination. Constraint (13) prevents the creation of loops in the path. Constraints (14) to (17) are constraints on the resources and power of switches. Since both sides of equations (4) to (7) and (14) to (17) are not homogeneous, coefficients α and β are respectively used to normalize these equations.

Now, consider binary variable B_{kl}^{ij} so that if $B_{kl}^{ij} = 1$, it means that the request from server i to server j can be transferred to switch l through switch k. Otherwise ($B_{kl}^{ij} = 0$), switches k and l do not participate in the transfer of requests from server i to j. In this case, although the objective function is linear, the existence of non-linear constraints and binary variable B_{kl}^{ij} convert the problem into a Mixed Integer Non-linear Programming (MINLP) problem and, in general, NP-hard [15]. This problem suffers from a high complexity; so we break it down into several sub-problems: network sizing, choosing the proper server, and choosing the proper path. By precisely determining the network size, the energy consumption is reduced. By choosing the right server and path, the load balance of the network is realized. In this regard, the next section introduces a novel modular framework based on SDN and NFV in which these subproblems are imbedded as modules in its controller.

4 Software-defined Internet of Multimedia Things

IoMT is a global and massive network of multimedia things with numerous connections for sending the multimedia data and receiving services from multimedia cloud servers through a network of switches. Today, multimedia equipment is considered simple, and its intelligence is considered in the multimedia cloud. The processing power for the IoMT multimedia equipment is limited, so it is impossible to process multimedia services on this equipment, and it should be done in the multimedia cloud.

As a result, given the high volume of IoMT traffic, we need the routing, load distribution, and resource and energy management techniques in the multimedia cloud servers. Multimedia demands depend on a high level of data processing. This leads to increased power consumption on the servers. Multimedia communication also needs the QoS requirements specific to the IoMT applications. The transfer of multimedia content from things to the application server, which is located on the IoMT clouds, imposes significant traffic management requirements compared to data collection from scalar resources. In particular, these requirements are critical in real-time multimedia communication where delay may not be tolerable depending on various applications. For example, in smart businesses and smart medical systems, delays may not be tolerable. The IoMT servers, a critical part of IoT, have potential capacity to offer various applications. Overall, it can be concluded that the integrated management of IoMT cloud resources, including servers and switches, requires a new framework: Software-defined Internet of Multimedia Things (Fig. 3).

This framework is based on the two-tier cloud computing, including core cloud and edge cloud. The core cloud includes the Energy-efficient and Load-balanced Resource Controller (ELRC). The edge cloud provides the IoMT virtual servers and switches. This separation of controller from servers and switches stems from the SDN logic. The servers are all based on the NFV technology, which means that it allows the creation of VNFs on separate PMs using a hypervisor and VM layer. For simplicity, there is a one-to-one correspondence between VMs and VNFs. Thus, the IoMT virtual servers are as VNF and the switches are as Open vSwitch (OVS).

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/JIOT.2021.3095237, IEEE Internet of Things Journal

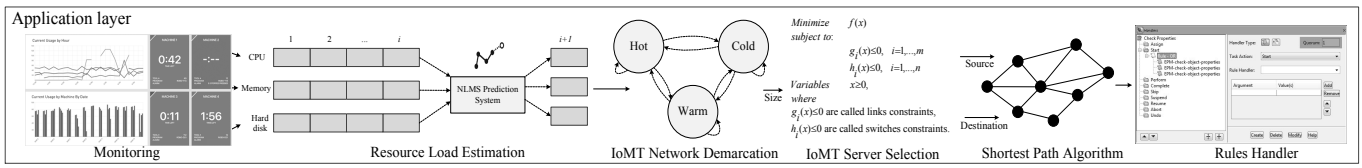


Figure 4. ELRC controller modules (application layer).

The ELRC has two parts: control and application. The SDN controller and NFV orchestration are responsible for controlling the OVSs and VNFs of the IoMT servers, respectively. They also issue the necessary rules based on the application layer modules. The three modules of *Network Monitoring*, *Flow Monitoring*, and *Resource Monitoring* explore and gather information using the Link Layer Discovery Protocol (LLDP). The controller is predictive-driven to respond more quickly to the load changes. In this regard, the *Resource Load Estimation* module estimates the future load of the network according to history. Also according to history, the *IoMT Network Demarcation* module determines the new size of the network. For this size, the *IoMT Server Selection* module selects the best server, and the *Path Selection* module determines the appropriate path to reach the selected server. Finally, the decisions made are converted into appropriate rules by the *Rules Handler* module and delivered to the control layer. In the following, the details of ELRC modules are presented (Fig. 4).

At the beginning of each alternating time interval τ, information is collected from physically active machines (CPU, memory, and hard disk) and creates a history. Using these statistics and the Normalized Least Mean Square (NLMS) predictor, the Resource Load Estimation module (subsection 4.1) predicts the future demand of VNFs for CPU, memory, and hard disk. In other words, it predicts the future load of active PMs based on the previous statistics. Then, the IoMT Network Demarcation module (subsection 4.2) resizes the network using a three-mode finite state machine (hot, warm, cold) through the migration and consolidation strategies. If the state is hot, overload decreases by migrating some VNFs of overloaded points to other PMs or adding a new PM to the network (size increases). If the state is cold, then the VNFs on underloaded points are transferred to another PM in order to turn off these points for energy savings (size decrease). Now that the network size and active PMs have been specified, the best server for servicing the next τ requests is selected by the IoMT Server Selection module and the Least Load algorithm. This results in a fair distribution of load between the IoMT virtual servers. The Path Selection module finds the best path to the selected server among the OVS switches using the Shortest Path algorithm. Finally, the Rules Handler module delivers all decisions made by previous modules in the form of appropriate OpenFlow rules to the controller for appropriate action.

4.1 Resource Load Estimation module

ρ of the last values instantiated from the CPU, memory and hard disk of active PMs at time intervals τ are maintained in the functions f_{cpu}, f_{mem} and f_{hard}. Thus, the functions f^1_{cpu}, f^1_{mem} and f^1_{hard} contain ρ of the last values of CPU usage, memory usage and hard disk usage of PM1. In other words, $f^1_{cpu}(x) = [x^1_i, x^1_{i-1}, ..., x^1_{i-\rho+1}]$ is a vector that contains the history of the CPU usages of PM1. For PM1, x^1_i is the value of the i-th CPU usage ($0 \leq x^1_i \leq 1$). Moreover, y^1_i is the value of the i-th memory usage and z^1_i is the value of the the i-th hard disk usage ($0 \leq y^1_i, z^1_i \leq 1$). As a result, $f^1_{mem}(y) = [y^1_i, y^1_{i-1}, ..., y^1_{i-\rho+1}]$ and $f^1_{hard}(z) = [z^1_i, z^1_{i-1}, ..., z^1_{i-\rho+1}]$ are vectors that contain the history of the memory usages and hard disk usages of PM1. The goal is to obtain an estimate of x_{i+1} ($\widehat{x}_{i+1}$), y_{i+1} ($\widehat{y}_{i+1}$), and z_{i+1} ($\widehat{z}_{i+1}$) for each active PM, according to the f functions and a prediction method. The most important prediction methods include time-series analyses. Among these methods, the Normalized Least Mean Square (NLMS) predictor is the main one providing the best trade-off between complexity, accuracy, and responsiveness. That is why we apply this predictor as the prediction system in the Resource Load Estimation module. Next, we want to obtain $\widehat{x}_{i+1}$ for active PM k using the NLMS algorithm ($\widehat{x}^k_{i+1}$).

Given a vector of ρ observations of x^k_i, $f(x^k_i) = [x^k_i, x^k_{i-1}, ..., x^k_{i-\rho+1}]$, generate an estimation $\widehat{x}^k_{i+1}$ of the value x^k_{i+1}. The NLMS filter coefficients are time differing and are tuned on the basis of the feedback information taken by the error ε_i that $\varepsilon^k_i = x^k_{i+1} - \widehat{x}^k_{i+1}$. The vector of filter coefficients with h_i is specified. The values of h adjust dynamically in order to decrease the Mean Square Error. The NLMS works the following: 1- Initialize the coefficient h_0; 2- For each new data, update the filter h_i based on the recursive equation:

$$h^k_{i+1} = h^k_i + \mu \frac{\varepsilon^k_i f(x^k_i)}{\| x^k_i \|^2} \tag{18}$$

where $\| x^k_i \|^2 = f(x^k_i) f^T(x^k_i)$ and μ is a fixed parameter called step size. Pursuant to [16], NLMS converges so long as $0 < \mu < 2$. At time i, the values x^k_{i+1}, hence ε^k_i, aren't known. So, the value ε^k_{i-1} is used, instead, and the one step NLMS predictor update equation becomes:

$$h^k_{i+1} = h^k_i + \mu \frac{\varepsilon^k_{i-1} f(x^k_{i-1})}{\| x^k_{i-1} \|^2} \tag{19}$$

Overall, the NLMS algorithm formulation can be summarized as:

- Parameters: ρ=filter order, μ=step size,
- Initialization: $h_0 = 0$,
- Computation for i:

$$f(x^k_i) = [x^k_i, x^k_{i-1}, ..., x^k_{i-\rho+1}] \tag{20}$$

$$\widehat{x}^k_{i+1} = h^k_i \times f^T(x^k_i) \tag{21}$$

$$h^k_i = h^k_{i-1} + \mu \frac{\varepsilon^k_{i-1} f(x^k_{i-1})}{\| x^k_{i-1} \|^2} \tag{22}$$

$$\varepsilon^k_i = x^k_{i+1} - \widehat{x}^k_{i+1} \tag{23}$$

$$\| f(x^k_i) \|^2 = f(x^k_i) f^T(x^k_i) \tag{24}$$

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/JIOT.2021.3095237, IEEE Internet of Things Journal

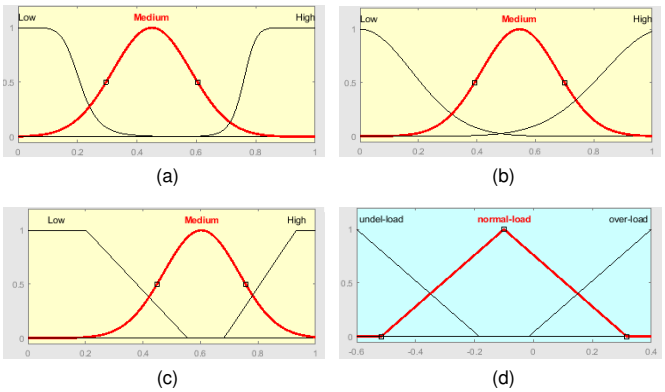


Figure 5. a) Input membership function for $\widehat{x}_{i+1}^k$ [0,1] b) Input membership function for $\widehat{y}_{i+1}^k$ [0,1] c) Input membership function for $\widehat{z}_{i+1}^k$ [0,1] d) Output membership function for $class^k$ [-0.6,0.4].

Where $f(x_i^k)$ and $\widehat{x}_{i+1}^k$ are input and output of the predictor, respectively. Similarly, $\widehat{y}_{i+1}^k$ and $\widehat{z}_{i+1}^k$ can be obtained for each active PM.

4.2 IoMT Network Demarcation module

To determine the appropriate size, the network state in terms of load must be specified, which depends on the predicted consumption resources of active PMs ($\widehat{x}_{i+1}^k$, $\widehat{y}_{i+1}^k$, $\widehat{z}_{i+1}^k$). For this reason, PMs are first classified by the fuzzy logic. Three classes of *under load*, *normal load* and *overload* are defined. The input of this classifier for active PM k is $\widehat{x}_{i+1}^k$, $\widehat{y}_{i+1}^k$ and $\widehat{z}_{i+1}^k$, and the output is one of the three mentioned classes ($class^k$). Membership functions are fine-tuned using experience and repeated tests (Fig. 5). The fuzzy inference method is Mamdani and the Center of Gravity (CoG) is used for defuzzification. Since Mamdani systems and CoG have more intuitive and easier-to-understand rule bases, they are well-suited to IoMT system applications.

The next step in designing a fuzzy system is to design a fuzzy rule base. These rules are designed to express the fact that if consumption resources are high, the load is high, and as a result, the PM class is *overload* (and vice versa). This base is a set of If-Then rules, each of which is responsible for part of the inference and decision. It should be noted that too many rules complicate the fuzzy system and reduce its speed. The proposed rules are as follows:

1) *If ($\widehat{x}_{i+1}^k$ is Low) and ($\widehat{y}_{i+1}^k$ is Low) and ($\widehat{z}_{i+1}^k$ is Low) Then ($class^k$ is under load)*
2) *If ($\widehat{x}_{i+1}^k$ is Low) and ($\widehat{y}_{i+1}^k$ is Low) and ($\widehat{z}_{i+1}^k$ is Medium) Then ($class^k$ is under load)*
3) *If ($\widehat{x}_{i+1}^k$ is Low) and ($\widehat{y}_{i+1}^k$ is Low) and ($\widehat{z}_{i+1}^k$ is High) Then ($class^k$ is normal load)*
4) *If ($\widehat{x}_{i+1}^k$ is Low) and ($\widehat{y}_{i+1}^k$ is Medium) and ($\widehat{z}_{i+1}^k$ is Low) Then ($class^k$ is under load)*
5) *If ($\widehat{x}_{i+1}^k$ is Low) and ($\widehat{y}_{i+1}^k$ is Medium) and ($\widehat{z}_{i+1}^k$ is Medium) Then ($class^k$ is normal load)*
6) *If ($\widehat{x}_{i+1}^k$ is Low) and ($\widehat{y}_{i+1}^k$ is Medium) and ($\widehat{z}_{i+1}^k$ is High) Then ($class^k$ is normal load)*
7) *If ($\widehat{x}_{i+1}^k$ is Low) and ($\widehat{y}_{i+1}^k$ is High) and ($\widehat{z}_{i+1}^k$ is Low) Then ($class^k$ is normal load)*
8) *If ($\widehat{x}_{i+1}^k$ is Low) and ($\widehat{y}_{i+1}^k$ is High) and ($\widehat{z}_{i+1}^k$ is Medium) Then ($class^k$ is normal load)*
9) *If ($\widehat{x}_{i+1}^k$ is Low) and ($\widehat{y}_{i+1}^k$ is High) and ($\widehat{z}_{i+1}^k$ is High) Then ($class^k$ is overload)*
10) *If ($\widehat{x}_{i+1}^k$ is Medium) and ($\widehat{y}_{i+1}^k$ is Low) and ($\widehat{z}_{i+1}^k$ is Low) Then ($class^k$ is under load)*
11) *If ($\widehat{x}_{i+1}^k$ is Medium) and ($\widehat{y}_{i+1}^k$ is Low) and ($\widehat{z}_{i+1}^k$ is Medium) Then ($class^k$ is normal load)*
12) *If ($\widehat{x}_{i+1}^k$ is Medium) and ($\widehat{y}_{i+1}^k$ is Low) and ($\widehat{z}_{i+1}^k$ is High) Then ($class^k$ is normal load)*
13) *If ($\widehat{x}_{i+1}^k$ is Medium) and ($\widehat{y}_{i+1}^k$ is Medium) and ($\widehat{z}_{i+1}^k$ is Low) Then ($class^k$ is normal load)*
14) *If ($\widehat{x}_{i+1}^k$ is Medium) and ($\widehat{y}_{i+1}^k$ is Medium) and ($\widehat{z}_{i+1}^k$ is Medium) Then ($class^k$ is normal load)*
15) *If ($\widehat{x}_{i+1}^k$ is Medium) and ($\widehat{y}_{i+1}^k$ is Medium) and ($\widehat{z}_{i+1}^k$ is High) Then ($class^k$ is overload)*
16) *If ($\widehat{x}_{i+1}^k$ is Medium) and ($\widehat{y}_{i+1}^k$ is High) and ($\widehat{z}_{i+1}^k$ is Low) Then ($class^k$ is normal load)*
17) *If ($\widehat{x}_{i+1}^k$ is Medium) and ($\widehat{y}_{i+1}^k$ is High) and ($\widehat{z}_{i+1}^k$ is Medium) Then ($class^k$ is overload)*
18) *If ($\widehat{x}_{i+1}^k$ is Medium) and ($\widehat{y}_{i+1}^k$ is High) and ($\widehat{z}_{i+1}^k$ is High) Then ($class^k$ is overload)*
19) *If ($\widehat{x}_{i+1}^k$ is High) and ($\widehat{y}_{i+1}^k$ is Low) and ($\widehat{z}_{i+1}^k$ is Low) Then ($class^k$ is normal load)*
20) *If ($\widehat{x}_{i+1}^k$ is High) and ($\widehat{y}_{i+1}^k$ is Low) and ($\widehat{z}_{i+1}^k$ is Medium) Then ($class^k$ is normal load)*
21) *If ($\widehat{x}_{i+1}^k$ is High) and ($\widehat{y}_{i+1}^k$ is Low) and ($\widehat{z}_{i+1}^k$ is High) Then ($class^k$ is overload)*
22) *If ($\widehat{x}_{i+1}^k$ is High) and ($\widehat{y}_{i+1}^k$ is Medium) and ($\widehat{z}_{i+1}^k$ is Low) Then ($class^k$ is normal load)*
23) *If ($\widehat{x}_{i+1}^k$ is High) and ($\widehat{y}_{i+1}^k$ is Medium) and ($\widehat{z}_{i+1}^k$ is Medium) Then ($class^k$ is overload)*
24) *If ($\widehat{x}_{i+1}^k$ is High) and ($\widehat{y}_{i+1}^k$ is Medium) and ($\widehat{z}_{i+1}^k$ is High) Then ($class^k$ is overload)*
25) *If ($\widehat{x}_{i+1}^k$ is High) and ($\widehat{y}_{i+1}^k$ is High) and ($\widehat{z}_{i+1}^k$ is Low) Then ($class^k$ is overload)*
26) *If ($\widehat{x}_{i+1}^k$ is High) and ($\widehat{y}_{i+1}^k$ is High) and ($\widehat{z}_{i+1}^k$ is Medium) Then ($class^k$ is overload)*
27) *If ($\widehat{x}_{i+1}^k$ is High) and ($\widehat{y}_{i+1}^k$ is High) and ($\widehat{z}_{i+1}^k$ is High) Then ($class^k$ is overload)*

Once the class of each active PM is specified, the overall state of the IoMT network can be specified. The network can be in one of the three states *hot*, *warm* or *cold*. If at least one PM is of the *overload* class, then the network state is *hot* and the "Size Increase" algorithm is executed. If there is no PM of the *overload* class and more than half of the PMs are of the *under load* class, then the network state is *cold* and the "Size Decrease" algorithm is executed. The network is in the *warm* state and does not need to be resized if no PM is overloaded and more than half of the PMs are of the *normal-load* class.

4.2.1 *Size increase algorithm*

The purpose of this algorithm is to change the network state from *hot* to *warm*. So it seeks to reduce the temperature of each overloaded PM. As long as there is an overloaded PM, it looks for a good destination PM to migrate a VNF from this hot PM. The best destination PM is an underloaded PM. In the absence of such a PM, a new PM is added to the network, and migration to it occurs. Due to the conservative nature of this method, the destination PM should not be of the *normal load* class because it is likely to change to the *overload* class.

Algorithm 1: Size increase

```
for (i = 1 to number of overloaded PMs) do
    if (There is a PM j whose class is under load) then
        Migrate a VNF from PM i to PM j;
    else
        PM k is added to the network and a VNF is
        migrated from PM i to PM k;
        if (There is switch l that is required) then
            Switch l is activated;
        end
    end
end
```

Algorithm 2: Size decrease

```
for (i = 1 to number of underloaded PMs) do
    if (There is a PM j whose class is under load) then
        Migrate a VNF from PM i to PM j;
        if (PM i lacks VNF) then
            Switch off PM i;
            if (There is switch l that is not used) then
                Switch off switch l;
            end
        end
    end
end
```

4.2.2 Size decrease algorithm

The purpose of this algorithm is to change the network state from *cold* to *warm* and switch off underloaded PMs as much as possible. In this regard, for the VNFs hosted on such PMs, it looks for a suitable destination PM. The best destination is a PM whose class is *under load*. If the origin PM is no longer a VNF host, then it switches off. In this algorithm, the destination PM should not be of the *normal load* class because it is likely to change to the *overload* class.

Three examples of dynamic demarcation of the IoMT network resources are shown in Fig. 6.

5 Implementation and Evaluation of Results

Fig. 7 shows the topology and the testbed, including 8 PMs for the IoMT VNFs, 2 PCs for the OVS switches and 1 PM for the ELRC controllers. PMs include the G9 (three), G8 (two), G7 (three), and G6 (one) HP servers. In the testbed, *Floodlight v1.2*, *Open vSwitch v2.4.1*, and *Kaa server v1.2* are used to implement the ELRC controller, the OpenFlow switches and the IoMT VNFs, respectively. Kaa is really a very adaptable open-source system for developing applications in the IoT. Floodlight is a powerful Java-based controller. The Open vSwitch is a virtual and software switch that supports the OpenFlow protocol. Floodlight runs on the G9 DL580 HP server and we change it according to what was described in the previous sections. This server is equipped with VMware ESXi hypervisor v6.0. The designed modules are executed on the controller. The Open vSwitches run on two PCs, each with an Intel Xeon 4-core 3.70GHz CPU and 16 GB of RAM. The Oprofile software is used to monitor the consumption resources of PMs. The iPerf is used to create multimedia sessions (traffic injection), and the StarTrinity is used to measure the network parameters and the quality of sessions. iPerf is a professional network monitoring and management software which sends packets at a constant or variable rate. It creates a bit rate stream of traffic. The bandwidth of the links is 10 Mbps. ρ and μ are chosen to be 30 and 0.8 because they create a satisfactory efficiency based on the analysis of the results. The comparison method is LBBSRT [17]. This new method using the SDN controller to obtain the response time of each server to select a server with the minimum response time. LBBSRT is used for the load balancing of servers in the SDN, but it lacks the energy management mechanism for servers. The QoS and QoE multimedia evaluation criteria include *throughput*, *delay*, *power consumption*, *number of VNFs and OVSs*, *average CPU usage*, *Mean Opinion Score (MOS)*, and *R Factor*.

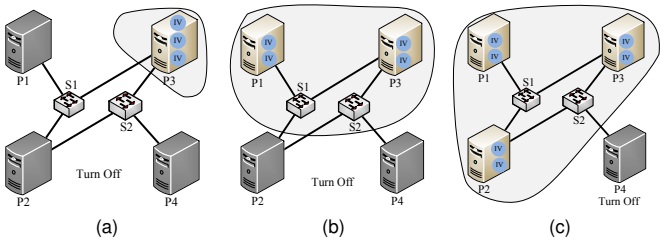


Figure 6. Three examples of dynamic demarcation of the IoMT network resources to optimize energy consumption with respect to load a) Network size = 1 active server b) Network size= 2 active servers and 2 active switches c) Network size = 3 active servers and 2 active switches.

As measuring multimedia quality is important to the service providers and end users, ITU-T provides two test methods subjective and objective testing. Subjective testing represents the earliest attempts on this issue to evaluate the speech quality by giving MOS. ITU-T Rec. P.800 presents the MOS test procedures as users can rate the speech quality from 1 (Poor) to 5 (Excellent) scale. Of course, the numbers of the listeners are an important factor in estimating accurate scores. Thus, subjective testing using MOS is time consuming, expensive and does not allow real time measurement. Consequently, in recent years, new methods were developed for measuring MOS scores in an objective way (without human perception): notably PESQ and the E-model. This paper use E-model and StarTrinity. The E-model is in accordance with the ITU-T Recommendation G.107. This model is widely used both to estimate quality during the planning stage of multimedia networks, and also during their operation. Based on E-model, R Factor is a value derived from metrics such as latency, jitter, and packet loss per ITU-T Recommendation G.107.

Next, we present the evaluation results of the proposed framework at both data and control planes. Each test is performed at least three times and their average is considered as the result.

5.1 Data plane performance evaluation

This section examines the data plane performance of the proposed framework. To do this, we inject traffic into the PM set in a period of 100 seconds in the form of four scenarios. Scenarios with *low*, *medium*, *high*, and *very high* offered loads (Figs. 8a to 8d). Figs. 8e to 8h show the throughput of servers

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/JIOT.2021.3095237, IEEE Internet of Things Journal

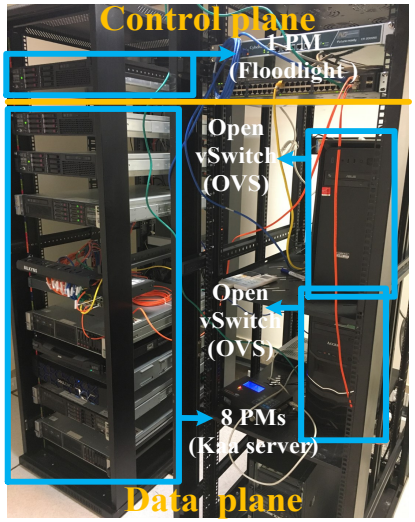


Figure 7. Software-defined Internet of Multimedia Things testbed.

in four scenarios for the four mechanisms *ELRC*, *LBBSRT*, *Round Robin (RR)*, and *No Control (NC)*.

Throughput is the number of communications serviced per unit time. Throughput follows the traffic pattern. That is, it increases with increasing the offered load. The ELRC achieved the highest throughput in all four scenarios. The LBBSRT, an SDN-based method, can only achieve half of the throughput obtained by the ELRC. This is due to the predictive nature of ELRC. The RR mechanism is in the next place and after that, the lowest throughput is related to the NC mechanism, which has no resource management. Note that the ELRC can achieve a very high throughput (very close to the offered load) even in very heavy traffic (Fig. 8h). This proves that the ELRC has done very well in both resource management and traffic routing. One of the consequences of heavy traffic (overload) is increased delay.

Figs. 8i to 8l show the delay of mechanisms at different loads. The maximum delay is related to the NC and RR methods and is proportional to the offered load. That is, as traffic increases, so does the delay. But this increase is very small in the ELRC due to global resource management, and consequently no servers are overloaded. For example, at an offered load of about $13,000$ cps (Fig. 8d), the delay of the ELRC method is less than 250 ms (Fig. 8l). However, the LBBSRT cannot achieve a delay of less than 1000 ms in these conditions. A comparison of Figs. 8h and 8l shows that the ELRC with intelligent resource management achieved a high throughput with low delay even in severe conditions.

The next evaluation parameter is power. Figs. 8m to 8p show power consumption in different mechanisms. Since the ELRC is equipped with a module to adjust the network boundary and uses virtual resources for switches and servers, its power consumption varies according to the offered load. At the peak of traffic, power consumption increases, and as traffic decreases, so does power consumption. This can be understood from the number of active OVSs and VNFs of IoMT (Figs. 8q to 8t). The ELRC manages energy by managing the number of active switches and servers. Conversely, other methods that lack such management, have a constant power consumption for a total period of 100 seconds. Figs. 8q to 8t show that the ELRC limits the network boundary as the offered load decreases, and expands the network boundary as the offered load increases.

In addition, the ELRC balances the loads between servers. As shown in Figs. 8u to 8x, the average CPU consumption of active servers in the ELRC is almost unchanged in 100 seconds (unlike other methods). The standard deviation of CPU consumption of servers in the ELRC is very small compared to other methods. Fig. 9 is a snapshot of the CPU usage of PMs, which shows that in the ELRC, the CPU consumption of all active servers is almost equal. There are similar results for memory and hard disk consumption, which are not presented here.

So far, we have evaluated the quality of service of the proposed framework infrastructure. Now we want to evaluate the quality of user experience of this framework in the form of two criteria, MOS and R Factor. The R Factor score, which is found in conjunction with multimedia assessment processes, can range between 1 (worst) to 100 (ideal) and is dependent on the number of users that are satisfied with the quality of a test multimedia signal after it has passed through a network from a source to a destination. Also, MOS is one of the quality metrics used in the area of multimedia experience quality. The lowest feasible MOS, representing complete lack of end-user satisfaction, is 1, and the highest possible MOS, representing ideal multimedia quality, is 5.

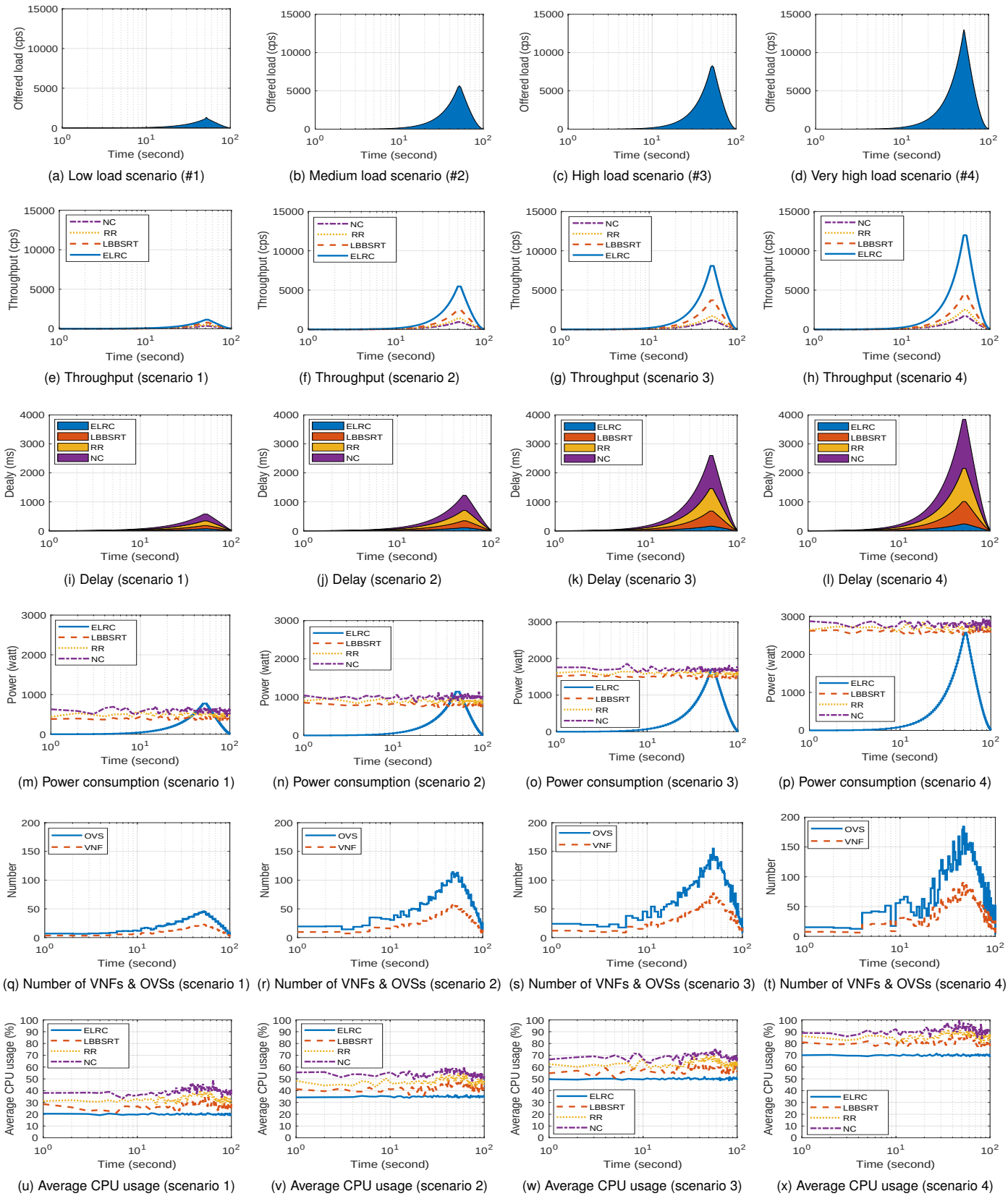
Figs. 8y to 8ab show the MOS in different methods. As can be seen, the most MOS belongs to the ELRC and in the worst case scenario, its average is not less than 4, which indicates a very high level of user satisfaction. This is also evident from the comparison of the R Factor in Figs. 8ac to 8af. As traffic increases, the R Factor decreases, but in the ELRC, it is not less than about 83. Conversely, in other methods, the R Factor not only drops sharply, but also has high oscillations. For example, in heavy traffic, the R Factor of the LBBSRT method drops to less than 70 (Fig. 8af).

5.2 Control plane performance evaluation

In this section, we examine the performance evaluation of the control plane of the proposed framework. Fig. 10 shows the details of processor consumption by the ELRC controller obtained through Oprofile. As can be seen, the Floodlight Kernel takes up more than half of the ELRC processor, but the proposed modules consume much less CPU power. This is evidence of the high scalability of the ELRC. This means that the proposed modules do not impose a high processing overhead on the controller. Among these modules, the Resource Load Estimation module consumes more CPU due to the implementation of the NLMS algorithm. The IoT Network Demarcation module takes the next place in terms of CPU consumption due to the implementation of the size increase and size decrease algorithms. This figure also shows that the ELRC processor is not saturated up to the heavy offered load of $20,000$ cps. Besides, the media quality parameters are very satisfying (Fig. 11). For example, the media delay is less than 800 ms (Fig. 11b) and the media completion rate is more than 96% (Fig 11d). This causes the media setup time and media retransmission rate to be very low even in $20,000$ cps (less than 650 ms and less than 400 cps, respectively (Figs. 11a, 11f)).

Table 1 shows that the ELRC throughput is very close to the offered load and it handles almost all the OpenFlow

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/JIOT.2021.3095237, IEEE Internet of Things Journal



(a) Low load scenario (#1) (b) Medium load scenario (#2) (c) High load scenario (#3) (d) Very high load scenario (#4)

(e) Throughput (scenario 1) (f) Throughput (scenario 2) (g) Throughput (scenario 3) (h) Throughput (scenario 4)

(i) Delay (scenario 1) (j) Delay (scenario 2) (k) Delay (scenario 3) (l) Delay (scenario 4)

(m) Power consumption (scenario 1) (n) Power consumption (scenario 2) (o) Power consumption (scenario 3) (p) Power consumption (scenario 4)

(q) Number of VNFs & OVSs (scenario 1) (r) Number of VNFs & OVSs (scenario 2) (s) Number of VNFs & OVSs (scenario 3) (t) Number of VNFs & OVSs (scenario 4)

(u) Average CPU usage (scenario 1) (v) Average CPU usage (scenario 2) (w) Average CPU usage (scenario 3) (x) Average CPU usage (scenario 4)

Figure 8. Continue to the next page

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/JIOT.2021.3095237, IEEE Internet of Things Journal

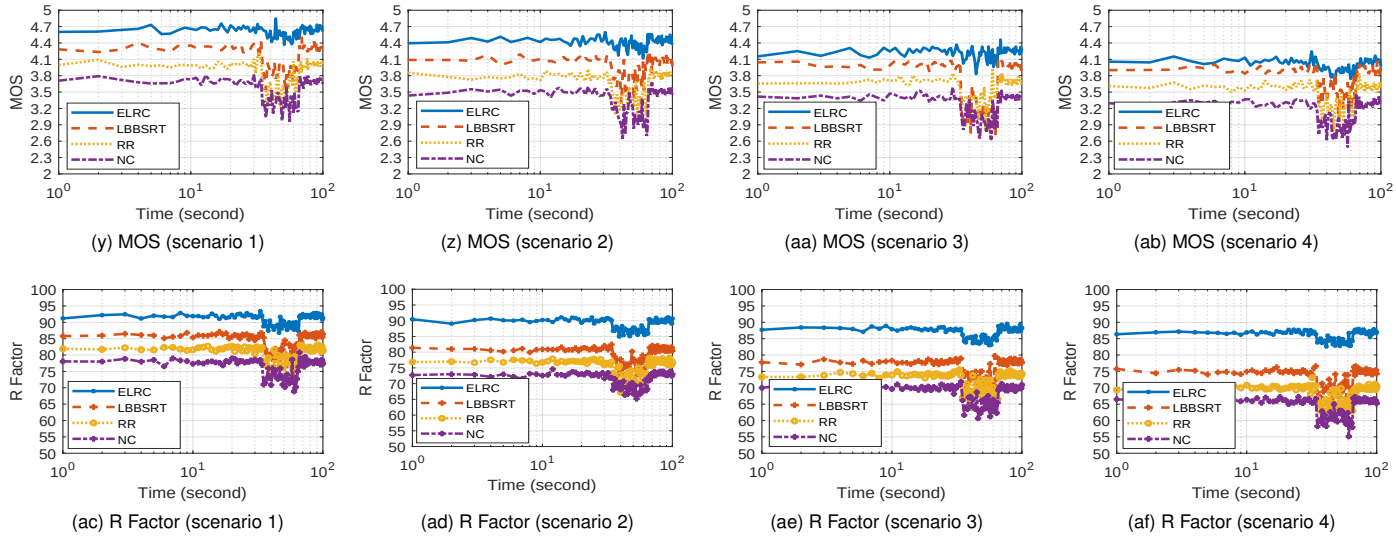


Figure 8. Efficiency evaluation of data plane in various mechanisms by the QoS and QoE evaluation parameters.

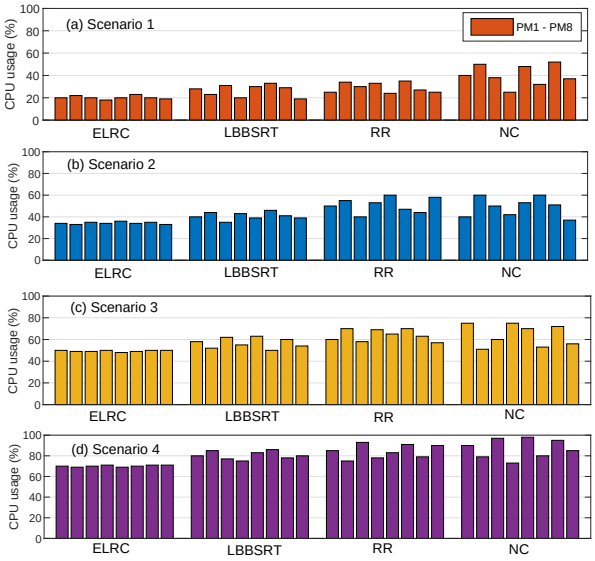


Figure 9. Snapshot of PMs CPU consumption in different scenario.

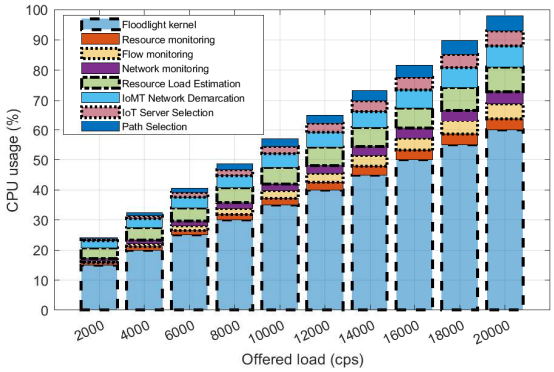


Figure 10. CPU consumption by the ELRC modules.

messages. For example, in the $20,000$ cps traffic, the ELRC throughput is approximately $19,588$ cps. In addition, the ELRC response time is less than 1000 ms. Also, the maximum memory consumption is 90%, which indicates that the proposed modules are not a bottleneck for the ELRC.

6 Conclusions and Future Works

IoMT has more challenges and requirements than IoT. Multimedia data requires higher processing capabilities than scalar data. In addition, multimedia data requires high bandwidth and large storage capacity. Moreover, similar to traditional multimedia networks, different IoMT-based applications require a specific quality of service to provide good user experiences. In this regard, the IoMT cloud servers have been developed to process multimedia data. Due to the huge and growing volume of multimedia traffic, the IoMT cloud has two major challenges for network resource management: load distribution and energy savings. In this article, we proposed the Software-defined Internet of Multimedia Things framework for integrated IoMT cloud resource management and meeting the mentioned requirements. In this framework, the ELRC controller makes a trade-off between efficiency and energy using proactive heuristic algorithms, including the network sizing method and the NFV technology. The IoMT network boundary, which contains active servers and switches, is constantly changing according to the offered load. The task of the ELRC is to precisely determine this boundary according to the load and energy constraints. That is, as the load decreases, the network boundary gets smaller, so that energy consumption is reduced, but at the same time, efficiency (such as server throughput) is not harmed. As the load increases, the network boundary becomes wider to satisfy efficiency. In addition, load is distributed between servers. We implemented the proposed framework in a real testbed and evaluated it with various scenarios. The results show that the ELRC, with respect to the load, both minimized the number of active switches and IoMT servers for energy savings and distributed the load. As a result, the QoS and QoE evaluation parameters, such as throughput, multimedia delay, MOS and R Factor, were greatly improved.

For future works, we intend to use the resource verti-

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/JIOT.2021.3095237, IEEE Internet of Things Journal

Table 1
Performance of the ELRC controller at different loads

Offered load (cps)	2000	4000	6000	8000	10000	12000	14000	16000	18000	20000
Throughput (cps) ∼	1988	3865	5868	7858	9822	11793	13750	15719	17684	19588
Response time (ms) ∼	56	198	245	385	427	584	695	765	846	957
Memory usage (%) ∼	18.67	27.65	39.75	48.53	55.76	60.97	68.98	76.56	84.45	90.65

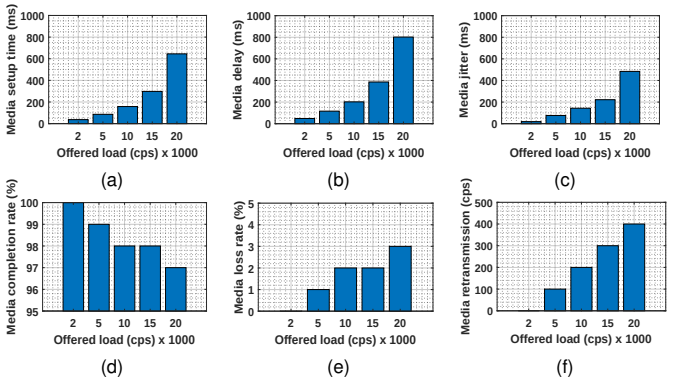


Figure 11. a) Media setup time b) Media delay c) Media jitter d) Media completion rate e) Media loss rate f) Media retransmission rate (ELRC).

cal scaling methods and compare them with the proposed method in this article (horizontal scaling). We will also study more sophisticated strategies to estimate the load of IoMT servers. We also plan to develop the ELRC and the proposed framework instead of using VMs in the Docker-Container or OpenStack and compare them with each other. We will also look at using the Dual Decomposition or Lagrangian Relaxation methods (instead of using heuristic ones) to reduce the complexity of the problem.

REFERENCES

[1] A. Nauman, Y. A. Qadri, M. Amjad, Y. B. Zikria, M. K. Afzal, and S. W. Kim, "Multimedia internet of things: A comprehensive survey," *IEEE Access*, vol. 8, pp. 8202–8250, 2020.

[2] S. A. Alvi, B. Afzal, G. A. Shah, L. Atzori, and W. Mahmood, "Internet of multimedia things: Vision and challenges," *Ad Hoc Networks*, vol. 33, pp. 87–111, 2015.

[3] M. A. Jan, M. Usman, X. He, and A. U. Rehman, "Sams: A seamless and authorized multimedia streaming framework for wmsn-based iomt," *IEEE Internet of Things Journal*, vol. 6, no. 2, 2018.

[4] K. Thiyagarajan, R. Lu, K. El-Sankary, and H. Zhu, "Energy-aware encryption for securing video transmission in internet of multimedia things," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 3, pp. 610–624, 2018.

[5] M. Usman, "Paal: A framework based on authentication, aggregation, and local differential privacy for internet of multimedia things," *IEEE Internet of Things Journal*, vol. 7, no. 4, 2019.

[6] M. Wang, D. Xiao, and Y. Xiang, "Low-cost and confidentiality-preserving multi-image compressed acquisition and separate reconstruction for internet of multimedia things," *IEEE Internet of Things Journal*, 2020.

[7] Y. Zhang and S. Liu, "A real-time distributed cluster storage optimization for massive data in internet of multimedia things," *Multimedia Tools and Applications*, vol. 78, no. 5, 2019.

[8] M. J. Piran, Q.-V. Pham, S. R. Islam, S. Cho, B. Bae, D. Y. Suh, and Z. Han, "Multimedia communication over cognitive radio networks from qos/qoe perspective: A comprehensive survey," *Journal of Network and Computer Applications*, p. 102759, 2020.

[9] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A software defined networking architecture for the internet-of-things," in *2014 IEEE network operations and management symposium (NOMS)*. IEEE, 2014, pp. 1–9.

[10] S. Bera, S. Misra, and A. V. Vasilakos, "Software-defined networking for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1994–2008, 2017.

[11] K. Kalkan and S. Zeadally, "Securing internet of things with software defined networking," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 186–192, 2017.

[12] F. Restuccia, S. DŠOro, and T. Melodia, "Securing the internet of things in the age of machine learning and software-defined networking," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4829–4842, 2018.

[13] X. Huang, R. Yu, J. Kang, Z. Xia, and Y. Zhang, "Software defined networking for energy harvesting internet of things," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1389–1399, 2018.

[14] T. Theodorou and L. Mamatas, "Coral-sdn: A software-defined networking solution for the internet of things," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2017, pp. 1–2.

[15] C. H. Papadimitriou, "On the complexity of integer programming," *Journal of the ACM (JACM)*, vol. 28, no. 4, pp. 765–768, 1981.

[16] S. S. Haykin, *Adaptive filter theory*. Pearson Education India, 2008.

[17] H. Zhong, Y. Fang, and J. Cui, "Reprint of lbbsrt: An efficient sdn load balancing scheme based on server response time," *Future Generation Computer Systems*, vol. 80, pp. 409–416, 2018.

Ahmadreza Montazerolghaem received the Ph.D. degree in computer engineering from the computer department, Ferdowsi University of Mashhad (FUM), Iran, in 2017. He was a postdoctoral researcher at FUM in 2018. Also, he was IT Director at the Quchan University of Technology, Quchan, Iran in 2020. He is currently an Assistant Professor at the Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran. He is an IEEE student member and quality manager at VoIP type approval laboratory in FUM. He is also a member of National Elites Foundation (Society of prominent students of the country). His research interests are in Software Defined Networking (SDN), Network Function Virtualization (NFV), Internet of Things (IoT) and, Multimedia Networking.