# Efficient Resource Allocation for Multimedia Streaming in Software-Defined Internet of Vehicles

Ahmadreza Montazerolghaem

# Efficient Resource Allocation for Multimedia Streaming in Software-defined Internet of Vehicles

Ahmadreza Montazerolghaem

**Abstract**

Due to the rapid growth of the Internet of Vehicles (IoV) and the rise of multimedia services, IoV networks' servers and switches are facing resource crises. Multimedia vehicles connected to the Internet of Things are increasing; there are millions of vehicles and heavy multimedia traffic in the IoV network. The network's scarcity of resources results in overload, which, in turn, leads to a degradation of both Quality of Service (QoS) and Quality of Experience (QoE). Conversely, when resources are abundant, it leads to unnecessary energy wastage. Managing IoV network resources optimally while considering constraints such as *Energy*, *Load*, *QoS*, and *QoE* is a complex challenge. To address this, the study proposes a solution by decomposing the problem and designing a modular architecture named $ELQ^2$. This architecture enables simultaneous control of the mentioned constraints, effectively reducing overall complexity. To achieve this objective, *Network Softwarization and Virtualization* concepts are employed. This modern architecture allows dynamically adjusting of the scale of the resources on demand, effectively reducing energy usage. Additionally, this architecture provides some other potentials, such as "the distribution of multimedia traffic among servers", "determining the route with high QoS for traffic", and "selecting a media with high QoE". A real test field is provided by *Floodlight Controller*, *Open vSwitch*, and *Kamailio Server* tools to evaluate the performance of $ELQ^2$. The findings suggest that the utilization of $ELQ^2$ holds promise in reducing the count of active servers and switches via effective resource management. Additionally, it demonstrates enhancements in various QoS and QoE parameters, encompassing throughput, multimedia delay, R Factor, and MOS, accomplished through load balancing strategies. As an illustration, the deployment of flows has achieved a commendable success rate of $95\%$ owing to the utilization of SDN-based and comprehensive management practices encompassing all network resources.

**Index Terms**

Software-defined Internet of Vehicles (SD-IoV), Network Functions Virtualization (NFV), Software-defined Networking (SDN), Efficient network resource allocation, Internet of multimedia vehicles.

## 1 Introduction

Multimedia traffic is highly increasing in smart cities [1]–[3]. Internet of Multimedia Vehicles (IoMV) is an IoV extension that aims at providing an appropriate field for high-quality multimedia streaming for vehicles [4]. Hence, many studies have been done to develop multimedia-based services and applications in IoMV [5]–[9]. Recent advancements in intelligent multimedia equipment production have resulted in a notable increase in the number of internet-connected vehicles capable of accessing real-time multimedia services. These services encompass a wide range of applications, including online games, videoconferences, remote surveillance via intelligent video cameras, real-time content analysis, security/surveillance services, innovative multimedia care services, and social networks. The IoMV network's servers and switches play a pivotal role in facilitating the transmission, storage, processing, and delivery of this substantial volume of multimedia traffic. (Fig. 1). Integrated equipment management can effectively ensure the network's QoS and user's QoE [10], [11]. Multimedia traffic, encompassing both video and audio data, possesses unique attributes in contrast to conventional network traffic. Real-time communication necessitates servers with ample storage resources and processing power to accommodate these multimedia demands effectively. Additionally, multimedia transfers between switches mandate higher bandwidth capabilities than those typically required by regular traffic.

Nevertheless, excessive allocation of network resources or over-provisioning can result in unnecessary energy consumption. Consequently, it becomes imperative to dynamically provision resources based on demand, thereby mitigating the risk of overload and its associated consequences during peak periods, as well as avoiding energy waste during off-peak times. To achieve an optimal energy-performance balance, efficient and network-aware methods are essential in managing these diverse demands within the multimedia ecosystem. In this regard, SDN

- *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/TITS.2023.3303404, IEEE Transactions on Intelligent Transportation Systems

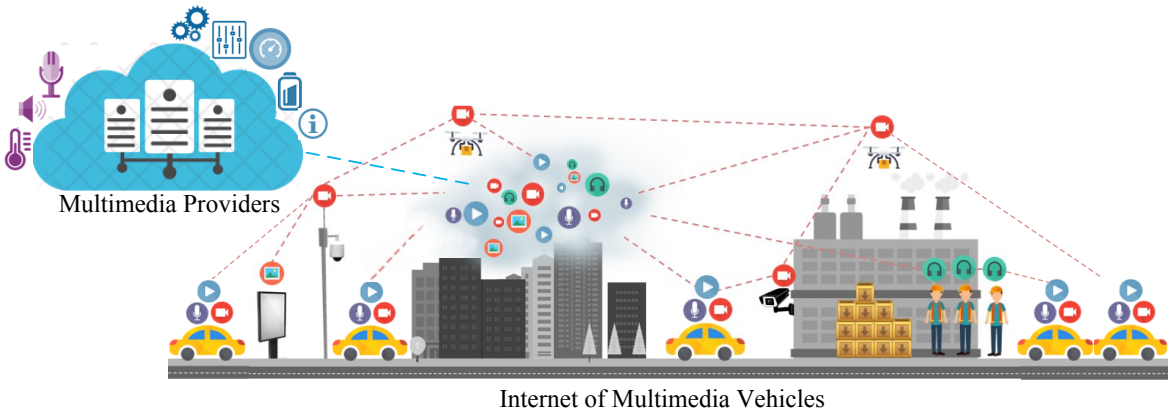



Figure 1. The high volume of multimedia traffic is transported between vehicles and servers through network switches for processing, storage, and servicing.

and NVF technologies can be used [12]. SDN, which has received significant attention recently, can provide a comprehensive view of IoMV network resources in a logically centralized controller for global resource management. In a more precise manner, SDN empowers sophisticated management of resources within the IoMV network by decoupling the data plane from the control plane and establishing a centralized, software-based control mechanism. Additionally, NFV plays a pivotal role in enabling the dynamic adjustment of resources through the virtualization of IoMV network resources.

- During peak times: a strategic approach involves activating idle virtual servers and switches within the IoMV network to prevent overload and effectively handle the increased demand for multimedia services;
- During off-peak times: a prudent measure involves inactivating unused virtual servers and switches within the IoMV network to reduce energy consumption and optimize resource utilization.

### 1.1 Motivations

Fig. 2 illustrates the components of the IoMV network, comprising smart cars, Roadside Units (RSUs), switches, multimedia provider servers, and transported multimedia streaming. The communication of multimedia content, be it demand messages or response messages, necessitates a specific route between the network switches. Given the highly dynamic nature of this mutual communication and topology, an intelligent and network-aware infrastructure becomes essential.

As per predictions on internet network traffic, there is a significant projected increase in multimedia traffic in the forthcoming years. The proliferation of high-speed 4G and 5G networks has further contributed to the continuous expansion and integration of IoMV [13], [14]. Nevertheless, it is worth noting that multimedia traffic within IoMV exhibits variability throughout the day and night. Servers within IoMV often encounter overload during peak hours, which can significantly degrade the quality of multimedia services provided. Conversely, during underload hours, energy is wasted due to low server utilization. This duality of challenges necessitates the development of effective strategies to manage server capacity and energy consumption optimally.

Fig. 3 depicts three *overload*, *underload*, and *normal load* modes in IoMV networks. Resource management solutions must be adopted for the first two modes to prevent their dire consequences.

Therefore, this paper has been organized based on the following motivations:

- Increasing growth of the IoMV network and high-volume multimedia traffic;
- Severe load variations of the IoMV network during different times (peak and non-peak hours);
- Decline in QoS and QoE during peak periods;
- Waste of energy during non-peak hours.

### 1.2 Contributions

The current study presents several noteworthy contributions:

1) Design and Implementation of $ELQ^2$ Architecture: The research introduces an innovative architecture called $ELQ^2$, specifically tailored for the integrated management of servers and switches within the IoMV network. The primary objective of this architecture is to provide users with high-quality multimedia streaming while optimizing energy usage.

2) Dynamic Resource Provisioning: $ELQ^2$ is equipped to dynamically respond to changes in traffic by automatically provisioning (virtual) resources as required. This feature ensures efficient resource allocation in real-time, enabling the network to adapt to fluctuating demands.
3) Reduction in Hardware Costs: The utilization of $ELQ^2$ architecture leads to a reduction in hardware costs. By optimizing resource allocation and virtualizing resources, the need for excessive physical hardware is minimized, thereby offering cost-saving benefits.
4) Modular Structure: The $ELQ^2$ architecture is designed with a modular structure, allowing for flexibility and scalability. Each module serves a specific purpose and is equipped with algorithms tailored to achieve their respective goals.

The development and integration of these features enable $ELQ^2$ to achieve the objectives of enhanced multimedia streaming quality, reduced energy consumption, cost efficiency, and adaptable resource management within the IoMV network.

The $ELQ^2$ architecture is organized into distinct modules, with each module equipped with specific algorithms tailored to achieve its objectives. The following are the key modules and their corresponding algorithms:

* Multimedia Traffic Prediction: This module utilizes predictive algorithms to forecast IoMV's multimedia traffic patterns. By anticipating traffic changes, the network can proactively adapt its resource allocation strategies.
* Network Size Determination: The module employs algorithms to precisely determine the optimal size of the IoMV network. This ensures that the network is appropriately sized to handle the expected traffic demands efficiently.
* Server Selection: Algorithms within this module assist in selecting the most suitable server for handling multimedia streaming requests. The goal is to minimize latency and optimize resource utilization.
* Route Selection: This module utilizes advanced algorithms to determine the optimal route for transmitting multimedia content to the selected server. By choosing the most efficient route, the network can enhance data delivery performance.
* Media Quality Selection: The module is responsible for selecting the highest-quality media content for transmission. Algorithms are employed to ensure that users receive superior multimedia streaming experiences.

The integration of these algorithms within their respective modules enables $ELQ^2$ to achieve its overarching goals of improved multimedia streaming quality, efficient resource management, and optimized network performance within the IoMV environment.

### 1.3 Organization

Section 2 of this study reviews the research background. Section 3 designs the framework and controller of $ELQ^2$ modular. New algorithms are suggested for each module. The subsections also explain more details. Section 4 implements $ELQ^2$, evaluates the results, and analyzes the $ELQ^2$ performance. Section 5 proposes a conclusion and further studies.

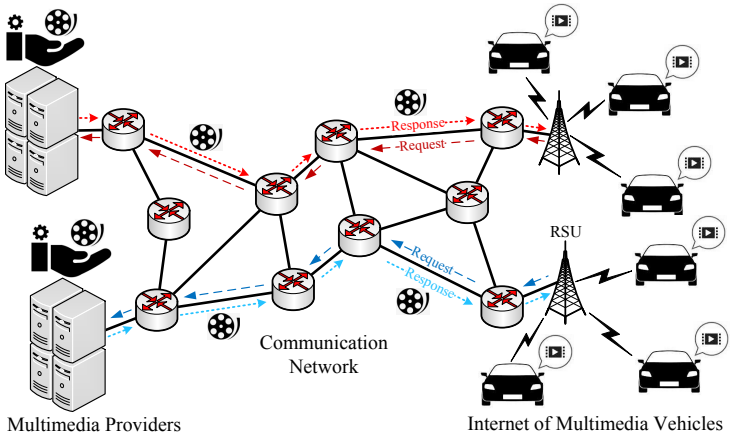



Figure 2. IoMV network infrastructure: smart vehicles, RSUs, switches, multimedia servers and multimedia streams.

## 2 Related work

In the work by Islam *et al.* [15], the authors provide a comprehensive taxonomy of Software-defined Vehicular Network (SDVN) architecture. They subsequently conducted a survey, classifying the state-of-the-art SDVN routing protocols. Additionally, the paper highlights the existing challenges associated with SDVNs. Boukerche *et al.* [16] propose a strategy for managing software-defined vehicular networks based on vehicles' mobility densities and communication latencies between switch-enabled access points. Their approach aims to optimize energy consumption within the network. In Zhang *et al.*'s study [17], the authors investigate the communication network for vehicles in the context of an Internet of Things (IoT)-based intelligent transportation system. They utilize a layered network simulation approach employing OPNET Modeler. The authors of [18] present a framework based on SDN and NFV for managing cloud multimedia centers with virtualized resources. This paper discusses overload and its impact on multimedia centers' QoS and energy consumption. Addis *et al.* [19] propose resource allocation strategies for virtualized cloud environments that minimize energy consumption while satisfying performance and availability guarantees. In this way, a scalable hierarchical framework based on mixed-integer nonlinear optimization of resource management is presented. Using virtualization capabilities in conjunction with SDN for virtual machines and traffic consolidation, Son *et al.* propose a dynamic overbooking strategy in [20]. By allocating more precise resources to virtual machines and traffic, the proposed strategy adapts to dynamically changing workloads. Using this strategy increases overbooking while still minimizing service-level agreement violations. SDN flexibility can also be used to solve the issues with traditional load balancing schemes by utilizing server response time [21]. By utilizing the real-time response time of each server measured by the SDN controller, [21] ensures that the load on each server is evenly distributed. The purpose of [22] is to examine the issue of load balancing in multimedia connections in order to prevent overloads. The load balancer in the proposed architecture of this paper has a window for each server. Each window displays the server's response time history over time. Load distribution is based on these windows. In the work by Jiang *et al.* [23], the authors introduce a counter-based load-balancing algorithm aimed at distributing requests to a cluster of multimedia servers. The algorithm assigns new requests to the server with the lowest counter, and the counter for that server is incremented by one. This approach optimizes the distribution of incoming requests across servers, enhancing overall system performance. The paper by Pokhrel *et al.* [24] presents a redesign of the wireless edge framework within the IoV to enable orchestration and automation. The authors propose extending SDN capabilities to vehicles by deploying mobile base stations with SDN functionality. This enhancement facilitates efficient network management and communication within the IoV ecosystem. Jiang *et al.* [25] aim to study the measurement and analysis technology for SDNs in the context of the IoV. They propose a new measurement framework for IoV heterogeneous networking utilizing SDN. Additionally, they present a performance measurement and analysis method to evaluate the efficiency of the proposed framework. Since OpenFlow switches have limited flow tables, IoV characteristics pose some challenges in rule installation. IoV scalability requires compact flow tables. As a result, Wang *et al.* [26] create a novel rule installation mechanism to reduce SD-IoV's amount of OpenFlow rules. To develop a reliable connectivity framework for SD-IoV, [27] proposes an algorithm for scalable link optimization. Additionally, an analysis of the importance of vehicular networks is conducted in order to establish

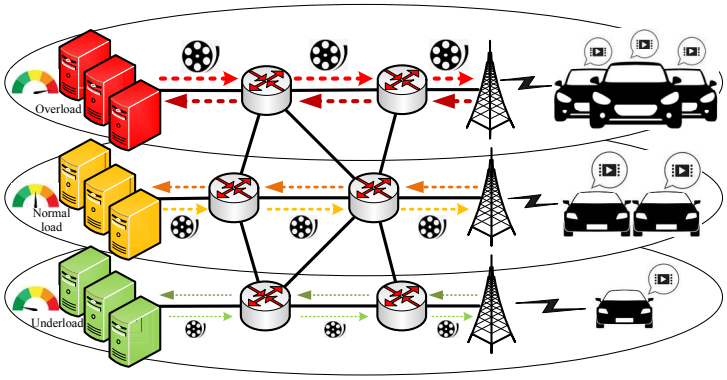



Figure 3. The increasing rise of intelligent vehicles' multimedia traffic during peak hours causes an *overload* in multimedia servers. On the other hand, energy is lost during *underload* times due to the idleness of multimedia servers. This study aims at adjusting IoMV resources according to need.

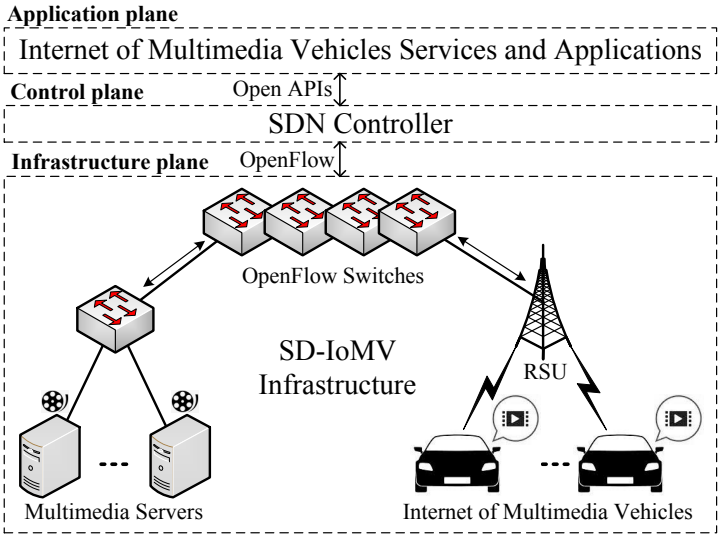



Figure 4. Overview of $ELQ^2$ for IoMV network.

a use case for a smart city with stable and reliable connectivity. An approach for caching multimedia content in a vehicular network with limited resources is proposed in [28]. A two-layer hierarchical network architecture is used in the proposed scheme to allow caching only at edge nodes. Also, to improve network performance, both caching location and cached content are considered. In their research, Chen *et al.* [29] explore service-oriented dynamic vehicular connection management in SD-IoV. Their focus is on designing a system capable of supporting multiple concurrent requests while simultaneously improving resource utilization efficiency and flexibility. By integrating SDN with the IoV, their proposed architecture illustrates the potential of creating advanced intelligent transportation systems for the future. In [30], a technique for maximizing vehicular QoE is proposed by combining video quality selection and resource allocation. In order to provide seamless video playback at end users, it leverages the queueing dynamics and channel states. Sodhro *et al.* presented a video transmission rate control algorithm and the development of a cloud-based video transmission framework in their work [31]. The work presented in reference [32] is primarily concerned with the adoption of specific strategies to extend battery life while facilitating on-demand variable bit rate video transmission from the medical video server to the base station.

Amidst the considerable research examining the advantages of employing Software-Defined Networking (SDN) in vehicular communication networks, only limited literature has delved into the practical implications of integrating SDN and Network Functions Virtualization (NFV) in the context of multimedia Internet of Vehicles (IoV). Consequently, it is crucial and opportune to investigate this approach, especially given the rapid expansion of multimedia applications and the utilization of SDN/NFV technologies.

## 3 Proposed Framework: $ELQ^2$

The IoMV constitutes an extensive network of multimedia-equipped vehicles engaging in communication to exchange multimedia streaming with multimedia servers through a network of switches. This section of the study introduces a novel framework named $ELQ^2$ specifically designed for this expansive IoMV network. To achieve this, SDN technology is utilized to provide a centralized, global view of the entire IoMV network's resources, facilitating integrated resource management and efficient traffic distribution. Fig. 4 illustrates an overview of the $ELQ^2$ framework, which encompasses three planes. The *infrastructure plane* consists of multimedia servers and vehicles interconnected via OpenFlow switches, a replacement for traditional switches. These OpenFlow switches are programmable through the widely used OpenFlow protocol and an SDN controller. The OpenFlow protocol encompasses messages such as `Features-request/reply`, `Packet-In`, and `Flow-mod`, enabling queries to switches and command installations on them. Moving to the *control plane*, the SDN controller collects control data from the infrastructure plane. This data includes multimedia request volumes, available server resources, switch topology, link capacities, and other relevant information. The SDN controller communicates with the infrastructure plane using the OpenFlow protocol messages and also governs the behavior of switches. On the

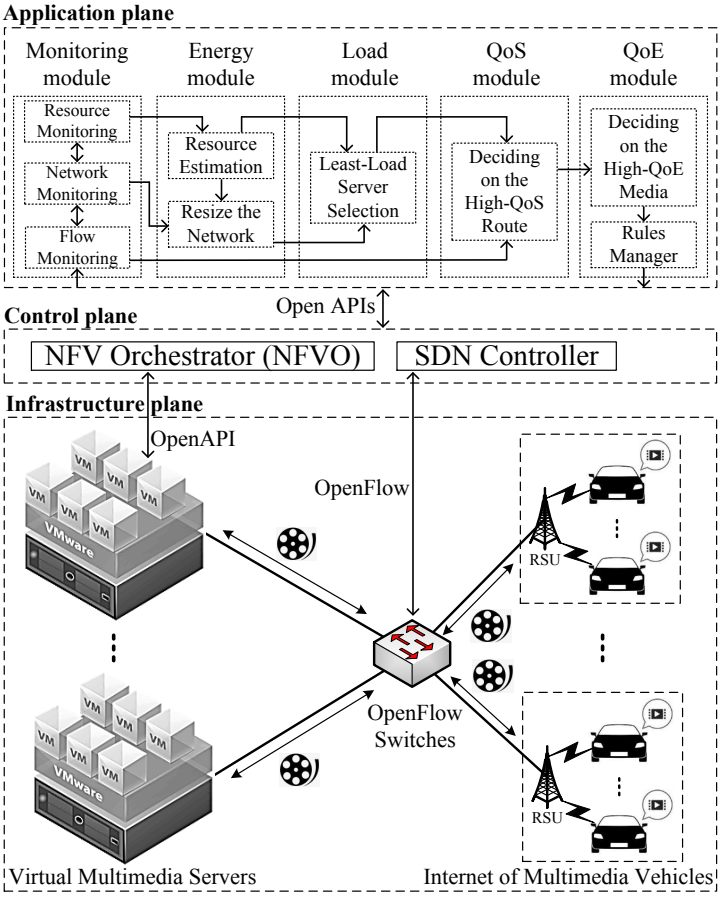



Figure 5. The modular framework of proposed $ELQ^2$ in close view.

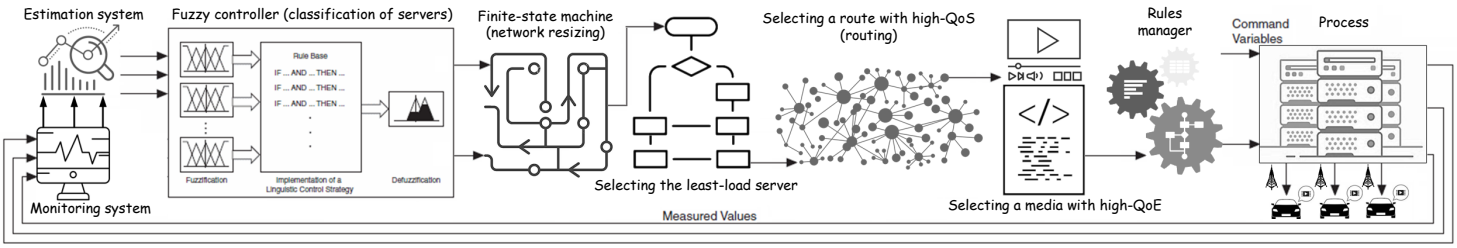



Figure 6. The cycle of proposed modules.

other hand, the *application plane* plays a pivotal role in the decision-making process, based on the data collected from the control plane. It possesses complete information about the IoMV network and acts as the orchestrator of the system. Within this plane, intelligent algorithms, such as routing, load prediction, load balancing, etc., are executed to optimize network operations.

The application and control planes are interconnected through standard Open APIs, ensuring seamless communication and coordination between them. This structured and interconnected architecture enables the $ELQ^2$ framework to efficiently manage multimedia connections within the IoMV network, optimizing resource utilization and enhancing overall performance.

Fig. 5 depicts more details about $ELQ^2$. As seen in this figure, switches and servers are controlled centralized so that multimedia traffic streaming reaches intelligent vehicles timely with high quality. Servers are naturally virtual and use NFV technology. It means that these servers enable the creation of Virtualized Network Functions (VNF) on the autonomous Physical Machines (PM) by using a hypervisor layer and the potential of Virtual Machines (VM). One VNF may consist of one or more VMs that run a specific network function (e.g., IoMV

servers) on a PM[1]. In this case, purpose-specific hardware equipment is not required for functions of the IoMV network. It is possible to dynamically create, deploy, or migrate some instances of VNFs. One also can delete the instances based on the network's conditions. VM live migration technology allows changing the mapping between VNFs and PMs without any service interruption. NFV Orchestrator (NFVO) manages VNFs. NFVO modifies the size of the network through informed allocation based on the demands of VNFs. NFVO activates the inactive PMs during peak times and assigns VNF to them for servicing. In this case, the network's capacity and size are increased. The idle PMs are turned off during non-peak times. This case reduces the network size and energy usage. NFVO and the SDN controller depend on the decisions made by the application plane's modules to manage servers and switches.

The application plane consists of five key modules: *monitoring*, *energy*, *load*, *QoS*, and *QoE*. Within these modules, there are submodules for *resource*, *network*, and *flow monitoring*, which explore and collect data from lower-plane equipment using the Link Layer Discovery Protocol (LLDP). In the *monitoring* submodules, statistics and records are gathered, providing valuable insights into network performance and resource utilization. The *energy* module utilizes this collected data to predict the required resources and subsequently orchestrates the network size based on these predictions. Once the network size is determined, the *load* module selects the optimal PM to handle the incoming multimedia requests. The *QoS* module comes into play to determine the highest-quality route that reaches the selected PM, ensuring efficient and reliable data transmission. Moreover, the *QoE* module considers the highest-quality media available from the selected PM to be sent to the respective vehicle, enhancing the end-user experience. The *rules handler* submodule translates the decisions made within the application plane into suitable commands, which are then delivered to the control layer for implementation and execution. By effectively coordinating the various modules and submodules, the application plane orchestrates dynamic resource management and optimized multimedia streaming within the IoMV network.

Modules of $ELQ^2$ have a cycle (Fig. 6), i.e., they do their tasks within intermittent intervals $\tau$. The *monitoring system* captures the network. Data associated with active PMs' resources (CPU, memory, and hard-disc) are collected and recorded. These data are used by the *estimation system* to predict the future demand of VNFs for resources. In other words, it forecasts the future load of active PMs. Then, the component of *classification of servers* classifies PMs by using fuzzy logic. Next, the *network resizing* component modifies the network size and the set of active PMs by using a finite state machine through migration and consolidation. The component of *selecting the least-load server* determines the least-load active PM for servicing multimedia requests. This leads to a load balance between active PMs. The *routing* component specifies a route with high QoS that reaches the selected PM. The next component selects a multimedia with high QoE to deliver to the demander vehicle. Finally, the *rule manager* component transforms the decisions into OpenFlow messages and delivers them to lower layers to be run and processed.

1. A one-by-one correspondence exists between VMs and VNFs for simplicity.

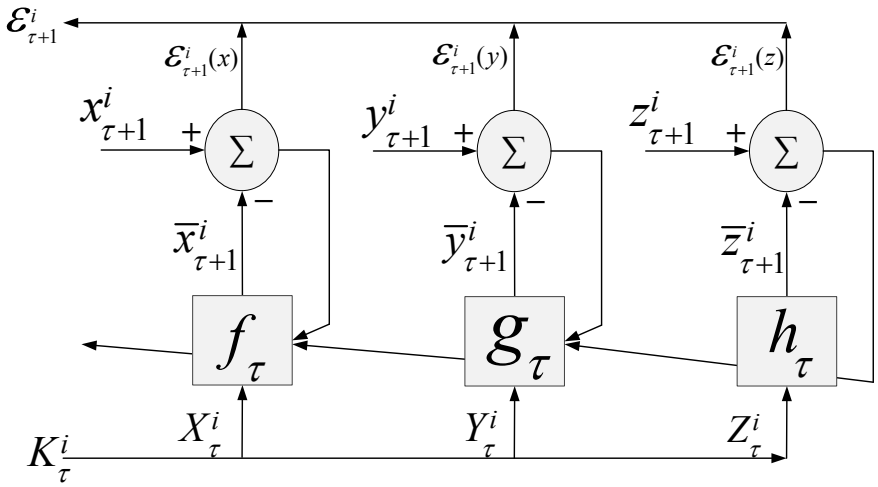



Figure 7. The proposed NLMS system for the estimation of IoMV resources.

### 3.1 Estimation system

An Normalized Least Mean Square (NLMS) algorithm-based predictor is designed in this phase to predict the future load of active PMs. Based on the consumption pattern of CPU, memory and hard-disk of PMs at time $\tau$, this system estimates their values at $\tau + 1$. Fig. 7 depicts the system details. The system input is a three-row matrix $K^i_\tau$, which includes the sampled history of CPU, memory, and hard-disc of PM $i$. The first row consists of the vector $X^i_\tau = [x^i_\tau, x^i_{\tau-1}, ..., x^i_{\tau-\phi+1}]$, the second row indicates the vector $Y^i_\tau = [y^i_\tau, y^i_{\tau-1}, ..., y^i_{\tau-\phi+1}]$, and the third row is the vector $Z^i_\tau = [z^i_\tau, z^i_{\tau-1}, ..., z^i_{\tau-\phi+1}]$ that are $\phi$ samples of CPU, memory, and hard-disc of PM $i$, respectively. It must be noted that all values in this system are normalized ($0 \leq x^i_\tau, y^i_\tau, z^i_\tau \leq 1$). This system tends to minimize the mean square error ($\varepsilon^i_{\tau+1}$). The output of this system comprises estimated values for the next $\tau$ that equal $\bar{x}^i_{\tau+1}$, $\bar{y}^i_{\tau+1}$, and $\bar{z}^i_{\tau+1}$, respectively. These parameters are measured based on the equations below and $f_\tau$, $g_\tau$, and $h_\tau$ filters:

$$\bar{x}^i_{\tau+1} = f_\tau \times [x^i_\tau, x^i_{\tau-1}, ..., x^i_{\tau-\phi+1}]^T \tag{1}$$

$$\bar{y}^i_{\tau+1} = g_\tau \times [y^i_\tau, y^i_{\tau-1}, ..., y^i_{\tau-\phi+1}]^T \tag{2}$$

$$\bar{z}^i_{\tau+1} = h_\tau \times [z^i_\tau, z^i_{\tau-1}, ..., z^i_{\tau-\phi+1}]^T \tag{3}$$

The filters are updated based on the following recursive equations for each new data:

$$f_\tau = f_{\tau-1} + \mu \frac{\varepsilon^i_{\tau-1}(x)[x^i_{\tau-1}, x^i_{\tau-2}, ..., x^i_{\tau-\phi}]}{\| [x^i_{\tau-1}, x^i_{\tau-2}, ..., x^i_{\tau-\phi}] \|^2} \tag{4}$$

$$g_\tau = g_{\tau-1} + \mu \frac{\varepsilon^i_{\tau-1}(y)[y^i_{\tau-1}, y^i_{\tau-2}, ..., y^i_{\tau-\phi}]}{\| [y^i_{\tau-1}, y^i_{\tau-2}, ..., y^i_{\tau-\phi}] \|^2} \tag{5}$$

$$h_\tau = h_{\tau-1} + \mu \frac{\varepsilon^i_{\tau-1}(z)[z^i_{\tau-1}, z^i_{\tau-2}, ..., z^i_{\tau-\phi}]}{\| [z^i_{\tau-1}, z^i_{\tau-2}, ..., z^i_{\tau-\phi}] \|^2} \tag{6}$$

Moreover, the errors are calculated based on the equations below:

$$\varepsilon^i_\tau(x) = x^i_{\tau+1} - \bar{x}^i_{\tau+1} \tag{7}$$

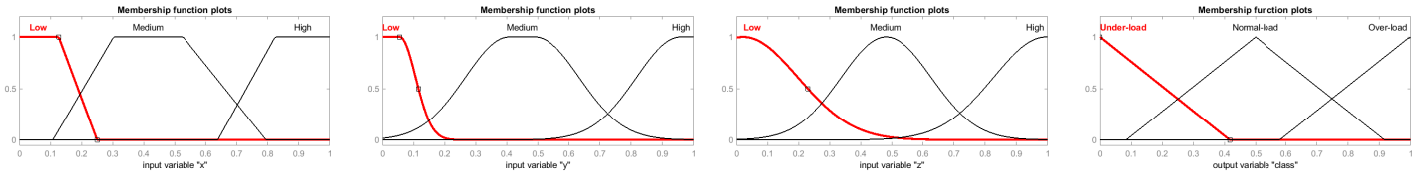
$$\varepsilon^i_\tau(y) = y^i_{\tau+1} - \bar{y}^i_{\tau+1} \tag{8}$$

$$\varepsilon^i_\tau(z) = z^i_{\tau+1} - \bar{z}^i_{\tau+1} \tag{9}$$

The $f_0$, $g_0$, and $h_0$ are usually initialized to zero; $\mu$ also is a constant parameter that is called step size and equals $0 < \mu < 2$.

### 3.2 Network resize and energy management

A classifier is designed for PMs by using fuzzy logic according to Fig. 8. PMs are classified into three classes based on their predicted resource: $under - load$, $normal - load$, and $over - load$. $\bar{x}^i_{\tau+1}$, $\bar{y}^i_{\tau+1}$, and $\bar{z}^i_{\tau+1}$ are the classifier's inputs and their membership functions have been fine-tuned through repeated testing. Mamdani and Center of Gravity (CoG) have been used for the fuzzy inference and defuzzification, respectively. The fuzzy rule base is also given in Table 1 and has been designed so that the PM class becomes $under - load$ if resources of PM $i$ are low, and vice versa.




(a) (b) (c) (d)

Figure 8. The proposed fuzzy system: a) input membership function for $\bar{x}^i_{\tau+1}$ $[0, 1]$ b) input membership function for $\bar{y}^i_{\tau+1}$ $[0, 1]$ c) input membership function for $\bar{z}^i_{\tau+1}$ $[0, 1]$ d) output membership function for $class^i$ $[0, 1]$.

Table 1
The proposed fuzzy rule base to determine the class of PM $i$

| if | | | then |
|---|---|---|---|
| $\bar{x}^i_{\tau+1}$ | $\bar{y}^i_{\tau+1}$ | $\bar{z}^i_{\tau+1}$ | $class^i$ |
| Low | Low | Low | $under-load$ |
| | | Medium | $under-load$ |
| | | High | $normal-load$ |
| | Medium | Low | $under-load$ |
| | | Medium | $normal-load$ |
| | | High | $normal-load$ |
| | High | Low | $normal-load$ |
| | | Medium | $normal-load$ |
| | | High | $over-load$ |
| Medium | Low | Low | $under-load$ |
| | | Medium | $normal-load$ |
| | | High | $normal-load$ |
| | Medium | Low | $normal-load$ |
| | | Medium | $normal-load$ |
| | | High | $over-load$ |
| | High | Low | $normal-load$ |
| | | Medium | $over-load$ |
| | | High | $over-load$ |
| High | Low | Low | $normal-load$ |
| | | Medium | $normal-load$ |
| | | High | $over-load$ |
| | Medium | Low | $normal-load$ |
| | | Medium | $over-load$ |
| | | High | $over-load$ |
| | High | Low | $over-load$ |
| | | Medium | $over-load$ |
| | | High | $over-load$ |

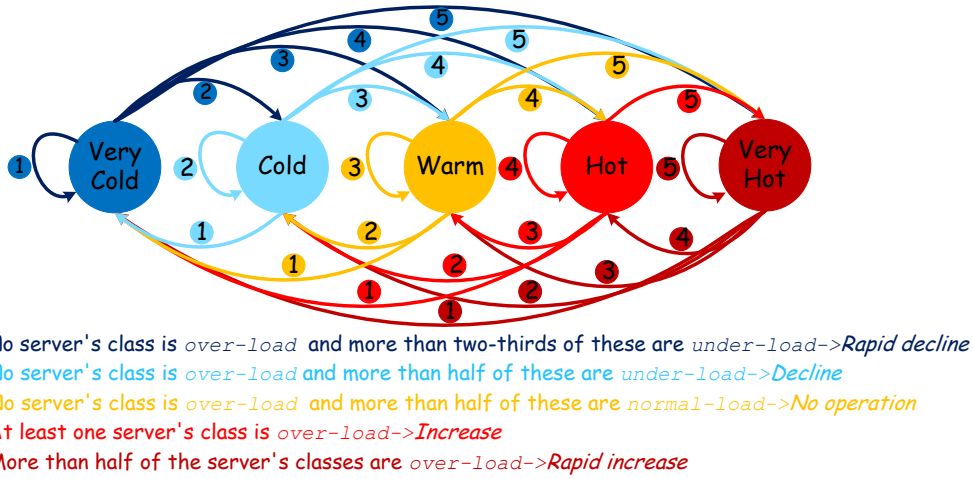



Figure 9. According to the class of PMs, the IoMV network is placed in one of these five states and the size of the network changes accordingly.

Now, according to the class of PMs, the IoMV network is categorized in one of five states: *very hot*, *hot*, *warm*, *cold*, or *very cold* (Fig. 9). Algorithm (1) is the categorization algorithm of the IoMV network in one of the abovementioned states.

In this stage, based on the state of the IoMV network and utilizing algorithms (2) to (5), modifications are made to the network size. For instance, if the state is categorized as *hot*, the *size increase* algorithm (3) is executed. In this case, migrating some VNFs from overloaded PMs to other PMs or adding new PMs to the network helps alleviate the overload condition.

Conversely, if the state is classified as *cold*, the *size decline* algorithm (4) is initiated. This algorithm facilitates the transfer of VNFs from underloaded PMs to other PMs. Consequently, idle PMs can be turned off to conserve

**Algorithm 1:** categorization of the IoMV network

**for** *($i = 1$ to number of PMs)* **do**
  **if** *(more than half of PMs are from $over-load$ class)* **then**
    {the network state is *very hot* and the *rapid increase* algorithm (2) runs};
  **else if** *(at least one PM is from $over-load$ class)* **then**
    {the network state is *hot* and the *increase* algorithm (3) runs};
  **else if** *(there are no PMs from the $over-load$ class and more than half are from the $normal-load$ class)* **then**
    {the network state is *warm* and does not need to be resized};
  **else if** *(there are no PMs from the $over-load$ class and more than half are from the $under-load$ class)* **then**
    {the network state is *cold* and the *decline* algorithm (4) runs};
  **else if** *(there are no PMs from the $over-load$ class and more than two-thirds of PMs are from the $under-load$ class)* **then**
    {the network state is *very cold* and the *rapid decline* algorithm (5) runs};
  **end**
**end**

**Algorithm 2:** rapid increase in size of IoMV network

**for** *($i = 1$ to number of overloaded PMs)* **do**
  {PM $k$ is added to the network and half of the VNFs of PM $i$ are migrated to PM $k$};
  **if** *(switch $l$ exists that is required)* **then**
    {switch $l$ is turned on};
  **end**
**end**

energy.

The main goal of algorithms (2) to (5) is to bring the network to a *warm* state, where energy and efficiency are in equilibrium, thereby eliminating the need for further adjustments. This state ensures optimal network performance and resource utilization without being underloaded or overloaded.

### 3.3 Load distribution between servers

After the size of the network and active PMs of IoMV are determined, the load is distributed among active PMs in a way that active PMs are balanced in terms of load. This occurs when the new multimedia request is sent to the least-load PM. If the least-load PM is selected within each interval $\tau$, PMs will be balanced through time. The load of each active PM is affected by all three $\bar{x}^i_{\tau+1}$, $\bar{y}^i_{\tau+1}$, and $\bar{z}^i_{\tau+1}$ (CPU, memory, hard-disc), while these variables' gender is different. Hence, formulas (10), (11), and (12) are used to make them similar:

$$Load^i_{\tau+1}(CPU) = \begin{cases} 0, & \bar{x}^i_{\tau+1} < \nabla\bar{x}^i_{\tau+1} \\ \frac{\bar{x}^i_{\tau+1} - \nabla\bar{x}^i_{\tau+1}}{\Delta\bar{x}^i_{\tau+1} - \nabla\bar{x}^i_{\tau+1}}, & \nabla\bar{x}^i_{\tau+1} < \bar{x}^i_{\tau+1} \leq \Delta\bar{x}^i_{\tau+1} \\ 1, & \bar{x}^i_{\tau+1} \geq \Delta\bar{x}^i_{\tau+1} \end{cases} \tag{10}$$

$$Load^i_{\tau+1}(Memory) = \begin{cases} 0, & \bar{y}^i_{\tau+1} < \nabla\bar{y}^i_{\tau+1} \\ \frac{\bar{y}^i_{\tau+1} - \nabla\bar{y}^i_{\tau+1}}{\Delta\bar{y}^i_{\tau+1} - \nabla\bar{y}^i_{\tau+1}}, & \nabla\bar{y}^i_{\tau+1} < \bar{y}^i_{\tau+1} \leq \Delta\bar{y}^i_{\tau+1} \\ 1, & \bar{y}^i_{\tau+1} \geq \Delta\bar{y}^i_{\tau+1} \end{cases} \tag{11}$$

$$Load^i_{\tau+1}(HardDisk) = \begin{cases} 0, & \bar{z}^i_{\tau+1} < \nabla\bar{z}^i_{\tau+1} \\ \frac{\bar{z}^i_{\tau+1} - \nabla\bar{z}^i_{\tau+1}}{\Delta\bar{z}^i_{\tau+1} - \nabla\bar{z}^i_{\tau+1}}, & \nabla\bar{z}^i_{\tau+1} < \bar{z}^i_{\tau+1} \leq \Delta\bar{z}^i_{\tau+1} \\ 1, & \bar{z}^i_{\tau+1} \geq \Delta\bar{z}^i_{\tau+1} \end{cases} \tag{12}$$

There are fixed upper and lower limits for CPU of the active PM $i$. These are referred to as $\Delta\bar{x}^i_{\tau+1}$ and $\nabla\bar{x}^i_{\tau+1}$ thresholds. The $\Delta\bar{y}^i_{\tau+1}$ and $\nabla\bar{y}^i_{\tau+1}$ thresholds also indicate limits of the memory of PM $i$. Moreover, $\Delta\bar{z}^i_{\tau+1}$ and

**Algorithm 3:** increase in size of IoMV network

```
for (i = 1 to number of overloaded PMs) do
    if (PM j from the under − load class exists) then
        {a VNF migrates from PM i to PM j};
    else
        {PM k is added to the network and one VNF is migrated from PM i to PM k};
        if (switch l exists that is required) then
            {switch l is turned on};
        end
    end
end
```

**Algorithm 4:** decline in size of IoMV network

```
for (i = 1 to number of underloaded PMs) do
    if (PM j from the under − load class exists) then
        {a VNF migrates from PM i to PM j};
        if (PM i lacks VNF) then
            {PM i is turned off};
            if (switch l exists that is idle) then
                {switch l is turned off};
            end
        end
    end
end
```

$\nabla \bar{z}^i_{\tau+1}$ are upper and lower thresholds for the hard-disc of PM $i$. Therefore, the load of active PM $i$ is calculated within interval $\tau + 1$ by using Eq. (13):

$$\begin{aligned} Load^i_{\tau+1} = \alpha Load^i_{\tau+1}(CPU) + \beta Load^i_{\tau+1}(Memory) \\ +\gamma Load^i_{\tau+1}(HardDisk), \quad Load^i_{\tau+1} \in [0,1] \end{aligned} \tag{13}$$

The coefficients $\alpha$, $\beta$, and $\gamma$ indicate the weight of CPU, memory, and hard-disc on the PM load ($\alpha+\beta+\gamma = 1$). It is possible to change the importance rate of CPU, memory, and hard-disc on the load by changing the above-mentioned coefficients. For example, if you increase the value of $\alpha$, it will increase the importance of CPU usage in determining the load, making CPU resources more influential in the overall load calculation[2]. Finally, the PM $i$ with the least $Load^i_{\tau+1}$ is selected, and the request for new multimedia is sent to it.

### 3.4 Select the route with a high QoS

This module aims to find a route that reaches the selected PM with high QoS. To do this, all potential routes are firstly considered as the candidates (set $P$). Then the bandwidth and delay are calculated for all candidate routes based on the available information in the controller. The route with the highest bandwidth and lowest delay is selected as the route with the high QoS among the candidate routes. In this lieu, the bandwidth and delay of each route of the candidate routes are calculated as follows:

$$b_p = min\{b_{i,j} | (i,j) \in p, p \in P\} \tag{14}$$

$$d_p = \sum_{(i,j)\in p} d_{i,j} \tag{15}$$

2. By tuning these coefficients, the load calculation can be customized based on the specific requirements and characteristics of your system, allowing effective management and optimization of resource allocation according to your desired priorities. This flexibility is beneficial in tailoring the system's behavior to different scenarios and workloads.

**Algorithm 5:** rapid decline in size of IoMV network

```
for (i = 1 to number of underloaded PMs) do
    if (PM j from the under − load class exists) then
        {migration of all VNFs from PM i to PM j};
        {PM i is turned off};
        if (switch l exists that is idle) then
            {switch l is turned off};
        end
    end
end
```

Eq. (14) explains the bandwidth measurement in the path, where $b_p$ is the available bandwidth for the path $p$ and $b_{i,j}$ is the available bandwidth for any link $(i, j)$ in the path. Available link bandwidth is calculated based on the link utilization rate $u_{i,j}$ and maximum possible link capacity $B_{i,j}$ through equation $b_{i,j} = B_{i,j} \times (1 - u_{i,j})$. Eq. (15) formulates the delay in the path $p$ where $d_p$ refers to the path delay and $d_{i,j}$ refers to the delay of each link $(i, j) \in p$.

### 3.5 Select the multimedia with high QoE

We use E-model to decide about the QoE. This model is designed based on the recommendation of ITU-T (ITU-T G.107) and is widely used to estimate the quality in the planning phase of multimedia networks and during operations. This model calculates the *Rating Factor (R)* range $(0-100)$ regarding the different parameters affecting media quality. How to calculate *R* has been expressed in the paper [33]. $R = 0$ indicates the worst quality, while $R = 100$ implies the best quality. The $R$ is converted to the *Mean Opinion Score (MOS)* based on the Eq. (16). MOS rate specifies the multimedia quality according to the *R Factor* [34].

$$MOS = \begin{cases} 1, R < 0 \\ 1 + 0.0535R + R(R-60)(100-R) \times 7 \times 10^{-6}, 0 \leq R \leq 100 \\ 4.5, R > 100 \end{cases} \tag{16}$$

MOS rate varies between $1$ and $4.5$, which $1$ and $4.5$ indicate the lowest and highest quality, respectively. In this module, according to the MOS of available multimedia in the selected PM, the highest one is chosen to be provided for the requesting vehicle.

## 4 Implementing and Performance Evaluation

Fig. 10 illustrates the field of testing the $ELQ^2$ framework. This field comprises six PMs for implementing VNFs of IoMV (P1-P6), two PMs for OpenFlow switches (S1 and S2), and one PM for implementing the proposed controller (C1). The hardware of these PMs is heterogeneous, as detailed in Table 2. The software used includes *Floodlight v1.2* [35] for the controller, *Open vSwitch v2.4.1 (OvS)*[3] [36] for the switches, and *Kamailio v4.3.6* [37] for the multimedia VNFs of IoMV. The modules designed in Section 3 are executed on the controller (C1). To evaluate the performance, *iPerf* software is utilized to create multimedia traffic, while *StarTrinity* software measures various network parameters and multimedia quality metrics. The links within the test field have a bandwidth of 10 Mbps, and the parameters $\phi$ and $\mu$ are set to 30 and 0.8, respectively. Moreover, the values of $\alpha$, $\beta$, and $\gamma$ are chosen as 0.33. The results obtained from evaluation and comparison between $ELQ^2$ and articles [18]–[23] have been provided regarding four part: *energy*, *load*, *QoS*, and *QoE*. These articles were reviewed in Section 2.

### 4.1 Evaluation of energy

Energy is evaluated based on some criteria, including power usage, the number of VNFs, the number of OvSs, and the number of active PMs and switches. As seen in Fig. 11, six servers (P1-P6) are linked through two

3. OpenFlow virtual Switches

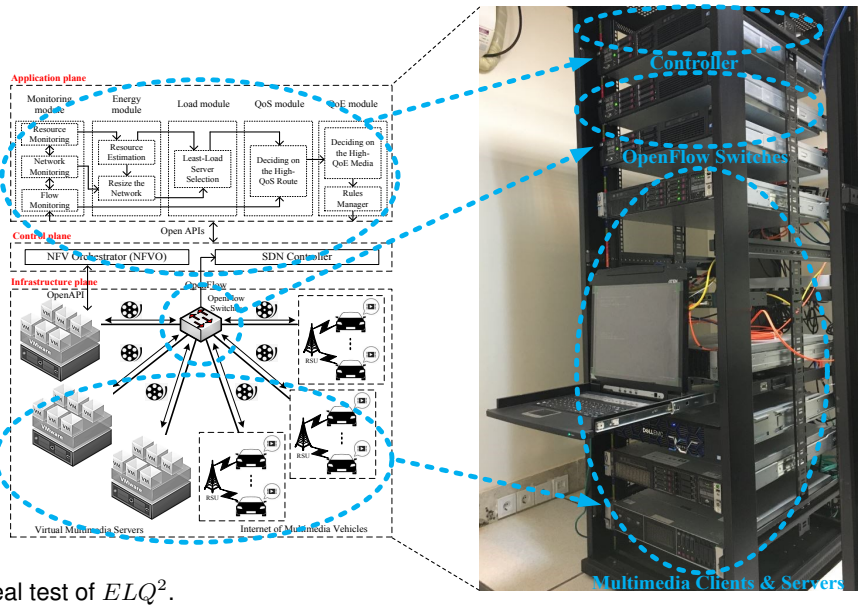


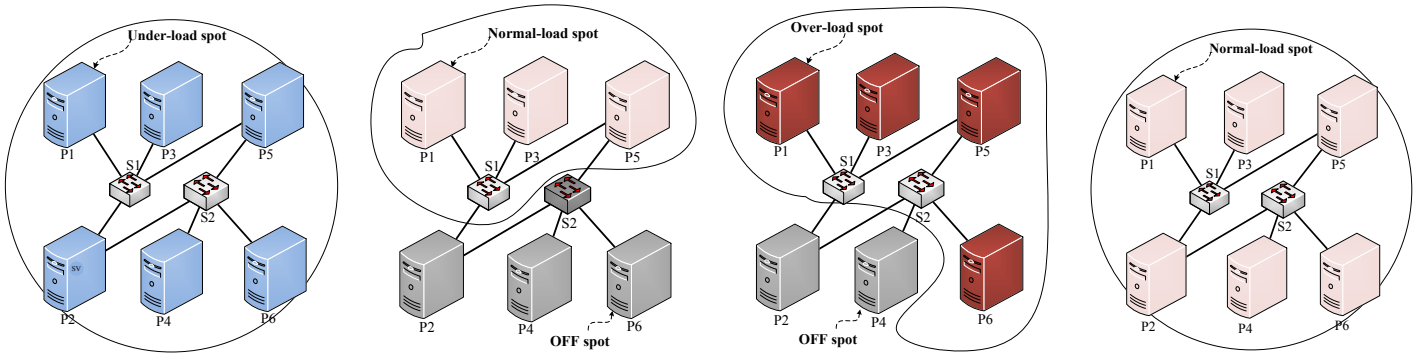

Figure 10. The field for the real test of $ELQ^2$.

Table 2
$ELQ^2$ testbed characteristics

| Name | Hardware | | | Software | Role |
|---|---|---|---|---|---|
| | Memory | CPU | HDD | | |
| C1 | *32 GB DDR3* | *Intel Xeon E5-2600* | *150 GB SATA* | *Floodlight v1.2* | *Controller* |
| S1 | *8 GB DDR3* | *Intel Xeon E5500* | *75 GB SATA* | *Open vSwitch v2.4.1* | *Switch* |
| S2 | *8 GB DDR3* | *Intel Xeon E5500* | *75 GB SATA* | *Open vSwitch v2.4.1* | *Switch* |
| P1 | *8 GB DDR3* | *Intel Xeon E5500* | *75 GB SATA* | *Kamailio v4.3.6* | *Server* |
| P2 | *16 GB DDR3* | *Intel Xeon E5600* | *100 GB SATA* | *Kamailio v4.3.6* | *Server* |
| P3 | *32 GB DDR3* | *Intel Xeon E5-2600* | *150 GB SATA* | *Kamailio v4.3.6* | *Server* |
| P4 | *32 GB DDR* | *Intel Xeon E5-2600* | *150 GB SATA* | *Kamailio v4.3.6* | *Server* |
| P5 | *8 GB DDR3* | *Intel Xeon E5500* | *75 GB STAT* | *Kamailio v4.3.6* | *Server* |
| P6 | *16 GB DDR3* | *Intel Xeon E5600* | *100 GB STAT* | *Kamailio v4.3.6* | *Server* |

switches (S1 and S2). The management of this infrastructure is the responsibility of the $ELQ^2$ controller, which has been able to manage the size and resources of the network well, and change the state of the network from *cold* and *hot* states (Fig. 11a and 11c) to *warm* state (Fig. 11b and 11d). The *warm* (normal) state is the desired mode for the network.

Fig. 12 depicts the detailed results of energy evaluation. Among papers [18]–[23], only papers [18] and [20] considered energy too. Hence, these two papers are compared with $ELQ^2$ regarding energy metrics. To do this, variable multimedia traffic is injected into the network for 2000 seconds based on the stochastic pattern of Fig. 12a. Figs. 12b and 12c depict the number of VNFs and OvSs, respectively. As can be seen, these numbers follow the traffic pattern. However, $ELQ^2$ relies on fewer VNFs and OvSs for its network than [18] or [20]. For instance,




(a) before decrease size (cold state) (b) after decrease size (warm state) (c) before increase size (hot state) (d) after increase size (warm state)

Figure 11. Proper resize of IoMV network and moving towards the warm state (from a to b and c to d).

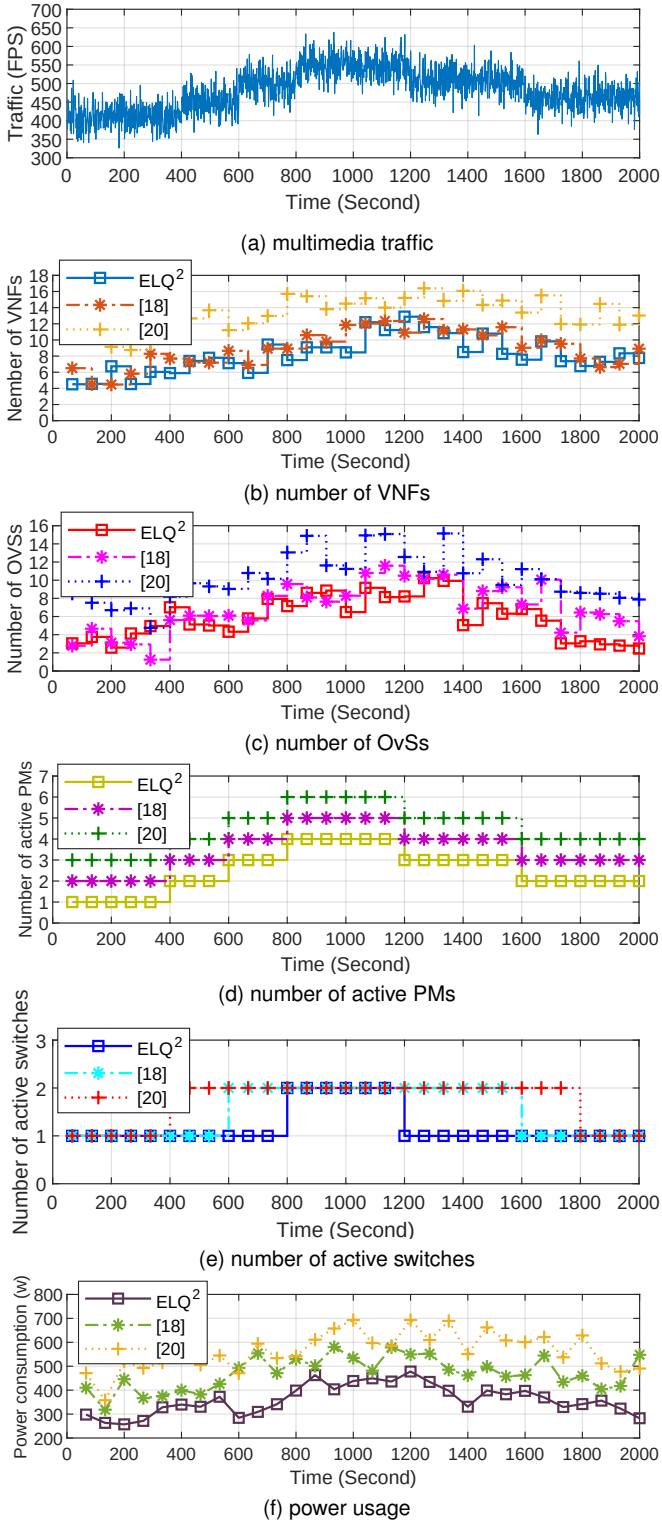
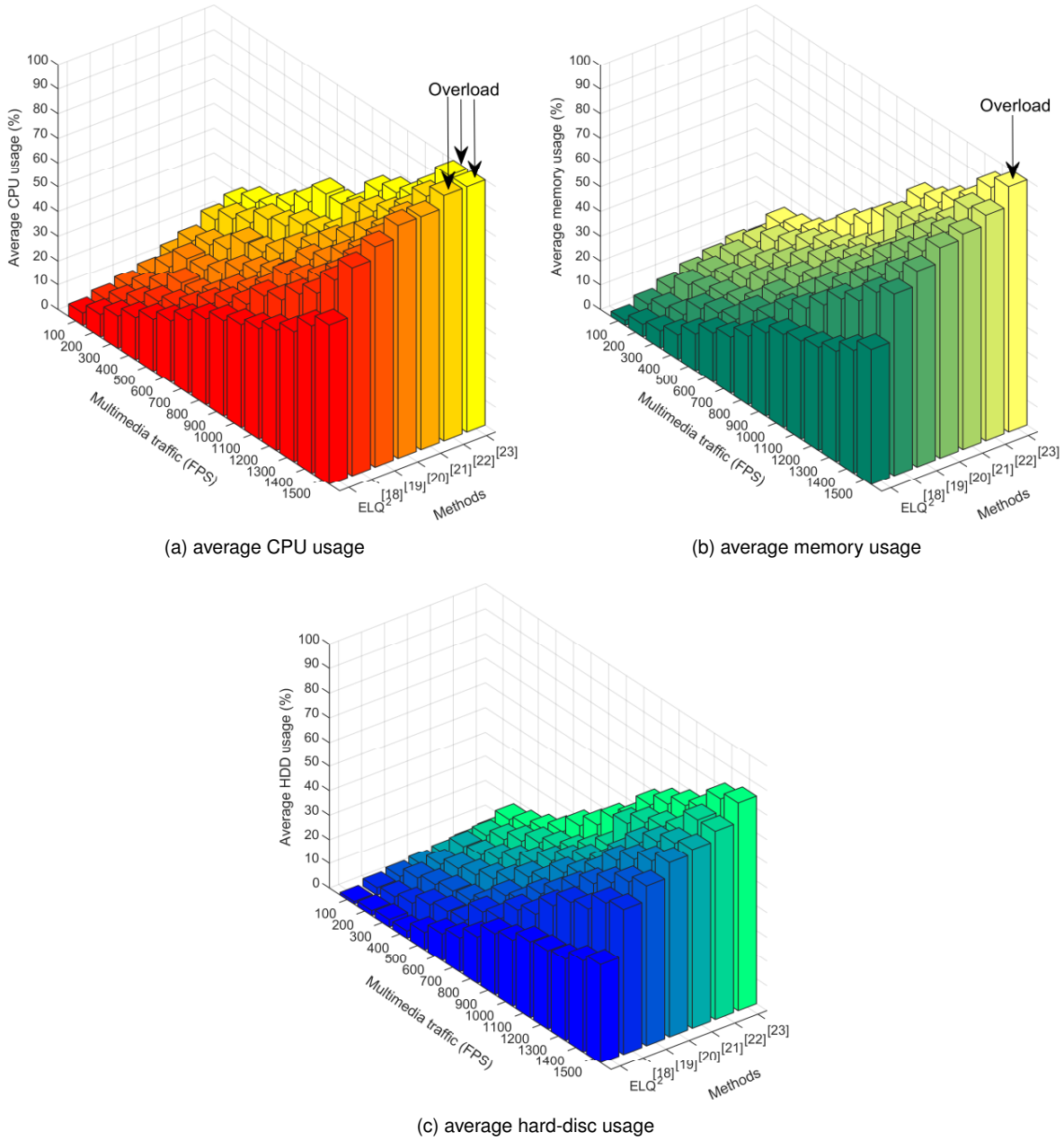



Figure 12. Energy evaluation based on the input multimedia traffic.

at the 1000th second, when traffic is almost 550 Flow per Second (FPS), the number of VNFs in the $ELQ^2$ method equals 8 while this value equals 12 and 14 in studies [18] and [20], respectively. The number of OvSs at the 1000th second is 6, 8, and 11 respectively for these three methods. It should be noted that the curve [18] more resembles the $ELQ^2$ curve because both methods (unlike method [20]) are predictive and predict the load pattern. Nevertheless, $ELQ^2$ provides a higher prediction precision and better performance.

In addition to an assessment of the number of virtual functions and switches (VNFs and OvSs), the extant study examines the number of physically active PMs and switches. Accordingly, one can find the amount of physical hardware required to run VNFs and OvSs in different methods. As shown in parts d and e of Fig. 12,



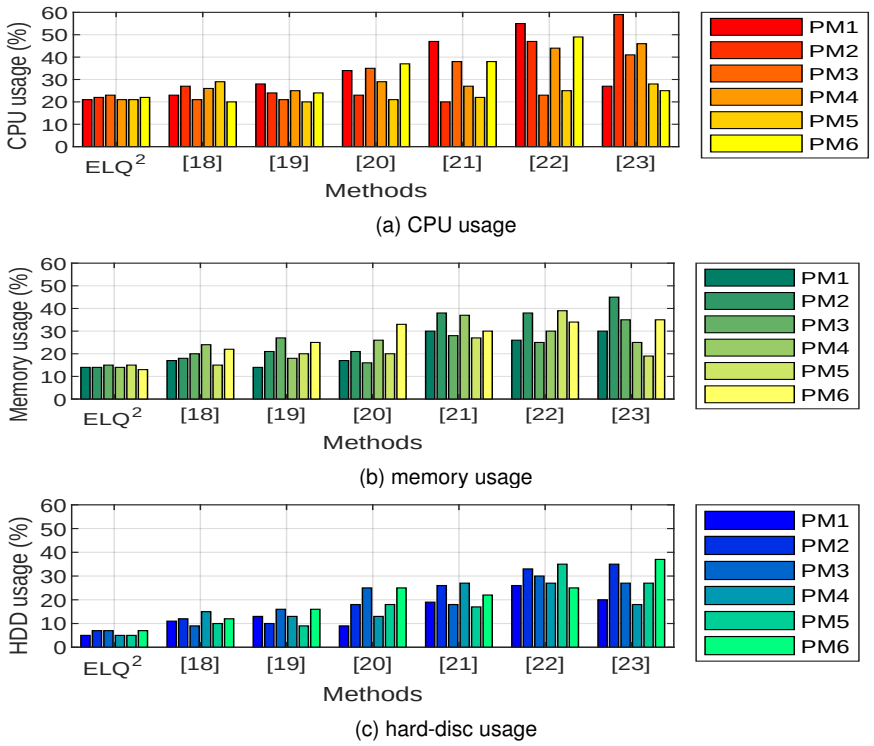

(a) average CPU usage

(b) average memory usage

(c) average hard-disc usage

Figure 13. The average usage of PMs' resources in different methods and traffics.

the obtained results demonstrate a correlation between the number of physically active PMs and switches and the traffic pattern. As it was expected, these numbers also follow the traffic pattern. Since the $ELQ^2$ method uses fewer VNFs and OvSs, this method also employs fewer PMs and switches compared to the other methods. For instance, the method [20] activates all $6$ available PMs during the $800$-$1200$s when the traffic pattern is in peak time. On the contrary, there are $5$ and $4$ active PMs in methods [18] and $ELQ^2$, respectively; it means $ELQ^2$ drives the highest traffic with the fewest active PMs ($4$ active PMs and $2$ inactive PMs to save energy). Also, there are fewer active switches in $ELQ^2$ (Fig. 12e). Of the two available switches, the method [20] uses both switches during the $400$-$1800$s interval. Moreover, the method [18] keeps both switches active within $600$-$1600$s interval. However, method $ELQ^2$ use both switches only during the short-term $800$-$1200$s interval while handling its task only with one switch during the rest of the time (to save energy). In total, all of the mentioned cases make the power consumption of the $ELQ^2$ far less than the other two methods (Fig. 12f).




(a) CPU usage

(b) memory usage

(c) hard-disc usage

Figure 14. PMs' resources consumption in traffic with $500$ FPS load.

## 4.2 Evaluation of load and resources

In this subsection, evaluation metrics comprise the average consumption of CPU, memory, hard-disc, and how the load is distributed between PMs. Fig. 13 illustrates the results. This figure indicates the average usage of PMs' resources (CPU, memory, and hard-disc - axis z) through traffic rise (axis x) in different methods (axis y). Traffic begins from $100$ FPS and reaches $1500$ FPS. Any increase in traffic leads to higher resource use, although this is not an identical increase in different methods. The lowest increase rate in resource use occurs in $ELQ^2$ method. Unlike the $ELQ^2$ method, methods [22] and [23] indicate the highest growth because only these two methods - among methods used in studies [18]–[23] - are not SDN-based. However, the method [19] also is not SDN-based, but its logic is centralism and has a hierarchical structure. Because SDN has a centralized controller and a global view of the network, it has a better integrated management of the network's resources. Of course, the proper design of components and modules of controller is effective in improving the efficiency of SDN-based methods. This issue is evident in the results of $ELQ^2$ and [18]–[21] methods in Fig. 13.

An important aspect to note is that all the mentioned methods, such as those in [22] and [23], are CPU-centric, meaning they heavily rely on CPU processing rather than focusing on memory or hard-disc usage. Consequently, during periods of heavy traffic, the CPUs of PMs become saturated, leading to system overload. Methods [22] and [23] are overloaded (regarding CPU) in the $1500$ FPS load and $1400$ FPS load, respectively. Moreover, PMs' memory are overloaded in the $1500$ FPS load in method [23]. The overload causes devastating consequences, such as a throughput decline for the system. However, the $ELQ^2$ method handles the resource usage very well, even in heavy traffic of $1500$ FPS. Accordingly, less than half of the resources are used without saturation in the $ELQ^2$ method. This advantage contributes to an optimal throughput (investigated in the next subsection), which $ELQ^2$ achieves substantially.

Fig. 14 shows the resource consumption of individual PMs in different methods. This figure depicts how the load is distributed among PMs during traffic with $500$ FPS load. As you can see, in the $ELQ^2$ method, not only the consumption of resources is less, but also the load distribution between PMs is balanced. In contrast, for example in the method [22], PM1 and PM2 have high CPU usage, while CPU usage is low in PM3 (Fig. 14a). A milder aspect of this case is true in the case of memory and hard-disc (Figs. 14b and 14c).

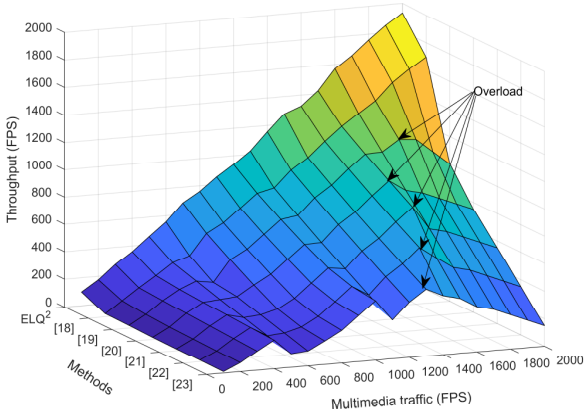



Figure 15. Average PMs' throughput.

### 4.3 Evaluation of QoS

Throughput and delay are metrics used to evaluate QoS. The load 100 - 2000 FPS is injected into the system in experiments of this section. Figs. 15 and 16 depict the average throughput and delay of PMs in different methods. Throughput indicates the number of multimedia flows that have been established successfully. An increase in load leads to a throughput increase until the system's capacity is full and resources are saturate. This is when an overload occurs. Overload leads to descending throughput. Most researches aim to keep the throughput fixed in the overload or postpone it through resource management. Fig. 15 indicates the throughput of different methods and their successfulness rate in network resource management. As can be seen, only $ELQ^2$ and [18] methods have not been overloaded up to the 2000 FPS load. However, $ELQ^2$ achieves a higher throughput compared to the method used in the paper [18]. For instance, the throughput of $ELQ^2$ and [18] equal 1758 FPS and 1530 FPS, respectively in 1800 FPS traffic. The reason for the superiority of $ELQ^2$ over [18] is its controller simplicity and agility. Five other methods have been overloaded, and their throughput has declined in different traffics due to the lack of centralized and intelligent controller or sophistication of resource management processes in these methods. The overload point of these methods has been illustrated in the figure. As saw in Fig. 13a, methods used in studies [22] and [23] faced CPU saturation in 1500 FPS and 1400 FPS traffic, respectively. As demonstrated in Fig. 15, the mentioned methods, which are CPU-centric, have encountered overload and experienced a decline in throughput under the same traffic conditions. The overload condition not only reduces the throughput but also results in an increase in delay, as shown in Fig. 16. In contrast, the $ELQ^2$ method, which utilizes resource-aware management, exhibits significantly lower delay and linear growth in delay. The resource-aware management approach effectively balances the network's resources, ensuring efficient utilization and optimal performance even under heavy traffic loads. This results in a stable and consistently low delay, leading to improved overall system performance and user experience. On the other hand, other methods that do not employ resource-aware management suffer from high delay with exponential growth, which worsens under overload conditions. The lack of efficient resource management in these methods leads to performance degradation and a decline in the quality of service, especially when the network faces heavy traffic. Overall, the $ELQ^2$ method's resource-aware management strategy offers a more robust and scalable solution, delivering superior performance and maintaining low delay even in demanding scenarios, while other CPU-centric methods struggle to cope with overload conditions and experience significant performance degradation.

The next experiment of this section examine the performance of $ELQ^2$'s controller more precisely within five scenarios. According to Fig. 17, five scenarios (from very low to very high load) are run for 200 seconds (40 seconds for each scenario). The blue curve represents the input traffic, and the red curve is the goodput of $ELQ^2$'s controller. As it is seen, the goodput is highly near to the input load in all five scenarios. It means that the $ELQ^2$'s controller has handled almost all input traffic well (even in the very high load scenario). For instance, the controller has deployed 1140 flows out of 1200 flows in the second 161 (fifth scenario). Accordingly, the flows have been deployed with a 95% success rate because of the SDN-based and all-around management of all network resources.

In Fig. 18, the resource usage of $ELQ^2$'s controller is monitored, and it is observed that the resource consumption, specifically CPU usage, follows a linear growth pattern in accordance with the traffic pattern. Even in the fifth scenario with the highest input traffic, resource consumption reaches a maximum rate of 56%.

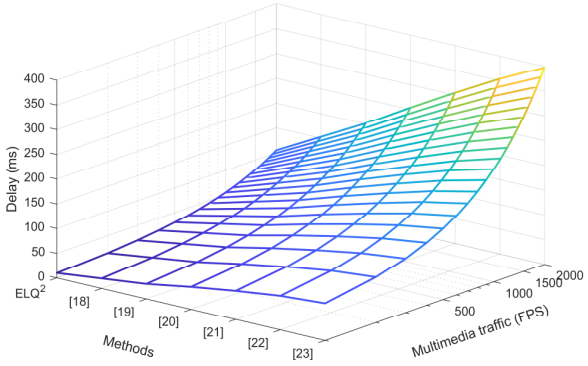



Figure 16. Average PMs' delay.

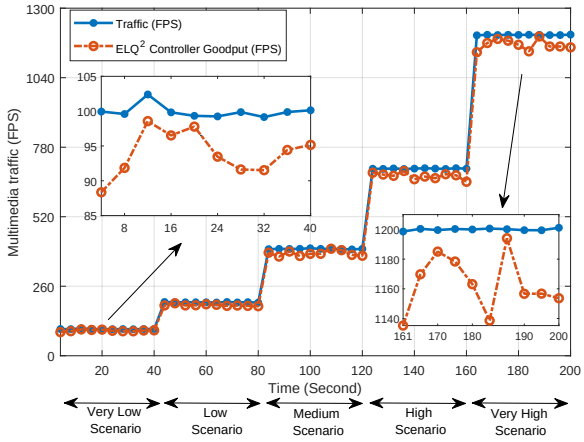



Figure 17. $ELQ^2$'s controller goodput.

It is essential to note that almost half of this rate (25%) is attributed to the controller kernel, while the proposed modules account for the remaining portion.

So, the designed modules within $ELQ^2$ do not saturate the controller's resources, thereby avoiding the controller from becoming a bottleneck in the system. Among the modules, the *Energy* module exhibits the highest CPU usage (10%) during the very high-load scenario due to executing the NLMS algorithm and fuzzy engine. However, this value is typically lower in other scenarios.

In the rest of this subsection, we investigate whether $ELQ^2$'s controller can cope with the network equipment failure and prevent the subsequent throughput decline. The experiment of Figs. 19 and 20 is designed to answer

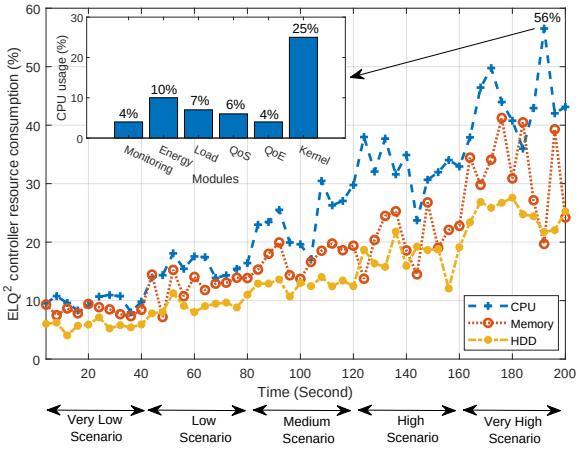



Figure 18. Resource usage by $ELQ^2$'s controller (and resource usage by designed modules).

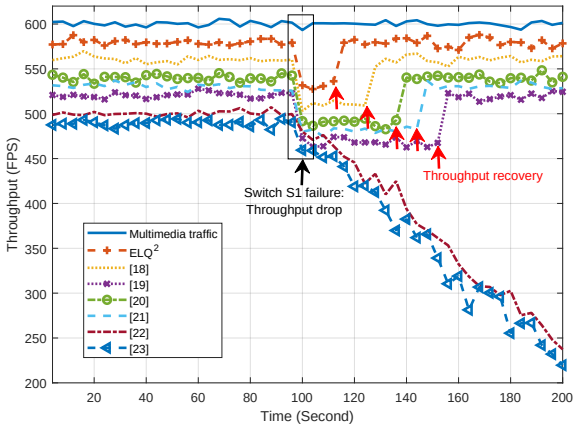



Figure 19. Analyzing throughput of different methods after switch S1 idleness at second 100.

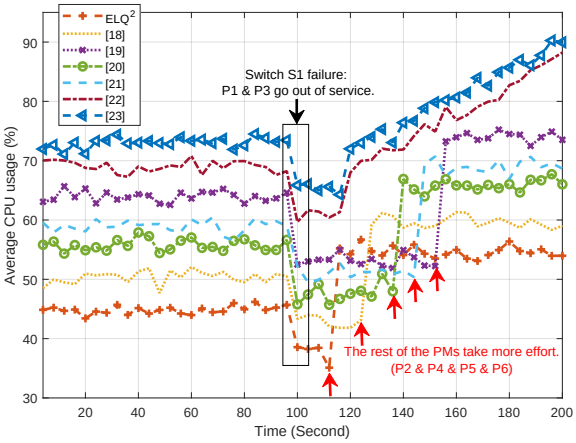



Figure 20. Average consumption of PMs' CPU in different methods after switch S1 idleness at second 100.

the mentioned question. In this case, 600 FPS traffic is injected into the IoMV network for 200 seconds, and the S1 switch will fail at the second 100. Servers P1 and P3 leave the network at this moment regarding the network topology (Fig. 11). This subject leads to a throughput decline (Fig. 19) and a temporary reduction of CPU usage (Fig. 20). Among the methods, only $ELQ^2$ and [18]–[21] methods can recover throughput because these methods handle the resources through a centralized or hierarchical structure. In contrast, methods [22] and [23] are only used for load balance between servers and do not have a cohesive mechanism for orchestrated network resource management. Moreover, these methods do not have up-to-date information about the network switches' state. Hence, the network faces an overload after switch S1 failure in these two methods. In this case, these methods cannot prevent throughput decline (Fig. 19) despite the high resource usage (Fig. 20). Among methods that can recover throughput, $ELQ^2$ does this as quickly as possible (< the 20s), while this takes more than 50 seconds in the study [19] (Fig. 19). The high operating speed of $ELQ^2$ is rooted in its centralized controller that can rapidly collect new network information (servers, switches, and links) and export the required commands for flows' redirect towards the remaining servers (P2, P4, P5, P6). Therefore, average resource usage will be increased because four servers exist in the circuit instead of six servers. However, throughput returns to the pre-failure value of S1.

### 4.4 Evaluation of QoE

MOS and R Factor are used to evaluate the QoE of multimedia. MOS varies between 1 and 4.5; the higher the MOS, the higher the QoE value. R Factor varies between 0 and 100. The higher the R Factor, the higher the QoE will be. Table 3 reports the values of these two metrics for different methods and scenarios. Upon reviewing the values presented in this table, it becomes evident that the $ELQ^2$ method, incorporating the QoE module in its controller, has achieved the highest MOS and R Factor values among the different methods and scenarios

Table 3
Evaluation of MOS and R Factor parameters in different scenarios and methods

| Scenario | Very low laod | | Low laod | | Medium laod | | High load | | Very high load | |
|---|---|---|---|---|---|---|---|---|---|---|
| Traffic | 100 FPS | | 200 FPS | | 400 FPS | | 700 FPS | | 1200 FPS | |
| Factor | MOS | R Factor | MOS | R Factor | MOS | R Factor | MOS | R Factor | MOS | R Factor |
| $ELQ^2$ | 4.4919 | 94.2350 | 4.4774 | 91.8580 | 4.4266 | 88.6457 | 4.3503 | 84.2956 | 4.2234 | 80.5897 |
| [18] | 4.4882 | 90.6483 | 4.3933 | 87.5348 | 4.2853 | 81.9424 | 4.2364 | 75.3044 | 4.1057 | 70.7278 |
| [19] | 3.8865 | 81.4678 | 3.7424 | 74.6393 | 3.5692 | 70.4190 | 3.2918 | 65.9624 | 3.0908 | 60.6436 |
| [20] | 4.4507 | 88.2956 | 4.2054 | 85.6384 | 3.8946 | 80.0279 | 3.6248 | 73.6497 | 3.4776 | 68.3772 |
| [21] | 3.9023 | 85.6375 | 3.7832 | 80.2280 | 3.6124 | 75.9346 | 3.5033 | 70.4308 | 3.3334 | 64.8468 |
| [22] | 3.7933 | 80.0297 | 3.6655 | 71.5357 | 3.4806 | 68.7464 | 3.2524 | 60.7246 | 3.0567 | 55.9363 |
| [23] | 3.7754 | 78.4844 | 3.6067 | 74.7356 | 3.4248 | 65.4705 | 3.114 | 56.7329 | 2.6889 | 51.8705 |

evaluated. This achievement, along with ensuring QoS, optimizing energy consumption, and managing load, is a turning point in this field of research. The highest MOS equaled $4.4919$, indicating the high multimedia quality that IoMV users experience. Another interesting point of the $ELQ^2$ method is that traffic increase does not exclude the multimedia quality from the standard range, and MOS and R Factor are not decreased to less than $4.2234$ and $80.5897$. This is while in other methods even a very low value of $2.6889$ is observed for MOS.

## 5 Conclusion and Further Studies

The extant study examined a critical and sophisticated problem in the IoMV network due to the increasing expansion of multimedia traffic in this network. This problem considered IoMV network resource management to achieve the following objectives simultaneously: 1) ensuring QoS and QoE, 2) load balance, and 3) optimal energy usage. To do this, the new SDN and NFV-based $ELQ^2$ framework was designed, implemented, and evaluated. The modular controller of this framework is equipped with a robust NLMS algorithm for traffic prediction and an efficient fuzzy engine for the classification of PMs. Some algorithms were also designed for network resizing: 1) the network border is enhanced (active equipment is increased), and efficiency is increased during peak times. 2) the network border is decreased (active equipment is reduced), and energy usage is saved during non-peak times. Moreover, some other algorithms were designed as modules of this controller: selecting the least-load PM, selecting a route with high QoS, and selecting the multimedia with high QoE. The $ELQ^2$ was implemented on a real testing field, and the results indicated that this method works better, more rapidly, and more informed in resource management rather than other methods. Various scenarios were designed and implemented to approve this topic. Moreover, various criteria, including power usage, throughput, delay, resource usage, and MOS were examined and analyzed. The evaluation of $ELQ^2$ has demonstrated its intelligent and rapid response to network failures, effectively preventing any decline in throughput. The controller's ability to quickly identify and address network failures contributes to maintaining a stable and reliable multimedia streaming experience. Additionally, the assessment of throughput and delay further highlights $ELQ^2$'s capabilities in dealing with overload situations and resource saturation. The framework's resource-aware management approach allows it to efficiently allocate resources and optimize performance, even under heavy traffic loads. In further studies, we want to use machine learning concepts for the automatic training of $ELQ^2$ controller. Moreover, we want to develop $ELQ^2$ in a way that can control urban and network traffics simultaneously. In this way, vehicles routing and network flows routing are done intelligently in a centralized method. On the other hand, one can gather urban traffic data when collecting network traffic data. In this way, the controller will have a global view of urban transportation infrastructure and subsequent incidents and also the network infrastructure. Therefore, the optimal route of vehicles (in terms of delay, for example) is discovered simultaneously when an optimal route is found for network streaming.

## References


[1] L. J. G. Villalba, J. A. P. Fernández, and A. L. S. Orozco, "Analysis of mp4 videos in 5g using sdn," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, pp. 2668–2677, 2022.

[2] M. Xing, J. He, and L. Cai, "Maximum-utility scheduling for multimedia transmission in drive-thru internet," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 4, pp. 2649–2658, 2015.

[3] S. Zou, H. Chen, H. Feng, G. Xiao, Z. Qin, and W. Cai, "Traffic flow video image recognition and analysis based on multi-target tracking algorithm and deep learning," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[4] K. N. Qureshi, S. Din, G. Jeon, and F. Piccialli, "Internet of vehicles: Key technologies, network model, solutions and challenges with future aspects," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1777–1786, 2020.

[5] X. Xu, Z. Fang, L. Qi, X. Zhang, Q. He, and X. Zhou, "Tripres: Traffic flow prediction driven resource reservation for multimedia iov with edge computing," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 17, no. 2, pp. 1–21, 2021.

[6] D. Wu, L. Deng, H. Wang, K. Liu, and R. Wang, "Similarity aware safety multimedia data transmission mechanism for internet of vehicles," *Future Generation Computer Systems*, vol. 99, pp. 609–623, 2019.

[7] C. Chen, X. Liu, T. Qiu, L. Liu, and A. K. Sangaiah, "Latency estimation based on traffic density for video streaming in the internet of vehicles," *Computer Communications*, vol. 111, pp. 176–186, 2017.

[8] K. Lin, F. Xia, and G. Fortino, "Data-driven clustering for multimedia communication in internet of vehicles," *Future Generation Computer Systems*, vol. 94, pp. 610–619, 2019.

[9] Z. Lv, L. Qiao, and H. Song, "Analysis of the security of internet of multimedia things," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 16, no. 3s, pp. 1–16, 2020.

[10] Q. Xin, M. Alazab, R. G. Crespo, and C. E. Montenegro-Marin, "Ai-based quality of service optimization for multimedia transmission on internet of vehicles (iov) systems," *Sustainable Energy Technologies and Assessments*, vol. 52, p. 102055, 2022.

[11] T. Abar, A. Rachedi, A. ben Letaifa, P. Fabian, and S. El Asmi, "Fellowme cache: Fog computing approach to enhance (qoe) in internet of vehicles," *Future Generation Computer Systems*, vol. 113, pp. 170–182, 2020.

[12] R. Saha, C. Roy, and S. Misra, "Soft-safe: Software defined safety-as-a-service for intelligent transportation system," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[13] S. Wan, R. Gu, T. Umer, K. Salah, and X. Xu, "Toward offloading internet of vehicles applications in 5g networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4151–4159, 2020.

[14] B. Cao, Z. Sun, J. Zhang, and Y. Gu, "Resource allocation in 5g iov architecture based on sdn and fog-cloud computing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3832–3840, 2021.

[15] M. M. Islam, M. T. R. Khan, M. M. Saad, and D. Kim, "Software-defined vehicular network (sdvn): A survey on architecture and routing," *Journal of Systems Architecture*, vol. 114, p. 101961, 2021.

[16] A. Boukerche and N. Aljeri, "An energy-efficient controller management scheme for software-defined vehicular networks," *IEEE Transactions on Sustainable Computing*, vol. 7, no. 1, pp. 61–74, 2021.

[17] H. Zhang and X. Lu, "Vehicle communication network in intelligent transportation system based on internet of things," *Computer Communications*, vol. 160, pp. 799–806, 2020.

[18] A. Montazerolghaem, M. H. Yaghmaee, and A. Leon-Garcia, "Green cloud multimedia networking: Nfv/sdn based energy-efficient resource allocation," *IEEE Transactions on Green Communications and Networking*, vol. 4, no. 3, pp. 873–889, 2020.

[19] B. Addis, D. Ardagna, B. Panicucci, M. S. Squillante, and L. Zhang, "A hierarchical approach for the resource management of very large cloud platforms," *IEEE Transactions on Dependable and Secure Computing*, vol. 10, no. 5, pp. 253–272, 2013.

[20] J. Son, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya, "Sla-aware and energy-efficient dynamic overbooking in sdn-based cloud data centers," *IEEE Transactions on Sustainable Computing*, vol. 2, no. 2, pp. 76–89, 2017.

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

[22] A. Montazerolghaem, S.-K. Shekofteh, M. Yaghmaee, and M. Naghibzadeh, "A load scheduler for sip proxy servers: design, implementation and evaluation of a history weighted window approach," *International Journal of Communication Systems*, vol. 30, no. 3, p. e2980, 2017.

[23] H. Jiang, A. Iyengar, E. Nahum, W. Segmuller, A. N. Tantawi, and C. P. Wright, "Design, implementation, and performance of a load balancer for sip server clusters," *IEEE/ACM transactions on networking*, vol. 20, no. 4, pp. 1190–1202, 2012.

[24] S. R. Pokhrel, "Software defined internet of vehicles for automation and orchestration," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3890–3899, 2021.

[25] D. Jiang, Z. Wang, L. Huo, and S. Xie, "A performance measurement and analysis method for software-defined networking of iov," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3707–3719, 2020.

[26] X. Wang, C. Wang, J. Zhang, M. Zhou, and C. Jiang, "Improved rule installation for real-time query service in software-defined internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 2, pp. 225–235, 2016.

[27] A. H. Sodhro, J. J. Rodrigues, S. Pirbhulal, N. Zahid, A. R. L. de Macedo, and V. H. C. de Albuquerque, "Link optimization in software defined iov driven autonomous transportation system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3511–3520, 2020.

[28] D. Gupta, S. Rani, S. H. Ahmed, S. Garg, M. J. Piran, and M. Alrashoud, "Icn-based enhanced cooperative caching for multimedia streaming in resource constrained vehicular environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4588–4600, 2021.

[29] J. Chen, H. Zhou, N. Zhang, W. Xu, Q. Yu, L. Gui, and X. Shen, "Service-oriented dynamic connection management for software-defined internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 10, pp. 2826–2837, 2017.

[30] H. Khan, S. Samarakoon, and M. Bennis, "Enhancing video streaming in vehicular networks via resource slicing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 3513–3522, 2020.

[31] A. H. Sodhro, A. Gurtov, N. Zahid, S. Pirbhulal, L. Wang, M. M. U. Rahman, M. A. Imran, and Q. H. Abbasi, "Toward convergence of ai and iot for energy-efficient communication in smart homes," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9664–9671, 2020.

[32] A. H. Sodhro, Y. Li, and M. A. Shah, "Green and friendly media transmission algorithms for wireless body sensor networks," *Multimedia Tools and Applications*, vol. 76, pp. 20 001–20 025, 2017.

[33] S. Jelassi, G. Rubino, H. Melvin, H. Youssef, and G. Pujolle, "Quality of experience of voip service: A survey of assessment approaches and open issues," *IEEE Communications surveys & tutorials*, vol. 14, no. 2, pp. 491–513, 2012.

[34] A. A. Barakabitze, N. Barman, A. Ahmad, S. Zadtootaghaj, L. Sun, M. G. Martini, and L. Atzori, "Qoe management of multimedia streaming services in future networks: a tutorial and survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 526–565, 2019.

[35] "Floodlight controller," https://github.com/floodlight/floodlight, accessed: 2022-10-30.

[36] "Open vswitch," www.openvswitch.org, accessed: 2022-10-30.

[37] "Kamailio," https://www.kamailio.org/, accessed: 2022-10-30.