The Journal of Supercomputing (2022) 78:14471–14503
https://doi.org/10.1007/s11227-022-04448-w

Softwarization and virtualization of VoIP networks

Ahmadreza Montazerolghaem[1]

Accepted: 13 March 2022 / Published online: 4 April 2022

Abstract
Nowadays, voice over IP (VoIP) is a cost-effective and efficient technology in the communications industry. Free applications for transferring multimedia on the Internet are becoming more attractive and pervasive day by day. Nevertheless, the traditional, close, and hardware-defined nature of the VoIP networks' structure makes the management of these networks more complicated and costly. Besides, its elementary and straightforward mechanisms for routing call requests have lost their efficiency, causing some problems, such as SIP servers' overload. In order to tackle these problems, we introduce VoIP network softwarization and virtualization and propose two novel frameworks in this article. In this regard, we take advantage of the SDN and NFV concepts such that we separate data and control planes and provide the possibility for centralized and softwarized control of this network. This matter leads to effective routing. The NFV also makes this network's dynamic resource management possible by functions virtualization of the VoIP network. The proposed frameworks are implemented in a real testbed, including Open vSwitch and Floodlight, examined by various scenarios. The results demonstrate an improvement in signaling and media quality in the VoIP network. As an example, the average throughput and resource efficiency increased by at least 28% and the average response time decreased by 34%. The overall latency has also been reduced by almost 39%.

Keywords Softwarized VoIP networks · Software-defined networking (SDN) · Network function virtualization (NFV) · OpenFlow

1 Introduction

Nowadays, the VoIP network [1, 2] is highly expanding. Broadness and variety of IP services have led various networks to the VoIP network. This network consists of two main phases: signaling and media.

✉ Ahmadreza Montazerolghaem
a.montazerolghaem@comp.ui.ac.ir

[1] Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran

The SIP protocol [3, 4] is for the VoIP calls signaling. Besides, by the expansion of the wireless 5G technologies [5] by the carriers, the SIP protocol has been employed significantly. The information required for holding a session is exchanged through SIP signaling. The SIP signaling is carried out by forwarding the request and response messages through SIP proxy servers between the caller (or UAC) and callee (or UAS). In order to reach the destination (callee), the messages need routing.

After the signaling phase, the media phase is for media exchange. The route of signaling messages is independent of the media route. The media is transferred through the network switches and the RTP protocol [6]. The signaling messages pass the network switches and also the SIP proxies. The SIP proxies are responsible for routing the seventh layer of messages, and the network's switches are responsible for routing the third layer of messages [7]. Improper routing of messages leads to the wrong usage of SIP proxies' resources and also network switches [8]. Since the two mentioned phases operate independently, the routing and resource management are also carried out independently in two levels of “SIP proxies" and “network switches". In addition, this kind of routing is distributed among all switches and proxies. The lack of a global view over the entire network leads to the overload phenomenon, directly affecting the network's throughput [9]. The network's throughput drops to zero in this case. Therefore, centralized routing and resource management is highly required.

This article proposes two softwarized centralized management frameworks for the VoIP network by employing the SDN [10, 11] and NFV [12, 13] concepts. These frameworks integrally manage all resources of proxies and switches.

By employing the SDN, the entire network and its components are considered as a virtual unit network, which becomes controllable using software and special APIs, such as OpenFlow protocols [14]. The SDN architecture consists of a control plane and a data plane [15, 16]. The data plane consists of the network devices, including routing devices, switches, etc., such that they lack the controlling or softwarized sections for automated decision-making. The intelligent part of the network is located in the SDN's softwarized controllers keeping the general structure of the network (the control plane) [17]. The controller consists of a set of applications, including routing, firewall, load balance, supervision, etc. The connection protocol between planes is a kind of standard Open APIs, including OpenFlow protocols. In case there are such interfaces, the controller is capable of dynamic programming of the network's non-identical devices [18]. A method called LBBSRT (Load Balancing by Server Response Time) [19] is used for the load balancing between the servers in SDN. LBBSRT using the SDN controller to obtain the response time of each proxy server to select a proxy server with a minimum response time. We use this method to compare performance.

The NFV technology is strongly complementary to the SDN. The NFV aims to employ virtualization technologies to virtually implement various devices and network functions, called Virtualized Network Functions or VNF [20]. A VNF can consist of one or several Virtual Machines (VMs) implementing special software on Physical Machines (PMs) in order to provide a network function (instead of considering separate hardware for each function) [21]. Thereby, the NFV separates the network functions from the single-purpose hardware, enabling these functions to operate in a softwarized way and under the virtual machines' control. Consequently, the resource management of the network's proxies can be carried out proactively by employing this technology [22]. The instances of VNF can be dynamically defined and employed where needed, or dynamically migrated, copied in several duplicates, or eliminated according to the existing condition in the network [23]. So, the frameworks proposed for the VoIP network will have some features, including efficiency improvement, effective routing, and intelligent load balance, integrated resource management, traffic management, programming capability, reducing costs.

At the end of this section, let us mention that we did a re-extensive search on the existing work on new papers about VoIP networks. We have categorized these articles as follows:

- Encrypted VoIP classification [24],
- Security functions for VoIP [25],
- Priority queueing for VoIP [26],
- QoE monitoring for VoIP [27],
- Multimedia applications on 5G systems [28],
- Multi-domain case for VoIP [29].

SDN and NFV technologies have been used in all the above articles. But because they are outside the scope of this article, no further explanation is given to maintain the integrity of the paper. The current paper is especially about the design of a novel framework for managing VoIP network resources.

1.1 Contribution

We are looking for the following objectives:

- Increasing the VoIP network's throughput,
- Reducing the delay in making a call,
- Enhancing the quality and evaluating multimedia quality parameters, including Mean Opinion Score (MOS) [30] and R factor [31],
- The integrated resources management of the VoIP network,
- The centralized routing of signaling and media messages of the VoIP network.

In this regard, the major contributions of this article are indicated as follows:

- Designing two centralized frameworks based on SDN and NFV for VoIP resource and network traffic management at each two signaling and media phases,

    1. First approach: designing SDN and NFV controller for the VoIP network,

2. Second approach: designing SDN controller+ for the VoIP network,

- Implementing the proposed frameworks on a real testbed and evaluating their performances under various scenarios.

The use of SDN and NFV technologies due to having a centralized control and global view of all segments and resources of the VoIP network can help achieve the above goals, including increasing throughput, reducing latency, and improving call quality.

1.2 Organization

In Sect. 2, two approaches towards softwarized VoIP networks are proposed, and in Sect. 3, the function of these two approaches is evaluated under different scenarios and criteria. Ultimately, the conclusion and future works are presented (Sect. 4).

2 Proposed frameworks

In this section, two frameworks for softwarized VoIP networks are designed. The first one consists of an SDN and NFV controller, whose SDN part manages the network's switches and the NFV part manages VNFs of VoIP. Both parts interact with each other. In the second framework, an SDN controller, named SDN controller+, is proposed for the VoIP network that carries out the whole routing (SIP and IP) process softwarized and at the control plane. The details of these two frameworks are presented in the following.

2.1 The first proposed framework: SDN and NFV controller for VoIP

This section aims designing a novel framework for the VoIP networks by taking advantage of the SDN and NFV concepts. The schematic of this design is presented in Fig. 1. As can be seen in this figure, VoIP network includes various domains. A server (PM), several OpenFlow switches, and a controller are responsible for handling signaling and media at each domain. Instead of single-purpose hardware SIP proxies, the virtual SIP proxies, named SIP VNFs, are provided on the server.

Similar to the SIP proxies, these SIP VNFs are responsible for routing the seventh layer of signaling traffic. The number of these SIP VNFs depends on the network's input signaling traffic, meaning that the number of these SIP VNFs declines at the low load and increases at the high load. The controller is responsible for managing these SIP VNFs. According to its comprehensive knowledge of the entire VoIP network, the controller properly allocates the SIP VNFs. Another responsibility of the controller is traffic flow management. Employing the statistic the OpenFlow switches provide, the controller performs the traffic flow management and routing. It must be noted that the signaling traffic of each domain for routing must pass through the SIP VNF pertinent to that domain, meaning

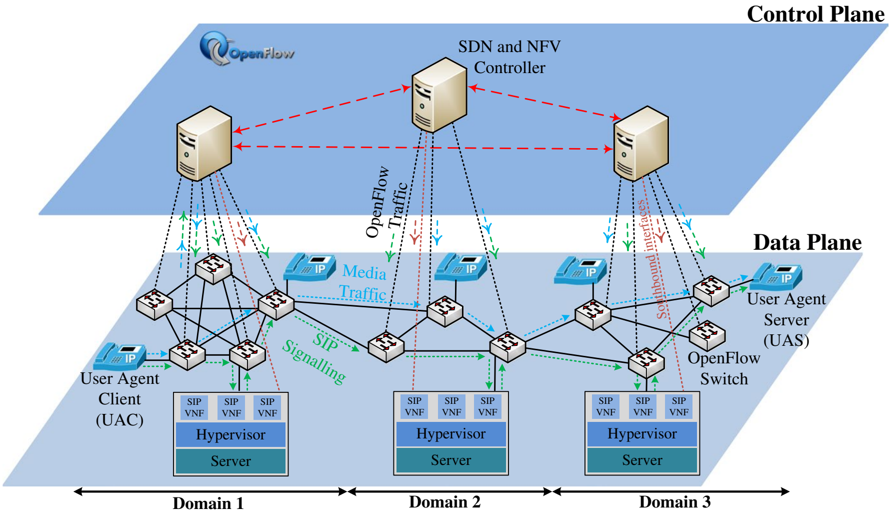


Fig. 1 The first proposed framework

that signaling traffic passes through the middle SIP VNFs hop by hop to reach the destination. On the contrary, media traffic does not need to pass through the SIP VNFs to reach the destination, and it merely passes through the OpenFlow switches to reach the destination. The controller is responsible for routing traffic media among the OpenFlow switches. The controllers' information is synced at certain times. This flexible structure is employed in both dynamic resources management and centralized effective traffic routing. This matter leads to fairly load distribution and also calls' quality enhancement. In the following, the proposed controller's architecture in this framework will be examined.

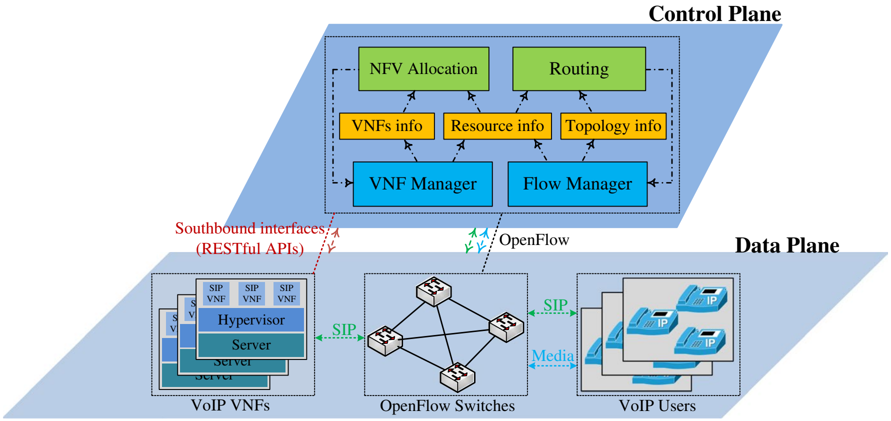


Fig. 2 The architecture of the first proposed controller

The architecture and the components of the proposed controller are indicated in Fig. 2. The proposed framework separated the control and data planes in the VoIP network. The data plane includes the VoIP Users, the OpenFlow switches, and VoIP VNFs. The control plane includes the centralized controller. The relationship between these two planes is through the RESTful APIs [32] and also the OpenFlow protocol. In the data plane, the signaling is handled by the SIP protocol; the media by the RTP protocol. The components and the modules of the controller are categorized into three groups:

(1) routing and NFV allocation,
(2) collecting and keeping information,
(3) flow and VNFs management.

The thorough information of the network and switch topology, the server resource information, and the VNFs status information are collected in the controller. Using this information, the *Routing* and *NFV Allocation* modules make the proper decisions. These decisions are provided to the *Flow Manager* and *VNF Manager* modules in the form of rules in order to be transferred to the data plane in the OpenFlow and RESTful API messages format. The details of the message passing are indicated in the proposed framework as follows. As shown in Fig. 2, such integrated and centralized architecture brings about the deliberate distribution of the input load (by the users) among the components of the network (switches and servers).

Figure 3 demonstrates the message passing. As indicated, the message passing is executed in five steps: *initiate session*, *registration*, *media*, *tear down*, and *time out*. The first step is the registration of a user in the SIP VNFs, starting with the `Register` message. The controller receives the updated information of the PMs and VNFs, and their resources; with respect to that, adding to or subtracting from the VNFs is carried out if needed. A proper VNF is chosen, and the registration message is sent to it. In this step, the exchanged RESTful API messages between the controller and PM are defined in red color. The second step is making a call. The `Invite` message is delivered to the controller and SIP VNF via a `Packet-In` message. The order of the other messages for making a call is `180 Ringing`, `200 Ok`, and `Ack`. Only one duplicate of the `Invite` message is sent to the controller in the `Packet-In` message format for each session. The third step is the media passing. For this purpose, the duplicate of the first RTP message is sent to the controller. The commands are installed on the OpenFlow switches via `Flow-Mod` messages. The RTP messages reach the destination via OpenFlow switches. The fourth step is the tear down of the session, performed by using the `Bye` message. In the end, the fifth step is called time out, leading to the update of the server's information and VNFs in the controller periodically and at the end of a session.

The order of the message passing is indicated in Fig. 4 (from No. 1 to 10, respectively). As demonstrated in this figure, the signaling messages pass the VoIP VNFs, but the media messages only pass through the OpenFlow switches. The relationship between the controller and VoIP VNFs is via RESTful API (No. 3 and 4). The relationship between the controller and the switches is via OpenFlow.

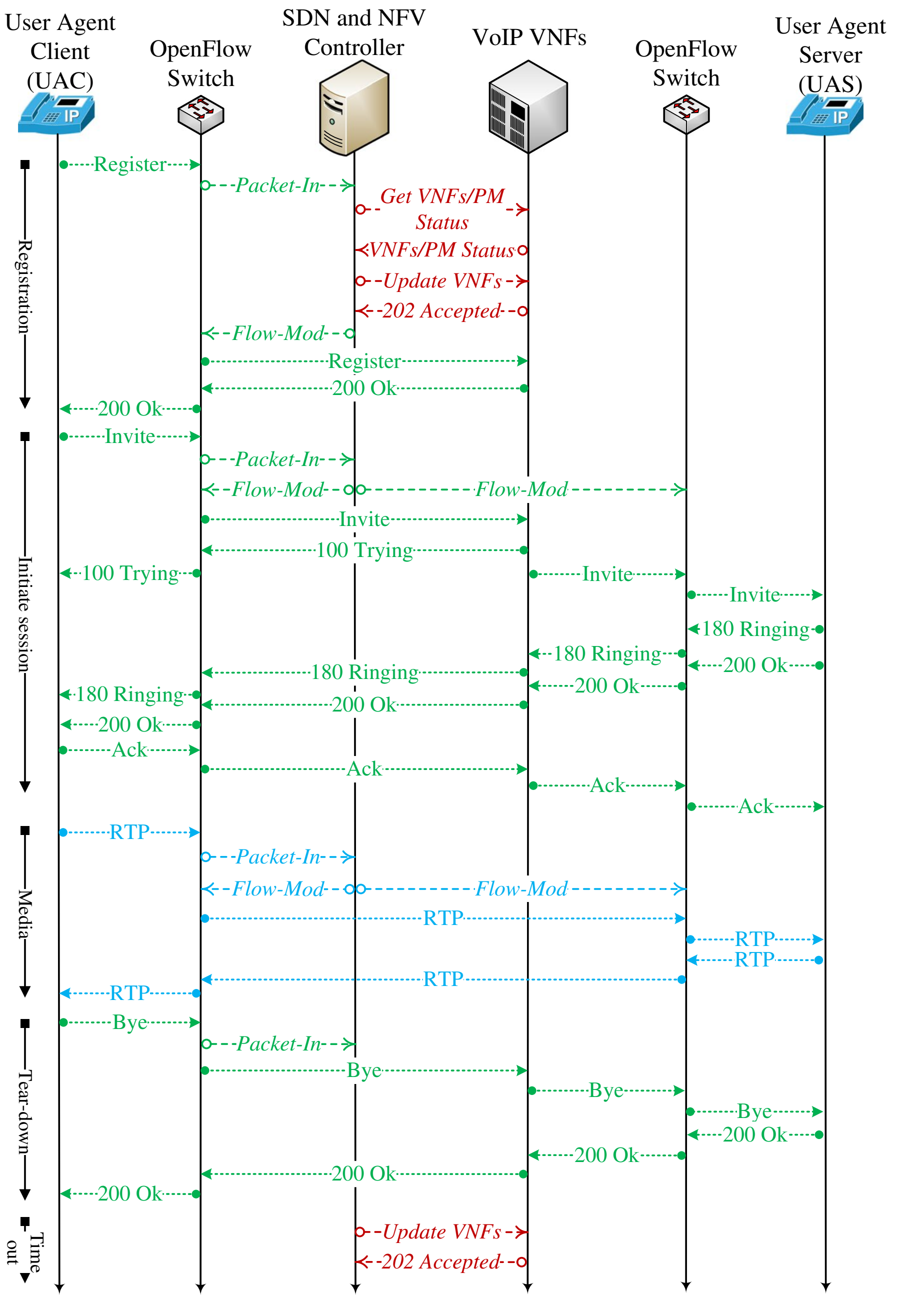


Fig. 3 Message passing in the first proposed framework

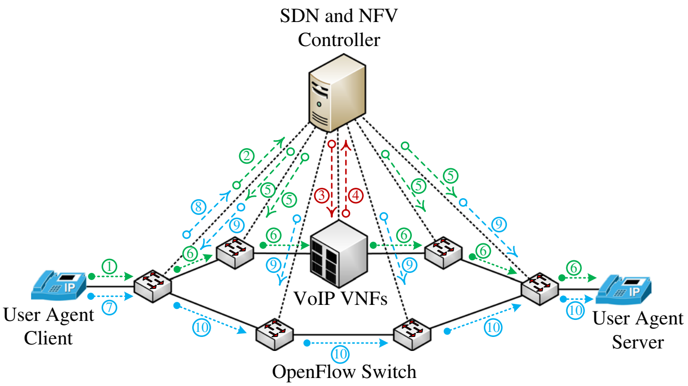


Fig. 4 The order of the message passing (from No. 1 to 10)

2.2 The second proposed framework: SDN controller+ for VoIP

In the previously proposed framework, the seventh layer routing of the SIP signaling messages was performed by the VoIP VNFs. The controller was also responsible for managing the numbers of SIP VNFs. In this section, the previous framework is developed such that its complexity is mitigated in the data plane. Also, the layer-seven routing of the signaling messages is executed similar to the layer-three routing of the other messages in a centralized way in the controller. It leads to the centralization and integration in the routing of all messages. Moreover, the dependency on the hardware is lowered, and the softwarization of the routing operations has taken place. In this case, we are getting closer to the softwarized VoIP networks. Accordingly, a controller named SDN controller+ is designed and implemented. The general framework of this design is indicated in Fig. 5. As shown in this figure, the relationship between the control and data

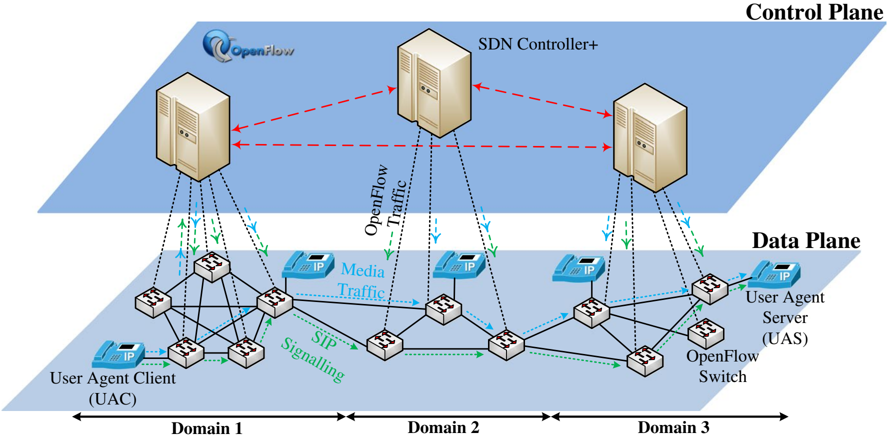


Fig. 5 The second proposed framework

planes is merely OpenFlow, and there is no need for the RESTful API messages. It means that the data plane has been simplified, and the relationships between the control and data planes have been constrained to the OpenFlow. However, the modules of the controller have been changed. We will introduce the SDN controller+ modules as follows.

Figure 6 demonstrates SDN controller+ modules. These modules are categorized into three groups: (1) the routing modules, (2) information keeping modules (3) flow management modules. Since the VoIP VNF does not exist in the data plane, the controller routes layer-seven signaling messages through the *SIP Routing* module. For routing layer-three media messages, the *IP Routing* module is used. These two modules are fed by the *Resource info*, the *SIP user info*, and *Topology info* modules. In the end, the *SIP Flow Manager* and *Media Flow Manager* modules provide the routing decisions for the switches in the form of OpenFlow commands and get new statistics of the network status.

The message passing through four steps (*media*, *initiate session*, *registration*, and *tear down*) is demonstrated in Fig. 7. The SDN controller+ plays the main part whether in the new users' registration, in the call requests routing, or the media transfer. Of course, it is worth mentioning that the first message of each step is sent to the SDN controller+. The remaining messages of that flow follow the decisions made for the first message.

The order of the exchanged messages between UAS and UAC is demonstrated in Fig. 8. There is no difference between the signaling and media messages in the data plane, and all of them pass through the OpenFlow switches. However, according to the SDN controller+ decisions, it is likely that their routes are different from each other. It should be noted that the data plane has become simpler in comparison with the previous method.

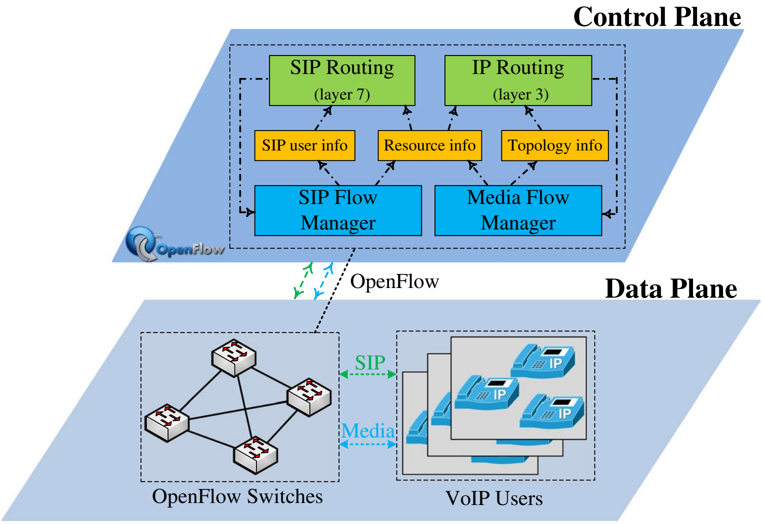


Fig. 6 The architecture of the second proposed controller

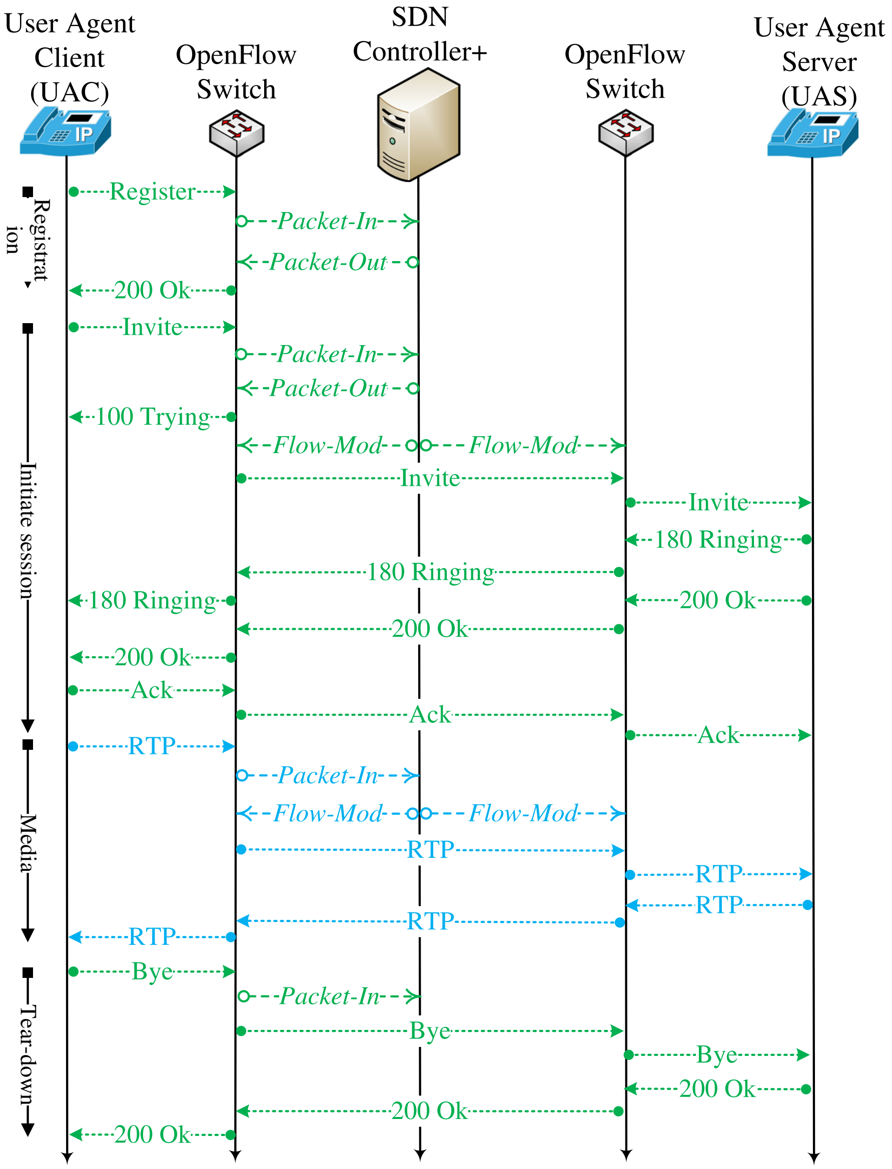


Fig. 7 Message passing in the second proposed framework

3 Performance assessment

In this part, the testbed for implementing the proposed framework will be explained elaborately. Afterward, under various scenarios, the performance of both proposed frameworks will be assessed, and their results will be provided. Figure 9 indicates the topology and the testbed, including 8 PMs for the VNFs of the VoIP, 3 PCs

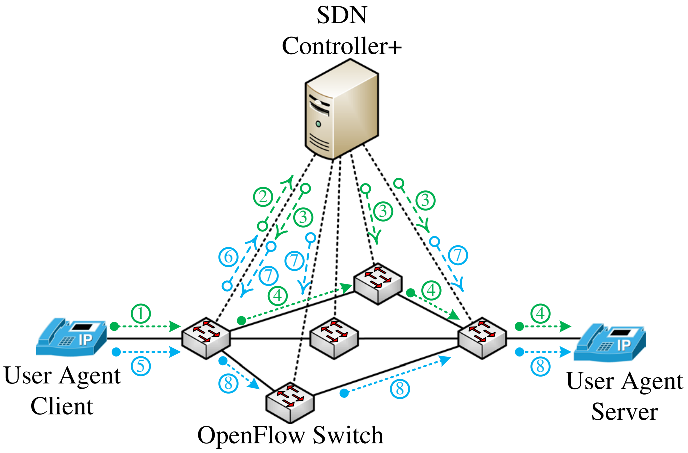


Fig. 8 The order of the message passing (from No. 1 to 8)

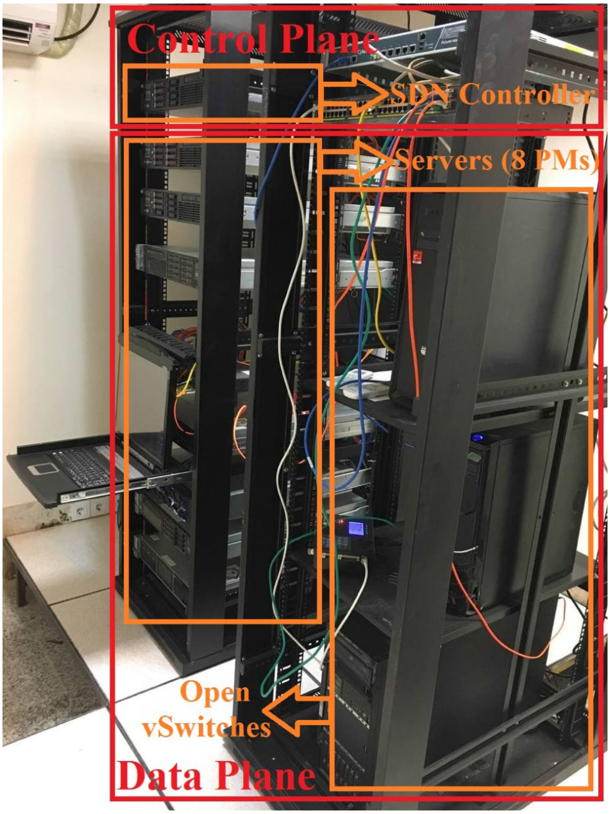


Fig. 9 The testbed to assess the proposed frameworks

for the Open vSwitch (OVS), and one PM for the proposed controller's execution. The PMs are inhomogeneous and include HP G9 servers (3 units), G8 (2 units), G7 (3 units), and G6 (one unit). In the testbed, the Floodlightv1.2 [33], Open vSwitch v2.4.1 [34], and OpenSIPS [35] are used for the implementation of proposed controllers, the OpenFlow switches, and VNFs of the VoIP, respectively. The Floodlight is executed on an HP G9 DL580 server, and we changed it according to what is explained in the previous sections. The designed modules are added to it. This server is equipped with VMware ESXi hypervisor v6.0. The OVSs are executed on 3 PCs, each of which is equipped with an Intel Xeon 4-core 3.70GHz CPU and 16 GB RAM. The Oprofile [36] software is also used for tracking the consumed resources of PMs. The SIPp [37] is used for VoIP sessions generation, and the StarTrinity [38] is employed to measure the network's parameters and the quality of the sessions. The bandwidth of the links is 10 Mbps. The comparison method is LBBSRT [19]. As stated earlier, this method is used for the load balancing between the servers in SDN. LBBSRT using the SDN controller to obtain the response time of each server to select a server with a minimum response time. The procedure of this method to obtain the response time for load balancing includes the following three steps:

- Step1: Send `Packet-out` message to the switches.
- Step2: Switches handles the `Packet-out` message.
- Step3: The server sends the reply message, from which the controller obtains the server response time.

Details of these steps are given in the [19].

The assessment criteria include the Quality of Service (QoS) and the Quality of Experience (QoE) of the VoIP network, such as throughput, delay, numbers of VNFs and OVSs, average CPU and memory consumption, MOS, and R factor. The assessment results of the proposed framework are provided in two data and control planes and two signaling and media phases. Each experiment is conducted at least three times, and their average is considered the result.

3.1 Evaluating the performance of the first framework: SDN and NFV controller for VoIP

In the experiments of this section, we have considered four load scenarios indicating the offered load or the input traffic of the network: low, medium, high, and very high. The amount and distribution of this traffic in 100 seconds are shown in Fig. 10. As an illustration, the maximum load in the low-load scenario is approximately 1500 cps, while it is 13,000 cps in the very high-load scenario. These scenarios lead the proposed framework to be assessed in the low, medium, and high-load conditions.

The throughput of the four scenarios are indicated in Fig. 11. The results are comparing the "LBBSRT", "Round Robin", and "proposed" methods. In all scenarios, the throughput pattern follows the offered load pattern. However, the throughput of the proposed method is much closer to the offered load. This issue is much more evident in the third and fourth scenarios. As the load increases and when moving from

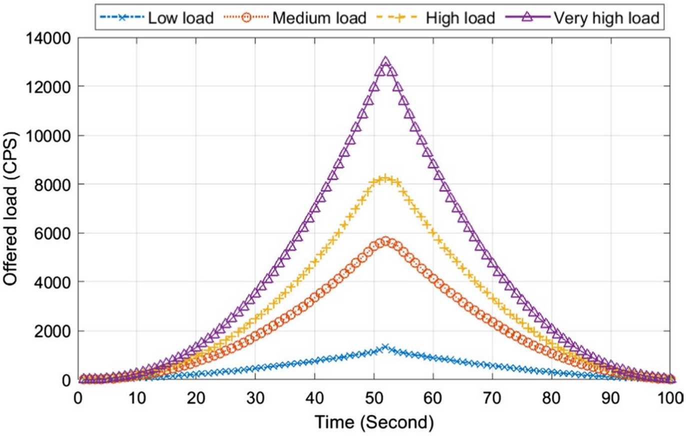
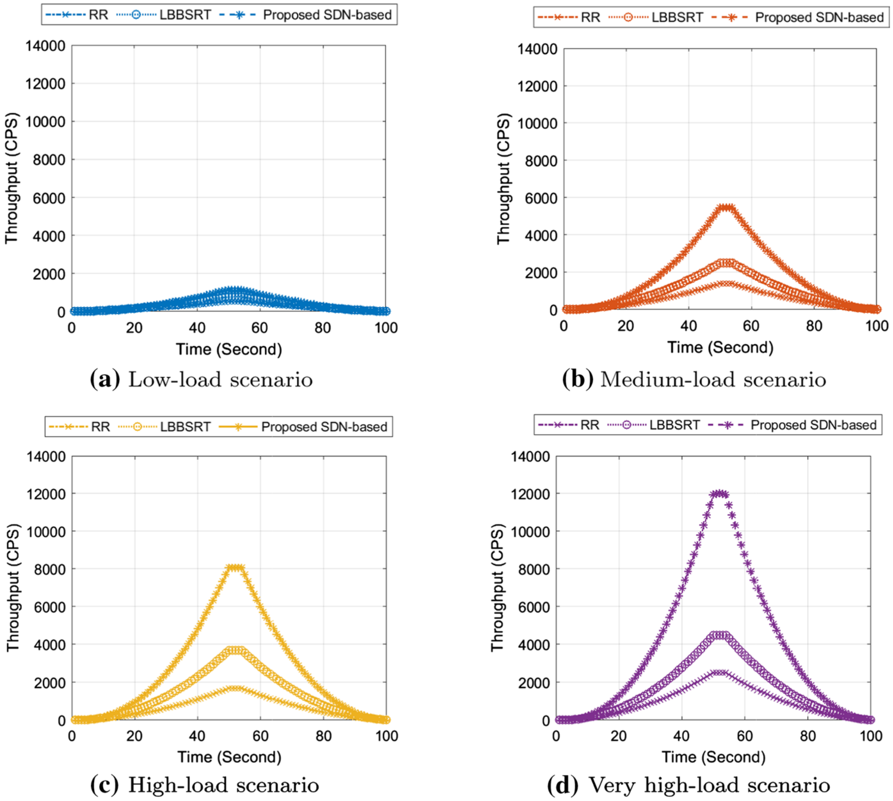


Fig. 10 Input traffic of the VoIP network (call requests)



(a) Low-load scenario

(b) Medium-load scenario

(c) High-load scenario

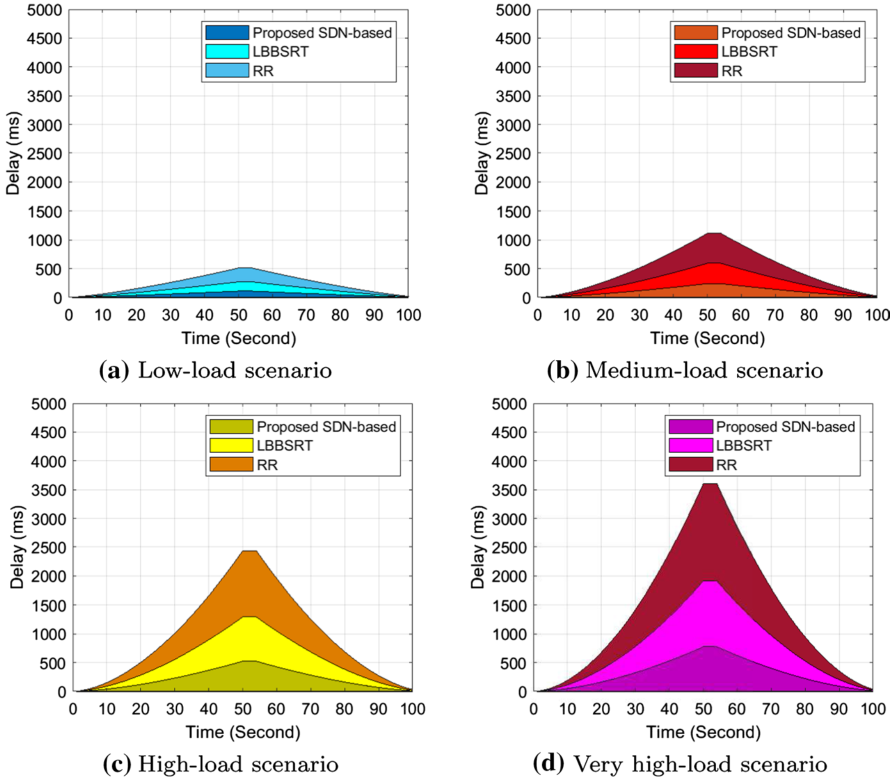
(d) Very high-load scenario

Fig. 11 The throughput of the proposed mechanism compared to LBBSRT and RR

the low-load scenario to the high-load scenario, the throughput drop of the LBBSRT and RR methods increases. Nonetheless, thanks to having a global view, the proposed method could prevent the throughput drop even in the high-load scenario.

The results of the delay in making a call are indicated in Fig. 12. As the offered load increases, the delay increases, as well. However, its growth is different for different methods. The increase of the delay in the RR method is extremely high due to the lack of traffic management. On the other hand, the proposed method based on SDN has been able to prevent excessive delay increase and service quality decline by intelligent traffic management and keep the delay less than 1000 ms. In the worst case, the delay in the RR method is more than 3500 ms and approximately 2000 ms in the LBBSRT method. Excessive delay leads to service quality decline.

The reason behind the excessive increase in the delay of the fourth scenario and the RR mechanism is the saturation of servers' resources. This issue is examined in Fig. 13. This figure demonstrates the average CPU usage of eight PMs. As can be seen, the highest resource usage belongs to the RR mechanism. On the other hand, the least resource usage belongs to the proposed mechanism, which is because of conscious resource management. In the RR mechanism and at the second 50 of the fourth scenario, the CPU of the servers are almost saturated. This issue is the reason



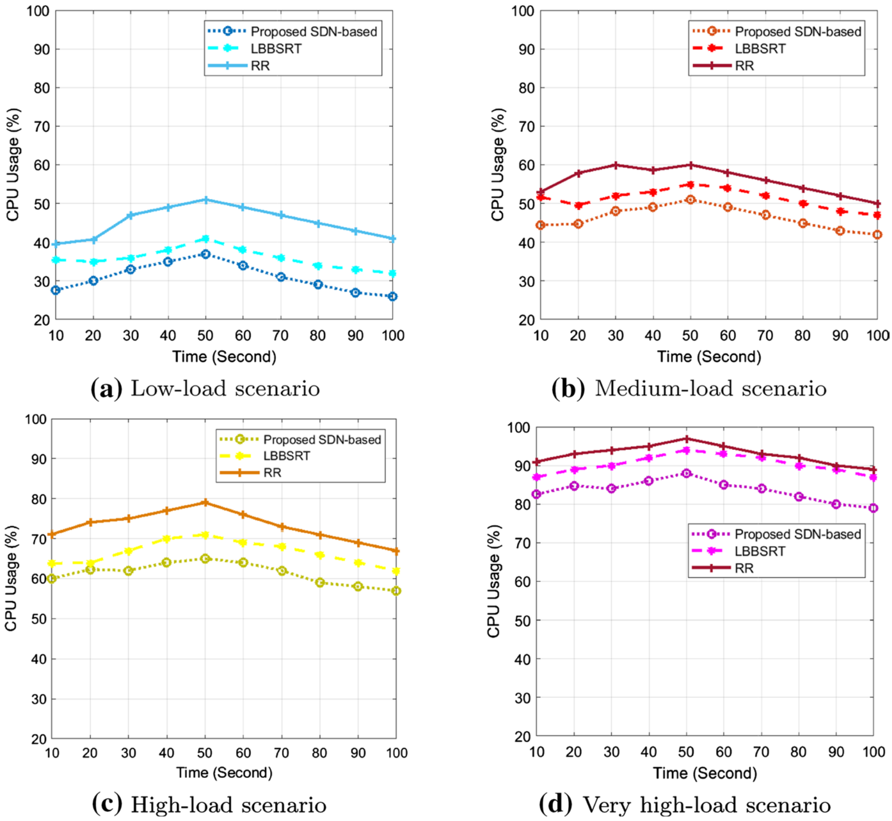
(a) Low-load scenario

(b) Medium-load scenario

(c) High-load scenario

(d) Very high-load scenario

Fig. 12 Delay in making a call in the proposed mechanism compared to the LBBSRT and RR



(a) Low-load scenario

(b) Medium-load scenario

(c) High-load scenario

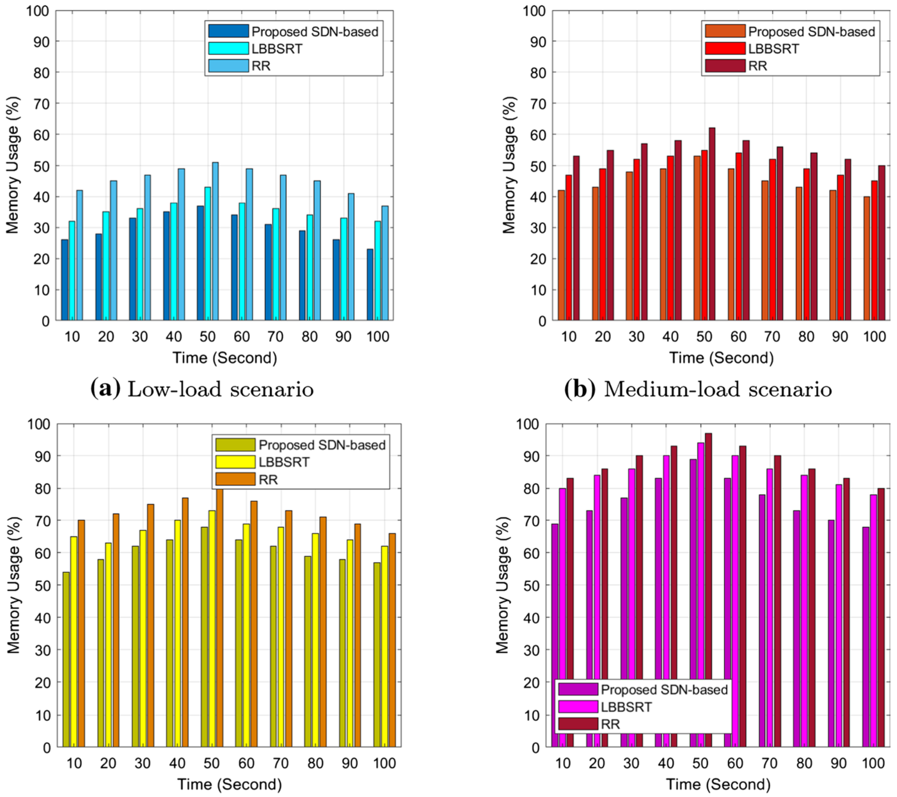
(d) Very high-load scenario

Fig. 13 Average CPU usage of the PMs in the proposed mechanism compared to LBBSRT and RR

behind this mechanism's delay increase of up to 3500 ms. In the worst case, the maximum CPU usage of the servers in the proposed method is 88%. To put it differently, by proper resource management, the proposed controller does not allow servers' CPUs to be saturated even in the high-load scenario.

Resources management is not only is for CPUs but also includes memory. Since the number of SIP VNFs and servers is managed by the controller, the servers' average memory is also used in accordance with a logic pattern. This pattern is indicated in Fig. 14. However, in RR, the PMs' average memory usage is high even in the low-load scenario. The highest memory usage of the servers belongs to the RR method in the fourth scenario and at the second 50 (almost saturated). However, in this scenario and at the same exact moment, the servers' average memory usage does not exceed 87% in the proposed method.

Note Fig. 15 in order to accurately understand the controller's performance in the management of the VNFs and switches. As can be seen, the number of VNFs and OVSs over time follows the offered load pattern (Fig. 10). This means that the number of VNFs and OVSs is low at the low offered load, and at the high offered load,



(a) Low-load scenario

(b) Medium-load scenario

(c) High-load scenario

(d) Veru high-load scenario

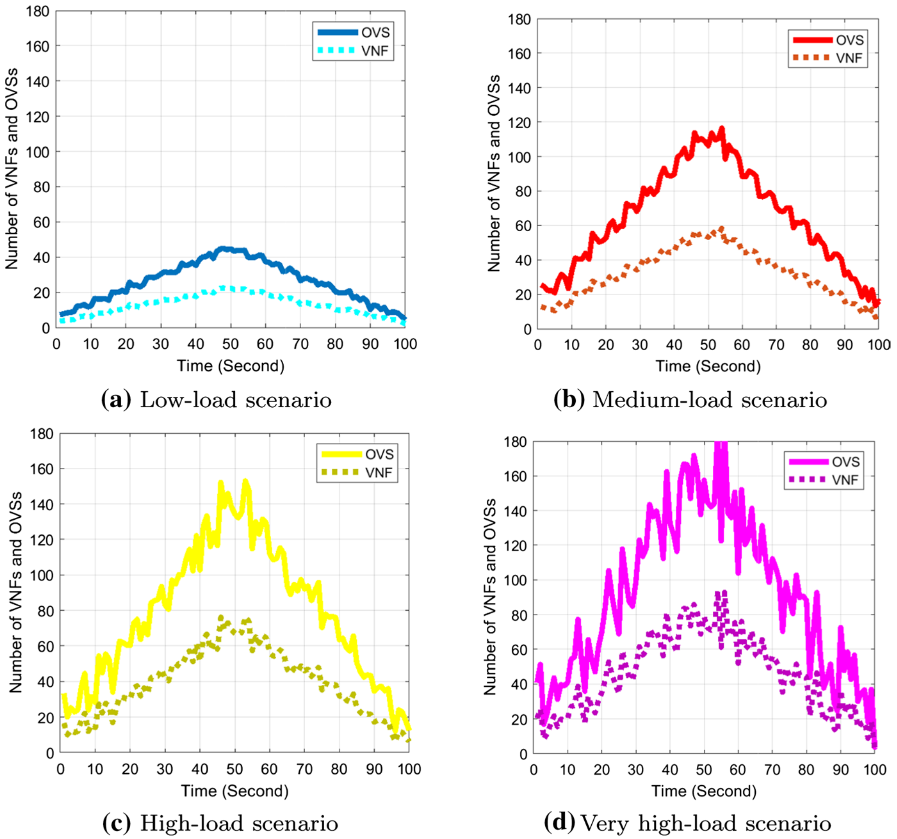
Fig. 14 The PMs' average memory usage in the proposed mechanism compared to LBBSRT and RR

the number of VNFs and OVSs also increases (unlike the LBBSRT and RR methods in which the servers and resources are constant all the time). This intelligent management in the accurate adjustment of resources with respect also to traffic leads to a decline in energy consumption, meaning that a high number of VNFs and OVSs that lead to energy waste at low-load is not required. Therefore, the constancy of the resources is accompanied by energy waste, and the proposed model has tried to prevent it by employing NFV.

The QoS parameters pertaining to the signaling phase have been examined so far. In the following, the QoE parameters pertinent to the media phase will be examined. Figure 16 indicates the MOS of media in different conditions.

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 1.0, and the highest possible MOS, representing ideal multimedia quality, is 5.0.

Figure 16a–d demonstrate the MOSs at different scenarios. As can be seen, the highest MOS belongs to the proposed method, and in the worst case, it never drops to less than 3.5, and it is higher than 4 on average. This issue indicates the high



(a) Low-load scenario

(b) Medium-load scenario

(c) High-load scenario

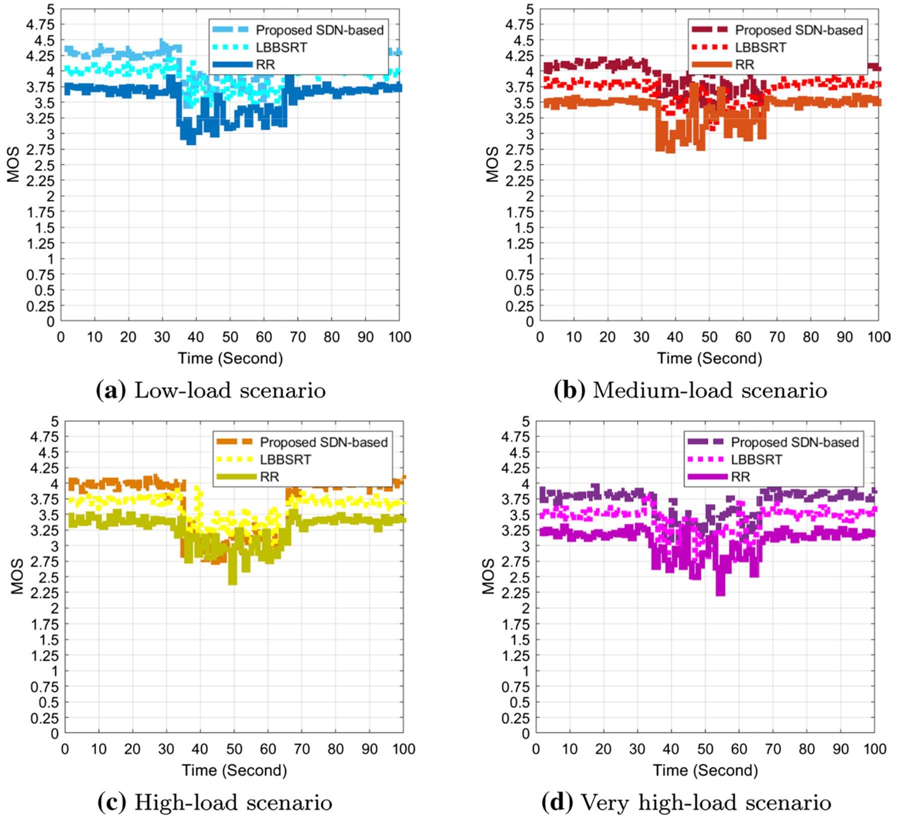
(d) Very high-load scenario

Fig. 15 The number of VNFs and OVSs in the proposed framework

quality of media in the proposed method. However, in the other methods, as load increase, not only does MOS drops extremely, but also it has sharp fluctuation. For instance, in the heavy traffic scenario, the MOS of the LBBSRT method even drops to less than 2.5 (Fig. 16d).

In the following, the details pertinent to PMs' used resources (CPU and memory) of the three "proposed", "LBBSRT", and "RR" mechanisms will be addressed. Fig. 17 is a snapshot from the PMs' CPU consumption in the medium-load scenario (Fig. 13b). Comparing Fig. 17a–c reveals that the load among the eight PMs is well balanced. Nonetheless, in the two other methods, the difference in CPU usages in the PMs is significant and noticeable. Besides, the average CPU usages of PMs in the proposed framework is approximately 40%, which is equal to 50 and 60% for the LBBSRT and RR methods, respectively.

Figure 18 demonstrates a snapshot from the PMs' used memory in the medium-load scenario. At the seconds 40 and 60, when the offered load is higher, the used memory of the PMs is also higher, while at the seconds 10 and 100, when the offered load is lesser, the used memory of the PMs is also lesser, consequently.



(a) Low-load scenario

(b) Medium-load scenario

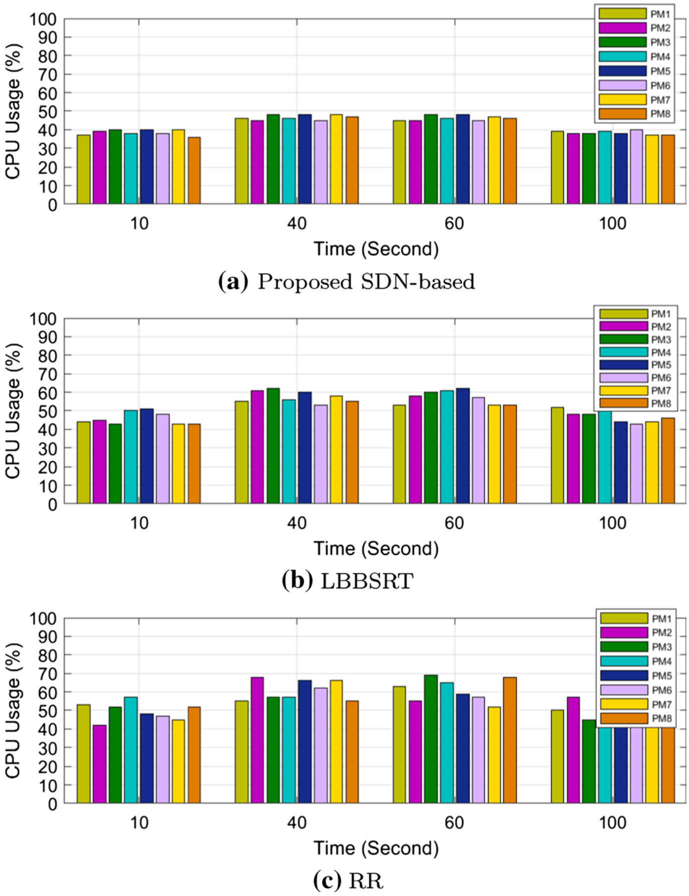
(c) High-load scenario

(d) Very high-load scenario

Fig. 16 Media MOS in the proposed method compared to LBBSRT and RR

Figure 18a indicates that in the proposed method, the load is well balanced among the PMs. Besides, the average memory usage is approximately 45%. Figure 18b, c demonstrate that the load balance among PMs by the LBBSRT and RR methods is not as good as the proposed model. In addition, the average memory usage in these two methods is more than the proposed method.

In the following, the message exchange in the real testbed is monitored by sniffing tools (Wireshark). The result is demonstrated in Fig. 19. In this figure, the type and order of messages and also the details of the exchanged message (such as time, the source and destination IPs, protocols, etc.) are indicated. As can be seen, the exchanged messages are in accordance with Fig. 3 and the pattern of the proposed method. The ending bytes of IP addresses of UAC, OVS1, controller, PM, OVS2, and UAS are 100, 101, 102, 103, 104, and 105, respectively. Receiving the `Register` message from the UAC (192.168.1.100), the OVS1 switches (192.168.1.101) forwards it to the controller (192.168.1.102) in the form of a `Packet-In` message. The controller also starts negotiation with PM (192.168.1.103) to determine or allocate a suitable VNF. Afterward, it forwards the `Register` message to VNF



(a) Proposed SDN-based

(b) LBBSRT

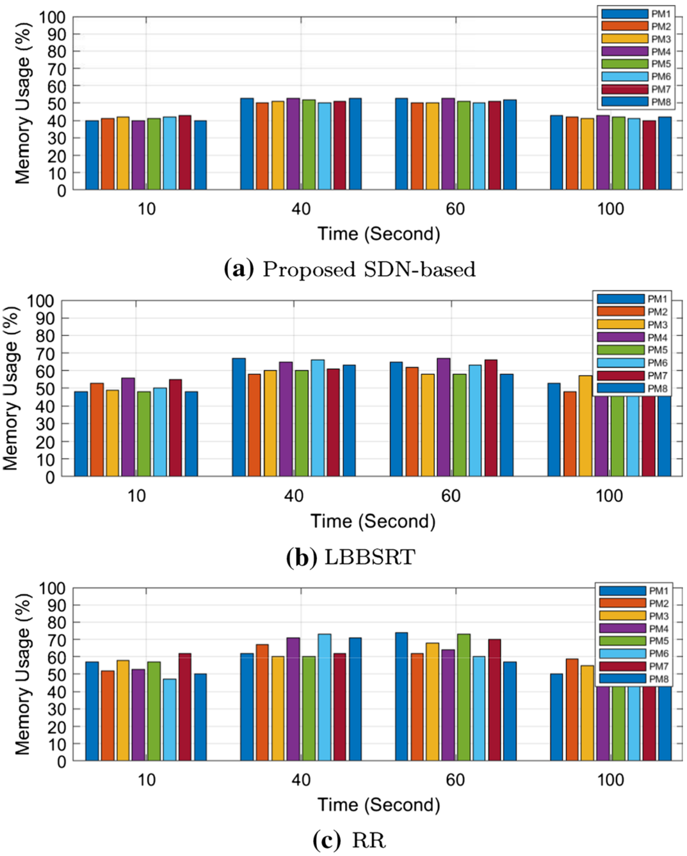
(c) RR

Fig. 17 CPU usage of PMs

and the confirmation response (`200 Ok`) to UAC. Then, it enters the initiate session stage and the following, as can be seen in the figure.

It is worth mentioning that the RESTful API messages exchanged between the controller and PM include the Put, Get, and Post methods that are indicated in the figure by the numbers 3 to 6. The controller sends the routs to the OpenFlow switches via the `Flow-Mod` message.

So far, we have presented the results of the data plane. Now, we want to evaluate the control plane. In order to do that, we examine the CPU usage by the Kernel and the modules designed in the controller and make a comparison between them. The results are presented in Table 1. The offered load increases from 1000 cps to 10,000 cps. The highest CPU usage belongs to the Linux Kernel and not the designed modules. This issue indicates that, firstly, the proposed controller is scalable at least to the 10,000 cps offered load, approximately. Secondly, the designed modules are not bottleneck, meaning that they do not have unjustified CPU usage.



(a) Proposed SDN-based

(b) LBBSRT

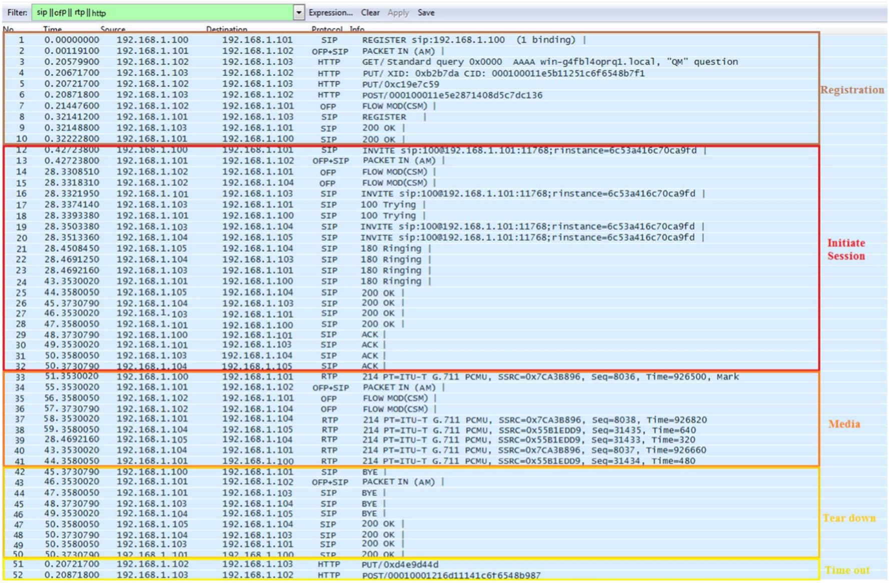
(c) RR

Fig. 18 The PMs' used memory

For instance, the *VNFs Manager* module has a maximum CPU usage of 11.10%. Among the modules, the modules that are responsible for collecting and storing the network information (similar to *VNFs info*) have more resource usage than other modules.

3.2 Evaluating the performance of the second framework: SDN controller+ for VoIP

In this section, we evaluate the performance of the second framework. The first evaluation criterion is the throughput over time (Fig. 20). The offered load increases in four steps and during 10-second periods. At the first ten seconds, it is low-load (1000 cps), at the second ten seconds, it is medium-load (6000 cps), at the third ten seconds, it is high-load (8000 cps), and at the fourth ten seconds, it is very high-load (12,000 cps). SDN controller+ could keep the throughput very close to the offered load at every four stages. However, the throughput drop of


Filter: sip ||ofP || rtp || http
Expression... Clear Apply Save
Registration
Initiate Session
Media
Tear down
Time out

```
No.  Time        Source         Destination    Protocol  Info
1    0.00000000  192.168.1.100  192.168.1.101  SIP       REGISTER sip:192.168.1.100  (1 binding) |
2    0.00119100  192.168.1.101  192.168.1.102  OFP+SIP   PACKET IN (AM) |
3    0.20579900  192.168.1.102  192.168.1.103  HTTP      GET/ Standard query 0x0000  AAAA win-g4fbl4oprq1.local, "QM" question
4    0.20671700  192.168.1.103  192.168.1.102  HTTP      PUT/ XID: 0xb2b7da CID: 000100011e5b11251c6f6548b7f1
5    0.20721700  192.168.1.102  192.168.1.103  HTTP      PUT/ 0xc19e7c59
6    0.20871800  192.168.1.103  192.168.1.102  HTTP      POST/ 000100011e5e2871408d5c7dc136
7    0.21447600  192.168.1.102  192.168.1.101  OFP       FLOW MOD(CSM) |
8    0.32141200  192.168.1.101  192.168.1.103  SIP       REGISTER    |
9    0.32148800  192.168.1.103  192.168.1.101  SIP       200 OK |
10   0.32222800  192.168.1.101  192.168.1.100  SIP       200 OK |
12   0.42723800  192.168.1.100  192.168.1.101  SIP       INVITE sip:100@192.168.1.101:11768;rinstance=6c53a416c70ca9fd |
13   0.42723800  192.168.1.101  192.168.1.102  OFP+SIP   PACKET IN (AM) |
14   28.3308510  192.168.1.102  192.168.1.101  OFP       FLOW MOD(CSM) |
15   28.3318310  192.168.1.102  192.168.1.104  OFP       FLOW MOD(CSM) |
16   28.3321950  192.168.1.101  192.168.1.103  SIP       INVITE sip:100@192.168.1.101:11768;rinstance=6c53a416c70ca9fd |
17   28.3374140  192.168.1.103  192.168.1.101  SIP       100 Trying |
18   28.3393380  192.168.1.101  192.168.1.100  SIP       100 Trying |
19   28.3503380  192.168.1.103  192.168.1.104  SIP       INVITE sip:100@192.168.1.101:11768;rinstance=6c53a416c70ca9fd |
20   28.3513360  192.168.1.104  192.168.1.105  SIP       INVITE sip:100@192.168.1.101:11768;rinstance=6c53a416c70ca9fd |
21   28.4508450  192.168.1.105  192.168.1.104  SIP       180 Ringing |
22   28.4691250  192.168.1.104  192.168.1.103  SIP       180 Ringing |
23   28.4692160  192.168.1.103  192.168.1.101  SIP       180 Ringing |
24   43.3530020  192.168.1.101  192.168.1.100  SIP       180 Ringing |
25   44.3580050  192.168.1.105  192.168.1.104  SIP       200 OK |
26   45.3730790  192.168.1.104  192.168.1.103  SIP       200 OK |
27   46.3530020  192.168.1.103  192.168.1.101  SIP       200 OK |
28   47.3580050  192.168.1.101  192.168.1.100  SIP       200 OK |
29   48.3730790  192.168.1.100  192.168.1.101  SIP       ACK |
30   49.3530020  192.168.1.101  192.168.1.103  SIP       ACK |
31   50.3580050  192.168.1.103  192.168.1.104  SIP       ACK |
32   50.3730790  192.168.1.104  192.168.1.105  SIP       ACK |
33   51.3530020  192.168.1.100  192.168.1.101  RTP       214 PT=ITU-T G.711 PCMU, SSRC=0x7CA3B896, Seq=8036, Time=926500, Mark
34   55.3530020  192.168.1.101  192.168.1.102  OFP+SIP   PACKET IN (AM) |
35   56.3580050  192.168.1.102  192.168.1.101  OFP       FLOW MOD(CSM) |
36   57.3730790  192.168.1.102  192.168.1.104  OFP       FLOW MOD(CSM) |
37   58.3530020  192.168.1.101  192.168.1.104  RTP       214 PT=ITU-T G.711 PCMU, SSRC=0x7CA3B896, Seq=8038, Time=926820
38   59.3580050  192.168.1.104  192.168.1.105  RTP       214 PT=ITU-T G.711 PCMU, SSRC=0x55B1EDD9, Seq=31435, Time=640
39   28.4692160  192.168.1.105  192.168.1.104  RTP       214 PT=ITU-T G.711 PCMU, SSRC=0x55B1EDD9, Seq=31433, Time=320
40   43.3530020  192.168.1.104  192.168.1.101  RTP       214 PT=ITU-T G.711 PCMU, SSRC=0x7CA3B896, Seq=8037, Time=926660
41   44.3580050  192.168.1.101  192.168.1.100  RTP       214 PT=ITU-T G.711 PCMU, SSRC=0x55B1EDD9, Seq=31434, Time=480
42   45.3730790  192.168.1.100  192.168.1.101  SIP       BYE |
43   46.3530020  192.168.1.101  192.168.1.102  OFP+SIP   PACKET IN (AM) |
44   47.3580050  192.168.1.101  192.168.1.103  SIP       BYE |
45   48.3730790  192.168.1.103  192.168.1.104  SIP       BYE |
46   49.3530020  192.168.1.104  192.168.1.105  SIP       BYE |
47   50.3580050  192.168.1.105  192.168.1.104  SIP       200 OK |
48   50.3730790  192.168.1.104  192.168.1.103  SIP       200 OK |
49   50.3580050  192.168.1.103  192.168.1.101  SIP       200 OK |
50   50.3730790  192.168.1.101  192.168.1.100  SIP       200 OK |
51   0.20721700  192.168.1.102  192.168.1.103  HTTP      PUT/ 0xd4e9d44d
52   0.20871800  192.168.1.103  192.168.1.102  HTTP      POST/00010001216d11141c6f6548b987
```

Fig. 19 Wireshark capture of message flow on the first proposed framework

Table 1 CPU usage by the Kernel and designed modules in the first proposed controller

Offered load (cps)	1000	2000	3000	4000	5000	6000	7000	8000	9000	10,000
Kernel	8.25	9.36	10.58	11.36	12.25	13.75	15.95	18.85	20.25	21.12
NFV Allocation module	1.25	2.65	3.85	4.75	5.85	6.36	7.45	8.88	9.52	10.01
Routing module	1.05	2.85	3.45	4.78	5.65	6.35	7.78	8.75	9.95	10.04
VNFs info module	3.32	4.52	5.78	6.65	7.75	8.36	9.42	10.87	11.15	12.26
Resource info module	3.12	4.85	5.45	6.36	7.25	8.45	9.62	10.25	11.14	12.03
Topology info module	2.25	3.78	4.65	5.12	6.23	7.12	8.02	9.52	10.04	11.12
VNFs Manager module	2.04	3.74	4.52	5.36	6.41	7.23	8.66	9.14	10.75	11.10
Flow Manager module	2.21	3.20	4.23	5.02	6.77	7.85	8.15	9.23	10.52	11.05

the LBBSRT and RR mechanisms increases over time (especially as the offered load increases). For instance, when the offered load is 12,000 cps, the maximum throughput of the proposed mechanism in the period of seconds 30–40 is approximately 12,000 cps. However, the throughput of the LBBSRT is approximately 6000 cps, and the throughput of the RR mechanism is approximately 3800 cps.

The average response time of the PMs is demonstrated in Fig. 21. The response time rises as the offered load increases. However, it does not exceed approximately 40 ms. This delay is pertinent to the load of 12,000 cps. The delay in such a load is higher for the other methods, which can be observed in the next figures.

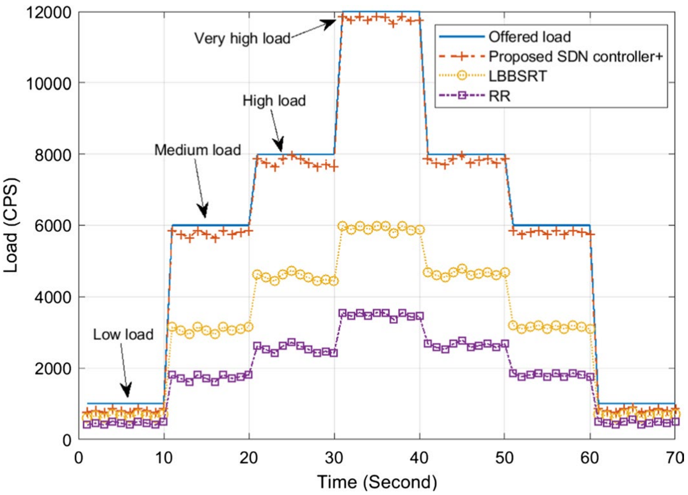


Fig. 20 The throughput comparison of the second proposed mechanism with LBBSRT and RR (during a time)

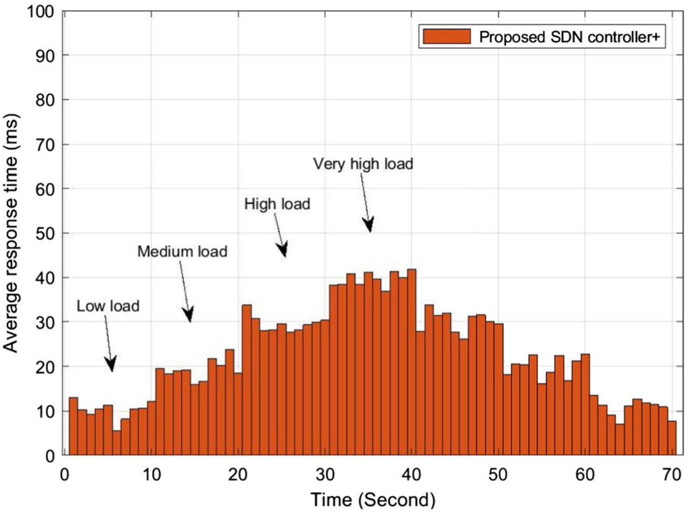


Fig. 21 The average response time during the time in the proposed mechanism

Figure 22 belongs to the LBBSRT method. The response time of the PMs in each of the four steps is greater in this method than in the proposed method. Particularly, as the offered load increases, such that the input load reaches its maximum amount from seconds 30 to 40, the response time reaches exceeds 200 ms, which is four times higher than the proposed method.

Figure 23 indicates the response time during the time in the RR mechanism. As shown, compared to the proposed and LBBSRT methods, the response time of

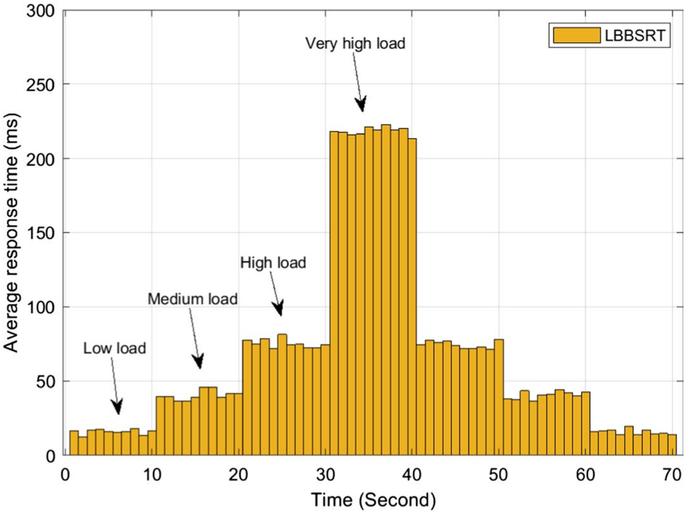


Fig. 22 The average response time during the time in the LBBSRT mechanism

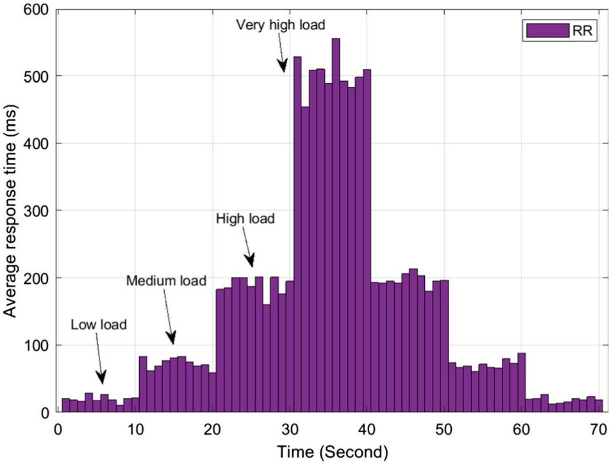


Fig. 23 The average response time during the time in the RR mechanism

the PMs is much higher. Comparing Fig. 20 to Fig. 23 reveals that not only does the proposed method reach a higher throughput, but also it does not lead to delay increase, even in heavy traffic. This is due to the centralized management nature of the network by this method. On the contrary, not only is the throughput lower in the distributed RR and LBBSRT methods but also making contact encounters a great deal of delay. This is due to the lack of integrated resource management and deliberate routing. It should be noted that SDN controller+ carried out the routing of the layer 3 and 7 simultaneously. This issue affects the throughput enhancement and delay reduction.

The injected traffic to the network was merely the offered load so far, and the PMs did not have any background traffic. By having the background traffic, the efficiency evaluation of the methods will be conducted in the form of three new scenarios. In each of the three scenarios, the offered load is 8000 cps.

- The first scenario: the background traffic of each PM is 100 cps (each of the eight PMs is equal to 100 cps).
- The second scenario: the background traffic of the PM1 is equal to 100 cps. The background traffic of the PM2 is two times of the PM1. The background traffic of the PM3 is two times of the PM2, and so on (until PM8).
- The third scenario: the background traffic of the PM1 is equal to 100 cps. The background traffic of the PM2 is three times of the PM1. The background traffic of the PM3 is three times of the PM2, and so on (until PM8).

Figures 24, 25 and 26 demonstrate the throughput of the PMs in different mechanisms and three scenarios. When there is equal background traffic (the first scenario), the average throughput of the proposed, the LBBSRT, and the RR mechanisms are 7500, 5500, and 3500 cps, respectively. This issue indicates that even with equal background traffic, the proposed framework's throughput is greater than the other two methods.

When the background traffic is added to the network in the form of a second scenario, the throughput difference between the proposed method and the other two methods increases; as an illustration, the throughput difference between the proposed method and the LBBSRT exceeds 3000 cps. In both first scenarios, the proposed method's throughput is approximately constant (7500 cps) and very close to the offered load.

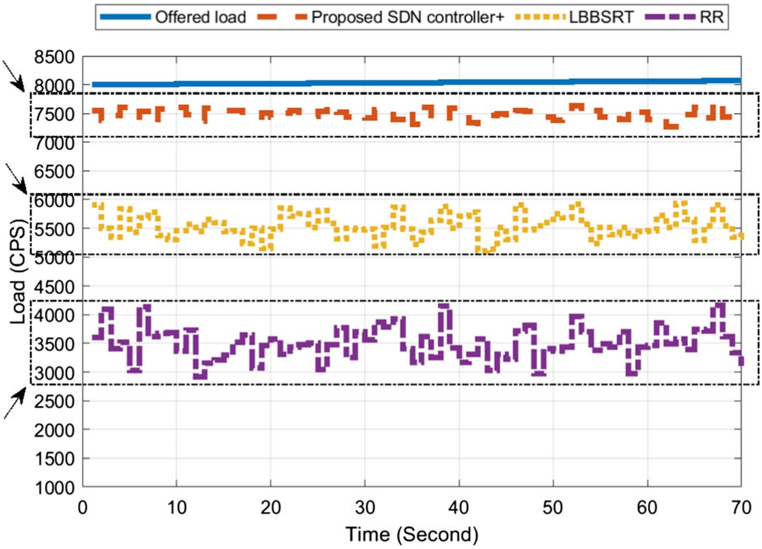


Fig. 24 The throughput in the first scenario (the background traffic is equal for each PM)

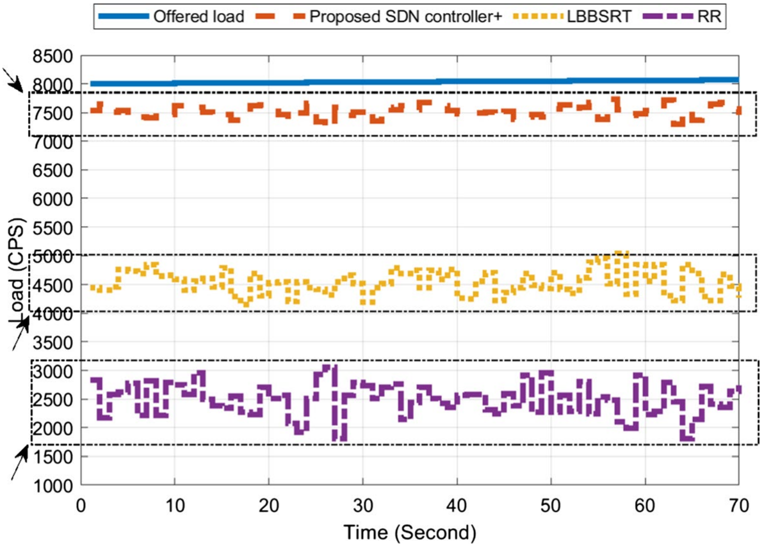


Fig. 25 The throughput in the second scenario (the background traffic of the PM is two times of the preceding PM)

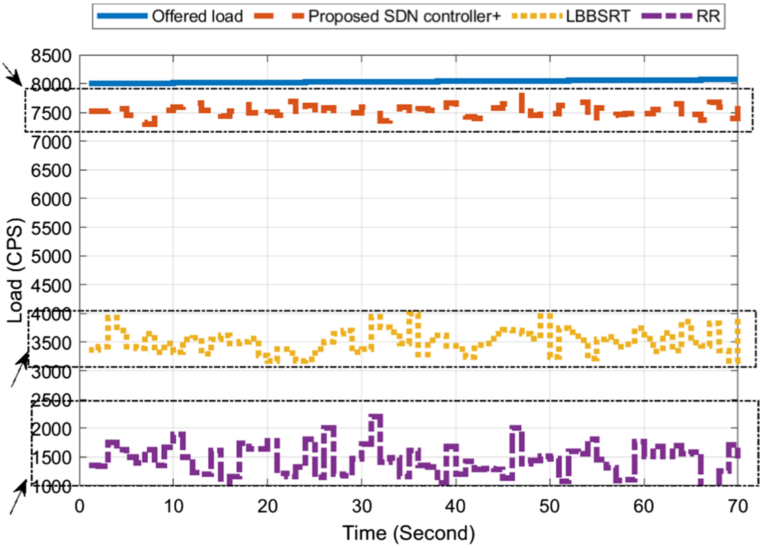


Fig. 26 The throughput in the third scenario (the background traffic of the PM is three times of the preceding PM)

In the third scenario, the throughput reduction of the LBBSRT and RR methods continues, and their throughput reaches approximately 3500 and 1500 cps, respectively (Fig. 26). The throughput reduction of these methods is due to the background traffic enhancement of the PMs and their inability to (1) handling the traffic and

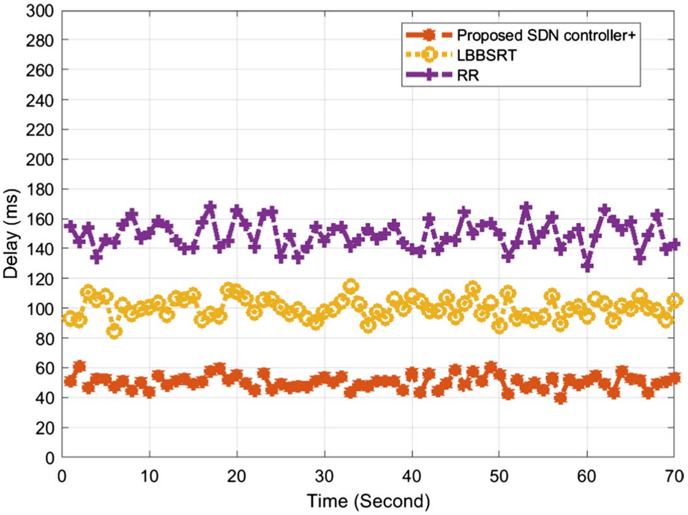


Fig. 27 The delay in the first scenario (the background traffic is equal for all PMs)

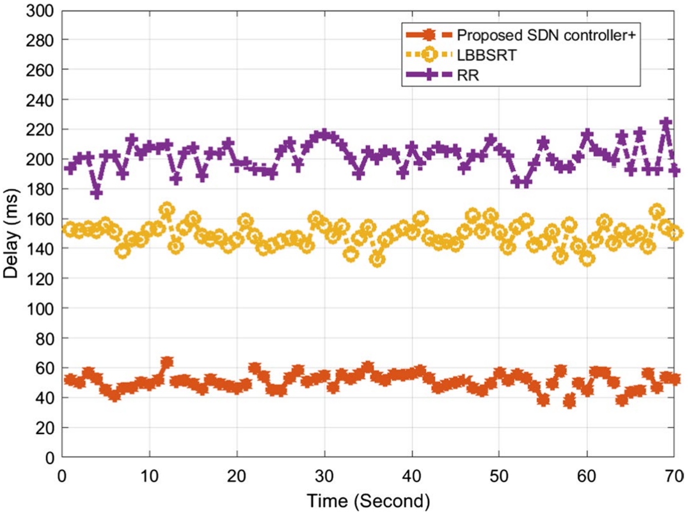


Fig. 28 The delay in the second scenario (the background traffic of the PM is two times of the preceding PM)

(2) smart resource management. Due to the lack of a global view over the whole network, they cannot distinguish the offered load traffic from the background traffic. However, thanks to the SDN controller+, the proposed framework is completely aware of the details of the network and traffic and manages the VoIP network concerning this fact.

Figures 27, 28 and 29 demonstrate the methods' delay in three scenarios. The proposed mechanism does not face extra delay due to the deliberate management

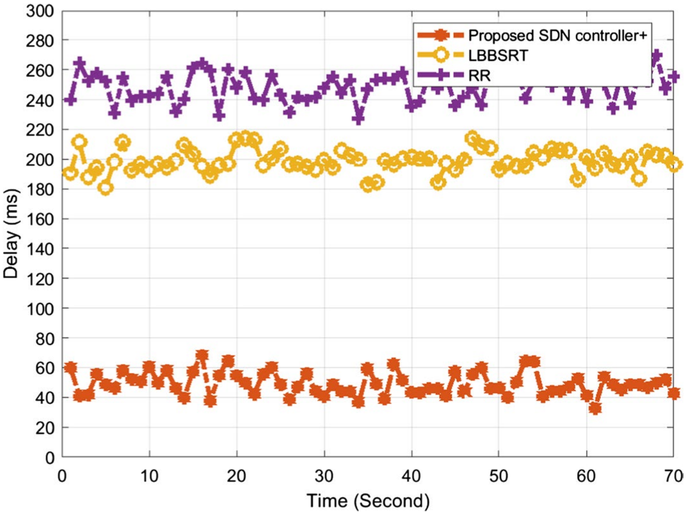


Fig. 29 The delay in the third scenario (the background traffic of the PM is three times of the preceding PM)

of the traffic. In this case, its delay in every three scenarios is approximately constant and less than 60 ms. Nevertheless, subject to the background traffic, the delay of the two other mechanisms increases. It means that as the background traffic increases, the delay rises, as well. For instance, the delay in the LBBSRT of the three scenarios is 100, 160, and 200 ms, respectively, which is much higher than that of the proposed method. Also, Fig. 30 illustrate overall latency.

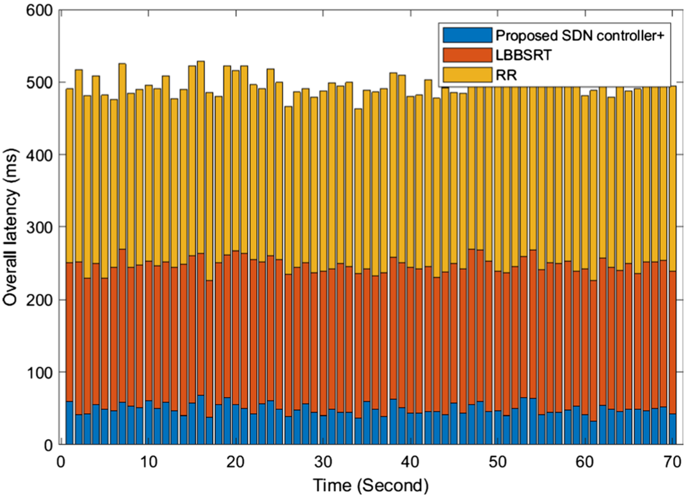


Fig. 30 The overall latency

In the next experiment, the R factor of the media will be examined. 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.

Figure 31 indicates the R factor in various methods. The maximum R factor belongs to the proposed framework, and in the worst case, it would not be less than 92, indicating a great deal of satisfaction among VoIP users. On the contrary, in the other methods, the R factor is lower, having fluctuation.

Figure 32 demonstrates the exchanged messages in the second proposed framework. Examining this figure that is captured by the Wireshark software indicates that the contact between UAC and UAS is according to Fig. 7. It means that without the hardware SIP proxy, only using the SDN controller+ and OpenFlow switches. The last number of IP addresses of the UAC, OVS1, SDN controller+, OVS2, and UAS are 100, 101,102, 103, and 104, respectively. In each one of the four steps *registration*, *initiate session*, *media*, and *tear down* only one message (green color underlines in the figure) is sent to the SDN controller+ (102). The decision is made only for that one message, and the remaining flow follows that decision. This issue indicates that the exchanged messages between the control and data planes are not overhead and bottlenecking in the proposed framework. This is an important issue that the SDN controller+ can achieve high throughput and low delay with the lowest messages received from the data plane.

Table 2 indicates the CPU usage by the Kernel and the SDN controller+ modules. The used CPU of the designed modules is lower than that of the Linux Kernel. In terms of CPU usage, none of the modules does lead to the controller's resource saturation. Also, the CPU usage by the modules is linear, not exponential. Therefore,

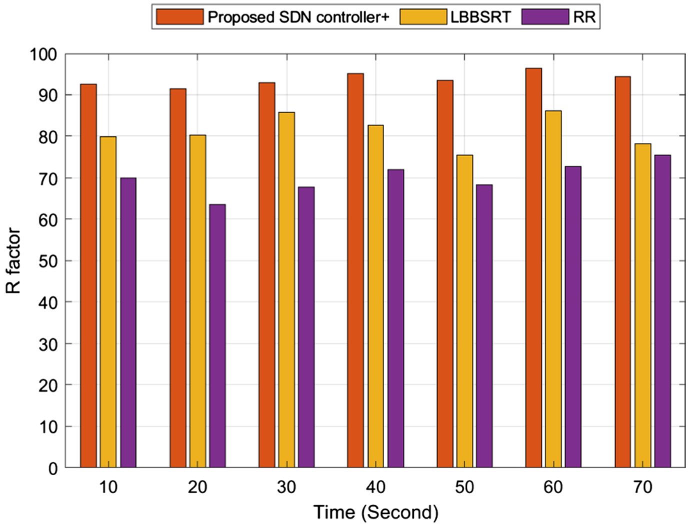
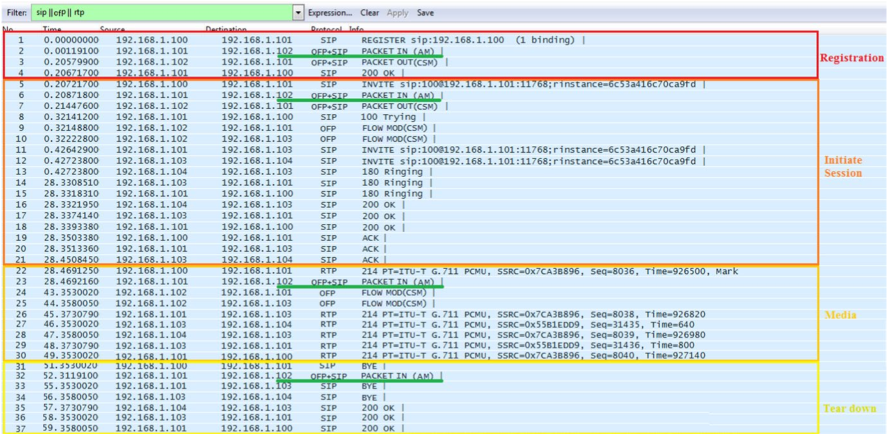


Fig. 31 The R factor of the media in the second proposed framework, compared to the LBBSRT and RR



```
No. Time        Source          Destination     Protocol Info
 1  0.00000000  192.168.1.100   192.168.1.101   SIP      REGISTER sip:192.168.1.100  (1 binding) |
 2  0.00119100  192.168.1.101   192.168.1.102   OFP+SIP  PACKET IN (AM) |
 3  0.20579900  192.168.1.102   192.168.1.101   OFP+SIP  PACKET OUT(CSM) |
 4  0.20671700  192.168.1.101   192.168.1.100   SIP      200 OK |
 5  0.20721700  192.168.1.100   192.168.1.101   SIP      INVITE sip:100@192.168.1.101:11768;rinstance=6c53a416c70ca9fd |
 6  0.20871800  192.168.1.101   192.168.1.102   OFP+SIP  PACKET IN (AM) |
 7  0.21447600  192.168.1.102   192.168.1.101   OFP+SIP  PACKET OUT(CSM) |
 8  0.32141200  192.168.1.101   192.168.1.100   SIP      100 Trying |
 9  0.32148800  192.168.1.102   192.168.1.101   OFP      FLOW MOD(CSM) |
10  0.32222800  192.168.1.102   192.168.1.103   OFP      FLOW MOD(CSM) |
11  0.42642900  192.168.1.101   192.168.1.103   SIP      INVITE sip:100@192.168.1.101:11768;rinstance=6c53a416c70ca9fd |
12  0.42723800  192.168.1.103   192.168.1.104   SIP      INVITE sip:100@192.168.1.101:11768;rinstance=6c53a416c70ca9fd |
13  0.42723800  192.168.1.104   192.168.1.103   SIP      180 Ringing |
14  28.3308510  192.168.1.103   192.168.1.101   SIP      180 Ringing |
15  28.3318310  192.168.1.101   192.168.1.100   SIP      180 Ringing |
16  28.3321950  192.168.1.104   192.168.1.103   SIP      200 OK |
17  28.3374140  192.168.1.103   192.168.1.101   SIP      200 OK |
18  28.3393380  192.168.1.101   192.168.1.100   SIP      200 OK |
19  28.3503380  192.168.1.100   192.168.1.101   SIP      ACK |
20  28.3513360  192.168.1.101   192.168.1.103   SIP      ACK |
21  28.4508450  192.168.1.103   192.168.1.104   SIP      ACK |
22  28.4691250  192.168.1.100   192.168.1.101   RTP      214 PT=ITU-T G.711 PCMU, SSRC=0x7CA3B896, Seq=8036, Time=926500, Mark
23  28.4692160  192.168.1.101   192.168.1.102   OFP+SIP  PACKET IN (AM) |
24  43.3530020  192.168.1.102   192.168.1.101   OFP      FLOW MOD(CSM) |
25  44.3580050  192.168.1.102   192.168.1.103   OFP      FLOW MOD(CSM) |
26  45.3730790  192.168.1.101   192.168.1.103   RTP      214 PT=ITU-T G.711 PCMU, SSRC=0x7CA3B896, Seq=8038, Time=926820
27  46.3530020  192.168.1.103   192.168.1.104   RTP      214 PT=ITU-T G.711 PCMU, SSRC=0x55B1EDD9, Seq=31435, Time=640
28  47.3580050  192.168.1.104   192.168.1.103   RTP      214 PT=ITU-T G.711 PCMU, SSRC=0x7CA3B896, Seq=8039, Time=926980
29  48.3730790  192.168.1.103   192.168.1.101   RTP      214 PT=ITU-T G.711 PCMU, SSRC=0x55B1EDD9, Seq=31436, Time=800
30  49.3530020  192.168.1.101   192.168.1.100   RTP      214 PT=ITU-T G.711 PCMU, SSRC=0x7CA3B896, Seq=8040, Time=927140
31  51.3530020  192.168.1.100   192.168.1.101   SIP      BYE |
32  52.3119100  192.168.1.101   192.168.1.102   OFP+SIP  PACKET IN (AM) |
33  55.3530020  192.168.1.101   192.168.1.103   SIP      BYE |
34  56.3580050  192.168.1.103   192.168.1.104   SIP      BYE |
35  57.3730790  192.168.1.104   192.168.1.103   SIP      200 OK |
36  58.3530020  192.168.1.103   192.168.1.101   SIP      200 OK |
37  59.3580050  192.168.1.101   192.168.1.100   SIP      200 OK |
```

Fig. 32 Wireshark capture of message flow on the second proposed framework

we can conclude that the designed modules do not lead to the bottlenecking of the SDN controller+. According to the numbers of this table, the SDN controller+ is scalable at least to the 10000 cps load. It should be noted that the modules collecting and keeping the information have greater CPU usage compared to the others.

CPU usage is almost 28% better than the traditional method (without SDN). Note that on average more than 18% of the processor is consumed by the Kernel. In contrast, designed modules consume less than 7% of CPU power (almost one-third).

At the end of this section, we point out that the improvement in the results discussed above is due to a fully centralized framework with an intelligent controller that is aware of the state of the entire network, which we designed and analyzed in this article. It would not have been possible without the use of up-to-date and comprehensive technologies such as SDN and NFV as well as the OpenFlow protocol.

4 Conclusion and further works

A proper design of the network and load distribution among the SIP proxies is one of the factors to prevent the breakdown of the VoIP services. For each of the load control methods of the SIP, there is an overload limit, and if the input traffic exceeds that limit, the method encounters failure. This failure leads to throughput instability. In this paper, employing the SDN and NFV technologies, we have designed and implemented two frameworks for the centralized control of the VoIP network traffic in two signaling and media phases. In the first framework, the SIP proxies are provided in the form of VoIP VNFs and based on the demand and the input load of the network. When the load is low, the numbers of the VoIP VNFs are low, and when the load is high, the VoIP VNFs numbers are greater. The signaling traffic is routed by using the allocated VoIP VNFs. The media traffic is also routed by the controller. The second framework provides the whole process of making a call and also the media exchange in the centralized controller. In other words, we will be

Table 2 The CPU usage by the Kernel and the designed modules in the second proposed controller

Offered load (cps)	1000	2000	3000	4000	5000	6000	7000	8000	9000	10,000
Kernel	7.78	9.25	10.02	10.36	12.41	13.85	14.44	18.15	21.01	21.11
SIP Routing module	1.88	2.45	2.75	4.65	4.95	6.25	6.75	8.63	10.32	10.74
IP Routing module	1.12	3.27	3.84	4.72	5.43	6.18	6.99	8.08	9.44	9.12
SIP user info module	2.11	4.27	4.85	6.45	6.79	7.95	9.33	10.07	11.04	11.90
Resource info module	3.22	3.74	5.14	5.22	7.18	8.72	10.64	10.98	11.15	11.84
Topology info module	2.11	3.56	4.36	6.08	6.25	6.64	8.14	9.75	10.65	11.36
SIP Flow Manager module	2.88	4.14	4.65	5.12	6.10	6.54	8.78	8.99	9.69	11.05
Media Flow Manager module	2.28	3.35	4.41	4.85	6.45	7.73	7.19	9.46	10.28	10.10

in no need of the hardware proxies of the SIP in the data plane. It means that the routing of signaling and media traffic (layers 3 and 7 routing) is executed in an integrated way in the controller. Both frameworks are implemented in the real testbed, and the evaluation performance and the results of both were provided in detail. The results indicate that productivity has been improved dramatically compared to the conventional structure of the VoIP networks. For example, the average throughput and resource efficiency increased by at least 28% and the average response time decreased by 34%. The overall latency has also been reduced by almost 39%. All this is due to the centralized and integrated control of the entire VoIP network by the proposed SDN-based controllers. For further works, we will generalize and develop the proposed controller for the IP Multimedia Subsystem (IMS) [39]. Also, we will implement it by using the Open IMS. The development of proposed frameworks for non-VoIP networks is also on our agenda. In this regard, it is possible to improve networks such as the Internet of Things or 5G networks by using softwarization and virtualization of networks concepts.

Author contributions All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by AM. The first draft of the manuscript was written by AM, and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding No funding was received.

Data availability statement My manuscript will be freely available to any researcher wishing to use them for non-commercial purposes, without breaching participant confidentiality (on demand).

Declarations

Confict of interest There is no conflict of interest.

References

1. Hu Z, Yan H, Yan T, Geng H, Liu G (2020) Evaluating QoE in VoIP networks with QoS mapping and machine learning algorithms. Neurocomputing 386:63–83
2. Nazih W, Hifny Y, Elkilani W, Abdelkader T, Faheem H (2019) Efficient detection of attacks in sip based VoIP networks using linear l1-SVM classifier. Int J Comput Commun Control 14(4):518–529
3. Johnston AB (2015) SIP: understanding the session initiation protocol. Artech House, Norwood
4. Ahson SA, Ilyas M (2018) SIP handbook: services, technologies, and security of session initiation protocol. CRC Press, Boca Raton
5. Coronado E, Khan SN, Riggio R (2019) 5G-empower: a software-defined networking platform for 5G radio access networks. IEEE Trans Netw Serv Manag 16(2):715–728
6. Sarker Z, Perkins C, Singh V, Ramalho M (2021) RTP control protocol (RTCP) feedback for congestion control. Internet RFC, no 8888
7. Montazerolghaem A, Moghaddam MHY, Leon-Garcia A (2017) OpenSIP: toward software-defined SIP networking. IEEE Trans Netw Serv Manag 15(1):184–199
8. Gurbani VK, Chiang T-C, Kumar S (2002) SIP: a routing protocol. Bell Labs Tech J 6(2):136–152
9. Montazerolghaem A (2021) Optimizing VoIP server resources using linear programming model and autoscaling technique: an SDN approach. Concurr Comput Pract Exp 33(21):e6424

10. Yang L, Ng B, Seah WK, Groves L, Singh D (2020) A survey on network forwarding in software-defined networking. J Netw Comput Appl 102947
11. Nisar K, Welch I, Hassan R, Sodhro AH, Pirbhulal S (2020) A survey on the architecture, application, and security of software defined networking. Internet Things 100289
12. Yang S, Li F, Trajanovski S, Yahyapour R, Fu X (2020) Recent advances of resource allocation in network function virtualization. IEEE Trans Parallel Distrib Syst 32(2):295–314
13. Cheng X, Wu Y, Min G, Zomaya AY (2018) Network function virtualization in dynamic networks: a stochastic perspective. IEEE J Sel Areas Commun 36(10):2218–2232
14. McKeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Shenker S, Turner J (2008) Openflow: enabling innovation in campus networks. ACM SIGCOMM Comput Commun Review 38(2):69–74
15. Hu T, Lan J, Zhang J, Zhao W (2019) EASM: efficiency-aware switch migration for balancing controller loads in software-defined networking. Peer-to-Peer networking and applications 12(2):452–464
16. Lu J, Zhang Z, Hu T, Yi P, Lan J (2019) A survey of controller placement problem in software-defined networking. IEEE Access 7:24290–24307
17. Belgaum MR, Musa S, Alam MM, Suud MM (2020) A systematic review of load balancing techniques in software-defined networking. IEEE Access 8:98612–98636
18. Rafique W, Qi L, Yaqoob I, Imran M, Rasool RU, Dou W (2020) Complementing IoT services through software defined networking and edge computing: a comprehensive survey. IEEE Commun Surv Tutor 22(3):1761–1804
19. Zhong H, Fang Y, Cui J (2018) Reprint of LBBSRT: an efficient SDN load balancing scheme based on server response time. Future Gener Comput Syst 80:409–416
20. Khebbache S, Hadji M, Zeghlache D (2017) Virtualized network functions chaining and routing algorithms. Comput Netw 114:95–110
21. Gedia D, Perigo L (2018) Performance evaluation of SDN-VNF in virtual machine and container. In: 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN). IEEE, pp 1–7
22. Hawilo H, Jammal M, Shami A (2019) Network function virtualization-aware orchestrator for service function chaining placement in the cloud. IEEE J Sel Areas Commun 37(3):643–655
23. Yi B, Wang X, Li K, Huang M et al (2018) A comprehensive survey of network function virtualization. Comput Netw 133:212–262
24. Ilievski G, Latkoski P (2020) Efficiency of supervised machine learning algorithms in regular and encrypted VoIP classification within NFV environment. Radioengineering 29(1)
25. Hyun D, Kim J, Jeong JP, Kim H, Park J, Ahn T (2016) SDN-based network security functions for VoIP and volte services. In: 2016 International Conference on Information and Communication Technology Convergence (ICTC). IEEE, pp 298–302
26. Olariu C, Zuber M, Thorpe C (2017) Delay-based priority queueing for VoIP over software defined networks. In: 2017 IFIP/IEEE symposium on integrated network and service management (IM). IEEE, pp 652–655
27. Xezonaki M-E, Liotou E, Passas N, Merakos L (2018) An SDN QoE monitoring framework for VoIP and video applications. In: 2018 IEEE 19th international symposium on “A World of Wireless, Mobile and Multimedia Networks” (WoWMoM). IEEE, pp 1–6
28. Barakabitze AA, Sun L, Mkwawa I-H, Ifeachor E (2019) A novel QoE-aware SDN-enabled, NFV-based management architecture for future multimedia applications on 5G systems. arXiv preprint arXiv:1904.09917
29. Koné B, Kora AD (2021) Management and orchestration for network function virtualization in a VoIP testbed: a multi-domain case. In: 2021 44th International Conference on Telecommunications and Signal Processing (TSP). IEEE, pp 372–376
30. Streijl RC, Winkler S, Hands DS (2016) Mean opinion score (MOS) revisited: methods and applications, limitations and alternatives. Multimed Syst 22(2):213–227
31. Manousos M, Apostolacos S, Grammatikakis I, Mexis D, Kagklis D, Sykas E (2005) Voice-quality monitoring and control for VoIP. IEEE Internet Comput 9(4):35–42
32. Ray PP, Kumar N (2021) SDN/NFV architectures for edge-cloud oriented IoT: a systematic review. Comput Commun
33. Floodlight controller. https://floodlight.atlassian.net/wiki. Accessed: 2021-01-30
34. Open vswitch. www.openvswitch.org. Accessed: 2021-01-30
35. Open source sip proxy server. https://opensips.org/. Accessed: 2021-01-30

36. Network monitoring tool. https://oprofile.sourceforge.io. Accessed: 2021-01-30
37. Open source test tool, traffic generator for the sip protocol. http://sipp.sourceforge.net/. Accessed: 2021-01-30
38. Software to test VoIP systems and IP networks. https://startrinity.com/. Accessed: 2021-01-30
39. Di Mauro M, Galatro G, Postiglione F, Tambasco M (2021) Performability of network service chains: stochastic modeling and assessment of softwarized IP multimedia subsystem. IEEE Trans Depend Secure Comput

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.