Journal of Ambient Intelligence and Humanized Computing (2023) 14:12981–13001
https://doi.org/10.1007/s12652-022-04230-z

ORIGINAL RESEARCH

Optimized software-defined multimedia framework: networking and computing resource management

Ahmadreza Montazerolghaem[1]

Received: 25 June 2021 / Accepted: 6 June 2022 / Published online: 13 July 2022

Abstract
Today, the multimedia over IP (MoIP) network has become a cost-effective and efficient alternative to the public switched telephone network (PSTN). Free applications for multimedia transmission over the Internet have become increasingly popular, gaining considerable popularity around the world. This communication consists of two phases, i.e., signaling phase and media exchange phase. The SIP protocol is responsible for the MoIP network signaling to provide services such as VoIP, voice and video conferencing, video over demand (VoD), and instant messaging. This application layer protocol has been standardized by the IETF for initiating, managing, and tearing down multimedia sessions and has been widely used as the main signaling protocol on the Internet. The signaling and media are handled by SIP proxies and network switches, respectively. One of the most critical challenges in MoIP is the overloading of SIP proxies and network switches. Because of these challenges, a wide range of network users experiences a sharp drop in service quality. Overload occurs when there are not enough processing resources and memory to process all the messages due to the lack of proper routing. This study aims to model the routing problem in MoIP by providing a framework based on software-defined networking (SDN) technology and a convex mathematical programming model to prevent overload. The proposed framework is simulated and implemented using various scenarios and network topologies. The results show that throughput, latency, message retransmission rate, and resource consumption have improved using the proposed approach.

Keywords Networking and computing resource allocation · Multi-objective optimization · Software-defined networking · Overload control · Voice over IP (VoIP) · SIP servers

1 Introduction

Nowadays, interest in multimedia over IP (MoIP) networks has increased significantly. The breadth and variety of services provided by IP networks have resulted in the network integration using various technologies to provide MoIP networks (Rosenberg et al. 2002, 2008). Communication in this network consists of two main phases, i.e., signaling and media phases. The SIP protocol has developed extensively for VoIP-based voice and video call signaling (Nahum et al. 2007). SIP is considered as a signaling protocol in the IP multimedia subsystem (IMS), which is the proposed signaling platform for the next generation of the networks (Ohta 2008; Noel and Johnson 2007). Furthermore, 3G and 4G wireless technologies are now being used more and more by the carriers. As a result, almost all mobile phones and devices will have to support the SIP protocol as the signaling protocol for multimedia sessions soon (Ohta 2004; Gurbani and Jain 2004; Pascual 2009). After the signaling phase, the media phase is considered for the multimedia exchange. The information required for initiating a session between the caller (or UAC) and callee (or UAS) is exchanged before the session through SIP signaling. SIP signaling is performed through the transfer of request and response messages by the SIP proxy servers (Hilt and Widjaja 2008). The route of the request and response messages does not depend on that of the media. The media is transmitted through network switches by the RTP protocol. Therefore, signaling messages pass through both network switches and SIP proxies, while media messages only pass through network switches. Messages require routing to reach the destination, i.e., callee (McGregor et al. 2006; Hilt et al. 2011; Gurbani et al. 2014; Shen and Schulzrinne 2010).

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

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

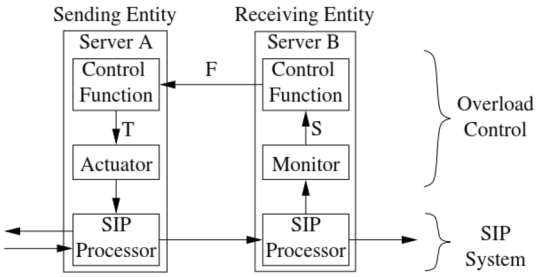


Fig. 1 Traditional SIP proxy structure

SIP proxies and network switches are responsible for routing the 7th and 3rd layers of the messages, respectively (Schulzrinne et al. 2002). Improper routing of messages results in the misuse of SIP proxy resources as well as network switches. Since the two phases operate independently, routing and resource management are also performed independently at two levels, i.e., SIP proxies and network switches (Lindholm et al. 2007; Cortes et al. 2004). Besides, this type of routing is distributed among all switches and proxies. Therefore, improper routing eliminates a global view of the entire network as a single network, which results in an overload phenomenon that directly affects the network throughput. The overload phenomenon in MoIP Leads to a drop in the network throughput to zero (Ohta 2008; Wanke et al. 2007). Therefore, using proper and centralized resource management is necessary. In other words, resources in MoIP are divided into two categories of computing resources (SIP proxy servers) and networking resources (network switches), which should be centrally managed to perform a proper resource-based routing.

Therefore, the MoIP network consists of a network of SIP proxies that route signaling messages to the destination hop-by-hop using the header of the 7th layer (Gokhale and Lu 2005; Chi et al. 2008). The network itself is located on a network of traditional switches that are responsible for routing messages through the header of the 3rd layer (Chi et al. 2008). Unfortunately, in this traditional structure, routing is distributed and is performed without the integrated consideration of computing and networking resources, as well as having remarkable complexities (Pesch et al. 2005). As mentioned earlier, improper routing causes an overload, which has serious consequences for the network.

The traditional structure of SIP proxies is shown in Fig. 1. In this structure, each SIP proxy independently has an overload control unit, including a load control function, a monitor, and an actuator (Homayouni et al. 2009). The SIP processor is also responsible for investigating the SIP messages and their routing. In this structure, the detection and elimination of the overload occur in the collaboration of all proxies. The traffic load statistics are shared locally among the proxies. The lack of a central controller in this structure causes unawareness about the topology and total resources of network (Ram et al. 2008; Xia et al. 2014; Montazerolghaem et al. 2017).

In this paper, a centralized overload control framework for the SIP network is proposed using the SDN concept (Nunes et al. 2014). This framework is an integrated concept for the management of all computing and networking resources. Using SDN, which is a new network architecture, the entire network and its components are seen as a single virtual network that can be controlled using designed software and application programming interfaces (APIs) (Kreutz et al. 2014; Bera et al. 2019). Therefore, a communication protocol between software and hardware is necessary, which led to considering the OpenFlow protocol (Qu et al. 2020). Any network component that supports this protocol can be located on the SDN network. The SDN architecture has several features, namely, configuration upgrade, performance improvement, dynamics, integrated resource and power management, traffic management, programming capability, cost reduction, and high accessibility (He et al. 2019). SDN architecture consists of three main components, i.e., data plane, control plane, and application plane. The data plane includes network devices such as routing devices, switches, etc., so that they do not have a control or software unit for automatic decision making. The intelligent unit of the network is located in the SDN controllers that maintain the overall structure of the network (control plane). The application plane includes a group of applications such as routing, firewall, load balancing, monitoring, and so on. The communication protocol between the planes is a set of standard Open APIs, including the OpenFlow protocol. With such interfaces, the controller is capable of programming the heterogeneous devices of the network dynamically (Bonfim et al. 2019).

1.1 Contributions

The main contributions of this study are as follows:

- Preventing overload and load distribution in the SIP network through integrated resource management;
- Designing an SDN-based centralized framework for routing at both switch and proxy levels;
- Mathematical modeling of the routing problem in SIP networks in two phases of signaling and media;
- Developing the proposed mathematical model for routing to satisfy complementary constraints, including energy constraints and link costs;
- Implementing and simulating the proposed framework in a real experiment platform and evaluating the performance of the framework using a variety of scenarios;

Therefore, the following objectives can be achieved:

(a). Increasing the total throughput of the SIP network,
(b). Reducing the latency in initiating a session,
(c). Routing in proxies and switch routes for load distribution.

1.2 Organization

The paper is organized as follows: in Section 2, we provide an overview of the related works. Proposed approach, system model, and problem formulations are given in Section 3. Section 4 presents the performance evaluation. The conclusion and future work are given Section 5.

2 Related Work

According to the literature, a large part of previous studies in this area has tried to solve the overload problem by redesigning the traditional overload control system shown in Fig. 1. The proper network design and load distribution among SIP proxies are one of the factors to prevent service failure in MoIP. An approach to eliminate the overload is to distribute new input traffic among the SIP servers based on their available capacity using a load balancer (Montazerolghaem et al. 2017; Jiang et al. 2012; Singh and Schulzrinne 2007). Therefore, the possibility of overload occurrence in a specific server decreases. However, these methods' performance depends on the load balancer performance since all the signaling traffic of the SIP network pass through it. In the case of severe overload, the occurrence of the overload even in the load balancer is possible, and therefore, its capacity should be increased using effective methods. Jiang et al. (2012) introduced three load distribution algorithms, namely, call-join-shortest-queue (CJSQ), transaction-join-shortest-queue (TJSQ), and transaction-least-work-left (TLWL). In CJSQ, the amount of work that a server should do is estimated based on the number of sessions assigned to the server. The load balancer has a counter to count the assigned calls for each server. When a new `Invite` request (corresponding to a new call) is received, the request is assigned to a server that its counter shows the least number, and the number then increases by one. When the load balancer receives an `Ok` response to a `Bye` request in a call, the call process is completed by the server, and the counter number decreases by one. One of the limitations of this method is that the number of calls assigned to a server is not always an accurate measure of the server's load. There might be long stopping (load-free) times between the call transactions. Besides, calls may consist of different numbers of transactions.

Therefore, in TJSQ, the load balancer routes the new call load to the server with the least number of active transactions, instead of the one with the least number of active calls. In this situation, counters determine the number of transactions assigned to each server, and new calls are assigned to the servers with the least number shown by the counters. One of the limitations of this method is that all the transactions are weighted equally. In the SIP protocol, `Invite` transactions are more expensive than `Bye` transactions because the state machine of the `Invite` transaction is more complex than that of the `Bye` transaction. Experiments show that `Invite` transactions cost ca. 75% more than `Bye` transactions (Jiang et al. 2012). In TLWL, the new call is routed to the server that has the least work to do. Here, work (or load) is based on the relative estimates of the transaction costs. In other words, in this method, the counters determine the total weighted transactions assigned to each server. Therefore, the new calls are assigned to the servers that have the lowest counter number. For example, if a server is processing `Invite` (with a relative cost of 1.75) and `Bye` (with a relative cost of 1.0) transactions, the server load will be equal to 2.75.

Montazerolghaem et al. (2017) proposed the fixed weighted average response time (FWAR) and history weighted average response time (HWAR) algorithms for load distribution by the load balancer. These algorithms use the window and the response times of the SIP proxies to estimate the load of the proxies. In the FWAR algorithm, the window size is fixed. However, in HWAR, a history of response times is kept in the windows using a mathematical model. Each SIP proxy has a corresponding window in the load balancer. The contents of each window are the history of the proxy's response time over time. HWAR is capable of registering an immense number of response time values. Singh and Schulzrinne (2007) introduced a load balancer for SIP using the hash function, in which the requests are routed based on the callee. In Ram et al. (2008) and Wu et al. (2007), a redundancy model has been proposed for load distributors in the SIP network. OpenAIS (Cheng et al. 2008) is introduced to monitor the status of downstream proxies, and the detected damaged proxy is quickly replaced with another proxy. In Kambourakis et al. (2006), a structure called Snocer is introduced for load distribution in SIP networks, in which the user can be connected to the load distributor if the access is provided. Otherwise, the user is connected to one of the network proxies directly or through DNS. In Cheng et al. (2008) and Kambourakis et al. (2006), backup servers have been used as redundancies. In Montagna and Pignolo (2008), the redundancy is classified into four types, namely, customer-based, DNS-based, IP-based, and database repetition-based redundancy. Moreover, the types of load division and distribution methods have been introduced using DNS, SIP identifier, proxies with the same address, and translator of the network address, and their performance is evaluated on a two-level architecture consisting of SIP proxies.

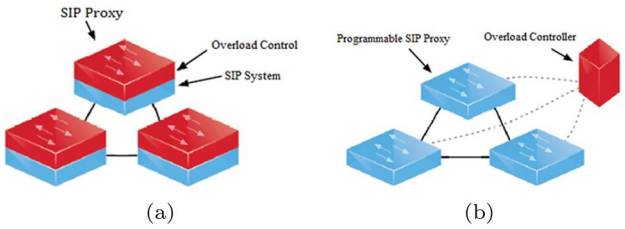


Fig. 2 Evolution in the SIP network (**a**) traditional SIP network, (**b**) SDN-based SIP network (proposed approach)

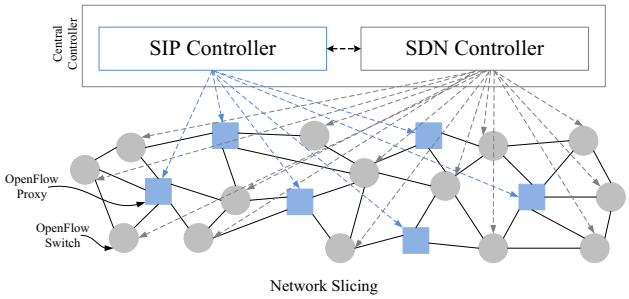


Fig. 3 The proposed framework consisting of a central controller for controlling OpenFlow switches and proxies

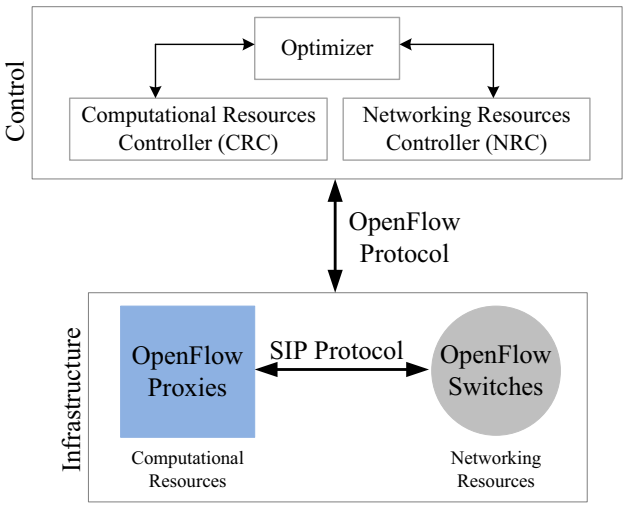


Fig. 4 The relationship among the infrastructure layer and the control layer, consisting of the optimizer, NRC, and CRC modules

As a conclusion of this section, it can be noted that almost all the previous approaches in this field are distributed, which results in complexity, performance decrement, and the lack of integrated and centralized resource management. Existing SIP products are not based on *SDN* and *network softwarization* concepts. As opposed to conventional hardware-centric products where hardware, protocols, and software components are embedded into a *closed* equipment, the SDN control plane is implemented in software in the form of a *logically centralized* controller. The controller runs on a single or a set of servers, has a global view of the network, and makes traffic management decisions according to operational policies. This is precisely what we intend to do in this paper. The success of VoIP depends heavily on *real-time communication* between the clients and the proxies. Data management is an important aspect of VoIP architecture. The handling of the vast amount of VoIP data is challenging using the usual data management methods due to the different constraints (such as resources, routing, QoS requirements, etc.). Hence, the *dynamic and scalable communication network architecture* is required to handle the seamless data generated by a sizably voluminous number of connected VoIP clients, which we have designed, implemented, and evaluated this architecture in this paper based on the *SDN* concepts. These concepts are some of the best techniques to control such vast data to have a robust, reliable, and efficient large-scale VoIP communication network.

3 Proposed approach

Figure 2 depicts the proposed approach in this study. Figure 2a shows the overall structure of a SIP proxy consisting of two main elements. Depending on the overload control method, these elements are distributed throughout the network to route messages and overload control. As mentioned, one of the most critical limitations of this network is its complexity and poor performance. To solve these problems and according to the concept of SDN, this study introduces an SDN-based SIP network (Fig. 2b). In this regard, SIP proxy components are converted into simple programmable equipment by OpenFlow agent. OpenFlow agent runs on network devices such as switches and SIP proxies. This interacts with a centralized SDN controller using the OpenFlow protocol. The OpenFlow agent translates OpenFlow commands into device specific actions. A centralized controller is also responsible for routing and controlling overloads throughout the network.

3.1 Proposed framework

Figures 3 and 4 show the proposed conceptual architecture and its details and modules, respectively. As can be seen, at the infrastructure level, there are two types of simple equipment based on the OpenFlow protocol, i.e., OpenFlow switch and OpenFlow proxy. In practice, network slicing occurs, and there is an integrated computing and networking equipment with the OpenFlow protocol at the infrastructure level. Furthermore, a central controller in the control level includes two units for simultaneous control of SIP traffic and media traffic. In practice, there is a centralized control

unit for integrated network management that has three modules, namely, *optimizer*, *computational resource controller (CRC)*, and *networking resource controller (NRC)* (Fig. 4). Communication between control and infrastructure is performed through OpenFlow protocol (Lara et al. 2013). We have used standard equipment that supports this protocol (version 1.5.1) (Nygren and et al. 2015).

There are two types of traffic in the infrastructure:

1- Signaling traffic (SIP messages) between proxies and switches;

2- Media traffic (RTP messages) between switches. A detailed description of the NRC, CRC, and optimizer modules is provided in the following.

– NRC:

The NRC module is responsible for collecting network information such as the topology of switches, links, as well as their available capacity and power.

– CRC:

The CRC module is responsible for collecting information about the topology of proxies, as well as their available capacity and power.

– Optimizer:

The optimizer module includes a linear programming mathematical model for optimal allocation of resources of proxies and switches. The input of this module is from NRC and CRC modules. The objective of this resource allocation model is to maximize the admission of calls so that they can be distributed among the proxies and switches without overloading. In other words, due to constraints of SIP network resources, this model should perform the following steps to increase the total network throughput:

(a) increasing the number of admitted call requests in the network, and then, (b) distributing the admitted requests among the existing routes from source to destination with no overload occurrence, and (c) maximizing the throughput in media phase according to available resources of switches.

As a result, the problem of optimal resource allocation is performed in signaling and media phases. Logically, there are three sub-problems of controlling the admission of call requests, determining the optimal route of admitted requests, and determining the route of the media. Therefore, optimal answers for resource consumption in proxies and switches, call admission rate, and signaling and media routes for admitted calls should be provided. Table 1 shows the nomenclature in the proposed model.

Note that, at the beginning of each time interval τ, each switch and proxy device transmits information to the controller (NRC, CRC) via the OpenFlow (data collection phase at time t_g - gathering phase). Depending on the data received and the network state, the optimizer calculates and notifies the switches and proxies of the optimal values of load and network traffic at time t_c (calculation phase) and at time t_n (notification phase). The optimizer then enters idle mode. In this method, the times t_n and t_g could be ignored. Switch and proxy devices are certainly operating non-stop at every single time intervals τ (service phase). SIP requests that reach the control unit at time τ must wait for admitting and processing until the beginning of the next τ time. This time could be ignored relative to the total time and could also be reduced by shortening the time τ.

3.2 System model

In general, the number of SIP proxies and switches with constrained processing and memory resources in a SIP network is equal to n and z, respectively. Each proxy or switch consumes its resources to initiate a session between its local users or users of different domains. In this system model, it is assumed that the binary and symmetric matrix L_{kl} includes the SIP network topology. In this matrix, $L_{kl} = 1$ indicates a physical link between the nodes k and l, while $L_{kl} = 0$ shows that there is no physical link between the two nodes. The elements in the main diagonal of the matrix are equal to zero. The square matrix $\mathbb{C}^{ij}$ indicates the number of call requests from the users in proxy i to the ones in proxy j. In this case, $\mathbb{C}^{ii}$ and $\mathbb{C}^{ij}$ are the number of local requests of proxy i and the outbound requests of proxy i. Assume that the optimal number of admitted calls from proxy i to proxy j is C^{ij}, so that $C^{ij} < \mathbb{C}^{ij}$. Furthermore, $\mathbb{D}^{ij}$ indicates the number of signaling requests from switch i to switch j, with an optimal value of D^{ij}. The number of media requests between switches i and j is $\bar{\mathbb{D}}^{ij}$ with an optimal value of $\bar{D}^{ij}$. X_{kv}^{ij} indicates the number of admitted requests from source i to destination j, which pass through two adjacent proxies of k and v ($X_{kv}^{ij} \leq C^{ij}$). Therefore, the total number of admitted requests from i to j is determined by C^{ij}, so that X_{kv}^{ij} indicates its distribution method among the routes between i and j. In this study, X_{kv}^{ij} is defined as the flow from source i to destination j, which pass through the routes k and v (Fig. 5).

Following is a list of categories of main notations.

- **Admission:**

– proxy:

∗ signaling:

·$\mathbb{C}^{ij}$: the number of *signaling* requests from the users in *proxy i* to the ones in *proxy j*.
·C^{ij}: the optimal number of admitted *signalings* from *proxy i* to *proxy j*.

Table 1 Key Nomenclatures

R	Set of OpenFlow Proxies
S	Set of OpenFlow Switches
H	Set of Links
τ	Time Slot
n	Number of Proxies
z	Number of Switches
h	Number of Links
γ	The proxy admission rate coefficient in the objective function
ρ	The switch admission rate coefficient in the objective function (signaling phase)
φ	The proxy resource coefficient in the objective function
δ	The switch resource coefficient in the objective function (signaling phase)
ζ	The switch admission rate coefficient in the objective function (media phase)
ς	The switch resource coefficient in the objective function (media phase)
ι	Link cost coefficient in the objective function
λ	The proxy power coefficient in the objective function
μ	The switch power coefficient in the objective function
α_1, β_1	Local call factor on proxy resources
α_2, β_2	Outbound call factor on proxy resources
ω, ψ	Call factor on switch resources
ξ, η	Media factor on switch resources
σ	Call factor on proxy power
ϱ	Call factor on switch power
L_{kl}	Network topology
ℓ_{kl}	Link weight
C^{ij}	The proxy call admission (number of calls established from proxy i to j)
$\mathbb{C}^{ij}$	The proxy call requests (number of calls requests from proxy i to j)
D^{ij}	The switch call admission (number of calls established from switch i to j)
$\mathbb{D}^{ij}$	The switch call requests (number of calls requests from switch i to j)
$\bar{D}^{ij}$	The switch media admission (number of media established from switch i to j)
$\bar{\mathbb{D}}^{ij}$	The switch media requests (number of media requests from switch i to j)
X_{kv}^{ij}	Signaling traffic flow (the number of calls from i to j, which must be relayed from k to v)
Y_{kv}^{ij}	Media traffic flow (the number of medias from i to j, which must be relayed from k to v)
p_v^{ϕ}	CPU usage of proxy v
P_v^{ϕ}	The strict bound of proxy v CPU
m_v^{ϕ}	Memory usage of proxy v

Table 1 (continued)

M_v^{ϕ}	The strict bound of proxy v memory
e_v^{ϕ}	Power usage of proxy v
E_v^{ϕ}	The strict bound of proxy v power
p_v^{θ}	CPU usage of switch v for signalling
P_v^{θ}	The strict bound of switch v CPU for signaling
m_v^{θ}	Memory usage of switch v for signalling
M_v^{θ}	The strict bound of switch v memory for signaling
$\bar{p}_v^{\theta}$	CPU usage of switch v for media
$\bar{P}_v^{\theta}$	The strict bound of switch v CPU for media
$\bar{m}_v^{\theta}$	Memory usage of switch v for media
$\bar{M}_v^{\theta}$	The strict bound of switch v memory for media
e_v^{θ}	Power usage of switch v
E_v^{θ}	The strict bound of switch v power

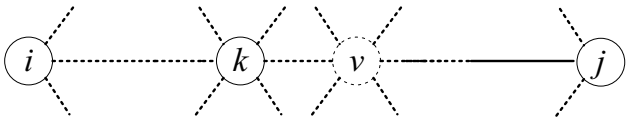


Fig. 5 Transmitting calls from the origin i to the destination j assisted by nodes k and v

– switch:

∗ signaling:

·$\mathbb{D}^{ij}$: the number of *signaling* requests from *switch i* to *switch j*.
·D^{ij}: the optimal number of admitted *signaling* from *switch i* to *switch j*.

∗ media:

·$\bar{\mathbb{D}}^{ij}$: the number of *media* requests between *switches i* and *j*.
·$\bar{D}^{ij}$: the optimal number of admitted *media* from *switch i* to *switch j*.

- **Routing:**

– X_{kv}^{ij}: the number of calls from origin i (*proxy i*) to destination j (*proxy j*), which must be relayed from *proxy k* to *proxy v*.

Each proxy or switch relies on the remaining amount of its resources, i.e., the processor and memory, to perform its operations. The remaining values of the processor and memory of the proxy or switch v are shown with P_v^{ϕ}, M_v^{ϕ}, P_v^{θ}, and M_v^{θ}, respectively. The controller is responsible for managing these resources. At the beginning of each time interval τ, each proxy or switch transmits the number of its new requests along with the P_i and M_i values to the controller through the OpenFlow protocol (data-gathering phase at the beginning of time interval τ). The controller computes the optimal values of the parameters X_{kl}^{ij}, D^{ij}, C^{ij}, and resources according to the L_{kl} matrix, the information received, and using the linear programming model introduced below (computing phase). It then notifies the proxies for applying the values (notifying phase).

We prove that the overload controlling problem of signaling and media in SIP networks is in the form of a mixed-integer linear program (MILP) and is, therefore, NP-hard and suffers from high time complexity in large-scale scenarios (Appendix A). So, by performing the relaxation on the proposed MILP model, the linear programming (LP) model can be presented as follows.

3.2.1 Signaling phase

In the following, a linear multi-objective model is introduced for the routing of signaling messages according to the problem constraints.

$$\begin{aligned} max\ & \gamma\left(\frac{\sum_{i=1}^{n}\sum_{j=1}^{n} C^{ij}}{\sum_{i=1}^{n}\sum_{j=1}^{n} \mathbb{C}^{ij}}\right) + \rho\left(\frac{\sum_{i=1}^{z}\sum_{j=1}^{z} D^{ij}}{\sum_{i=1}^{z}\sum_{j=1}^{z} \mathbb{D}^{ij}}\right) \\ & - \varphi\left(\frac{\sum_{g=1}^{n} p_g^{\phi}}{\sum_{g=1}^{n} P_g^{\phi}} + \frac{\sum_{g=1}^{n} m_g^{\phi}}{\sum_{g=1}^{n} M_g^{\phi}}\right) \\ & - \delta\left(\frac{\sum_{f=1}^{z} p_f^{\theta}}{\sum_{f=1}^{z} P_f^{\theta}} + \frac{\sum_{f=1}^{z} m_f^{\theta}}{\sum_{f=1}^{z} M_f^{\theta}}\right) \end{aligned} \tag{1}$$

Subject to:

Proxies constraints:

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

$$\sum_{k=1}^{n} L_{kv} X_{kv}^{ij} = \sum_{u=1}^{n} L_{vu} X_{vu}^{ij}, \forall i,j,v \in R, i \neq v, j \neq v \tag{3}$$

$$\sum_{k=1}^{n} L_{kv} X_{kv}^{iv} = C^{iv}, \forall i,v \in R, i \neq v \tag{4}$$

$$\sum_{u=1}^{n} L_{vu} X_{vu}^{vj} = C^{vj}, \forall j,v \in R, j \neq v \tag{5}$$

$$X_{kv}^{ii} = 0, \forall i,k,v \in R \tag{6}$$

$$X_{ki}^{ij} = 0, \forall i,j,k \in R \tag{7}$$

$$\alpha_1 C^{vv} + \alpha_2 (\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} L_{vk} X_{vk}^{ij} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} L_{kv} X_{kv}^{ij}) \leq p_v^{\phi}, \forall v \in R \tag{8}$$

$$\beta_1 C^{vv} + \beta_2 (\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} L_{vk} X_{vk}^{ij} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} L_{kv} X_{kv}^{ij}) \leq m_v^{\phi}, \forall v \in R \tag{9}$$

$$p_v^{\phi} \leq P_v^{\phi}, \forall v \in R \tag{10}$$

$$m_v^{\phi} \leq M_v^{\phi}, \forall v \in R \tag{11}$$

Switches constraints:

$$D^{ij} \leq \mathbb{D}^{ij}, \forall i,j \in S \tag{12}$$

$$\sum_{k=1}^{z} L_{kv} X_{kv}^{ij} = \sum_{u=1}^{z} L_{vu} X_{vu}^{ij}, \forall i,j,v \in S, i \neq v, j \neq v \tag{13}$$

$$\sum_{k=1}^{z} L_{kv} X_{kv}^{iv} = D^{iv}, \forall i,v \in S, i \neq v \tag{14}$$

$$\sum_{u=1}^{z} L_{vu} X_{vu}^{vj} = D^{vj}, \forall j,v \in S, j \neq v \tag{15}$$

$$X_{kv}^{ii} = 0, \forall i,k,v \in S \tag{16}$$

$$X_{ki}^{ij} = 0, \forall i,j,k \in S \tag{17}$$

$$\omega(\sum_{i=1}^{z}\sum_{j=1}^{z}\sum_{k=1}^{z} L_{vk} X_{vk}^{ij} + \sum_{i=1}^{z}\sum_{j=1}^{z}\sum_{k=1}^{z} L_{kv} X_{kv}^{ij}) \leq p_v^{\theta}, \forall v \in S \tag{18}$$

$$\psi(\sum_{i=1}^{z}\sum_{j=1}^{z}\sum_{k=1}^{z} L_{vk} X_{vk}^{ij} + \sum_{i=1}^{z}\sum_{j=1}^{z}\sum_{k=1}^{z} L_{kv} X_{kv}^{ij}) \leq m_v^{\theta}, \forall v \in S \tag{19}$$

$$p_v^{\theta} \leq P_v^{\theta}, \forall v \in S \tag{20}$$

$$m_v^{\theta} \leq M_v^{\theta}, \forall v \in S \tag{21}$$

Variables: $C^{ij}, D^{ij}, X_{kl}^{ij}, p_v^{\phi}, m_v^{\phi}, p_v^{\theta}, m_v^{\theta} \geq 0, \forall i,j,k,l,v$

In this model, the objective function is to maximize the throughput of proxies and switches, as well as reduce their resource consumption. $C^{ij}/\mathbb{C}^{ij}$ and $D^{ij}/\mathbb{D}^{ij}$ indicate the ratio of the number of admitted requests to the number of total requests for proxy and switch, respectively. The third and fourth terms in the objective function represent the consumption of the resources, i.e., processor and memory of the proxies and switches, respectively, which should be minimized. The coefficients γ, ρ, φ, δ indicate the degree of impact of throughput or resources in the objective function. Using these coefficients, a trade-off can be created between throughput and resources, so if the coefficients related to the throughput (γ and ρ) are higher, the objective function seeks to maximize the throughput by consuming more resources. In contrast, if the coefficients related to the resources (φ and δ) are higher, the objective function seeks to minimize resource consumption.

Constraints are divided into two general categories: proxies constraints and switches constraints. Constraint (2) indicates that the maximum number of admissions is equal to the number of requests ($C^{ij} \leq \mathbb{C}^{ij}$). Constraint (3) creates a balance between the input and output flows in each proxy, which means that the sum of the input flows to the proxy v from source i to destination j should be equal to that of the output flow. Constraint (4) requires the sum of the input flows to the proxy v from source i, passing through the proxies in the route k to be equal to C^{iv}. The next constraint distributes the sum of the output flows from server v to destination j among the servers adjacent to server v. Constraint (6) does not allow a flow of the same source and destination. Constraint (7) prevents forming loops in the route. Constraints (8) to (11) belong to the

allocation of proxy resources. Constraint (8) indicates that the processor of the proxies with coefficients α_1 and α_2 is consumed to handle local and outbound calls. Constraint (9) shows that the memory of the proxies with β_1 and β_2 coefficients is consumed to handle local and outbound calls.

The loose upper bounds of the optimal consumption of the processing and memory resources in the model are p_v and m_v, respectively, and their strict upper bound is equal to P_v and M_v, according to constraints (10) and (11).

Constraint (12) limits the admission of requests by switches to their capacity. Constraints (13) to (17) are defined to route traffic signaling in switches. Constraints (18) to (21) are also used to manage switch resources. Since the sides of equations (18) and (19) are not homogeneous, the coefficients ω and ψ are used for normalization.

Finally, without overloading, the model seeks to increase the total network throughput through optimal resource allocation. The total throughput is considered as the sum of the flows passing through the network. Therefore, according to the resources, the controller tries to maximize the admission of the calls so that it can distribute them among proxies and network switches.

3.2.2 Media phase

The next phase of the controller is media routing among the network switches. Therefore, the following linear model is proposed. The objective is to increase media throughput in the network of switches with minimal resource consumption.

$$max\ \zeta\left(\frac{\sum_{i=1}^{z}\sum_{j=1}^{z}\bar{D}^{ij}}{\sum_{i=1}^{z}\sum_{j=1}^{z}\bar{\mathbb{D}}^{ij}}\right) - \varsigma\left(\frac{\sum_{f=1}^{z}\bar{p}_f^{\theta}}{\sum_{f=1}^{z}\bar{P}_f^{\theta}} + \frac{\sum_{f=1}^{z}\bar{m}_f^{\theta}}{\sum_{f=1}^{z}\bar{M}_f^{\theta}}\right) \tag{22}$$

Subject to:

Switches constraints:

$$\bar{D}^{ij} \le \bar{\mathbb{D}}^{ij}, \forall i,j \in S \tag{23}$$

$$\sum_{s=1}^{z} L_{sl}Y_{sl}^{ij} = \sum_{e=1}^{z} L_{le}Y_{le}^{ij}, \forall i,j,l \in S, i \ne l, j \ne l \tag{24}$$

$$\sum_{s=1}^{z} L_{sl}Y_{sl}^{il} = \bar{D}^{il}, \forall i,l \in S, i \ne l \tag{25}$$

$$\sum_{e=1}^{z} L_{le}Y_{le}^{lj} = \bar{D}^{lj}, \forall j,l \in S, j \ne l \tag{26}$$

$$Y_{sl}^{ii} = 0, \forall i,s,l \in S \tag{27}$$

$$Y_{si}^{ij} = 0, \forall i,j,s \in S \tag{28}$$

$$\xi\left(\sum_{i=1}^{z}\sum_{j=1}^{z}\sum_{s=1}^{z} L_{ls}Y_{ls}^{ij} + \sum_{i=1}^{z}\sum_{j=1}^{z}\sum_{s=1}^{z} L_{sl}Y_{sl}^{ij}\right) \le \bar{p}_l^{\theta}, \forall l \in S \tag{29}$$

$$\eta\left(\sum_{i=1}^{z}\sum_{j=1}^{z}\sum_{s=1}^{z} L_{ls}Y_{ls}^{ij} + \sum_{i=1}^{z}\sum_{j=1}^{z}\sum_{s=1}^{z} L_{sl}Y_{sl}^{ij}\right) \le \bar{m}_l^{\theta}, \forall l \in S \tag{30}$$

$$\bar{p}_l^{\theta} \le \bar{P}_l^{\theta}, \forall l \in S \tag{31}$$

$$\bar{m}_l^{\theta} \le \bar{M}_l^{\theta}, \forall l \in S \tag{32}$$

Variables: $\bar{D}^{ij}, Y_{sl}^{ij}, \bar{p}_l^{\theta}, \bar{m}_l^{\theta}, \ge 0, \forall i,j,s,l$

In Eq. (22), the coefficients ζ and ς create a trade-off between media throughput and resources. If ζ is greater than ς, increasing the throughput will have a higher weight. Otherwise, the higher weight will belong to the reduction of resource consumption. Constraint (23) determines the maximum capacity of switches for media exchange. Constraints (24) to (28) are considered for media routing according to media traffic (Y_{le}^{ij}). Constraints (29) to (32) allocate resources to media routing. The coefficients ξ and η are used to normalize equations (29) and (30).

3.2.3 Complementary constraints

To generalize the problem of optimal resource allocation, the complementary constraints are introduced below. These constraints are considered to the proposed models described above. These complementary constraints are introduced in two parts: first, to consider the cost for each link, and second, to consider the power consumption of the proxies and switches.

The cost of each link: In this section, we look for a route with the lowest cost. Therefore, a weight is considered for each link (ℓ_{kl}). The matrix $\bar{L}_{kl}$ represents the total weight of all network topology links (Eq. 33).

$$\bar{L}_{kl} = \ell_{kl} \times L_{kl} \tag{33}$$

The objective function is also changed by Eq. (34) to achieve the route with the lowest cost. This equation seeks to reduce the cost of the route, in addition to maximizing the throughput and minimizing the resources consumed.

$$max\ \gamma\left(\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}C^{ij}}{\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbb{C}^{ij}}\right)+\rho\left(\frac{\sum_{i=1}^{z}\sum_{j=1}^{z}D^{ij}}{\sum_{i=1}^{z}\sum_{j=1}^{z}\mathbb{D}^{ij}}\right)-\varphi\left(\frac{\sum_{g=1}^{n}p_g^{\phi}}{\sum_{g=1}^{n}P_g^{\phi}}+\frac{\sum_{g=1}^{n}m_g^{\phi}}{\sum_{g=1}^{n}M_g^{\phi}}\right)-\delta\left(\frac{\sum_{f=1}^{z}p_f^{\theta}}{\sum_{f=1}^{z}P_f^{\theta}}+\frac{\sum_{f=1}^{z}m_f^{\theta}}{\sum_{f=1}^{z}M_f^{\theta}}\right)-\iota\left(\sum_{k=1}^{h}\sum_{l=1}^{h}\bar{L}_{kl}\right) \tag{34}$$

Energy: The objective function (Eq. 35) results in a route with maximum throughput and minimum energy consumption. For this purpose, power consumption reduction in proxies and switches should be considered. Constraint (36) consumes the remaining power of proxies on routing media messages[1]. The coefficient σ is used to normalize this equation. Eq. (37) determines the maximum power of the proxies. In constraint (38), the routing of the switches is limited to their remaining power, and ϱ normalizes the equation. Eq. (39) considers an upper bound for the power of the switches.

$$max\ \gamma\left(\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}C^{ij}}{\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbb{C}^{ij}}\right)+\rho\left(\frac{\sum_{i=1}^{z}\sum_{j=1}^{z}D^{ij}}{\sum_{i=1}^{z}\sum_{j=1}^{z}\mathbb{D}^{ij}}\right)-\lambda\left(\frac{\sum_{g=1}^{n}e_g^{\phi}}{\sum_{g=1}^{n}E_g^{\phi}}\right)-\mu\left(\frac{\sum_{f=1}^{z}e_f^{\theta}}{\sum_{f=1}^{z}E_f^{\theta}}\right) \tag{35}$$

$$\sigma\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}L_{lk}Y_{lk}^{ij}+\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}L_{kl}Y_{kl}^{ij}\right)\leq e_l^{\phi},\forall l\in R \tag{36}$$

$$e_l^{\phi}\leq E_l^{\phi},\forall l\in R \tag{37}$$

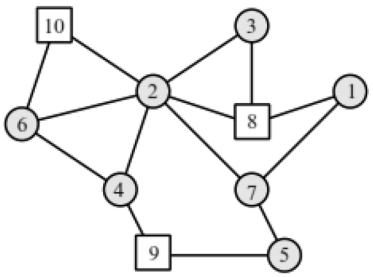
$$\varrho\left(\sum_{i=1}^{z}\sum_{j=1}^{z}\sum_{k=1}^{z}L_{lk}Y_{lk}^{ij}+\sum_{i=1}^{z}\sum_{j=1}^{z}\sum_{k=1}^{z}L_{kl}Y_{kl}^{ij}\right)\leq e_l^{\theta},\forall l\in S \tag{38}$$

$$e_l^{\theta}\leq E_l^{\theta},\forall l\in S \tag{39}$$

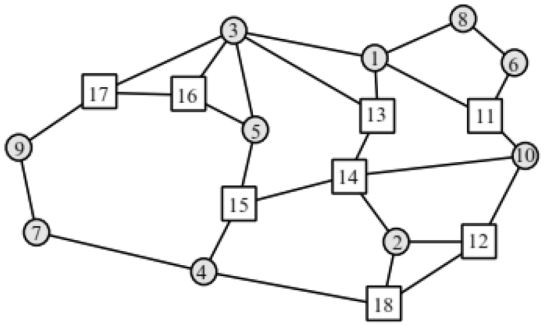
4 Performance evaluation

In this section, we describe the simulation process of the proposed approach and the implementation of its prototype. Also, the details and the results of the performance evaluation are presented.

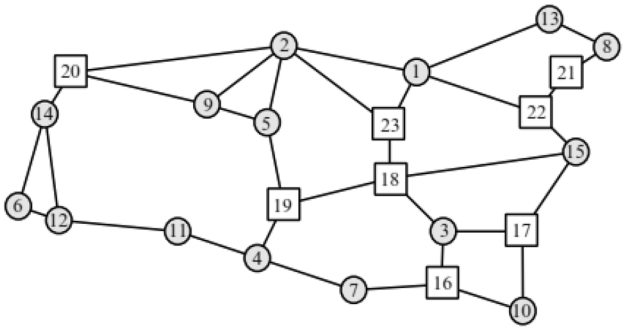
[1] Media is generally more energetic than signaling.



(a) The topology for the small-scale network (N10).



(b) The topology for the medium-scale network (N18).



(c) The topology for the fairly large-scale network (N23).

Fig. 6 The test topologies. Squares and circles indicate proxies and switches

4.1 Numerical results

To evaluate the proposed method, the topologies depicted in Fig. 6, along with three scenarios, namely, low-load, medium-load, and high-load scenarios, have been used in the MATLAB software environment. Scenarios ($\mathbb{C}^{ij}$,$\mathbb{D}^{ij}$,$\bar{\mathbb{D}}^{ij}$) are random values with the normal distribution. Also, each scenario is investigated with different states of γ and φ in the signaling phase (shown in $F1$ to $F4$), as well as different states of ρ and δ in the media phase (shown in $G1$ to $G4$). The initial random values considered for P_v, M_v, and E_v were equal to 100, 512, and 350, respectively.

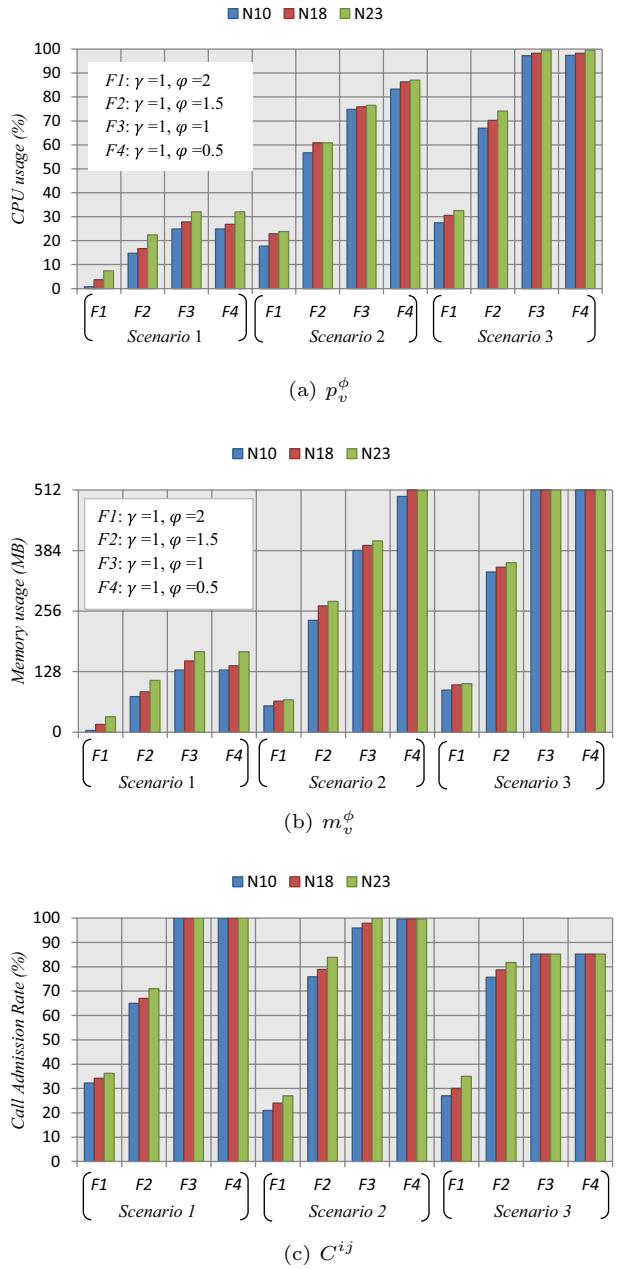


Fig. 7 The consumption of the processor and memory, along with admission rate for signaling phase in proxies

4.1.1 First experiment—signaling phase

In this experiment, we evaluate the performance of proxies and switches in the signaling phase. In this section, during the experiments, several assumptions are considered: $\alpha_1 = 0.02168, \alpha_2 = 0.07169, \beta_1 = 0.02329, \beta_2 = 0.08012,$ $\omega = 0.06345, \psi = 0.04635$. These parameters are tuned by the *training process*: we run data through the operations of the model, compare the resulting prediction with the actual value for each data instance, evaluate the accuracy, and adjust until we find the best values. Tuning is the process of maximizing a model's performance without over-fitting or creating too high a variance. The improved performance reveals which parameter settings are more favorable (tuned) or less favorable (untuned). In our experiments, we chose these values as a trade-off between fast convergence and responsiveness to input change. We ran various tests to tune these parameters.

Proxy: Fig. 7 shows the optimal values of p_v^{ϕ}, m_v^{ϕ}, and C^{ij} for different states of F and in three scenarios and three topologies. According to the figure, in all scenarios and topologies, with the change of state from $F1$ to $F4$, resource consumption and the percentage of call admission increase since the ratio of the coefficients γ and φ to each other indicates the importance of call admission or resource preservation. In $F1$, resource preservation is more important than in $F4$ and even causes a percentage of calls to be blocked. In contrast, in the state of $F4$, it is important to admit as many calls as possible, even if it costs the higher consumption of the resources. Using a trade-off for these coefficients, both the higher percentage of call admission and more appropriate resource consumption can be achieved.

Although moving from scenario 1 (low load) to scenario 3 (high load) increases the resource consumption, the input load to the network might increase so much that all the call cannot be initiated even with the consumption of all network resources (scenario 3 of $F3$ and $F4$ states).

Figure 8 shows the CPU consumption of proxies in the three topologies. Different combinations of F state and scenario are considered in this figure (e.g., $F1S1$ and $F4S2$). As can be seen, the load is distributed in a balanced form among the proxies, regardless of the type of topology and scenario. This is because the controller is aware of the remaining resources of the proxies and decides on load distribution and traffic routing accordingly. Similar trends are obtained for memory (Data are not shown).

Figure 9 depicts the effects of parameters α and β on processing resources, memory resources, local calls, and outbound calls. As shown in Figs. 9a, b, c, admitted local and outbound calls increase and decrease, respectively, by increasing α_1 compared to α_2. The processor consumption also decreases because α_1 is the coefficient of C^{vv} in Eq. (8) and handling local calls requires fewer resources. Since β_1 is the coefficient of C^{vv} in the equation of memory allocation (Eq. 9), an increase in β_1 increases the number of admitted local calls (Fig. 9e). Increasing local calls requires less memory compared to outbound calls (Fig. 9d).

Switch: Fig. 10 shows the optimal values of p_v^{θ}, m_v^{θ}, and D^{ij} for different states of G and in three scenarios and three topologies. G is a function of the coefficients ρ and δ, which are the weights of "admission of the flows" and "resources of the switches", respectively, in the objective function (Eq. 1). According to the figure, in all scenarios, by moving the state from $G1$ to $G4$, resource consumption and the percentage of call admission increase since the coefficient ρ

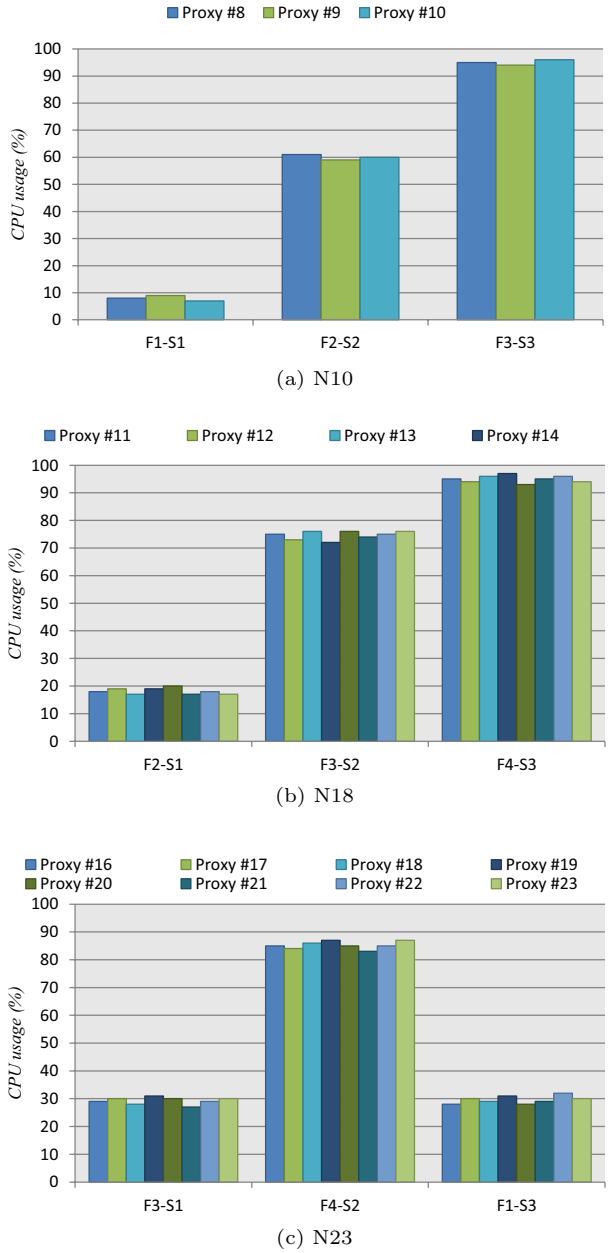


Fig. 8 CPU consumption of proxies in three topologies in the signaling phase

in $G4$ increases. Furthermore, moving from scenario 1 (low load) to scenario 3 (high load) increases resource consumption. However, the results are independent of the topology and the controller performance is not affected by the topology. This means that SDN can manage an extensive network of OpenFlow switches due to having a global view.

CPU and memory consumption of switches in the signaling phase for topologies N10 are given in Table 2. As can be seen in the table, the consumptions of resources in switches are very close to each other. For example, in scenario 1 and $G4$ state, the CPU consumption of all the seven switches

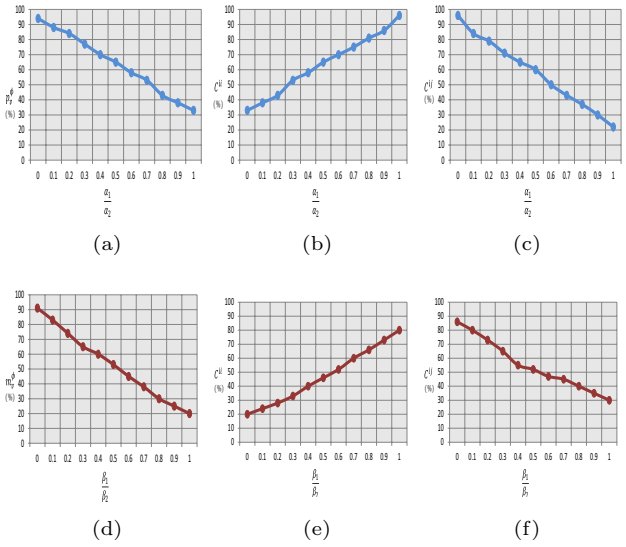


Fig. 9 The effects of α and β on p_v^{ϕ}, m_v^{ϕ}, C^{ii}, and C^{ij} of the proxies (sensitivity analysis): **a** Effects of α_1/α_2 on CPU. **b** Effects of α_1/α_2 on local calls. **c** Effects of α_1/α_2 on outbound calls. **d** Effects of β_1/β_2 on memory. **e** Effects of β_1/β_2 on local calls. **f** Effects of β_1/β_2 on outbound calls

is ca. 22%. Similar results are obtained for the memory. This indicates that the load is evenly distributed among the switches. Similar results are obtained for the N18 and N23 topologies (Data are not shown).

4.1.2 Second experiment—media phase

The media phase involves the switches, while the proxies have no role. Figure 11 illustrates the results of this phase for low-load, medium-load, and high-load scenarios. Z is a function of ζ and ς. The coefficients ς (resource consumption) and ζ (admission rate) are higher in $Z1$ and $Z2$, respectively. The optimal values of $\bar{p}_l^{\theta}$, $\bar{m}_l^{\theta}$, and $\bar{D}^{ij}$ are given in this figure for 600 s. In the first 200 s, the low-load scenario is performed, and then, in the second and third 200 s, the medium and high-load scenarios are performed, respectively. As can be seen, the resource consumption ($\bar{p}_l^{\theta}$ and $\bar{m}_l^{\theta}$) increases with increasing the load over time to keep the flow admission rate constant. This increase is higher for $Z2$ so that in the third scenario, no admission rates of more than 78% are achieved by the total resource consumption (Fig. 11a, b).

4.1.3 Third experiment—low-cost route

To investigate the complementary constraints in Eqs. (33) and (34), the cost of links (ℓ_{kl}) for the N10 topology is assumed as follows (Fig. 12).

Figure 13 shows the traffic of the links between the source (node 1) and destination (node 6) provided that they pass at least one proxy (proxy 8). In Scenario 1, traffic passes

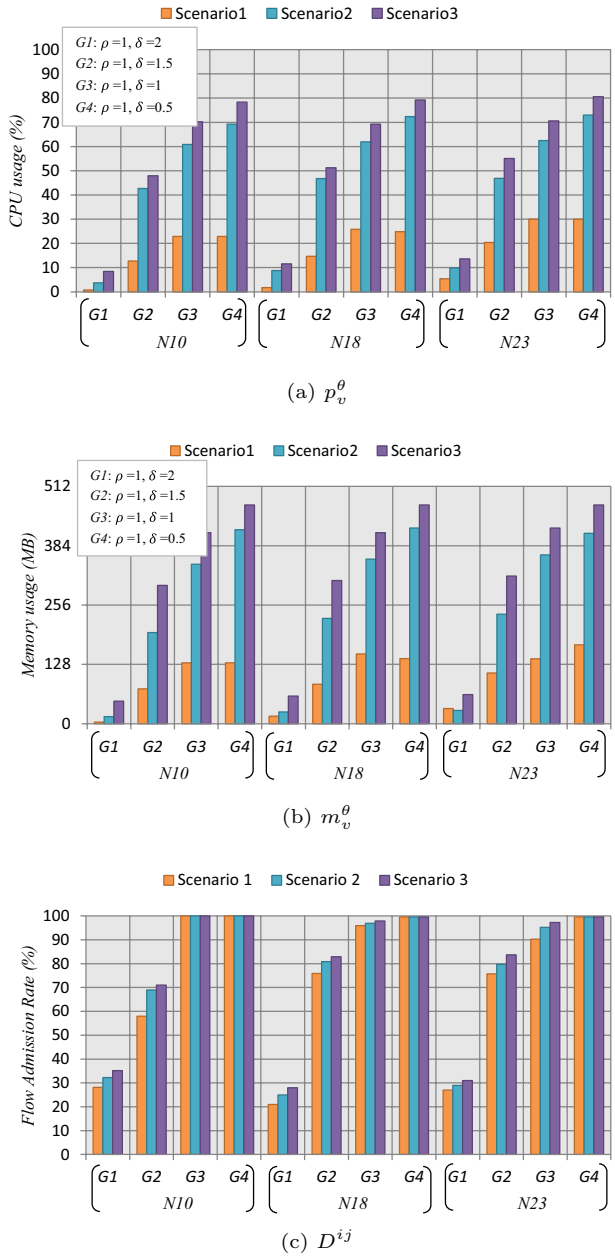


Fig. 10 The consumption of the processor and memory, along with admission rate for signaling phase in switches

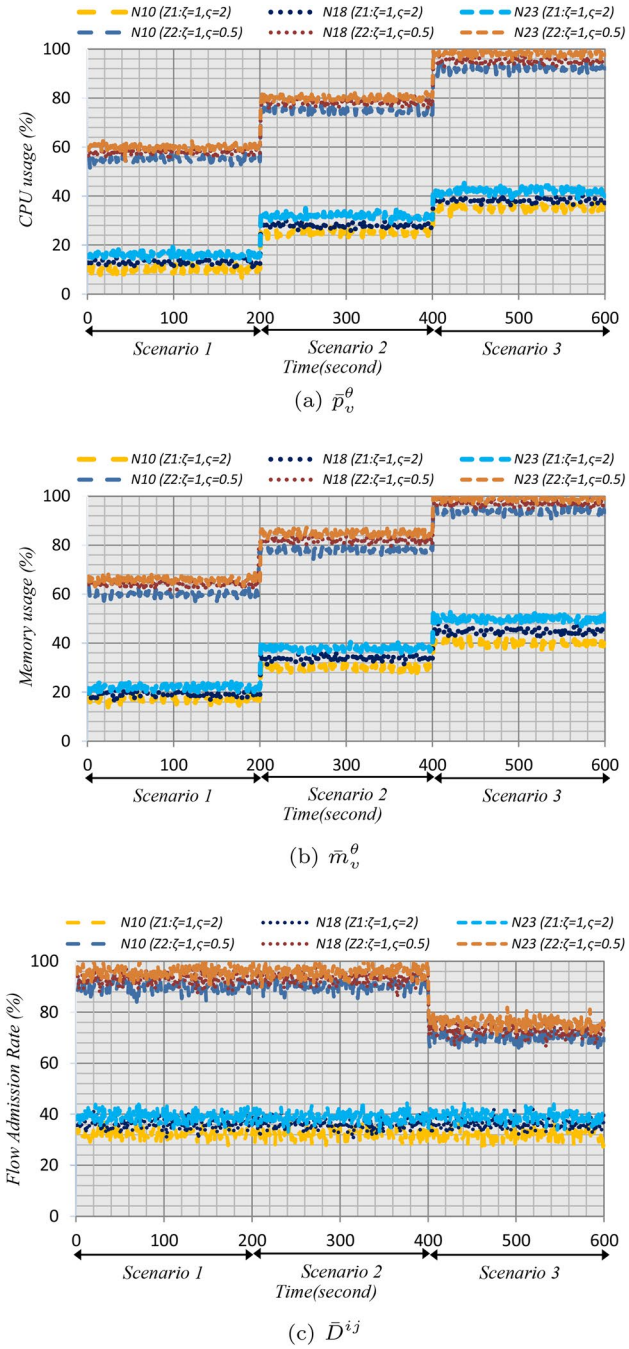


Fig. 11 Processor and memory consumption of switches, along with the admission rate in media phase

through links L1-8, L8-3, L3-2, and L2-6 (Fig. 13). According to Fig. 12, the least expensive route between nodes 1 and 6 is L1-8, L8-3, L3-2, and L2-6 (route cost = 19). Therefore, the controller is capable of transferring the traffic from the least expensive route correctly in the low-load scenario. However, in medium and high-load scenarios, traffic passes through L1-8, L8-3, L3-2, L8-2, and L2-6 (Fig. 13). That is, traffic is divided between the two routes of L8-3, L3-2, and L8-2. Although this is not the least expensive route, it is the optimal answer since the objective function (Eq. 34) considers the resource consumption reduction in addition to reducing the cost of the route (Fig. 14).

The admission rate is 100% for all three scenarios, from node 1 to node 6. The proposed mathematical model maximizes the objective function, so it considers not only a reduction in the cost of the route but an increase in admission rate and a reduction in resource consumption (Eq. 34). Therefore, as can be seen in Fig. 14, the consumption of the

Table 2 Resource consumption of switches in N10 topology (%)

		Scenario 1				Scenario 2				Scenario 3			
		*G*1	*G*2	*G*3	*G*4	*G*1	*G*2	*G*3	*G*4	*G*1	*G*2	*G*3	*G*4
Switch1	CPU	2	12	22	**22**	5	42	60	70	8	47	70	78
Switch2	CPU	3	11	23	**23**	5	41	62	69	**7**	46	71	79
Switch3	CPU	2	11	22	**22**	6	40	62	69	8	46	70	77
Switch4	CPU	3	13	21	**22**	4	43	63	68	7	45	72	76
Switch5	CPU	2	11	22	**22**	5	44	60	70	7	48	70	75
Switch6	CPU	3	10	21	**23**	6	42	61	69	6	46	71	77
Switch7	CPU	2	12	20	**22**	4	41	62	70	8	45	73	74
Switch1	memory	6	64	128	128	15	196	344	418	48	298	412	472
Switch2	memory	7	64	128	126	14	199	343	417	47	298	410	471
Switch3	memory	5	63	127	126	16	198	342	416	46	297	411	470
Switch4	memory	6	62	125	128	14	200	344	418	48	296	413	472
Switch5	memory	7	66	128	125	15	197	342	417	47	295	411	473
Switch6	memory	7	61	125	128	16	196	344	420	46	298	410	472
Switch7	memory	5	62	126	124	15	196	343	418	48	297	411	470

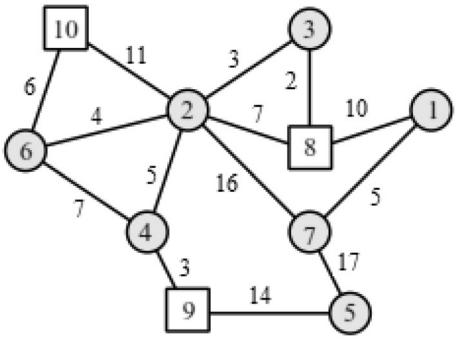


Fig. 12 The cost of links (ℓ_{kl}) in N10 topology

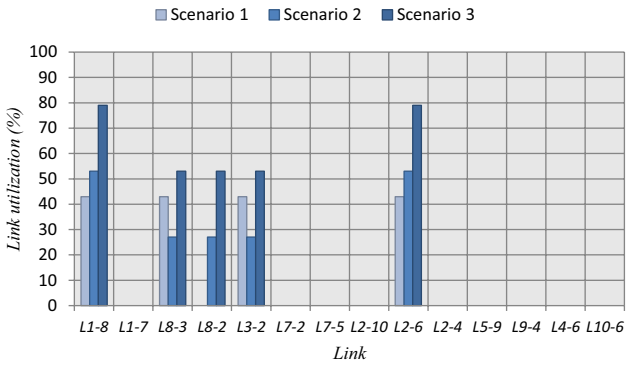


Fig. 13 Link utilization (X_{sl}^{ij})

resources is balanced. This phenomenon is also obvious in Fig. 13 since the load is divided between L8-3, L3-2, and L8-2 in Scenario 3. As mentioned before, this is to maximize the objective function, which is the weighted sum of the route cost, node resources, and admission rate.

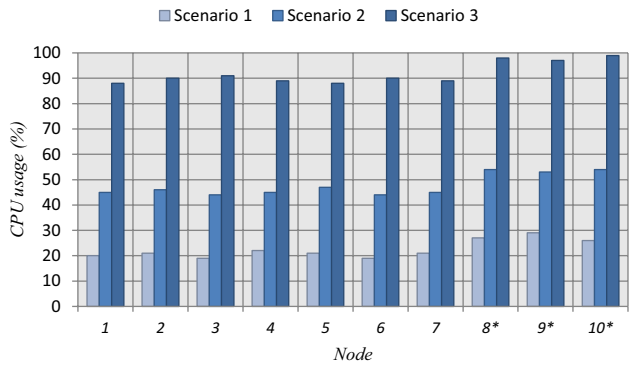
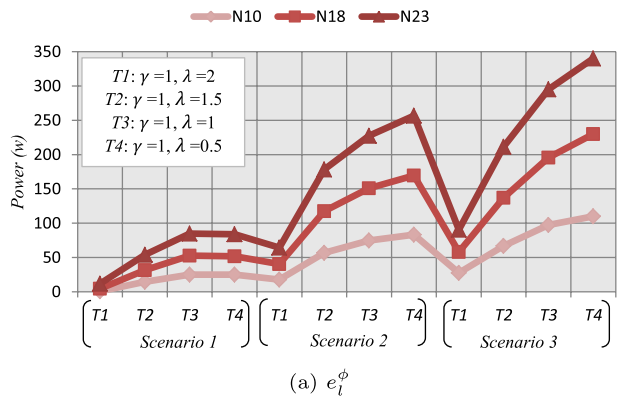


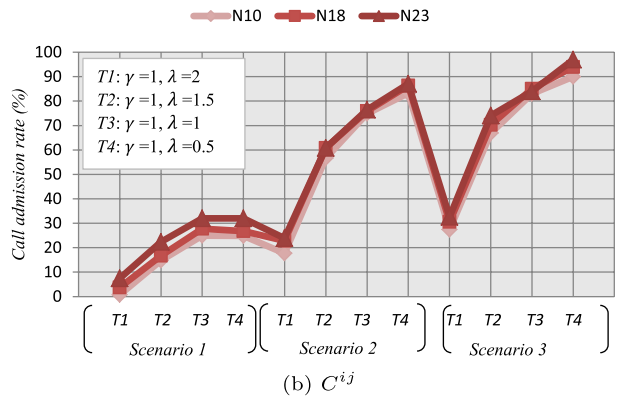
Fig. 14 p_v of the nodes, including proxies and switches (* indicates the proxy node)

4.1.4 Fourth experiment—Energy saving

To investigate the complementary constraints of energy (Eqs. 35 to 39), we evaluate the power consumption. Figure 15 shows the optimal values of e_l^{ϕ} and C^{ij} for different states of T, as well as three scenarios and three topologies. T is a function of γ and λ, which are the weights of "throughput" and "proxy power" in the objective function (Eq. 35), respectively. So, a trade-off between throughput and power is expected. As can be seen in Fig. 15, by moving from a low-load to a high-load scenario, the call admission rate increases in addition to an increase in power consumption. The same pattern is observed by moving from $T1$ to $T4$. In all the three topologies, although call admission rates are almost the same, the power consumptions are different. This is because the number of proxies

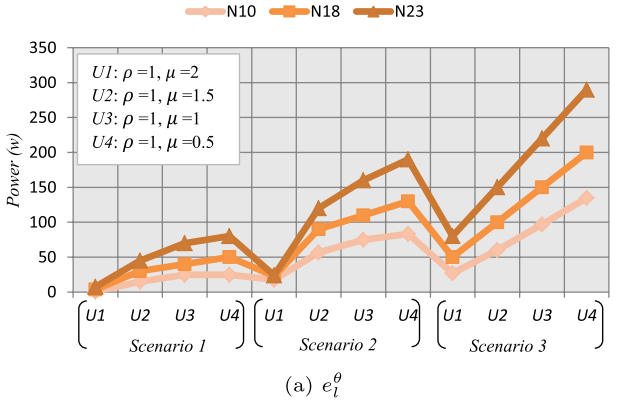


(a) e_l^{ϕ}

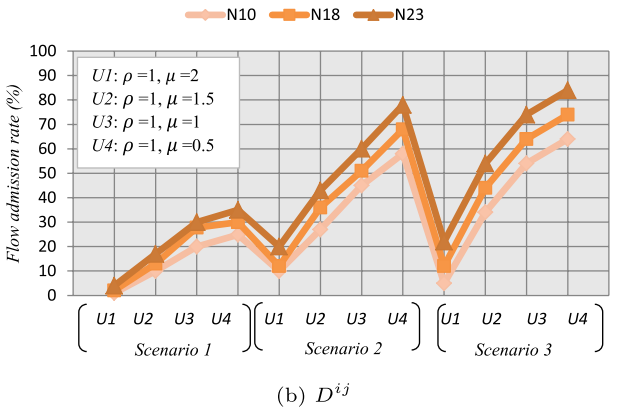


(b) C^{ij}

Fig. 15 Power consumption and admission rate of proxies



(a) e_l^{θ}



(b) D^{ij}

Fig. 16 Power consumption and admission rate of switches

varies in the three topologies, and consequently, the power consumption varies (Fig. 15a).

Figure 16 shows the optimal values of e_l^{θ} and D^{ij} for different states of U. The function U depends on ρ and μ, which are the weights of "throughput" and "switch power" in the objective function (Eq. 35), respectively. Figs. 15 and 16 have similar trends, except that the switches consume almost less power.

4.2 Experimental results

In this section, a prototype of the proposed design is implemented (Fig. 17). In the test platform, we use *Open vSwitch v2.4.1* and *Floodlight v1.2* to implement the OpenFlow switch and controller, and then, modify them according to what was described in the proposed approach section. Floodlight is a Java-based controller. Open vSwitch is a virtual switch that supports the OpenFlow protocol. We use *Kamailio v4.3.6* open-source software to implement SIP proxies equipped with the OpenFlow module. We use *SIPp* to implement user factors. *Oprofile* software has also been used to monitor resource consumption. The *iperf* is used for packet transfer at a fixed rate if the injection of background traffic among proxies is required. Furthermore, network

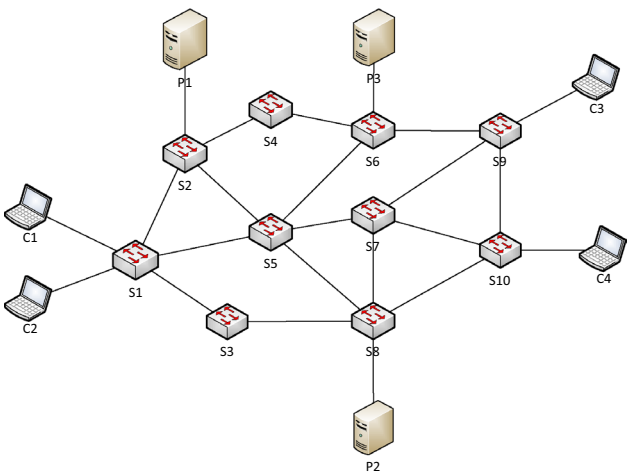


Fig. 17 The topology of the real experiment platform

packets are monitored using *Wireshark* software. We perform each experiment three times and report the average as a result. The algorithms for comparing the performance of the proposed approach are HWAR, FWAR (Montazerolghaem et al. 2017), and TLWL (Jiang et al. 2012). The TLWL algorithm sends a new call request to the SIP proxy

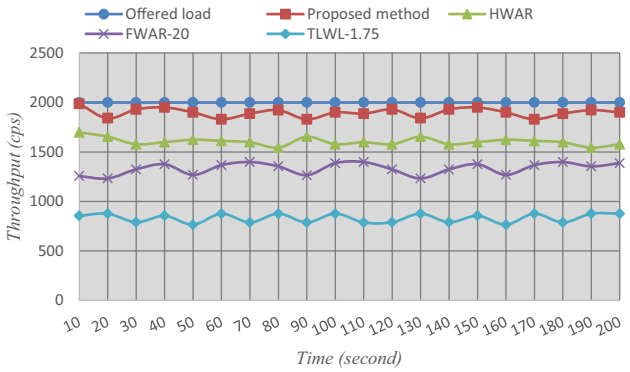


Fig. 18 Throughput over time

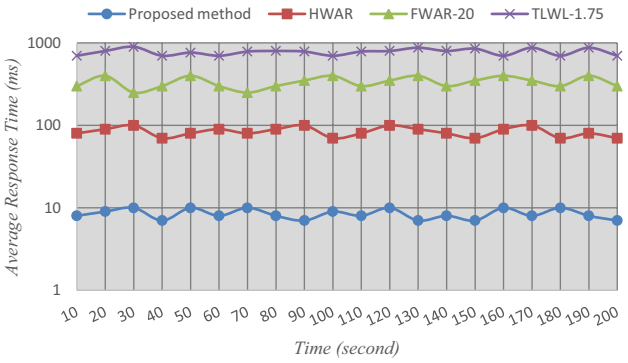


Fig. 19 Latency over time

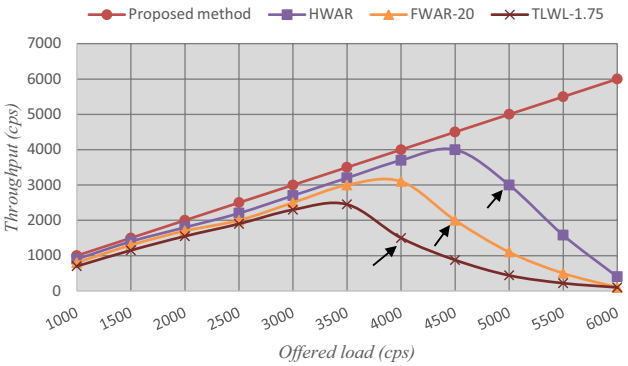


Fig. 20 Throughput at different offered loads

with minimum load using a load balancer. In this algorithm, counters determine the total weighted transactions allocated to each SIP proxy, and the new call is allocated to the proxy that has the lowest counter number. In TLWL-1.75, the transaction weight of `Bye` and `Invite` is 1.0 and 1.75, respectively. As previously mentioned, the FWAR and HWAR algorithms are proposed for load distribution by the load balancer. These algorithms use the window and the response times of the proxies to estimate the load of the proxies. In the FWAR algorithm, the window size is fixed. However, in HWAR, a history of response times is kept in the windows using a mathematical model. All three algorithms considered for comparison are distributed.

The results are presented at the two levels of infrastructure and control separately. At the infrastructure level, we evaluate the performance of proxies and switches in signaling and media phases. At the control level, we evaluate the performance of the controller.

4.2.1 Infrastructure level

The infrastructure includes proxies and switches in the OpenFlow, which their performance is discussed in signaling and media phases.

Signaling phase: Evaluation criteria in the signaling phase include throughput, latency, retransmission rate of `Invite`, and resource consumption of proxies, which are investigated below.

Throughput over time: Throughput is the number of successful calls per unit of time. Figure 18 shows a comparison of the performance of the proposed method with that of the HWAR, FWAR-20, and TLWL-1.75 algorithms over time. In this experiment, the offered load is 2000 calls per second (cps) over 200 seconds. The throughput of the proposed method is very close to the offered load, which indicates that the SDN-based framework is capable of achieving very high throughputs. The throughput of the proposed method in 200 s is more than that of the comparison algorithms. The TLWL-1.75 algorithm has the lowest throughput, which is because of its fully distributed nature as well as its lack of full knowledge of proxy resources.

Latency over time: Fig. 19 shows the average response time over 200 s. The *response time* for each request is the time period between sending the `Invite` message to the selected proxy and the receipt of the corresponding `200 OK` message. As can be seen, the latency of the proposed method is less than 10 ms. This value varies from 100 to 1000 ms in other methods, which is very high compared to the latency of the proposed method and has adverse effects on the quality of service of the call.

Throughput at different offered loads: In this experiment, the offered load starts from 1000 cps and increases linearly up to 6000 cps. As can be seen in Fig. 20, throughput of the proposed method also has a linear increasing trend, similar to the offered load. In contrast, other methods suffer from a drop in throughput at different offered loads. For example, a drop in the HWAR method starts at 5000 cps and continues until it reduces to zero. This drop starts at earlier points for the FWAR-20 and TLWL-1.75 methods (at 4500 cps and 4000 cps, respectively). Drop

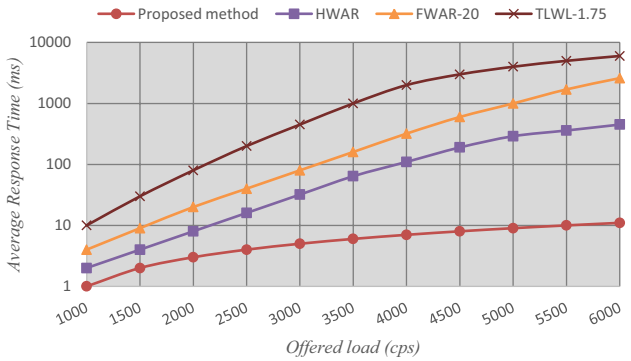


Fig. 21 Latency at different offered loads

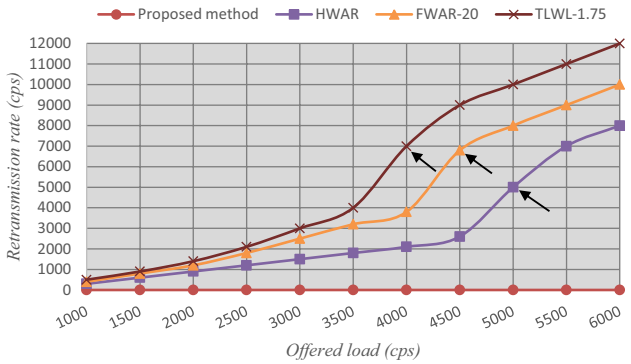


Fig. 22 `Invite` retransmission rate at different offered loads

in the throughput is a sign of overload. The consequences are such that it is not possible to return the normal situation, and the decreasing trend of the throughput continues.

Latency at different offered loads: Latency is another consequence of overload. Figure 21 shows the latency of the methods at different offered loads. As can be seen, the latency in the proposed method, even in high load, is less than 10 ms. In contrast; comparison methods exert an increase in latency by increasing the offered load.

`Invite` retransmission rate at different offered loads: As the overload occurs and proxies do not respond, the number of requests that are re-transmitted increases, which intensifies the overload (Fig. 22). The retransmission rate in the proposed method is almost zero since the throughput increases by increasing the offered load (Fig. 20) and does not suffer from overload. However, other methods, when they face a drop in the throughput, their retransmission rate increases. For example, the HWAR method faces a drop at 5000 cps (Fig. 20), and its retransmission rate increases rapidly at this point.

Proxy resources at different offered loads: The lack of sufficient resources for the process of messages is the reason for the overload. As can be seen in Table 3, none of the processors of the proxies is saturated in the proposed method. However, the processor of the proxies in the HWAR, FWAR-20, and TLWL-1.75 methods is saturated at 5000, 4500, and 4000 cps, respectively. This results in a drop in throughput and an increase in latency and retransmission rate.

Increasing the number of topology nodes: We performed several experiments to check the scalability with respect to the number of network nodes, which are presented below. Given the limited time we had to do so, we did some excruciating work on a very tight schedule to make sure that the experiments are precise and fair. In this experiment, the number of network nodes increases from 10 to 100 (N10 to N100). Figure 23 shows the throughput results. The proposed method is scalable up to N100. But the degree of scalability of other methods is less. For example, the HWAR method is scalable to N71. The degree of scalability of the FWAR-20 and TLWL-1.75 methods is even lower (N64 and

Table 3 CPU consumption of proxies in different methods (%)

		Offered load (cps)										
		1000	1500	2000	2500	3000	3500	4000	4500	5000	5500	6000
Proposed method	P1	12	21	27	34	43	56	62	70	77	84	95
	P2	12	20	26	35	43	55	62	70	77	85	95
	P3	13	21	27	35	43	55	61	69	77	85	94
HWAR	P1	13	24	29	38	37	59	68	74	100	100	100
	P2	20	12	20	27	40	50	61	64	100	100	100
	P3	12	22	18	37	47	44	58	61	100	100	100
FWAR-20	P1	22	29	37	44	59	56	78	100	100	100	100
	P2	12	20	27	32	39	60	80	100	100	100	100
	P3	24	28	39	40	58	67	86	100	100	100	100
TLWL-1.75	P1	20	29	39	49	69	86	100	100	100	100	100
	P2	12	10	29	48	71	92	100	100	100	100	100
	P3	31	33	37	28	55	74	100	100	100	100	100

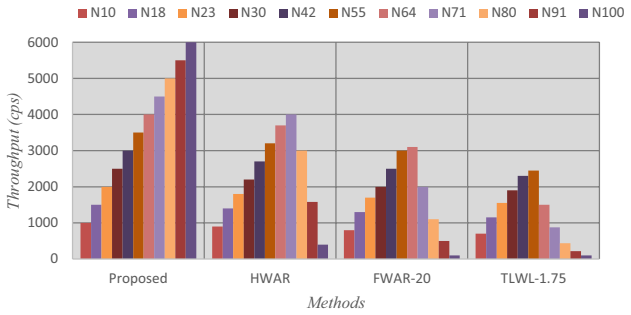


Fig. 23 Throughput at the different numbers of topology nodes

N55, respectively). This is because the proposed method is based on the SDN concept and is more aware and integrated than the entire network resources. In this case, it can manage more nodes. Of course, the proposed method also has a degree of scalability, which is more than the other three methods.

Media phase: In the media phase, we investigate the performance of switches. Table 4 shows the initiated media and latency of switches in different methods in this phase. The media initiated by the proposed method is 100% regardless of the offered load. However, other methods, because of trapping in overload, cannot initiate the total media. For example, the HWAR method is not capable of initiating the 100% of media after a load of 5000 cps, because as we saw in the previous section, it faces an overload at 5000 cps. Besides, the latency in media initiating depends on traffic and increases exponentially when facing overload. The latency in the proposed method is less than 13 ms in the media phase, while the latency in the HWAR method, for example, increase from 34.5 ms to 112.4 ms at 5000 cps.

Finally, the routes of signaling and media traffic for two sessions are shown as examples in Fig. 24. The first session is between C2 and C4 (C^{24}), while the second session is between C1 and C3 (C^{13}). The signaling route is selected so that it passes through at least one proxy as long as it satisfies the constraints of the proposed model. The media route only includes switches. The routing is based on the proposed linear programming model, according to the resources. The first signaling route is C2-S1-S3-S8-P2-S10-C4. In this session, the traffic has passed the proxy 2 (P2). The media traffic route in this session is C2-S1-S5-S7-S10-C4. It seems that even in terms of the number of hops, appropriate routes have been selected.

Table 4 Initiated media and latency in switches in the media phase

		Offered load (cps)										
		1000	1500	2000	2500	3000	3500	4000	4500	5000	5500	6000
Media established (%)	Proposed method	100	100	100	100	100	100	100	100	100	100	100
	HWAR	100	100	100	100	100	100	100	100	84	72	61
	FWAR-20	100	100	100	100	100	100	100	82	69	55	41
	TLWL-1.75	100	100	100	100	100	100	80	61	49	38	22
Delay (ms)	Proposed method	2.3	3.7	4.5	5.8	6.1	7.5	8.7	9.9	10.1	11.3	12.6
	HWAR	3.3	5.4	7.5	10.3	13.5	16.3	23.5	34.5	112.4	245.3	398.4
	FWAR-20	3.4	6.3	7.7	11.2	15.3	17.4	35.8	116.4	278.4	399.9	521.3
	TLWL-1.75	4.1	5.2	8.4	15.3	18.4	31.5	119.5	298.5	463.6	857.9	1467.8

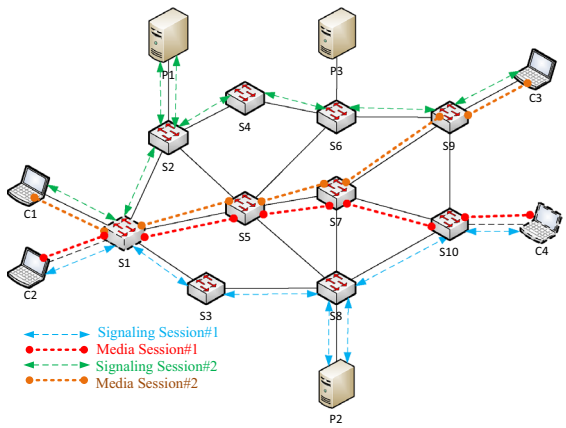


Fig. 24 Routing example in the signaling and media phases (X_{sl}^{ij}, Y_{sl}^{ij})

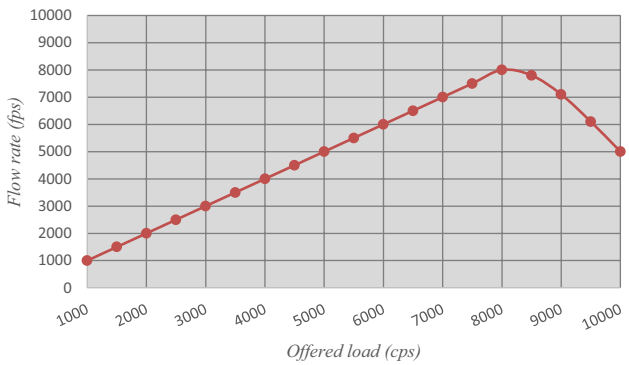


Fig. 25 The scalability of the proposed OpenFlow controller

4.2.2 Control level

The control level includes the proposed OpenFlow controller, which its performance is discussed.

OpenFlow Controller - Scalability: One of the most important evaluation criteria in the control level is controller scalability. Therefore, an experiment was designed to assess the scalability of the controller by increasing the offered load. Figure 25 shows that the proposed controller is scalable up to 8000 cps which means that the throughput of the controller increases by increasing the load up to 8000 cps and then starts to drop. This high throughput indicates the excellent scalability of the proposed controller, and the modules designed in the controller (simple NRC, CRC, and linear optimizer modules) do not impose excessive overhead on the controller.

5 Conclusions and future work

The overload of SIP proxies is one of the critical limitations of MoIP networks, which causes the saturation of resources due to improper routing of call requests in the application layer, and as a result, a severe drop in performance. Besides, in the media phase, the lack of centralized routing reduces the performance of switches and network throughput. This study provides an SDN-based framework for upgrading SIP networks to address these problems. In this regard, in the control layer, we designed an OpenFlow controller with three modules, namely, optimizer, CRC, and NRC, which manages the resources of proxies and switches centrally. OpenFlow proxies have joined the OpenFlow switches in the infrastructure layer. The optimizer module was designed based on the proposed linear programming models, the constraints of which are resource consumption and routing. Complementary constraints are also proposed to consider the cost of the link and power consumption. To evaluate the performance of the proposed framework, we first simulated it and then implemented it in a real experiment platform. Various experiments were performed in three scenarios and topologies. The performance of the proposed method is also compared with that of several conventional algorithms. The results show improvements in throughput, latency, and resource consumption in both signaling and media phases. For future work, we will generalize and develop mathematical models for the optimal allocation of resources for all network equipment in general. We will also equip the proposed framework with network functions virtualization (NFV) technology. In this case, the resources will be available virtually and on-demand. We will also implement the proposed method in the IP multimedia subsystem (IMS) using OpenIMS.

Appendix A

Theorem: Simultaneous control of overload in media and signaling for a set of $n > 2$ SIP proxy servers with limited CPU and memory resources is in the form of a mixed-integer nonlinear program (MINLP).

Proof: Let B_{kl}^{ij} be a binary variable. Assume that $B_{kl}^{ij} = 1$ denotes that a call request from proxy server i to proxy j could be transmitted from proxy k to proxy l and $B_{kl}^{ij} = 0$ denotes that proxy servers k and l do not participate in transferring request from proxy i to proxy j. Given the limited resources in the servers, the optimal values of C^{ij}, D^{ij}, $\bar{D}^{ij}$, X_{kl}^{ij}, and Y_{kl}^{ij} could be obtained through the following MINLP model:

$$max\ \gamma\left(\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}C^{ij}}{\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbb{C}^{ij}}\right)+\rho\left(\frac{\sum_{i=1}^{z}\sum_{j=1}^{z}D^{ij}}{\sum_{i=1}^{z}\sum_{j=1}^{z}\mathbb{D}^{ij}}\right)+\zeta\left(\frac{\sum_{i=1}^{z}\sum_{j=1}^{z}\bar{D}^{ij}}{\sum_{i=1}^{z}\sum_{j=1}^{z}\bar{\mathbb{D}}^{ij}}\right) \tag{A1}$$

Subject to:

$$C^{ij} \le \mathbb{C}^{ij}, \forall i,j \in R \tag{A2}$$

$$D^{ij} \le \mathbb{D}^{ij}, \forall i,j \in S \tag{A3}$$

$$\bar{D}^{ij} \le \bar{\mathbb{D}}^{ij}, \forall i,j \in S \tag{A4}$$

$$\sum_{k=1}^{n} B_{kv}^{ij} X_{kv}^{ij} = \sum_{u=1}^{n} B_{vu}^{ij} X_{vu}^{ij}, \forall i,j,v \in R, i \ne v, j \ne v \tag{A5}$$

$$\sum_{k=1}^{z} B_{kv}^{ij} X_{kv}^{ij} = \sum_{u=1}^{z} B_{vu}^{ij} X_{vu}^{ij}, \forall i,j,v \in S, i \ne v, j \ne v \tag{A6}$$

$$\sum_{s=1}^{z} B_{sl}^{ij} Y_{sl}^{ij} = \sum_{e=1}^{z} B_{le}^{ij} Y_{le}^{ij}, \forall i,j,l \in S, i \ne l, j \ne l \tag{A7}$$

$$\sum_{k=1}^{n} B_{kv}^{iv} X_{kv}^{iv} = C^{iv}, \forall i,v \in R, i \ne v \tag{A8}$$

$$\sum_{u=1}^{n} B_{vu}^{vj} X_{vu}^{vj} = C^{vj}, \forall j,v \in R, j \ne v \tag{A9}$$

$$\sum_{k=1}^{z} B_{kv}^{iv} X_{kv}^{iv} = D^{iv}, \forall i,v \in S, i \ne v \tag{A10}$$

$$\sum_{u=1}^{z} B_{vu}^{vj} X_{vu}^{vj} = D^{vj}, \forall j, v \in S, j \neq v \tag{A11}$$

$$\sum_{s=1}^{z} B_{sl}^{il} Y_{sl}^{il} = \bar{D}^{il}, \forall i, l \in S, i \neq l \tag{A12}$$

$$\sum_{e=1}^{z} B_{le}^{lj} Y_{le}^{lj} = \bar{D}^{lj}, \forall j, l \in S, j \neq l \tag{A13}$$

$$X_{kv}^{ii} = Y_{sl}^{jj} = 0, \forall i, k, v \in R, \forall j, s, l \in S \tag{A14}$$

$$X_{ki}^{iv} = Y_{lj}^{js} = 0, \forall i, k, v \in R, \forall j, s, l \in S \tag{A15}$$

$$\alpha_1 C^{vv} + \alpha_2 (\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} B_{vk}^{ij} X_{vk}^{ij} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} B_{kv}^{ij} X_{kv}^{ij}) \leq P_v^{\phi}, \forall v \in R \tag{A16}$$

$$\beta_1 C^{vv} + \beta_2 (\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} B_{vk}^{ij} X_{vk}^{ij} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} B_{kv}^{ij} X_{kv}^{ij}) \leq M_v^{\phi}, \forall v \in R \tag{A17}$$

$$\omega(\sum_{i=1}^{z}\sum_{j=1}^{z}\sum_{k=1}^{z} B_{vk}^{ij} X_{vk}^{ij} + \sum_{i=1}^{z}\sum_{j=1}^{z}\sum_{k=1}^{z} B_{kv}^{ij} X_{kv}^{ij}) \leq P_v^{\theta}, \forall v \in S \tag{A18}$$

$$\psi(\sum_{i=1}^{z}\sum_{j=1}^{z}\sum_{k=1}^{z} B_{vk}^{ij} X_{vk}^{ij} + \sum_{i=1}^{z}\sum_{j=1}^{z}\sum_{k=1}^{z} B_{kv}^{ij} X_{kv}^{ij}) \leq M_v^{\theta}, \forall v \in S \tag{A19}$$

$$\xi(\sum_{i=1}^{z}\sum_{j=1}^{z}\sum_{s=1}^{z} B_{ls}^{ij} Y_{ls}^{ij} + \sum_{i=1}^{z}\sum_{j=1}^{z}\sum_{s=1}^{z} B_{sl}^{ij} Y_{sl}^{ij}) \leq \bar{P}_l^{\theta}, \forall l \in S \tag{A20}$$

$$\eta(\sum_{i=1}^{z}\sum_{j=1}^{z}\sum_{s=1}^{z} B_{ls}^{ij} Y_{ls}^{ij} + \sum_{i=1}^{z}\sum_{j=1}^{z}\sum_{s=1}^{z} B_{sl}^{ij} Y_{sl}^{ij}) \leq \bar{M}_l^{\theta}, \forall l \in S \tag{A21}$$

$$B_{kl}^{ij} - L_{kl} \leq 0, \forall i, j, k, l \tag{A22}$$

$$\text{Variables:} \quad B_{kl}^{ij} \in \{0,1\}, C^{ij}, D^{ij}, \bar{D}^{ij}, X_{kl}^{ij}, Y_{kl}^{ij}, P_v^{\phi}, P_v^{\theta}, \bar{P}_l^{\theta}, M_v^{\phi}, M_v^{\theta}, \bar{M}_l^{\theta} \geq 0, \forall i, j, k, l$$

Although the objective function is linear, the nonlinear constraints and the binary variables B_{kl}^{ij} render the model an MINLP which is generally NP-hard and unsolvable in polynomial time. To overcome this drawback, we propose certain modifications, which remove the nonlinearity and allow the problem to take the form of an LP (Section 3.2).

References

Samaresh Bera, Sudip Misra, Abbas Jamalipour (2019) Flowstat: Adaptive flow-rule placement for per-flow statistics in sdn. IEEE J Selected Areas Commun 37(3):530–539

Bonfim Michel S, Dias Kelvin L, Fernandes Stenio FL (2019) Integrated nfv/sdn architectures: a systematic literature review. ACM Computing Surveys (CSUR) 51(6):1–39

Cheng Yun-Jung, Wang Kuochen, Jan Rong-Hong, Chen Chien, Huang Chia-Yuan (2008) Efficient failover and load balancing for dependable sip proxy servers. In *2008 IEEE Symposium on Computers and Communications*, pages 1153–1158. IEEE

Chi Caixia, Wang Dong, Hao Ruibing, Zhou Wei (2008) Performance evaluation of sip servers. In *2008 Third International Conference on Communications and Networking in China*, pages 674–679. IEEE

Mauricio Cortes, Robert Ensor J, Esteban Jairo O (2004) On sip performance. Bell Labs Tech J 9(3):155–172

Gokhale Swapna S, Lu Jijun (2005) Signaling performance of sip based voip: A measurement-based approach. In *GLOBECOM'05. IEEE Global Telecommunications Conference, 2005.*, volume 2, pages 5–pp. IEEE

Gurbani V, Hilt V, Schulzrinne H (2014) Session initiation protocol (sip) overload control. *draft-ietfsoc-overload-control-03*

Gurbani Vijay K, Rajnish Jain (2004) Transport protocol considerations for session initiation protocol networks. Bell Labs Tech J 9(1):83–97

TianZhang He, Toosi Adel N, Rajkumar Buyya (2019) Performance evaluation of live virtual machine migration in sdn-enabled cloud data centers. J Parallel Distributed Comput 131:55–68

Hilt Volker, Widjaja Indra (2008) Controlling overload in networks of sip servers. In *2008 IEEE International Conference on Network Protocols*, pages 83–93. IEEE

Hilt Volker, Noel E, Shen C, Abdelal A (2011) Design considerations for session initiation protocol (sip) overload control. *RFC6357*

Homayouni Maryam, Azhari Sayed Vahid, Jahanbakhsh Mojataba, Akbari Ahmad, Mansoori Alireza, Amani Nahid (2009) Configuration of a sip signaling network: An experimental analysis. In *2009 Fifth International Joint Conference on INC, IMS and IDC*, pages 76–81. IEEE

Hongbo Jiang, Arun Iyengar, Erich Nahum, Wolfgang Segmuller, Tantawi Asser N, Wright Charles P (2012) Design, implementation, and performance of a load balancer for sip server clusters. IEEE/ACM Trans Netw 20(4):1190–1202

Kambourakis Georgios, Geneiatakis Dimitris, Dagiuklas Tasos, Lambrinoudakis Costas, Gritzalis Stefanos (2006) Towards effective sip load balancing: the snocer approach. In *3rd Annual VoIP Security Workshop*

Diego Kreutz, Ramos Fernando MV, Esteves Verissimo Paulo, Esteve Rothenberg Christian, Siamak Azodolmolky, Steve Uhlig (2014) Software-defined networking: a comprehensive survey. Proc IEEE 103(1):14–76

Adrian Lara, Anisha Kolasani, Byrav Ramamurthy (2013) Network innovation using openflow: a survey. IEEE Commun Surveys Tutorials 16(1):493–512

Heikki Lindholm, Taneli Vähäkangas, Kimmo Raatikainen (2007) A control plane benchmark for telecommunications signalling applications. Sort 20:100

Patrick McGregor, Richard Kaczmarek, Vernon Mosley, Dennis Dease, Peter Adams (2006) National security/emergency preparedness and the next-generation network. IEEE Commun Mag 44(5):133–143

Montagna Sergio, Pignolo Maurizio (2008) Performance evaluation of load control techniques in sip signaling servers. In *Third International Conference on Systems (icons 2008)*, pages 51–56. IEEE

Ahmadreza Montazerolghaem, Shekofteh S-Kazem, Yaghmaee MH, Naghibzadeh Mahmoud (2017) A load scheduler for sip proxy servers: design, implementation and evaluation of a history weighted window approach. International Journal of Communication Systems 30(3):e2980

Nahum Erich M, John Tracey, Wright Charles P (2007) Evaluating sip server performance. ACM SIGMETRICS Performance Eval Rev 35(1):349–350

Noel Eric C, Johnson Carolyn R (2007) Initial simulation results that analyze sip based voip networks under overload. In *International Teletraffic Congress*, pages 54–64. Springer

Nunes Bruno Astuto A, Marc Mendonca, Xuan-Nam Nguyen, Katia Obraczka, Thierry Turletti (2014) A survey of software-defined networking: past, present, and future of programmable networks. IEEE Commun Surveys Tutorials 16(3):1617–1634

Nygren Anders, et al. (2015) Openflow switch specification, version 1.5.1. https://opennetworking.org. Accessed: 2022-01-01

Ohta Masataka (2004) Simulation study of sip signaling in an overload condition. In *Communications, Internet, and Information Technology*, pages 321–326

Ohta Masataka (2008) Performance comparisons of transport protocols for session initiation protocol signaling. In *2008 4th International Telecommunication Networking Workshop on QoS in Multiservice IP Networks*, pages 148–153. IEEE

Pascual Victor (2009) Sip server overload problem statement

Dirk Pesch, Isabel Pous Maria, Gerry Foster (2005) Performance evaluation of sip-based multimedia services in umts. Comput Netw 49(3):385–403

Kaige Qu, Weihua Zhuang, Qiang Ye, Xuemin Shen, Li Xu, Jaya Rao (2020) Dynamic flow migration for embedded services in sdn/nfv-enabled 5g core networks. IEEE Trans Commun 68(4):2394–2408

Ram Kaushik Kumar, Fedeli Ian C, Cox Alan L, Rixner Scott (2008) Explaining the impact of network transport protocols on sip proxy performance. In *ISPASS 2008-IEEE International Symposium on Performance Analysis of Systems and software*, pages 75–84. IEEE

Rosenberg J, et al. (2008) Requirements for management of overload in the session initiation protocol. *Work in Progress*

Rosenberg Jonathan, Schulzrinne Henning, Camarillo Gonzalo, Johnston Alan, Peterson Jon, Sparks Robert, Handley Mark, Schooler Eve, et al. (2002) Sip: session initiation protocol

Schulzrinne Henning G, Narayanan Sankaran, Lennox Jonathan, Doyle Michael (2002) Sipstone: Benchmarking sip server performance

Charles Shen, Henning Schulzrinne (2010) On tcp-based sip server overload control. In Principles, Systems and Applications of IP Telecommunications, pp 71–83

Kundan Singh, Henning Schulzrinne (2007) Failover, load sharing and server architecture in sip telephony. Comput Commun 30(5):927–942

Wanke Stephan, Scharf Michael, Kiesel Sebastian, Wahl Stefan (2007) Measurement of the sip parsing performance in the sip express router. In *Meeting of the European Network of Universities and Companies in Information and Communication Engineering*, pages 103–110. Springer

Wu Wei-Ming, Wang Kuochen, Jan Rong-Hong, Huang Chia-Yuan (2007) A fast failure detection and failover scheme for sip high availability networks. In *13th Pacific Rim International Symposium on Dependable Computing (PRDC 2007)*, pages 187–190. IEEE

Wenfeng Xia, Yonggang Wen, Heng Foh Chuan, Dusit Niyato, Haiyong Xie (2014) A survey on software-defined networking. IEEE Commun Surveys Tutorials 17(1):27–51