Green Cloud Multimedia Networking: NFV/SDN based Energy-efficient Resource Allocation

Ahmadreza Montazerolghaem, Mohammad Hossein Yaghmaee, Alberto Leon-Garcia

Green Cloud Multimedia Networking: NFV/SDN based Energy-efficient Resource Allocation

Ahmadreza Montazerolghaem, *Student Member, IEEE,* Mohammad Hossein Yaghmaee, *Senior Member, IEEE,* and Alberto Leon-Garcia, *Fellow, IEEE*

Abstract

The rapid growth of communications and multimedia network services such as Voice over Internet Protocol (VoIP) have caused these networks to face a crisis in resources from two perspectives: 1. Lack of resources and, as a result, overload; 2. Redundancy of resources and, as a result, energy loss. Cloud computing allows the scale of resources to be reduced or increased on demand. Many of the gains obtained from the cloud computing come from resource sharing and virtualization technology. On the other hand, the emerging concept of Software-Defined Networking (SDN) can provide a global view of the entire network for integrated resource management. Network Function Virtualization (NFV) can also be used to virtually implement a variety of network devices and functions. In this paper, we present an energy-efficient framework called GreenVoIP to manage the resources of virtualized cloud VoIP centers. By managing the number of VoIP servers and network equipment, such as switches, this framework not only prevents overload but also supports green computing by saving energy. Finally, GreenVoIP is implemented and evaluated on real platforms, including Floodlight, Open vSwitch, and Kamailio. The results suggest that the proposed framework can minimize the number of active devices, prevent overloading, and provide service quality requirements.

Index Terms

Energy-efficient VoIP Network, Green Networking, Cloud Multimedia Computing, Resource Allocation, SDN and NFV Orchestration, Large-scale VoIP Service Provider.

1 Introduction

Energy consumption in multimedia communication over IP is going to explode. Nowadays there is a strong tendency towards an "IP in everything and everything over IP" world. This lure of a common system for multimedia communications and its expected costs savings is showing to be a quite strong motivator

- *A. Montazerolghaem is with the Department of Computer Engineering, Quchan University of Technology, Quchan, Khorasan Razavi, Iran. MH. Yaghmaee is visiting professor at University of Toronto, Toronto, Canada. Alberto Leon-Garcia is with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, Canada.*
E-mail: Ar.montazer@qiet.ac.ir, mh.yaghmaeemoghaddam@utoronto.ca, alberto.leongarcia@utoronto.ca

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TGCN.2020.2982821, IEEE Transactions on Green Communications and Networking

IEEE Transactions on Green Communications and Networking, DOI 10.1109/TGCN.2020.2982821

for the proliferation of VoIP technology. VoIP is a communication system technology that enables multimedia communication via the IP [1]. VoIP comes in a wide variety of flavors such as desktop-based applications (e.g., Skype and Google Hangouts), systems replacing the Public Switched Telephony Network (PSTN) as a primary line voice service (e.g., Vonage), and, more recently, VoIP over smart or mobile phones (e.g., Viber, WhatsApp, and Line) [2].

In today's competitive world, VoIP Quality of Service (QoS) can pave the way to success for any business. VoIP is becoming increasingly common among providers and also, is mainly used in diverse industries. As stated by the business consulting company Frost & Sullivan, mobile VoIP alone (excluding fixed VoIP) constituted a $30 billion business worldwide by 2015 (up from just $600 million in 2008), and this trend continues. Because of this, Frost & Sullivan is incentivizing all mobile providers to start applying VoIP [3]. In November 2017, the number of concurrent VoIP users on Skype exceeded 300 million. Research has shown that there are approximately 200 million VoIP hard phone subscribers in the world and the number of mobile VoIP users exceed 150 million [4]. Furthermore, the emergence of 4G and 5G high-speed networks will contribute to the increasing growth and popularity of VoIP (VoLTE).

This popularity of using VoIP, on the one hand, and load changes in the network over time (for example, in a 24-hour span), on the other hand, make service providers face two major challenges:

- Under-provisioning: the shortage of resources and overload at peak overload times
- Over-provisioning: redundancy of resources and energy waste at underload times

By adopting virtualization technology, the Infrastructure as a Service (IaaS) model in cloud computing allows for the provision of the infrastructure as a service and on-demand, including computational resources (such as servers) and network resources (such as switches) (Fig. 1). Also, IaaS platforms provide dynamic resource scaling, which can be used by VoIP service providers [5]. There has been much discussion about the benefits and costs of the IaaS cloud model. The present paper discusses this issue along with the question of how the concept of an IaaS platform can help a VoIP service provider access the resources it needs in such a way that there is a trade-off between energy and performance. In other words, energy consumption is reduced and, at the same time, through the prevention of overload, the requirements of VoIP QoS are met. For this purpose, NFV and SDN [6]–[10] can be used.

Having gained much popularity, the two concepts of NFV and SDN can provide an abstract and centralized view of VoIP network resources. This can be very beneficial in the management of resources and with dealing with challenges, such as overload management and energy-efficient consumption [11]. More precisely, SDN allows for advanced management by separating data from control planes as well as centralized and software controls [12]. In addition, by virtualizing a variety of VoIP network devices and functions, NFV technology can help improve resource management. This can be achieved by optimally allocating the virtual functions

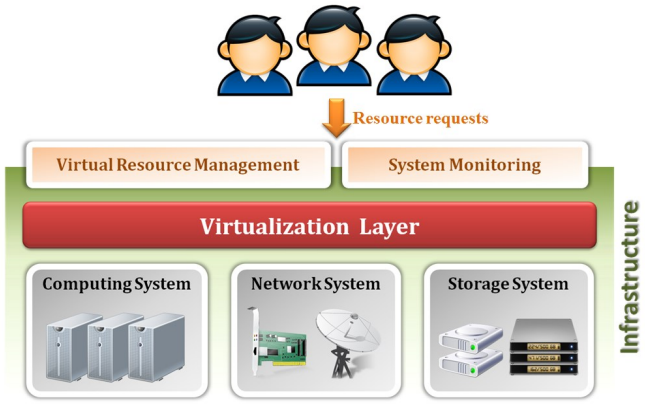


Figure 1. IaaS cloud platform.

of the network according to needs. For example, at peak load times, when the possibility of overload is high, inactive resources of the VoIP network, including servers, can be enabled and the virtual functions needed by the network can be assigned to them. Also, at times of a reduced load, inactive network equipment can be deactivated (removed from the circuit) to save energy [13].

1.1 Motivations

1) Increasing development and popularity of VoIP:

Having predicted the VoIP trend, a study by OVUM Telecom Research (a global business intelligence firm) states that VoIP volumes will rise [14]. Smartphones, mobile communication, and 4G LTE are the driving forces of this growth [15]. Also, the Internet of Things (IoT) plays a role in VoIP's increasing development. It is predicted that, by 2020, almost 24 billion things will be connected to the Internet [16]. This can imply connecting a cell phone to a coffee maker or other home appliances. Soon, VoIP will become essential to the Smart Home by using VoIP phones to control automation. For instance, with the VoIP softphone client on a smartphone or tablet, one will be able to adjust light, washing machine, home monitoring system and so on [17].

2) Increased energy prices and lower VoIP call prices:

The growth of communication industries has created a growing demand for VoIP. This, in turn, has led to the creation of large-scale VoIP centers that consume huge and significant amounts of electrical power. With such a large existing user and its growth, as well as constantly increasing energy costs, the energy efficiency of the systems must be determined. Despite the improvements and progress made in hardware energy efficiency, overall energy consumption is growing due to increased demand. Energy prices are also rising accordingly. For instance, in 2006, the cost of energy consumed by the IT infrastructure in the United States was $4.5 billion. This figure doubled by 2011. However, VoIP call prices are falling [2], [15]. Therefore, VoIP providers need to consider reducing the power consumption of their network through the virtualization of their functions.

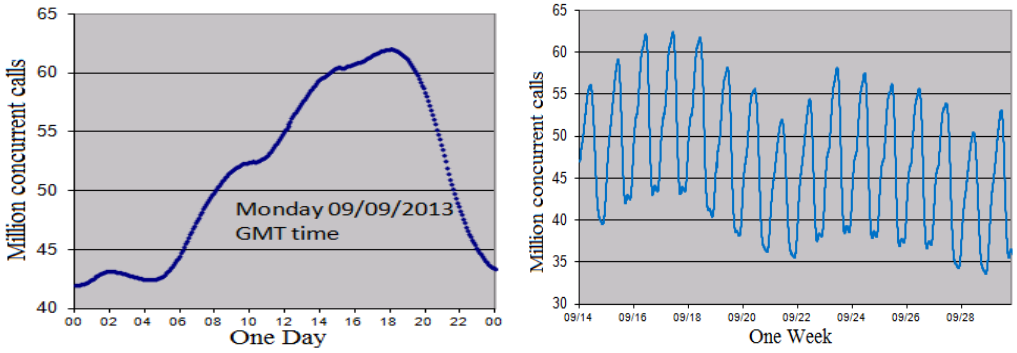
3) Load changes in the VoIP network:

Being the most popular VoIP application, Skype is attracting the attention of the research community [18]. Fig. 2a shows the number of Skype concurrent calls on Sept 9, 2013 [19]. This figure demonstrates that the load on this VoIP network varies considerably during a single day. Therefore, in peak hours, the servers of this network are overloaded, while there is a waste of power during underload hours, as the servers are used less. Fig. 2b reports Skype concurrent calls within a week. This figure also confirms that the network load varies considerably over time. Research shows that, due to over-provisioning for peak demand, VoIP servers are often used to a considerably lower degree than their capacity. To address this, the proposed cloud-based model aims to resolve this issue by the automatic supply of resources in response to load changes. In addition, this model also leads to a reduction in hardware costs.

In sum up, the VoIP network design has traditionally focused on the network Capital Expenditure (CapEx) minimization, which makes up for the equipment and installation costs of the network infrastructure. However, as Internet traffic and energy costs are exponentially rising, energy consumption is now a major concern for VoIP network providers for a number of reasons [4], [20]:

- The Operational Expenditure (OpEx), due to energy cost is significant
- ICT is a significant contributor to global energy consumption, which is necessary to down CO2 emissions
- Energy-efficient resource allocation relieves the heat dissipation problem.

So, energy minimization has become one of the main aims in designing VoIP communications networks.



(a) The number of Skype concurrent calls in a day. (b) The number of Skype concurrent calls within a week.

Figure 2. VoIP network load changes [19].

1.2 Organization

The current paper first presents a background of the VoIP network, its resources, and issues regarding its resource provisioning. Section 3 investigates related works of the research. Problem formulation is presented in Section 4. The proposed GreenVoIP framework and its modules and algorithms are presented in two phases in

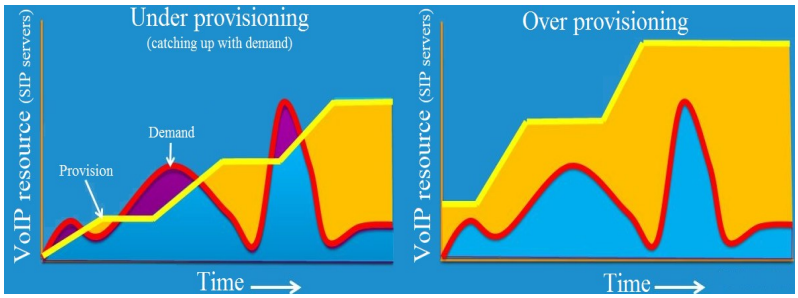


Figure 3. VoIP resources are not sufficient to meet the actual demand (bad user experience) or are not fully utilized (wasted energy).

Section 5. Section 6 describes the details of the testbed for the implementation of GreenVoIP and its performance evaluation results. Finally, in Section 7, the conclusions and future work are provided.

2 Background

VoIP is a technology of voice and multimedia transfer through the IP network. Its applications for transmitting sound, image, and texts on the Internet have become very popular. The core of VoIP network signaling is the SIP request/response protocol, standardized by institutions such as ITU (International Telecommunication Union), 3GPP (3rd Generation Partnership Project), and ETSI (European Telecommunications Standards Institute) [21].

The VoIP network also consists of two types of entities: User Agents (UA) and SIP servers. User agents are, in turn, divided into two categories: User Agent Client (UAC) and User Agent Server (UAS) which request and respond to the issuance of a message, respectively. Request messages contain information about the sender and receiver of the message generated and are sent by the UAC. Response messages are used to confirm that a request has been received and processed and contains the processing status [22].

SIP servers are also responsible for managing VoIP sessions between UAC and UAS. Servers have a limited capacity (e.g., 200 calls per second (cps)) and are interconnected through a network of switches. Due to the rapid growth of this network, one of the major challenges is the provision of needed on demand servers and load changes. What conventionally happens in this network is that resources, i.e., the number and capacity of SIP servers, are determined in advance. This causes over-provisioning or under-provisioning of these resources (Fig. 3).

When there is under-provisioning of resources, each SIP server may face overload. Overload, in turn, results in a severe drop in service quality (reduced throughput, delayed calls, and loss of calls). On the contrary, the over-provisioning of resources is one of the main reasons for low power efficiency in VoIP centers. Since these resources are intended for peak demand, most of the time their full capacity is not utilized. For example, it has been reported that the average utilization of SIP servers for large-scale VoIP centers is between approximately 10% and 30%, indicating that a significant amount of the capacity of these VoIP centers is idle [4], [15].

It should be taken into account that, in addition to having computational resources such as SIP servers, the VoIP network is located on a pool of network resources, such as switches. Therefore, it should also

be considered that the under/over provisioning of SIP servers also causes problems in the provisioning of switches.

The efficient use of the IaaS cloud concept with NFV and SDN technologies can help resolve the VoIP network resource management issues.

Fig. 4 shows the NFV architecture which takes advantage of virtualization technologies to turn network functions and services, such as VoIP, into Virtualized Network Functions (VNFs) [23]. These technologies will be implemented in software and executed as Virtual Machines (VMs) on commodity hardware and high-performance Physical Machines (PMs), namely Network Function Virtualization Infrastructures (NFVIs). In addition, NFV will possibly leverage cloud computing technologies. NFV Management and Organization (MANO) is responsible for the overall management of this structure [6]. A VNF may be composed of one or more VMs that perform a special function on PMs for providing a network function, such as SIP server, firewall or NAT functionality. To have network functions, there is no longer the need to buy and install special and separate hardware. Thus, the NFV independently separates the network functions from the special-purpose hardware and allows these functions to run controlled as software [24]. VNF instances can be created and used dynamically and on demand, or be dynamically migrated. Multiple copies of VNF instances can also be created or removed based on network conditions [25]. It should be noted that VM live migration technology provides the ability to change the mapping between VNFs and PMs while the services are running and without causing an interruption [26].

Fig. 5 illustrates the software-defined NFV architecture. In this architecture, there is a centralized management not only of VNFs, but also of switches. As a result, centralized and simultaneous management of servers and switches is possible. Through the OpenFlow protocol, the SDN controller can provide a global view of switch statuses. These switches lack a control function for automated decision making and are controlled by the SDN controller. The NFV platform is also managed by NFV MANO. The network's smart

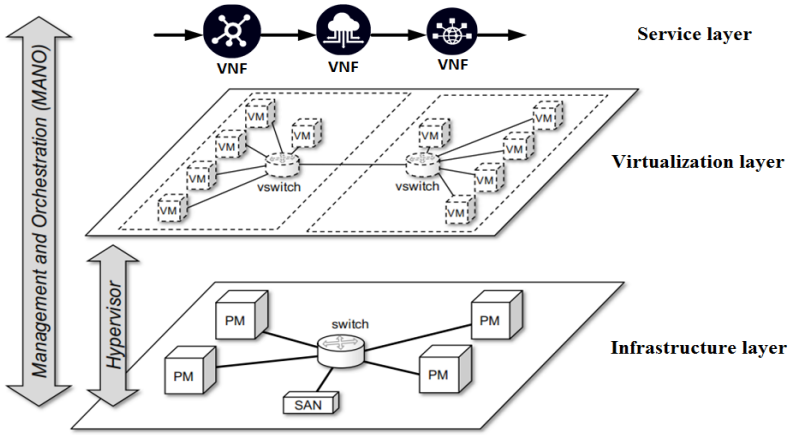


Figure 4. NFV layered architecture.

section is located in the control module, which sends control commands to the data layer in accordance with the policies and overall structure of the network. In order to achieve the desired QoS and save power as well, it is possible to enter or remove the equipment from the circuit according to the needs and changes in the input load; this is due to the integrated control and monitoring systems of this architecture [7], [8].

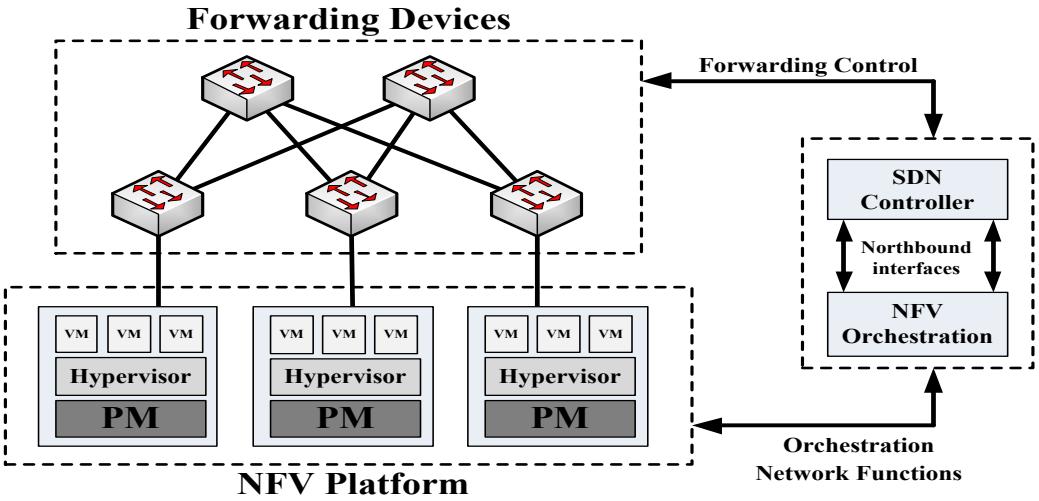


Figure 5. Software-defined NFV architecture.

3 Related Work

We review the related work in the following two sections:

3.1 VoIP Overload Control

The most prominent existing approaches to overcoming VoIP overload can be categorized as shown in Fig. 6. In this figure, one way to prevent overload is the distribution of the load between SIP servers based on their capacity [27]–[29]. Another categorized mechanism is overload control methods, which are divided into five general categories [30]–[35], the most important of which are local and distributed methods.

In local methods, each SIP server monitors its consumable resources independently and rejects additional calls without the need to interact with other servers [30].

In distributed methods, SIP servers prevent overload by interacting with each other and exchanging information. These methods are also divided into five categories, depending on the type of exchanged information [36]–[42]. For instance, [40] is a window-based method in which the load is sent to the downstream server only if the upstream server window has empty space. Also, message delays are used to adjust the size of the window.

Apart from the categorization in Fig. 6, the authors have proved in [43], [44] that the answer to VoIP overload control problem for n number of servers with limited memory and CPU is NP-complete. Thus,

a heuristic method is presented based on the linear programming model for call admission control. Also, the authors' article [45] introduced an SDN-based OpenSIP framework to create a load balance between SIP servers while making the least number of changes to the network through OpenFlow switches.

3.2 VoIP Energy Management

Most studies in this field are concerned with mechanisms for reducing energy consumption in wireless VoIP communications. These studies are mainly evaluated from the user's equipment side, such as SIP mobile devices, rather than from the provider's side, such as servers.

[2] performs a number of experiments to specify the power consumption of different VoIP parts, and shows that the presence of physical servers is the main obstacle to building energy-efficient VoIP systems. Also, in [46], the authors collect information about existing VoIP systems, evaluating their power consumption and relative energy efficiency through analysis a series of tests. They give a number of suggestions to make VoIP systems more energy-efficient. For example, some methods detected in mobile phones, such as Wake-on-LAN, may lead to more energy-efficient VoIP networks. Such devices can get into an energy saving mode in inactivity periods. They are remotely woken up by the arrival of an incoming call.

[47] considers that Wi-Fi interfaces have the high energy consumption (particularly when a device is idle). This is a considerable barrier to the widespread of VoIP over Wi-Fi. So, a practical energy management structure that leverages the cellular radio on a smartphone to apply wakeup is presented (called Cell2Notify).

Chen et al. [48] give a review of VoIP power saving methods defined in the IEEE 802.11 Wireless LAN (WLAN). In addition, [49] argues the issue of how to reduce energy consumption of the WLAN interface during a VoIP call all the while conserving the QoS. The authors in [50] propose an energy saving method for VoIP over wireless ad-hoc networks. The proposed solution in [51] uses frame aggregation and an asynchronous Power Save Mode (PSM) and, hence, improves VoIP QoS in WLANs. [52] attempts to determine whether speaker

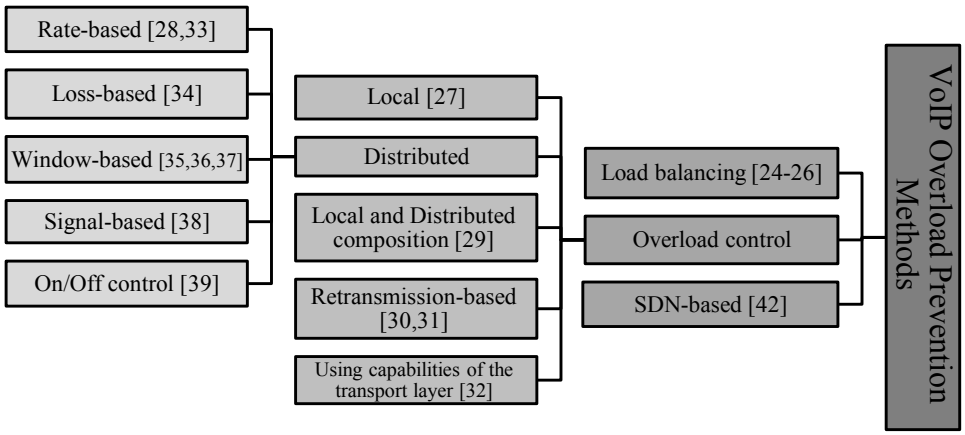


Figure 6. Categorization of papers on VoIP overload.

identity, coupled with the pause/gap distinction, can be used to improve the prediction of silent durations during VoIP calls. Such improvements can lead to energy savings and/or better quality of a call.

SiFi [53], a VoIP call energy savings method, focuses on conversational silences. In this work, energy is saved by setting the WiFi radio to PSM during periods of silence. In [54], the authors seek to reduce energy consumption in the VoIP network by designing an effective authentication protocol. Estepa et al. [55] investigate the effect of codec on the energy consumption of VoIP calls. Huaming Wu et al. argue for tradeoff between time and energy saving with cloud offloading in [56]. Also, these authors propose an energy-efficient offloading-decision algorithm based on Lyapunov optimization to balance the energy-delay tradeoff based on different offloading-decision criteria [57].

According to the above studies, firstly, energy management issues on the VoIP providers' side, such as SIP servers and switches, are not studied as thoroughly as the user equipment side, such as SIP mobile devices. Secondly, the problems of overload control and energy management, in the context of the VoIP network, have not been concurrently investigated. Moreover, to the best of our knowledge, there are no studies on a comprehensive approach combining server energy management with switch energy management for the standard VoIP network. Therefore, the exploration of such a mechanism is timely and crucial, especially considering the rapid development of VoIP applications and the emergence of NFV and SDN.

3.3 Contributions

Given the importance of the mentioned challenges in VoIP networks, the main purpose of the current paper is to design and implement an automated VoIP resource management system. With the help of the IaaS model, a trade-off between energy and overload is ensured. In this regard, the most important contributions of this paper include:

1) Defining the problem of preventing overload occurrence on VoIP servers and switches along with decreasing power consumption as an optimization problem,
2) Demonstrating NP-hardness of the defined optimization problem with limited resources,
3) Designing a framework called GreenVoIP for integrated resource management of VoIP cloud centers by using NFV and SDN concepts,
4) Designing algorithms that can effectively prevent overload in the VoIP network and, at the same time, minimize the number of servers and switches used,
5) Implementation and evaluation of the performance of the proposed approach in a real platform and under various scenarios.

IEEE Transactions on Green Communications and Networking, DOI 10.1109/TGCN.2020.2982821

4 PROBLEM FORMULATION

Assume that a VoIP communication network can be modeled as graph $G = (V, E)$, where V represents the set of switches, and E indicates the set of links. The notations used in this subsection are described in Table 1.

Table 1
Key Nomenclatures

V	Set of switches
E	Set of links
P	Set of all the routes from s to t
v	The number of switches
e	The number of links
l	The number of VoIP gateways
z	The number of VoIP servers
$d_{i,j}$	Delay of link $(i, j) \in E$
$b_{i,j}$	Bandwidth of link $(i, j) \in E$
r	The resource consumption of each VoIP server
ρ	The remaining resources of each VoIP server
w	The power consumption of each VoIP server
ω	The remaining power of each VoIP server
h	Hop count
s	Source node (the VoIP gateway)
t	Target node (the most appropriate VoIP server)
q_p	QoS of path $p \in P$
r_p	Resource consumption of server on path $p \in P$
w_p	Power consumption of server on path $p \in P$
b_p	Bandwidth of path $p \in P$
d_p	Delay of path $p \in P$
h_p	Length of path $p \in P$
B_p	The maximum network bandwidth
D_p	The maximum acceptable delay
H_p	The maximum number of permissible hops
R_p	The maximum server resources

Let $|V| = v$ and $|E| = e$ be the number of switches and number of links, respectively. This graph connects l VoIP gateways to z VoIP servers with power consumption w. VoIP gateway is a bridge which connects between communication network (servers/switches) and SIP user agent. It is responsible for protocol conversion and data fusion of different user data. Each link $(i, j) \in E$ has a delay $d_{i,j}$ and bandwidth $b_{i,j}$. The resource consumption of each VoIP server is also denoted by r. Here, ρ represents the remaining resources of each VoIP server. Also, the power consumption of each VoIP server is also denoted by w, and ω represents the remaining

IEEE Transactions on Green Communications and Networking, DOI 10.1109/TGCN.2020.2982821

power of each VoIP server. Generally h is considered as a hop count. The hop count and resources are assumed to be non-negative.

There are three types of VoIP traffic on this network (delay-centric, bandwidth-centric, and best-effort). The goal is to find the best path from the source node s (the VoIP gateway) to the target node t (the most appropriate VoIP server) which satisfies the resource and power limits of the servers as well as QoS of traffic. In other words, the objective of the problem is balancing the load of VoIP servers, and satisfying the QoS and energy in order to efficiently use resources and consequently achieve low delay and high throughput. In this regard, to guarantee QoS requirements, we are seeking the shortest path for best-effort traffic, minimum-delay path for delay-centric traffic, and maximum-bandwidth path for bandwidth-centric traffic.

Prior to proposing the SDN-based heuristic approach, we prove that the problem of load-balancing and energy-aware routing in the VoIP network with resource limitation is an ILP problem and is, therefore, NP-hard.

Proposition 1. *The joint load-balance and energy-aware routing problem in the VoIP communication network with limited server resources is an ILP problem.*

Proof: Let P be the set of all the routes from s to t. For any path $p \in P$, a binary variable u_p is introduced. Also, q_p, r_p, ρ_p, w_p, and ω_p are used to denote QoS, resource consumption, remaining resources, power consumption, and remaining power of p, respectively. The optimization problem is (Model 1):

IEEE Transactions on Green Communications and Networking, DOI 10.1109/TGCN.2020.2982821

$$\textbf{\textit{maximize}} \sum\nolimits_p f_p u_p \tag{1}$$

subject to:

$$f_p = \sum\nolimits_p (\alpha q_p - \beta r_p - \gamma w_p), \forall p \in P \tag{2}$$

$$q_p = \phi \frac{b_p}{B_p} - \delta \frac{d_p}{D_p} - \vartheta \frac{h_p}{H_p}, \ \forall p \in P \tag{3}$$

$$b_p \leq b_{i,j} u_p, \ (i,j) \in p, \forall p \in P \tag{4}$$

$$d_p = \sum\nolimits_{(i,j) \in p} d_{i,j} u_p, \forall p \in P \tag{5}$$

$$h_p = \sum\nolimits_{(i,j) \in p} |\{i,j\}| u_p, \forall p \in P \tag{6}$$

$$r_p \leq \frac{\rho_p}{R_p} u_p, \ \forall p \in P \tag{7}$$

$$w_p \leq \frac{\omega_p}{W_p} u_p, \ \forall p \in P \tag{8}$$

$$b_p \leq B_p, \ \ d_p \leq D_p, \ \ h_p \leq H_p, \ \ \rho_p \leq R_p, \ \ \omega_p \leq W_p, \tag{9}$$

$$0 \leq q_p \leq 1, \ \ 0 \leq r_p \leq 1, \ \ 0 \leq w_p \leq 1, \tag{10}$$

$$0 \leq \alpha \leq 1, \ \ 0 \leq \beta \leq 1, \ \ 0 \leq \gamma \leq 1, \tag{11}$$

$$\sum\nolimits_p u_p = 1, \tag{12}$$

$$\sum\nolimits_p (\phi + \delta + \vartheta) u_p = 1, \tag{13}$$

Variables: $u_p \in \{0,1\}, \phi, \delta, \vartheta \in \{0,1\}, q_p, r_p, w_p, b_p, d_p, h_p \geq 0$

This model seeks to find the path p, so that f_p is maximized (Eq. (1)). Here, f_p is a function of both the QoS, server consumable resources, and power consumption of the path p (Eq. (2)). The α, β, and γ coefficients provide a trade-off between QoS, resources, and energy. The objective is to achieve the highest weighted sum of QoS, resources, and energy. With f_p being maximized, a path can be explored that has the highest weighted sum of QoS and server resources and power. The q_p depends on the traffic class, while ϕ, δ and ϑ show which traffic is related to which of the three traffic classes (bandwidth-centric, delay-centric, and best-effort traffic, respectively). This model seeks to find the maximum-bandwidth, minimum-delay or shortest path by Eq. (3) and with respect to the traffic classes. Here, b_p, d_p and h_p represent the bandwidth, delay, and length of p, respectively (Eq. (4)-(6)). Eq. (7) ensures that the server consumable resources associated with the path p are less than its remaining resources. Also, Eq. (8) ensures that the server consumable power associated with the path p is less than its remaining power. Moreover, B_p, D_p, H_p, R_p, and W_p are the maximum network bandwidth, maximum acceptable delay, the maximum number of permissible hops, maximum server resources, and the maximum power of server respectively (Eq. (9)). The existence of these parameters causes the q_p, r_p, and w_p

values fall to between 0 and 1 (Eq. (10)), and therefore Eq. (2) normalizes. For this reason, α, β, and γ coefficients are between 0 and 1 as well (Eq. (11)). Eq. (12) guarantees that exactly one path is selected. Furthermore, each time the model is solved, the best path is obtained for one of the traffic classes (Eq. (13)). Although the objective function and constraints are linear, the binary variable u_p renders the model an ILP which is generally NP-hard, and cannot be solved in polynomial time [58]. To overcome this drawback, a SDN-based framework is proposed. ■

5 GREENVOIP

This section provides a framework for managing VoIP network resources based on the concept of IaaS and being equipped with NFV and SDN technologies. As stated earlier, the goal of the IaaS model is to provide resources based on demand, a concept with which the present study seeks to model its VoIP network. In other words, we present a VoIP resource management framework which is based on demand and aware of the load, thus potentially able to establish a trade-off between overload and energy issues.

For this purpose, we design a framework, called GreenVoIP, in two phases. In the first phase, the focus is on SIP server consolidation. The second phase, though, not only involves SIP server consolidation but also switch consolidation as well. It should be taken into account that consolidation must be performed in such a way that, while the amount of active equipment is minimized, overloading is also prevented and QoS is observed. The symbols used in this section are described in Table 2.

5.1 Phase 1: Load-aware Server Consolidation

Fig. 7 shows the GreenVoIP architecture in the first phase. This architecture is consistent with the NFV layered architecture described earlier and is made up of two main layers: infrastructure and MANO.

The infrastructure layer includes the physical layer, the hypervisor, and the virtual layer. The physical layer includes PMs or physical servers. In the proposed architecture, the functions of SIP servers are no longer hardware. Rather, each of these functions is virtual and a separate VNF in the virtual layer, called SIP-VNF, is provided by the VM feature. For simplicity, there is a one-to-one correspondence between VMs and SIP-VNFs.

The MANO layer manages the PMs and SIP-VNFs through migration and consolidation strategies and includes several components that run at alternate intervals. The *PM Resource Utilization Monitor* component monitors the CPU and the memory of the Active PM (APM)[1], in addition to retaining their history over time. According to this history, the *PM Load Estimation* component predicts the future demand of SIP-VNFs to the CPU and memory resources. In other words, it predicts the future load of APMs according to past statistics. As a result, the *PM Temperature Measurement* component obtains the future temperature of each APM and places

1. A server is considered active when it has at least one running SIP-VNF.

Table 2

Key Nomenclatures

x_i^k	Sample i of the CPU usage of server k
y_i^k	Sample i of the memory usage of server k
$\widehat{x}_{i+1}^k$	Estimate of x_{i+1}^k
$\widehat{y}_{i+1}^k$	Estimate of y_{i+1}^k
θ^k	CPU thresholds of active server k (θ_{low}^k & θ_{high}^k)
φ^k	Memory thresholds of active server k (φ_{low}^k & φ_{high}^k)
T_{CPU}^k	Normalization of $\widehat{x}_{i+1}^k$ ($0 \leq T_{CPU}^k \leq 1$)
T_{mem}^k	Normalization of $\widehat{y}_{i+1}^k$ ($0 \leq T_{mem}^k \leq 1$)
α	The weight of the CPU at server temperature
$Temp(k)$	The temperature of active server k
Δ_{hot}	Hot threshold (e.g., 0.9)
Δ_{cold}	Cold threshold (e.g., 0.25)
Δ_{green}	Green threshold (e.g., 0.4)
P	The set of total APMs
H	The set of hot APMs
C	The set of cold APMs

it into one of three categories: hot, warm, and cold. By knowing the future state of APMs, the *System State Determination* component can identify the future state of the entire system (the sum total of APMs). The future state of the system can be hot, warm, or cold. In the hot mode, the *Overload Control* component is executed and some SIP-VNFs hosted on hot points are transferred to another node to reduce the load on that point. It may be necessary to add a new PM to the system to reduce the temperature of the whole system. If the system state is cold, the *Green Computing* component is run and the SIP-VNFs hosted on cold points are transferred to another node. In this case, some of these PMs can be switched off to save energy. Finally, a migration list of SIP-VNFs is completed and passed on to the *SIP-VNF Manager* component to be executed.

Therefore, the goal is to appropriately determine the future size of the network, including active servers, according to the system temperature in such a way so as to satisfy energy and overload constraints. In the following, each of the MANO layer modules is explained in detail.

5.1.1 PM Resource Utilization Monitoring

This component is responsible for monitoring the utilization of PM resources, including the utilization of CPU and memory. These utilization metrics, which are monitored for any active physical server, are collected in this component to provide historical data needed for the next component.

In principle, this component samples the CPU and the memory of APMs at certain intervals and holds ρ number of the last sampled values in the f_{cpu} and f_{mem} functions. The f_{cpu}^1 function contains ρ number of the

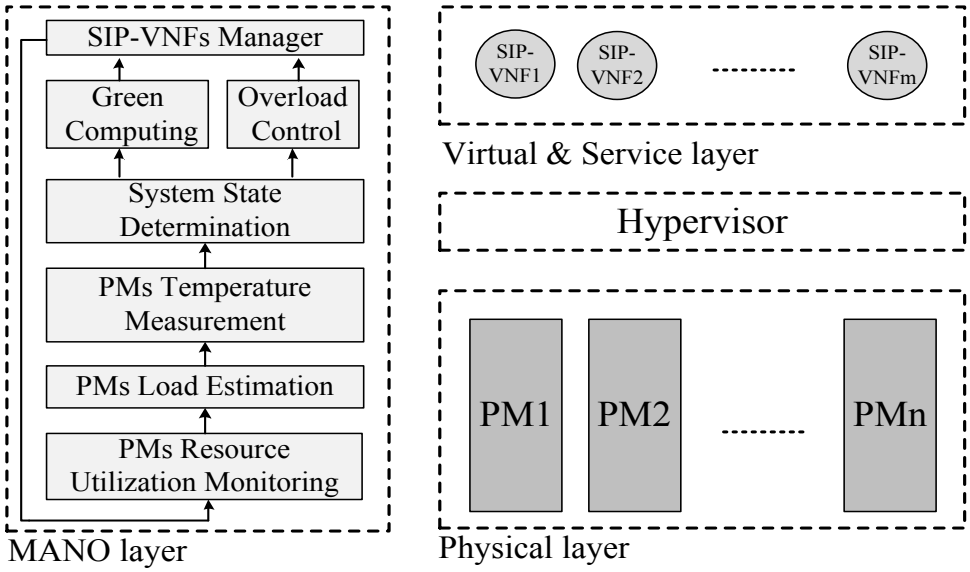


Figure 7. GreenVoIP architecture in the first phase.

last values of the CPU usage of physical machine 1. The f^1_{mem} function contains ρ number of the last values of the memory usage of physical machine 1.

Therefore, for PM1, x^1_i is the value of sample i of CPU usage and, as a result, $f^1_{cpu}(x) = [x^1_i, x^1_{i-1}, ...x^1_{i-\rho+1}]$ is a vector that contains the CPU usage history of PM1. Similarly, y^1_i is the value of sample i of memory usage and, as a result, $f^1_{mem}(y) = [y^1_i, y^1_{i-1}, ...y^1_{i-\rho+1}]$ is a vector that contains the memory usage history of PM1.

5.1.2 PM Load Estimation

Changes in the network input load affect the utilization of PM resources over time. Therefore, in order to know the future state of the load, one can use the resource utilization history of each PM, as this can be a reliable measure of a PM's future state. The aim is to obtain an estimate of x_{i+1} ($\widehat{x}^k_{i+1}$) and y_{i+1} ($\widehat{y}^k_{i+1}$) for each APM with regard to f functions and the prediction system.

Time-series analysis is one of the most important prediction methods. The *Normalized Least Mean Square (NLMS)* is the main predictor providing the best trade-off between complexity, accuracy, and responsiveness [59]. For this reason, the NLMS is applied as the prediction system in the *PM Load Estimation* component. The following intends to obtain $\widehat{x}_{i+1}$ for APM k by employing the NLMS algorithm ($\widehat{x}^k_{i+1}$).

Given the ρ observation vector of x^k_i, $f(x^k_i) = [x^k_i, x^k_{i-1}, \dots x^k_{i-\rho+1}]$ generates estimation $\widehat{x}^k_{i+1}$ of value x^k_{i+1}. The NLMS filter coefficients are time differing and tuned on the basis of the feedback information taken by error ε_i, in which $\varepsilon^k_i = x^k_{i+1} - \widehat{x}^k_{i+1}$. The vector of filter coefficients with h_i is specified. The values of h adjust dynamically in order to decrease the *Mean Square Error* [59]. The NLMS performs the following [44]:

1- Initializes coefficient h_0

2- For each new data, updates filter h_i based on the recursive equation:

$$h_{i+1}^k = h_i^k + \mu \frac{\varepsilon_i^k f(x_i^k)}{\| x_i^k \|^2} \tag{14}$$

where $\| x_i^k \|^2 = f(x_i^k) f^T(x_i^k)$ and μ is a fixed parameter called step size. Pursuant to [59], NLMS converges as long as $0 < \mu < 2$. At time i, values x_{i+1}^k, hence ε_i^k, are unknown. Therefore, value ε_{i-1}^k is used instead and the one step NLMS predictor update equation becomes:

$$h_{i+1}^k = h_i^k + \mu \frac{\varepsilon_{i-1}^k f(x_{i-1}^k)}{\| f(x_{i-1}^k) \|^2} \tag{15}$$

Overall, the NLMS algorithm formulation can be summarized as:

$$f(x_i^k) = [x_i^k, x_{i-1}^k, \dots x_{i-\rho+1}^k] \tag{16}$$

$$\widehat{x}_{i+1}^k = h_i^k f^T(x_i^k) \tag{17}$$

$$h_i^k = h_{i-1}^k + \mu \frac{\varepsilon_{i-1}^k f(x_{i-1}^k)}{\| f(x_{i-1}^k) \|^2}, \quad h_0 = 0 \tag{18}$$

$$\varepsilon_i^k = x_i^k - \widehat{x}_i^k \tag{19}$$

$$\| f(x_i^k) \|^2 = f(x_i^k) f^T(x_i^k) \tag{20}$$

where $f(x_i^k)$ and $\widehat{x}_{i+1}^k$ are the input and output of the predictor, respectively [44]. Similarly, $\widehat{x}_{i+1}$ and $\widehat{y}_{i+1}$ can be obtained for each APM. Details of the NLMS and rationale behind Eq. (14) to (20) are given in the appendix.

5.1.3 PM Temperature Measurement

The temperature of each active server is affected by both $\widehat{x}_{i+1}$ and $\widehat{y}_{i+1}$. However, the scale of these two variables differs. Therefore, with Eq. (21) and (22), they are normalized. Suppose one seeks the temperature of active server k.

$$T_{CPU}^k = \begin{cases} 0, & \widehat{x}_{i+1}^k < \theta_{low}^k \\ \frac{\widehat{x}_{i+1}^k - \theta_{low}^k}{\theta_{high}^k - \theta_{low}^k}, & \theta_{low}^k < \widehat{x}_{i+1}^k \leq \theta_{high}^k \\ 1, & \widehat{x}_{i+1}^k \geq \theta_{high}^k \end{cases} \tag{21}$$

$$T_{mem}^{k}=\begin{cases}0, & \widehat{y}_{i+1}^{k}<\varphi_{low}^{k}\\ \frac{\widehat{y}_{i+1}^{k}-\varphi_{low}^{k}}{\varphi_{high}^{k}-\varphi_{low}^{k}}, & \varphi_{low}^{k}<\widehat{y}_{i+1}^{k}\leq\varphi_{high}^{k}\\ 1, & \widehat{y}_{i+1}^{k}\geq\varphi_{high}^{k}\end{cases} \tag{22}$$

The θ_{low}^{k} and θ_{high}^{k} thresholds are fixed values that are considered for the CPU of active server k. The φ_{low}^{k} and φ_{high}^{k} thresholds are considered for server k's memory. As a result, the temperature of active server k is calculated using Eq. (23):

$$Temp(k)=\alpha T_{CPU}^{k}+(1-\alpha)T_{mem}^{k}, Temp(k)\in[0,1] \tag{23}$$

Coefficient α represents the weight or importance of the CPU or memory at server temperature. It is possible to alter the degree of importance of the CPU or memory at the temperature by changing coefficient α[2].

Finally, according to Eq. (24), an APM is called the *cold* point when its temperature is lower than cold threshold (Δ_{cold}), meaning that the utilization of each of its resources is low. This suggests that the server is mostly idle and considered a potential candidate for shutdown so as to save energy. If the temperature of APM k is higher than hot threshold (Δ_{hot}), it is placed in the *hot* category. This shows that, given the state of the resources, the server is not in a good condition and there is the possibility of overload; the temperature of a hot point indicates its overload. An active server is *warm* if its temperature is between these two thresholds, signifying that resource utilization is high enough to justify keeping the server running, but not running so much that it might be at risk of turning into a hot point.

$$Class(k)=\begin{cases}cold, & Temp(k)\leq\Delta_{cold}\\ warm, & \Delta_{cold}<Temp(k)\leq\Delta_{hot}\\ hot, & Temp(k)\geq\Delta_{hot}\end{cases} \tag{24}$$

In Eq. (24), two thresholds Δ_{cold} and Δ_{hot} were considered ($\Delta_{cold}\in[0,\Delta_{hot})$ and $\Delta_{hot}\in(\Delta_{cold},1]$). If $Temp(k)$ value, which is obtained from Eq. (23), is less than Δ_{cold}, the class of APM k is cold. It means that server is in a desirable condition according to the conditions of resources and there is no overload. If $Temp(k)$ value is more than Δ_{hot}, then there would be an overload in the server and the class of APM k is hot. When $Temp(k)$ is between Δ_{cold} and Δ_{hot} thresholds, the class of APM k is warm. This mechanism is a method of proactive management of resources to prevent the overload. If we consider a small value for these thresholds, the usage of resources would decrease and lead to lower productivity. But, if we consider larger values for them, more resources would be utilized and increase the probability of the overload due to lack of resources. Therefore, precisely setting the threshold values for maintaining trade-off between productivity and resources would be

2. This is due to the fact that the CPU and memory are not equally involved in an overload [43].

important. In its simplest and one of the most practical forms, productivity can be defined as a total throughput of different flows passing through a server. Consequently, we aim to provide a balance between usage of server resources and an increase in the throughput with the condition of overload and energy. The values of these thresholds which were obtained by several tests are presented in the performance evaluation section.

5.1.4 System State Determination

Once the temperature and the class of each APM are determined, the system state can then be determined. The *System State Determination* component can determine the state of the system according to the finite state machine in Fig. 8. According to this figure, the system can be in one of the three states: *hot*, *warm*, and *cold*.

When at least one APM is of the hot category, the system is in a hot state and an *Overload Control* algorithm must be run. In this case, the overload will be reduced by transferring some of the SIP-VNFs of hot points to other APMs or adding a new PM to the system, thus increasing network size.

If no APMs are from the hot category and the temperature of all APMs is lower than the green threshold (Δ_{green}), the system state is cold and the *Green Computing* algorithm must be run by the related component. In this case, the SIP-VNFs on the cold points are transferred to another APM so that these points can be turned off, therefore reducing network size.

Finally, if no APM is hot but the temperature of at least one APM is higher than Δ_{green}, the system is in a warm state and no action is required.

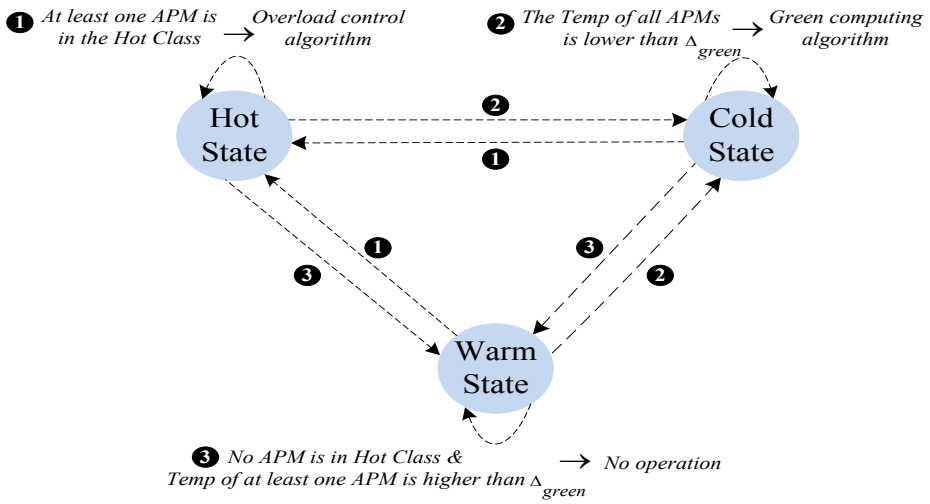


Figure 8. The finite state machine of the *System State Determination*.

5.1.5 Overload Control

In this component, the overload reduction algorithm is executed (Algorithm (1)). The purpose of this component is to reduce the temperature of all hot points as much as possible and so it aims to lower the temperature of each hot server. That is, as long as the temperature of a hot server is higher than the Δ_{hot}, the

Algorithm 1: Overload Control

```
for each APMi ∈ H do
    while Temp(i) ≥ Δ_hot do
        Migration of a SIP-VNF from PM i to PM ∈ P ≠ i that is cold and will stay cold;
        else
        Migration of a SIP-VNF from PM i to PM ∈ P ≠ i that is cold and becomes warm;
        else
        Migration of a SIP-VNF from PM i to PM ∈ P ≠ i that is warm and will stay warm;
        else
        A PM is added to the set P and a SIP-VNF is migrated to it from PM i;
    end
end
```

overload reduction algorithm tries to find a suitable destination server for a SIP-VNF from this hot server. After receiving this SIP-VNF, the destination server should not turn into a hot point. The priority of the destination server is "be cold and stay cold after receiving the SIP-VNF," or "be cold and become warm after receiving the SIP-VNF," or "be warm and stay warm after receiving the SIP-VNF," respectively.

The reason is that in the case of overload, the priority is to increase the load on a cold point. However, if necessary, a warm point is also acceptable as the destination server, provided that it does not turn into a hot point. Eventually, in the absence of a destination server with these qualities, a new PM is added to the set of APMs and this is considered as the SIP-VNF migration destination.

Note that each run of the algorithm transfers one SIP-VNF from the overloaded server. This does not necessarily eliminate the hot point, but at least the temperature is reduced. If the server continues to stay hot in the next run, the algorithm will repeat this process. There is the possibility of designing the algorithm in such a way so as to transfer multiple SIP-VNFs per run. However, this may cause overload in other servers.

In the end, a migration list of SIP-VNFs is prepared and sent to the *SIP-VNF Manager* component for execution.

5.1.6 Green Computing

In this component, a green computing algorithm is run (Algorithm (2)). The goal is to turn off the cold PMs, if possible, and transfer their SIP-VNFs to other servers. The challenge here is to reduce the number of active servers at the time of underload in such a way that performance is not compromised. This explains why the green computing algorithm is called upon when the temperature of all active servers is lower than the green threshold. This issue ensures that the system temperature is low enough to turn off the cold PMs without

IEEE Transactions on Green Communications and Networking, DOI 10.1109/TGCN.2020.2982821

Algorithm 2: Green Computing

```
for each APMi ∈ C & if state = cold do
    while Temp(i) ≤ Δ_cold & PM i has at least one SIP-VNF do
        Migration of a SIP-VNF from PM i to the PM ∈ P ≠ i that is warm and will stay warm;
        else
        Migration of a SIP-VNF from PM i to the PM ∈ P ≠ i that is cold and becomes warm;
        else
        Migration of a SIP-VNF from PM i to the PM ∈ P ≠ i that is cold and will stay cold;
    end
    The PM i is switched off;
end
```

adversely affecting performance or causing overload. This algorithm is executed for each cold point, provided that the state is still cold.

For each cold point, the "While" loop is repeated until at least one SIP-VNF is running on it. The loop looks for the appropriate destination for the SIP-VNFs located on this cold point. The type of server which is suitable as the destination server "is warm and stays warm after receiving a SIP-VNF." Otherwise, it will look for a server that "is cold and becomes warm after receiving a SIP-VNF" or a server that "is cold and stays cold."

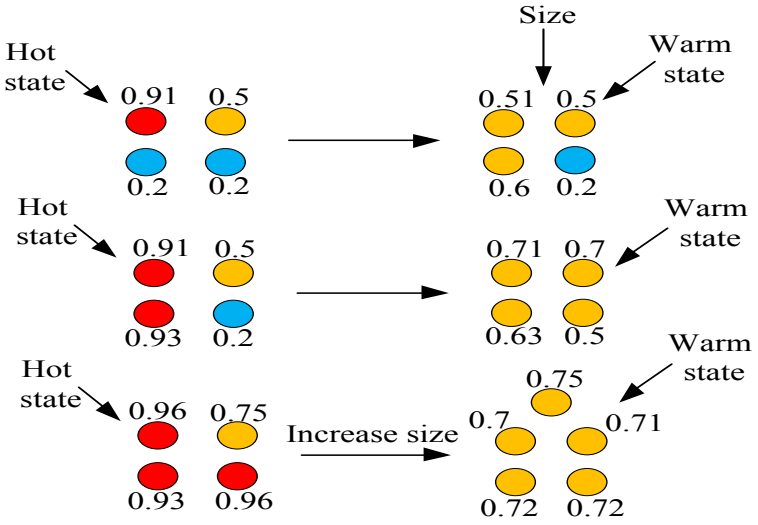
To explain, the load increase on a cold point reduces the likelihood of its being removed. However, if the need arises, the cold point is also acceptable as the destination server. Eventually, all the SIP-VNFs of this PM are transferred and this PM is removed from the set of APMs (P). A migration list of SIP-VNFs is prepared and sent to the *SIP-VNF Manager* component for execution. The list of cold points is also updated because some of them may no longer be cold points as a result of the SIP-VNF migration of in the above process.

From Algorithms 1 and 2, we find that the optimal state for the system is the warm state, for in this state there is neither the risk of system overload nor excessive of energy consumption. An example of a system state before and after the application of the above algorithms is shown in Fig. 9.

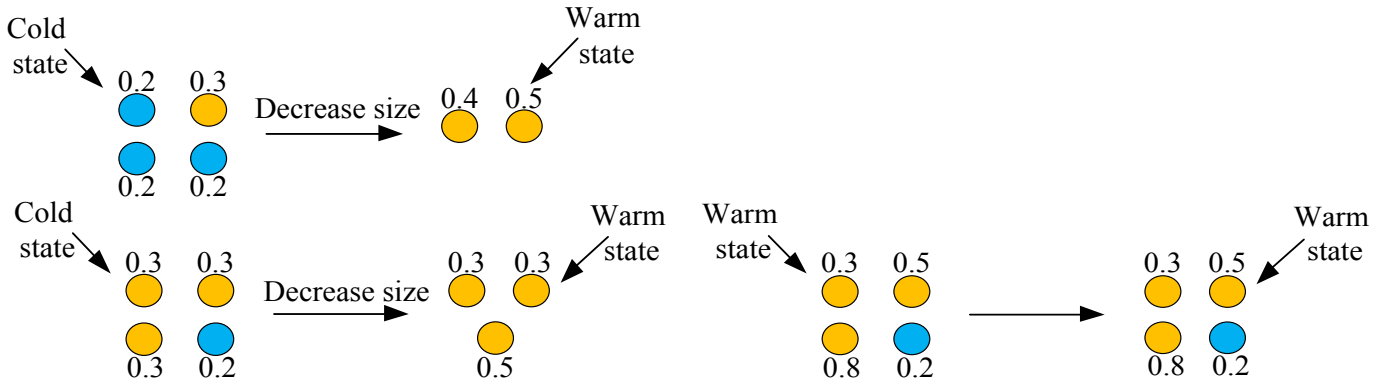
Note that the cold points are removed from the system only when the temperature of all active servers (APMs) is lower than the Δ_{green}. Otherwise, those cold points are left as potential destination servers for future offloading. This is consistent with our philosophy that green computing must be performed conservatively.

5.1.7 SIP-VNF Manager

This component receives and runs the migration list of the SIP-VNFs from the two previous components. That is, it sends the necessary commands for SIP-VNF migrations or for the switching off and/or on of PMs to the infrastructure layer equipment.



(a) System temperature change from hot to warm and a reduction in overload.



(b) System temperature change from cold to warm and a reduction in energy consumption.

(c) The warm system is optimal.

Figure 9. Three examples of hot, cold, and warm systems and consolidation servers (Δ thresholds values according to Table 2).

5.2 Phase 2: Load-aware Server+Switch Consolidation

In the first phase, it was hypothesized that the switches that provide server communications are conventional and therefore cannot be centrally managed. The architecture presented in the first phase can meet the requirements of VoIP networks with an infrastructure of conventional switches. However, with the advent of the SDN, in addition to servers, the network switches can be centrally managed in line with the objectives of the current paper.

Assuming that OpenFlow switches exist, Fig. 10, therefore, shows an overall display of this scheme in the second phase. This scheme is in accordance with the software-defined NFV architecture described earlier. In this scheme, SIP-VNFs are managed by VNF MANO and OpenFlow switches by the SDN controller. The infrastructure layer information is available to both VNF MANO and the SDN controller, the two of which communicate with each other.

An example of SIP server and switch consolidation is given in Fig. 11. Prior to consolidation, SIP-VNFs 1 through 6 are hosted on four servers and connected to each other via four switches. In this example, the P3 and

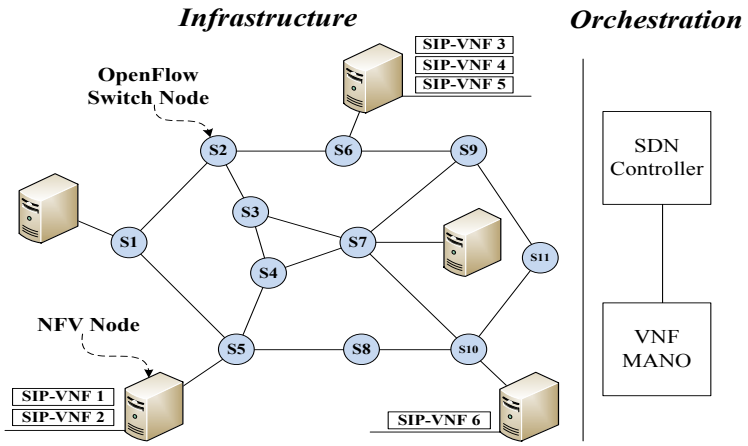


Figure 10. Infrastructure layer including NFV nodes, SIP-VNFs, and OpenFlow switches, and the orchestration layer including the SDN controller and the VNF MANO.

P4 servers are cold, P1 and P2 are warm, and the state of the system is cold. After SIP-VNF5 and SIP-VNF6 migrate to P1 and P2 respectively, the state of the system becomes warm. In this case, in addition to P3 and P4, the switches connected to them (S3 and S4) may also be turned off. This is possible with the SDN controller's global view.

Fig. 12 presents the GreenVoIP architecture in the second phase. This architecture consists of three layers: infrastructure, control, and application. The infrastructure layer includes PMs (or physical servers), SIP-VNFs, and OpenFlow switches. The control layer consists of two SDN and NFV controllers that maintain the overall structure of the network and manage the infrastructure layer. The necessary commands and decisions for the configuration of the infrastructure layer equipment are provided by the components of the application layer and given to the controllers. The communication protocol between the layers is a series of standard Open APIs (Application Programming Interface), including the OpenFlow protocol. The components of the application layer, which are related to the server management and SIP-VNFs, are similar to the GreenVoIP components in the first phase. However, to manage the switches, two components, namely *Network Discovery* and *Network*

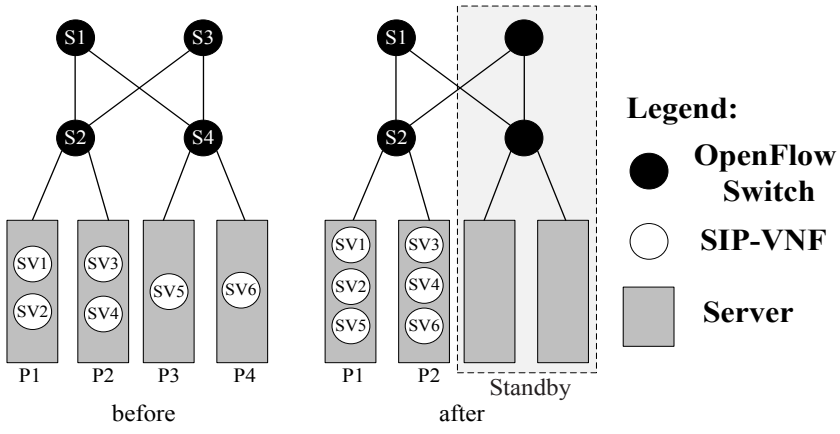


Figure 11. Example of SIP server and switch consolidation.

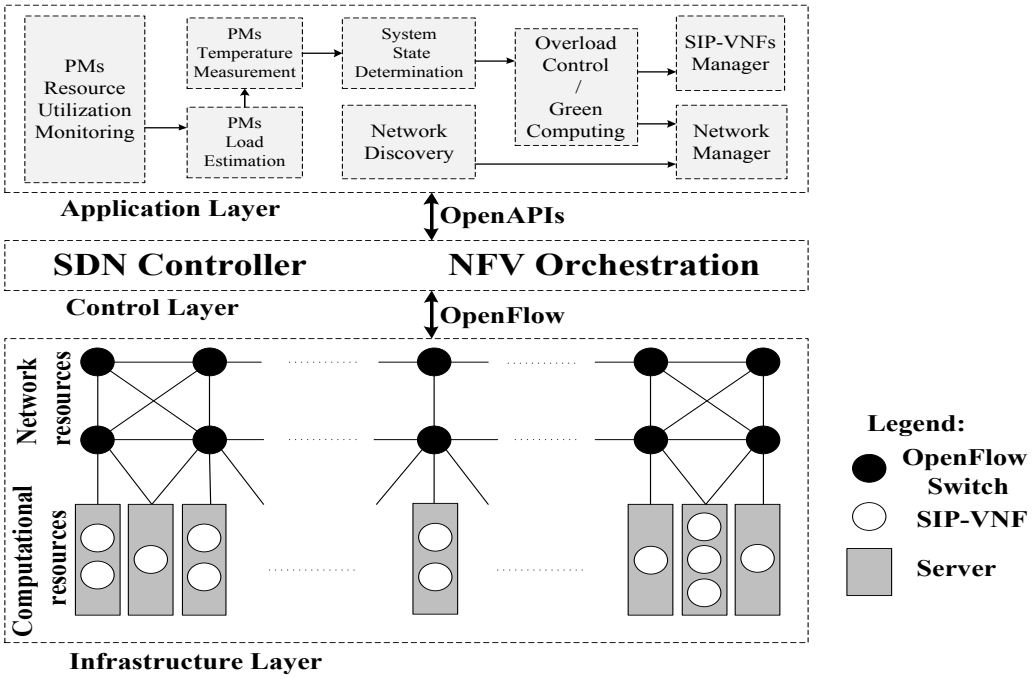


Figure 12. GreenVoIP architecture in the second phase.

Manager, are added to the application layer. By using the Link Layer Discovery Protocol (LLDP), the *Network Discovery* component obtains full information on the current state of the switches and their topology. The *Network Manager* component also provides a list of which switches should be activated and/or which should be deactivated, depending on the system state. That is, with regard to the *Overload Control/Green Computing* and the *Network Discovery* component information, when the system state is hot and a new PM must be added to the system, the nearest PM is added to the APM set. This requires the least number of switches to be activated. Also, when the system state is cold and a cold APM should be turned off, if a switch becomes idle, that switch is added to the list of turned-off switches.

6 Performance Evaluation

In this section, we first implement GreenVoIP on a real testbed and evaluate its performance. This testbed employs four servers and two OpenFlow switches. After that, the proposed framework is simulated in the *CloudSimSDN emulator* [60] in order to evaluate GreenVoIP performance by increasing the number of servers and switches (to the hundreds).

6.1 Implementation Testbed

Fig. 13 indicates the test topology in the infrastructure layer, which consists of four PMs and two OpenFlow switches. In addition, Fig. 14 provides an illustration of the testbed.

In this testbed, *Floodlight v1.2*, *Open vSwitch v2.4.1 (OVS)*, and *Kamailio v4.3.6* are employed to implement the controller, OpenFlow switches, and SIP-VNFs respectively. Also, VoIP calls are made with the *Spirent Abacus*

5000 device. The Floodlight controller is executed on an *HP G9 DL580* server, which is equipped with a *VMware ESXi hypervisor v6.0*. Open vSwitches are executed on two types of hardware, each of which is equipped with an Intel Xeon 4-core 3.70 GHz CPU and 16 GBs of RAM. Also, the execution of the Kamailio is on four *QSR Quanta* Servers (PM1 to PM4), each running an *Ubuntu Linux 14.04* OS and *Docker v1.12*, so that each *Container* contains one Kamailio (one Container for each SIP-VNF). The *Oprofile* Software is also used to observe the resources consumed by the PMs. In addition, the application plane is implemented as a module running on top of the controller. We run each experiment three times and take the average number as the result. The α parameter is tuned to 0.65. Also, ρ and μ are chosen to be 30 and 0.8 respectively. 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 overfitting or creating too high a variance. The improved performance reveals which parameter settings are more favorable (tuned) or less favorable (untuned).

6.2 Experimental Results

6.2.1 GreenVoIP Effectiveness

The first test evaluates the efficiency of GreenVoIP in overload reduction and green computing. For this purpose, the load is injected to the set of PMs in a 4000-second time period. As seen in Fig. 15, PM1 and PM2 are at first active and running five SIP-VNFs. Up to about 1000 seconds, both of these PMs are warm, since their average consumption of CPU and memory is moderate. After 1000 seconds, PM1 is overloaded and, as a result, SIP-VNF1 migrates to PM3. After 2000 seconds, the PM2 resource consumption increases and so this PM also starts heating, which results in SIP-VNF4 migrating to PM4. Over time and with load reduction, PM1 cools down after the 2500th second and both SIP-VNF2 and SIP-VNF3 migrate to PM2, thus releasing PM1. Similarly, PM3 and PM4 cool down after about the 2600th and the 3000th second respectively and their

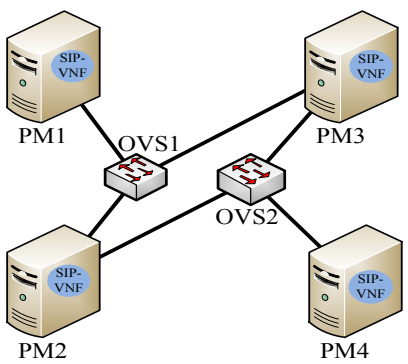


Figure 13. Test topology.

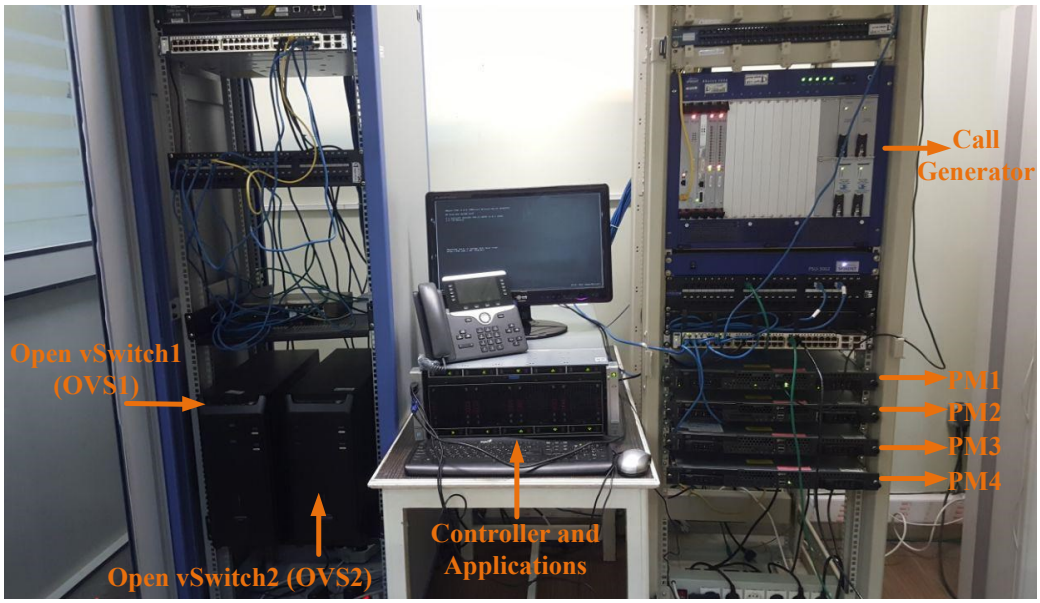


Figure 14. GreenVoIP testbed.

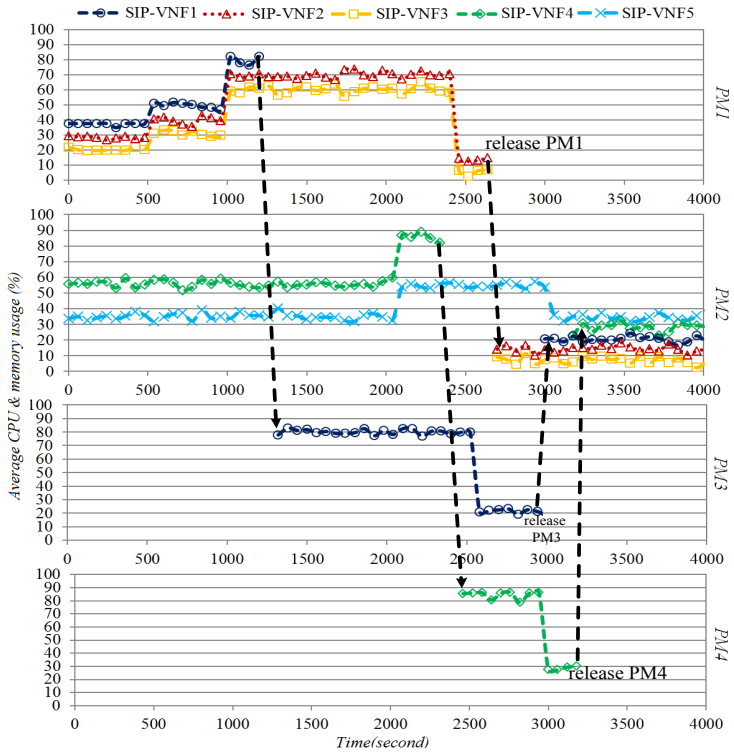
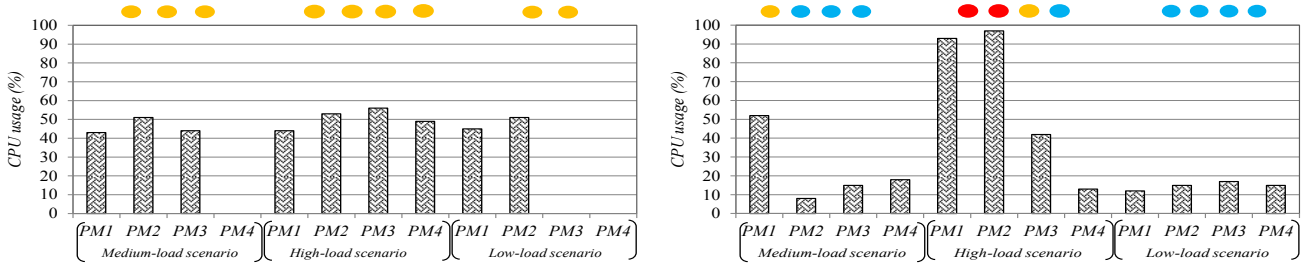


Figure 15. GreenVoIP efficiency.

SIP-VNFs migrate to PM2. After the 3200th second, only PM2 is active with regard to the input load and the other three PMs are inactive to save energy. Therefore, by increasing and decreasing the load, the number of APMs changes appropriately and the algorithm is repeated, spreading or consolidating the SIP-VNFs as needed.

Note that the OVS2 switch is activated after about the 1000th second and when PM3 is activated. This switch is deactivated after about the 3000th second when both PM3 and PM4 are released.



(a) In the presence of GreenVoIP

(b) In the absence of GreenVoIP

Figure 16. CPU usage of PMs.

The following investigates the presence and absence of GreenVoIP in a 24-hour time period, in three scenarios, and in more detail. These studies indicate that the call volume throughout the day is usually average in the morning, highest in the afternoon, and lowest at night. We have implemented these three time periods in the format of the three scenarios: *medium-load*, *high-load*, and *low-load* scenarios with call volumes of 1000, 2000, and 5000 cps respectively. The results are plotted in Fig. 16.

As seen in this figure, the CPU usage of PMs is almost equivalent in all three scenarios in the presence of GreenVoIP (Fig. 16a). Also, GreenVoIP is able to maintain the system in an optimal (warm) state in all three scenarios with the management of SIP-VNFs and APMs. In this regard, only in the high-load scenario are all four PMs active together. In the medium-load and low-load scenarios, the number of APMs is three and two respectively. On the contrary, all three scenarios have all four PMs active in the absence of GreenVoIP (Fig. 16b), which causes the system to be cold in the medium-load and low-load scenarios and hot in the high-load scenario. There is also a huge difference among the CPU usages of the PMs. Similar results are obtained regarding memory, but these will not be discussed here. The number of SIP-VNFs in the presence of GreenVoIP is illustrated in Fig. 17. As seen, the number of SIP-VNFs follows the load pattern. This makes the power consumption also intelligently follows the load pattern (Fig. 18a). As can be seen from Fig. 18, the lowest power consumption is in the low load scenario and the highest during the high load. The average power consumption in the presence of GreenVoIP is lower than in the absence of GreenVoIP. GreenVoIP can keep the system state in all three scenarios in a warm state and thus balances power consumption all the time. But without it, the system situation is cold in the low and medium load scenarios and is hot in the high load scenario which does not result in uniform power consumption.

By managing the number of APMs, GreenVoIP not only saves energy, but also prevents overload, as seen in Fig. 19. In the absence of GreenVoIP, PM1 and PM2 are overloaded in the high-load scenario (Fig. 16b). This results in a very high *average call setup delay* and *request retransmission rate*, since the CPU and memory of these two PMs are almost saturated. With GreenVoIP, the average call setup delay and request retransmission rate are very low. This is even true in the high-load scenario since GreenVoIP is an estimation-based method and constantly surveys PM resources.

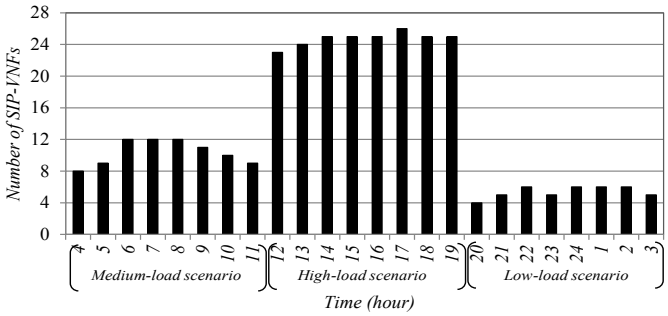
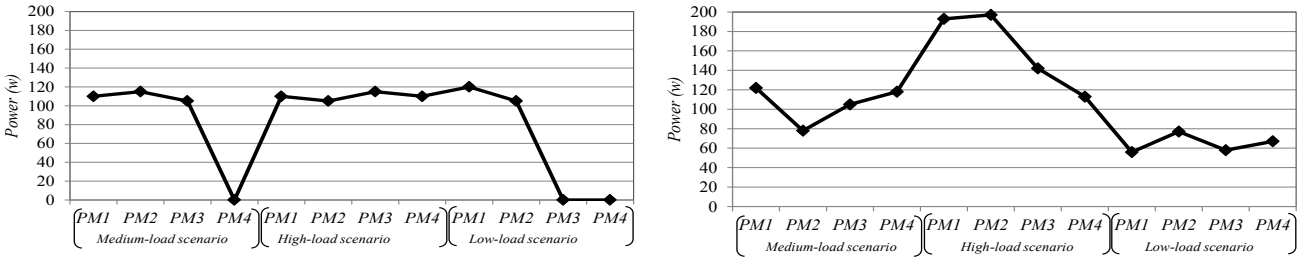


Figure 17. The number of SIP-VNFs in the presence of GreenVoIP.



(a) In the presence of GreenVoIP

(b) In the absence of GreenVoIP

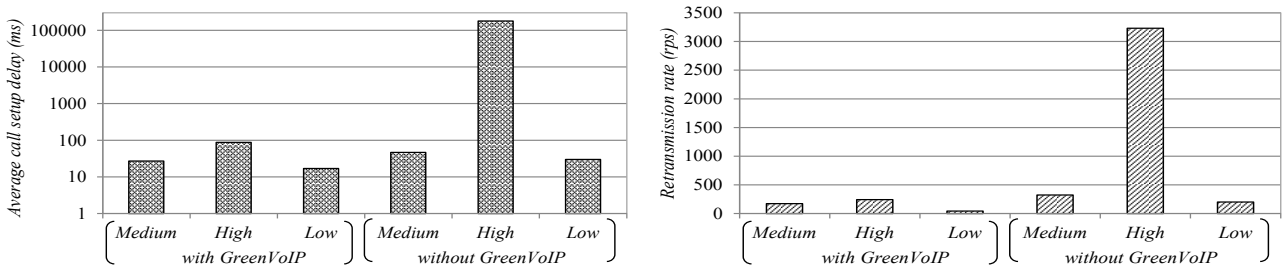
Figure 18. The power consumed by PMs

6.2.2 Comparison with other Methods

This section compares GreenVoIP with the *TLWL* [28], *FWAR*, and *HWAR* methods [27] and also the *without control* model. For this purpose, by increasing the offered load, we compare these five methods in terms of *server throughput*, *average response time*, *number of APMs*, *the number of active OVSs (AOVSs)*, and *the power consumption*. Server throughput is the number of successful calls per time unit. Response time is the time between the call request and the call. The TLWL, FWAR, and HWAR methods are used to control overload through load distribution between SIP servers.

With a load balancer, the TLWL algorithm routes a new call request to the SIP server with the least amount of load. In this algorithm, a counter is allocated to each server and any new call will be routed to the server with the least counter value. So, TLWL selects a server that has the least work, where work (i.e., load) is based on the relative estimates of transaction costs. In other words, TLWL estimates the server load based on the weighted average of transactions that are currently handled by a server. In this regard, this method maintains a set of counters includes the weighted number of transactions assigned to each server. A new request is allocated to the server with the lowest counter.

The two window-based FWAR and HWAR algorithms are used to estimate server load from server response times. In the FWAR algorithm, the window sizes are fixed. However, in the HWAR algorithm, a history of response times is stored in the windows with a mathematical formulation. In this case, a new call will be routed to the server with a history of the shortest response time. HWAR selects a server with the smallest



(a) Average call setup delay

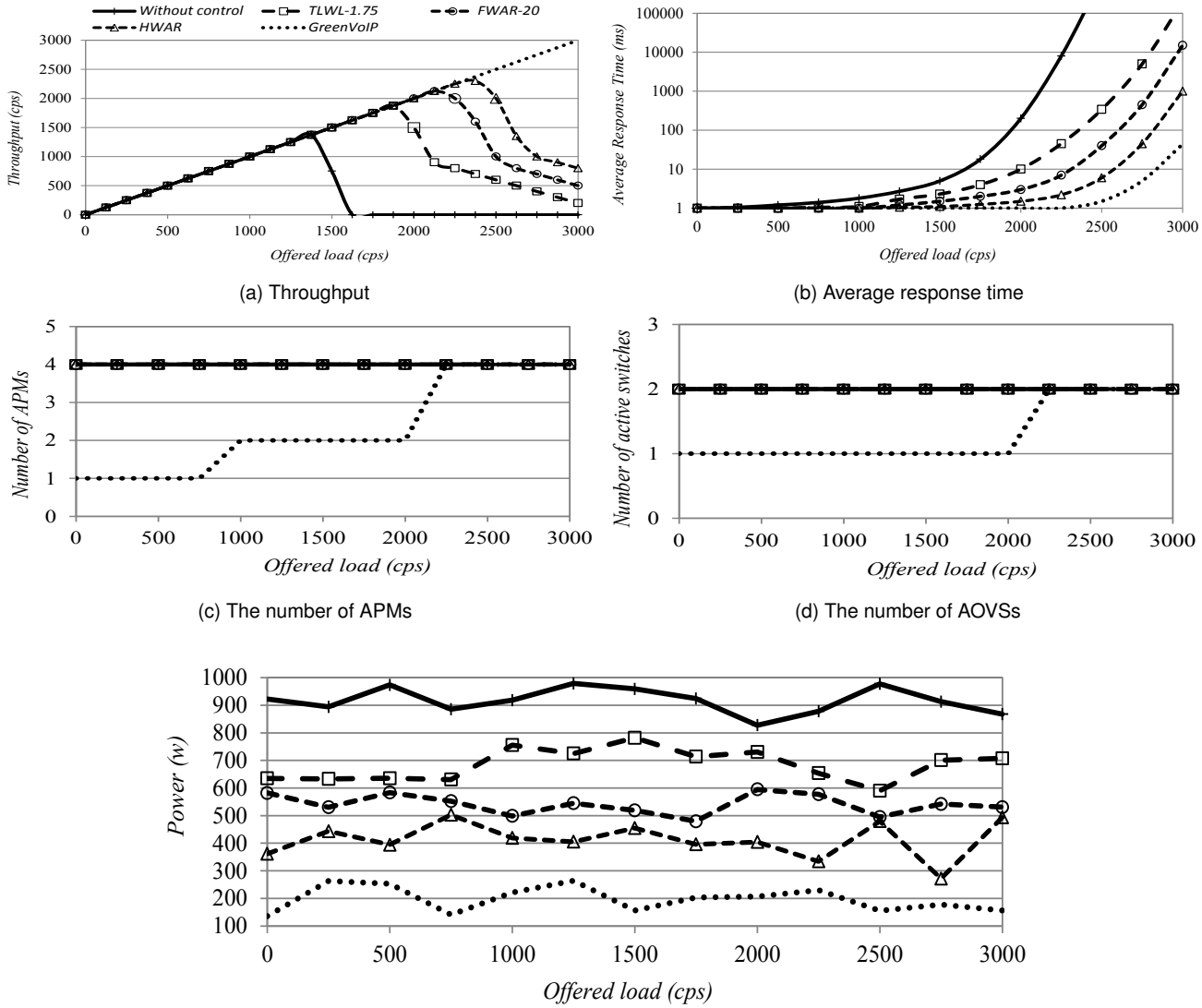
(b) Request retransmission rate

Figure 19. Overload effects.

amount of this history. It has an infinite memory capable of recording an immense number of response time values (keeps track of the whole history).

Fig. 20 illustrates the comparison results among these methods. As seen, the GreenVoIP throughput rises with the increase of the offered load. In contrast, the other methods encounter a throughput drop if the offered load passes a certain range of overload. For instance, TLWL faces overload when the offered load is about 2000 cps and, after that, there is a sharp throughput drop (Fig. 20a). Overload also results in a rise in response time. Fig. 20b indicates that the compared methods face an increase in the average response time by exceeding the overload threshold. On the other hand, the increase in average response time is minimum in GreenVoIP due to its use of intelligent resource management. Fig. 20c and 20d also indicate that GreenVoIP manages the amount of active equipment in the network to support green computing. As seen, GreenVoIP utilizes less equipment with a low offered load and increases the amount of active equipment when the offered load increases. However, in the other methods, all the equipment is active at all times. Fig. 20e shows the average power consumption of different methods. This figure acknowledges that GreenVoIP is less power consuming than other methods by managing the number of APMs. Similar results are obtained for AOVSs. We don't mention them anymore due to a lack of space. These results demonstrate GreenVoIP has been able to control both overload and power consumption in both servers and switches parts.

Table 3 compares the four methods for their CPU consumption of PMs and OVSs. This table provides an adequate view of load distribution among PMs using the four methods. According to this table, the APMs in GreenVoIP are always warm, while there is a great difference in PM CPU consumption in the other methods. For example, some PMs are saturated in a 2500 cps load, but some others are barely involved. In other words, some PMs are hot and some are cold, which is the reason for the lower throughput and higher response time in these methods. Also, the green computing algorithm in GreenVoIP successfully reduces the number of active equipment without damaging the throughput. This is due to GreenVoIP's integrated architecture which provides a global view of the entire network infrastructure and implements load balancing throughout the entire network so as to keep it warm and balanced. It is worth mentioning that similar results have been



(a) Throughput

(b) Average response time

(c) The number of APMs

(d) The number of AOVSs

(e) The average power consumed by APMs

Figure 20. Comparison between GreenVoIP and other methods.

obtained regarding memory consumption, but these will not be addressed here.

Previously presented results are related to servers and switches, but Table 4 plots the results of the GreenVoIP controller. As seen, the controller's throughput increases with the rise of the offered load, while its average response time is very low. Therefore, the GreenVoIP controller reaches high throughput with low delay. Also, the controller components (introduced in Sections 5.1 and 5.2) do not add any extra overhead to its resources, including CPU and memory, making the possibility of overload occurrence in the controller very low. This is evident in Table 4's controller CPU and memory consumption. The best results of GreenVoIP in Tables 3 and 4 are highlighted.

So, GreenVoIP has the following characteristics, which makes it different from other methods:

• Distributed throughout the network: If the switches are low-cost OpenFlow-enabled switches, then they can do load-balancing. It is therefore very scalable.

Table 3

Average CPU Consumption (%)

Load	500 (cps)				1500 (cps)				2500 (cps)			
Method	TLWL	FWAR	HWAR	GreenVoIP	TLWL	FWAR	HWAR	GreenVoIP	TLWL	FWAR	HWAR	GreenVoIP
PM1	22.41	24.09	22.34	30.54	44.34	34.34	31.75	39.24	67.74	56.34	**85.75**	49.74
PM2	18.36	9.83	31.05	0	19.64	25.46	39.74	37.53	25.95	**55.74**	52.47	48.95
PM3	8.64	21.30	29.75	0	43.47	34.46	34.84	0	**12.64**	82.03	**49.95**	45.75
PM4	37.06	30.36	17.99	0	10.02	24.44	39.53	0	**96.93**	**87.63**	85.58	41.85
OVS1	23.96	30.74	37.74	34.30	44.73	38.94	40.63	52.46	58.03	40.44	49.37	51.74
OVS2	40.35	19.74	23.54	0	55.28	29.02	39.74	0	64.04	55.85	51.02	54.65

Table 4

GreenVoIP Controller Performance

Offered load (cps)	500	1000	1500	2000	2500	3000
Throughput (fps) ∼	498	996	1457	1992	2494	2995
Average response time (ms) ∼	2.3	3.2	4.1	4.9	5.2	6.1
CPU consumption (%) ∼	12.48	17.75	24.87	31.65	39.65	46.55
Memory consumption (%) ∼	11.32	15.68	22.63	27.63	35.86	43.84

• Logically centralized: Load-balancing is defined in one place for the entire network - in the control plane app. It is easy to scale by replicating the control plane.

• Flexible: Each service in next generation SIP can have its own load-balancing algorithm. Each type of request can be load-balanced by a combination of oblivious and intelligent (flow-by-flow) schemes depending on the needs of the service, and the capabilities of the data-path and controller. The service creator is free to decide.

• Systematic: Acting according to an SDN-based system and implementing it in a real testbed. The results of performance appraisal in a real topology show that the evaluation criteria have improved significantly.

6.2.3 GreenVoIP Scalability

In order to test the scalability of GreenVoIP in more acute conditions, the number of PMs and SIP-VNFs must be increased. Therefore, we have emulated the GreenVoIP framework in *CloudSimSDN emulator*. In this case, a system is emulated with 50 OVSs, 100 PMs, and 500 SIP-VNFs, with the same characteristics mentioned in Section 6.1. Different loads are injected into the system in 100-second time periods, so that the offered load keeps increasing up to the time range of [500-600] sec, then it decreases and increases again at the range of [800-900] sec. The results are plotted in Table 5.

IEEE Transactions on Green Communications and Networking, DOI 10.1109/TGCN.2020.2982821

Table 5
GreenVoIP Scalability

Time (s)	0-100	100-200	200-300	300-400	400-500	500-600	600-700	700-800	800-900	900-1000
Offered load (cps)	3000	3500	4000	4500	5000	5500	4000	3000	6000	6500
Throughput (cps) $\sim$	2998	3496	3996	4495	4994	5493	3991	2990	5992	6494
Average response time (ms) $\sim$	15.09	18.76	23.65	27.75	32.54	41.01	25.12	19.01	50.59	62.37
The number of APMs	8	12	20	29	39	50	22	9	55	67
The number of SIP-VNFs	38	56	138	174	234	298	154	41	330	402
The number of AOVSs	4	6	11	15	20	26	10	4	26	33
Average number of migrations	5	7	10	14	19	24	11	6	30	35
Average decision time (ms) $\sim$	3.86	5.57	7.97	9.56	12.64	15.08	8.01	4.98	18.65	21.73

The number of APMs, SIP-VNFs, and AOVSs changes according to the offered load; that is to say, it follows the load pattern. Therefore, even with the highest load, the throughput is close to the offered load and the average response time does not reach 63 ms. In other words, GreenVoIP maintains a high throughput over time by adjusting the number of APMs and AOVSs even in very heavy traffic, such as 6500 cps. In addition, the number of migrations is low and grows almost linearly by enlarging the system size. In addition, the average decision time does not reach 22 ms even in the worst case, indicating that green computing and overload control algorithms have low complexities.

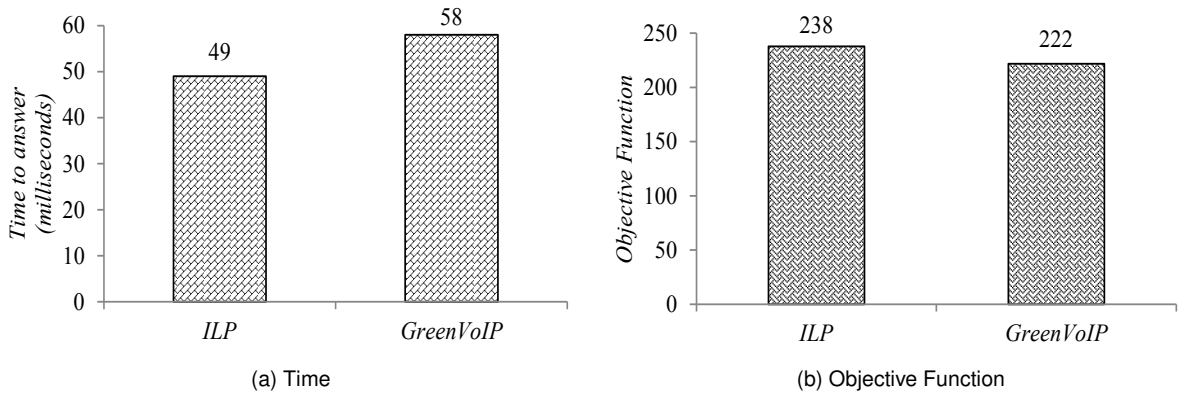
6.2.4 Complexity Analysis

As the joint load-balance and energy-aware routing problem in the VoIP communication network with limited server resources is an ILP problem and is, therefore, NP-hard and cannot be solved in polynomial time (Proposition 1), we can have a SDN-based algorithm (GreenVoIP) to solve the problem. In this regard, GreenVoIP decompose the problem into to subproblems and addressed them separately in polynomial time.

To compare them, the optimal solution (ILP Model 1) for limited cases (limited VoIP UA and server as sources and destinations) of the problem are compared with GreenVoIP. Note that given that the problem NP-hard, the optimal solution cannot be achieved in polynomial time and increasing the size of the problem exponentially increases the time of ILP Model 1.

In this experiment, l (the number of gateways) is 50 and z (the number of servers) is 10 which are connected. This way, the number of switches (v) and links (e) are 8 and 11. Moreover, W_p and R_p are 100 and 90 respectively. The remaining configuration is the same as that of section 6.1 (α, β, and γ are considered 0.65, 0.95 and 0.85, respectively. ρ and μ are chosen to be 30 and 0.8). 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 overfitting or creating too high a variance. The improved performance reveals which parameter settings are more favorable (tuned) or less favorable (untuned) . Finally, to solve ILP Model 1 we used CPLEX solver. Fig. 21 shows that GreenVoIP was able to closely resembles the result of ILP.



(a) Time (b) Objective Function

Figure 21. Objective function and time to answer of GreenVoIP and ILP

In Table 6, the solution value and time is provided for various inputs for the two methods. As is evident in this table, response time significantly increases for ILP method, whereas GreenVoIP has a slow growth. In fact GreenVoIP is a *heuristic* method to solve global load-balanced and energy-aware routing problem in a VoIP communication network.

Table 6

Comparison between GreenVoIP and ILP model 1

		$l=50, z=10$ $v=8, e=11$	$l=60, z=15$ $v=10, e=15$	$l=70, z=20$ $v=12, e=19$	$l=80, z=25$ $v=14, e=23$
Time to answer (ms)	ILP	95	755	4054	11465
	GreenVoIP	123	165	212	298
Objective Function ($\sum(f_p u_p)$)	ILP	199	345	456	576
	GreenVoIP	189	324	423	545

7 Conclusion and Future Work

VoIP systems are increasingly prevalent in modern life. Due to the rise in the number of VoIP-based apps being created, the advent of technologies such as 5G will encourage the trend towards mobile VoIP use as well. This has led to the foundation of large-scale VoIP call centers which consume huge amounts of electric power.

Most of the energy consumed in these centers is by servers and switches. This equipment also faces over-provisioning or under-provisioning in these centers, which, in themselves, cause problems such as overload or a lack of support of green computing. On the other hand, the emergence of SDN and NFV technologies have offered the possibility of the integrated management of network components.

The current study designed, implemented and evaluated a novel framework by the name of GreenVoIP, which is based on SDN and NFV technologies and employs the resource management approach for a trade-off between overload and energy in the VoIP network. In this regard, the SIP server functions are virtually presented by the name of SIP-VNFs. In addition, OpenFlow switches are used in the network infrastructure instead of traditional switches. As a result, the occurrence of overload is efficiently prevented and the amount of the active equipment in the network is minimized by the observance and management of SIP-VNFs and OpenFlow switches. GreenVoIP is evaluated in a real testbed, the results of which indicate that GreenVoIP can achieve optimal throughput and delay (QoS criteria) and minimize the number of active components to save energy, even in very heavy loads.

Future work includes a more detailed study on the parameters introduced in the present article, such as α, and the search for their optimum values. We also plan to study the effects of these parameters on each other. The θ and φ threshold values are important in the trade-off between overload occurrence and energy consumption. In the present study, the thresholds are considered to be fixed; however, in future work, intelligent Fuzzy methods will be presented to precisely determine these thresholds.

Note that although the proposed framework is designed for VoIP networks, it can also be used and evaluated for other types of network. Also, the future work regarding this article includes the design of an optimal mathematical model to determine the migration of SIP-VNFs. This model will include constraints, such as link delays or network Service-Level Agreement (SLA). We also intend to use machine learning to predict server loads.

REFERENCES

[1] Y. Amir, C. Danilov, S. Goose, D. Hedqvist, and A. Terzis, "An overlay architecture for high-quality voip streams," *IEEE Transactions on Multimedia*, vol. 8, no. 6, pp. 1250–1262, 2006.

[2] S. A. Baset and H. Schulzrinne, "Energy efficiency of voice-over-ip systems," 2010.

[3] M. Brandenburg. (2017) Growth opportunities in the voip access and sip trunking services market. [Online]. Available: http://www.frost.com/

[4] I. Ahmad and S. Ranka, *Handbook of Energy-Aware and Green Computing-Two Volume Set*. CRC Press, 2016.

[5] Y. Ran, J. Yang, S. Zhang, and H. Xi, "Dynamic iaas computing resource provisioning strategy with qos constraint," *IEEE Transactions on Services Computing*, vol. 10, no. 2, pp. 190–202, 2017.

[6] D. Cotroneo, L. De Simone, and R. Natella, "Nfv-bench: A dependability benchmark for network function virtualization systems," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 934–948, 2017.

[7] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[8] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2015.

[9] A. Bentaleb, A. C. Begen, R. Zimmermann, and S. Harous, "Sdnhas: An sdn-enabled architecture to optimize qoe in http adaptive streaming," *IEEE Transactions on Multimedia*, vol. 19, no. 10, pp. 2136–2151, 2017.

[10] H. E. Egilmez and A. M. Tekalp, "Distributed qos architectures for multimedia streaming over software defined networks," *IEEE Transactions on Multimedia*, vol. 16, no. 6, pp. 1597–1609, 2014.

[11] C. Assi, S. Ayoubi, N. El Khoury, and L. Qu, "Energy-aware mapping and scheduling of network flows with deadlines on vnfs," *IEEE Transactions on Green Communications and Networking*, vol. 3, no. 1, pp. 192–204, 2018.

[12] K. T. Bagci and A. M. Tekalp, "Dynamic resource allocation by batch optimization for value-added video services over sdn," *IEEE Transactions on Multimedia*, vol. 20, no. 11, pp. 3084–3096, 2018.

[13] C. Pham, N. H. Tran, S. Ren, W. Saad, and C. S. Hong, "Traffic-aware and energy-efficient vnf placement for service chaining: Joint sampling and matching approach," *IEEE Transactions on Services Computing*, 2017.

[14] A. Rakity. (2018) Ovum telecom research. [Online]. Available: https://ovum.informa.com/

[15] L. Sun, I.-H. Mkwawa, E. Jammeh, and E. Ifeachor, *Guide to voice and video over IP: for fixed and mobile networks*. Springer Science & Business Media, 2013.

[16] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[17] M. Coen. (2018) 5 must watch trends for enterprise voip in 2018. [Online]. Available: https://www.westuc.com/

[18] D. Bonfiglio, M. Mellia, M. Meo, and D. Rossi, "Detailed analysis of skype traffic," *IEEE Transactions on Multimedia*, vol. 11, no. 1, pp. 117–127, 2009.

[19] J. Mercier. (2018) Skype numerology. [Online]. Available: http://skypenumerology.blogspot.co.at/

[20] G. Shen and R. S. Tucker, "Energy-minimized design for ip over wdm networks," *Journal of Optical Communications and Networking*, vol. 1, no. 1, pp. 176–186, 2009.

[21] B. Goode, "Voice over internet protocol (voip)," *Proceedings of the IEEE*, vol. 90, no. 9, pp. 1495–1517, 2002.

[22] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "Sip: session initiation protocol," Tech. Rep., 2002.

[23] S. Agarwal, F. Malandrino, C. F. Chiasserini, and S. De, "Vnf placement and resource allocation for the support of vertical services in 5g networks," *IEEE/ACM Transactions on Networking*, vol. 27, no. 1, pp. 433–446, 2019.

[24] S. Agarwal, F. Malandrino, C.-F. Chiasserini, and S. De, "Joint vnf placement and cpu allocation in 5g," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 1943–1951.

[25] J. Elias, F. Martignon, S. Paris, and J. Wang, "Efficient orchestration mechanisms for congestion mitigation in nfv: Models and algorithms," *IEEE Transactions on Services Computing*, vol. 10, no. 4, pp. 534–546, 2017.

[26] F. Zhang, G. Liu, X. Fu, and R. Yahyapour, "A survey on virtual machine migration: Challenges, techniques and open issues," *IEEE Communications Surveys & Tutorials*, 2018.

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

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

[29] K. Singh and H. Schulzrinne, "Failover, load sharing and server architecture in sip telephony," *Computer Communications*, vol. 30, no. 5, pp. 927–942, 2007.

[30] L. De Cicco, G. Cofano, and S. Mascolo, "Local sip overload control: Controller design and optimization by extremum seeking," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 3, pp. 267–277, 2015.

[31] J. Wang, J. Liao, T. Li, J. Wang, J. Wang, and Q. Qi, "Probe-based end-to-end overload control for networks of sip servers," *Journal of Network and Computer Applications*, vol. 41, pp. 114–125, 2014.

[32] R. G. Garroppo, S. Giordano, S. Niccolini, and S. Spagna, "A prediction-based overload control algorithm for sip servers," *IEEE transactions on network and service management*, vol. 8, no. 1, pp. 39–51, 2011.

[33] Y. Hong, C. Huang, and J. Yan, "Modeling and simulation of sip tandem server with finite buffer," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 21, no. 2, p. 11, 2011.

[34] Y. Hong, C. Huang, and Yan, "Mitigating sip overload using a control-theoretic approach," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*. IEEE, 2010, pp. 1–5.

[35] C. Shen and H. Schulzrinne, "On tcp-based sip server overload control," in *Principles, Systems and Applications of IP Telecommunications*. ACM, 2010, pp. 71–83.

[36] J. Liao, J. Wang, T. Li, J. Wang, J. Wang, and X. Zhu, "A distributed end-to-end overload control mechanism for networks of sip servers," *Computer Networks*, vol. 56, no. 12, pp. 2847–2868, 2012.

[37] G. Mishra, S. Dharmaraja, and S. Kar, "Reducing session establishment delay using timed out packets in sip signaling network," *International Journal of Communication Systems*, vol. 29, no. 2, pp. 262–276, 2016.

[38] C. Shen, H. Schulzrinne, and E. Nahum, "Session initiation protocol (sip) server overload control: Design and evaluation," in *Principles, systems and applications of IP telecommunications. Services and security for next generation networks*. Springer, 2008, pp. 149–173.

[39] M. Homayouni, H. Nemati, V. Azhari, and A. Akbari, "Controlling overload in sip proxies: An adaptive window based approach using no explicit feedback," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*. IEEE, 2010, pp. 1–5.

[40] S. V. Azhari, M. Homayouni, H. Nemati, J. Enayatizadeh, and A. Akbari, "Overload control in sip networks using no explicit feedback: A window based approach," *Computer Communications*, vol. 35, no. 12, pp. 1472–1483, 2012.

[41] A. Abdelal and W. Matragi, "Signal-based overload control for sip servers," in *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE*. IEEE, 2010, pp. 1–7.

[42] E. Noel and C. R. Johnson, "Novel overload controls for sip networks," in *Teletraffic Congress, 2009. ITC 21 2009. 21st International*. IEEE, 2009, pp. 1–8.

[43] A. Montazerolghaem, M. H. Y. Moghaddam, and F. Tashtarian, "Overload control in sip networks: A heuristic approach based on mathematical optimization," in *Global Communications Conference (GLOBECOM), 2015 IEEE*. IEEE, 2015, pp. 1–6.

[44] A. Montazerolghaem, M. H. Yaghmaee, A. Leon-Garcia, M. Naghibzadeh, and F. Tashtarian, "A load-balanced call admission controller for ims cloud computing," *IEEE Transactions on Network and Service Management*, vol. 13, no. 4, pp. 806–822, 2016.

[45] A. Montazerolghaem, M. H. Y. Moghaddam, and A. Leon-Garcia, "Opensip: Toward software-defined sip networking," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 184–199, 2018.

[46] S. A. Baset, J. Reich, J. Janak, P. Kasparek, V. Misra, D. Rubenstein, and H. Schulzrinnne, "How green is ip-telephony?" in *Proceedings of the first ACM SIGCOMM workshop on Green networking*. ACM, 2010, pp. 77–84.

[47] Y. Agarwal, R. Chandra, A. Wolman, P. Bahl, K. Chin, and R. Gupta, "Wireless wakeups revisited: energy management for voip over wi-fi smartphones," in *Proceedings of the 5th international conference on Mobile systems, applications and services*. ACM, 2007, pp. 179–191.

[48] Y. Chen, N. Smavatkul, and S. Emeott, "Power management for voip over ieee 802.11 wlan," in *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE*, vol. 3. IEEE, 2004, pp. 1648–1653.

[49] V. Namboodiri and L. Gao, "Towards energy efficient voip over wireless lans," in *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2008, pp. 169–178.

[50] J. Kotwicki, "An analysis of energy-efficient voice over ip communication in wireless networks," Ph.D. dissertation, Citeseer, 2004.

[51] K.-W. Chin, "On maximizing voip capacity and energy conservation in multi-rate wlans," *IEEE Communications Letters*, vol. 14, no. 7, pp. 611–613, 2010.

[52] C. Kasten and G. Zhou, "Typed voip silence prediction for smartphone energy savings," *Wireless personal communications*, vol. 79, no. 3, pp. 1959–1973, 2014.

[53] A. J. Pyles, Z. Ren, G. Zhou, and X. Liu, "Sifi: exploiting voip silence for wifi energy savings insmart phones," in *Proceedings of the 13th international conference on Ubiquitous computing*. ACM, 2011, pp. 325–334.

[54] L. Zhang, S. Tang, and S. Zhu, "An energy efficient authenticated key agreement protocol for sip-based green voip networks," *Journal of Network and Computer Applications*, vol. 59, pp. 126–133, 2016.

[55] A. J. Estepa, J. M. Vozmediano, J. López, and R. M. Estepa, "Impact of voip codecs on the energy consumption of portable devices," in *Proceedings of the 6th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*. ACM, 2011, pp. 123–130.

[56] H. Wu, Q. Wang, and K. Wolter, "Tradeoff between performance improvement and energy saving in mobile cloud offloading systems," in *2013 IEEE international conference on communications workshops (ICC)*. IEEE, 2013, pp. 728–732.

[57] H. Wu, Y. Sun, and K. Wolter, "Energy-efficient decision making for mobile cloud offloading," *IEEE Transactions on Cloud Computing*, 2018.

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

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

[60] J. Son, A. V. Dastjerdi, R. N. Calheiros, X. Ji, Y. Yoon, and R. Buyya, "Cloudsimsdn: Modeling and simulation of software-defined cloud data centers," in *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on*. IEEE, 2015, pp. 475–484.

[61] A. Y. Nikravesh, S. A. Ajila, and C.-H. Lung, "Measuring prediction sensitivity of a cloud auto-scaling system," in *2014 IEEE 38th International Computer Software and Applications Conference Workshops*. IEEE, 2014, pp. 690–695.

[62] R. G. Garroppo, S. Giordano, M. Pagano, and G. Procissi, "On traffic prediction for resource allocation: A chebyshev bound based allocation scheme," *Computer Communications*, vol. 31, no. 16, pp. 3741–3751, 2008.

8 APPENDIX

We focus on proactive resource management that is based on predictions of future workload ($\widehat{c}^k_{\tau+1}$, $\widehat{m}^k_{\tau+1}$) based on the past workload ($X^k_\tau(CPU)$, $X^k_\tau(Memory)$). The most important method in this area is time-series analysis. Time-series analysis could be used to detect repeating patterns in the workload or to estimate future values for resource allocation. Consequently, the scaling action is done in advance. In order to apply time-series analysis in resource management or load balancing domains, a certain performance metric will be periodically sampled at fixed intervals (c^k_τ, m^k_τ). The result will be a time-series containing a sequence of last observations. Time-series methods extrapolate this sequence to predict future values. Some of the methods used for this purpose are Moving Average, Autoregression, Autoregressive-Moving-Average (ARMA), exponential smoothing, and machine learning techniques [61].

We utilize the work of [32], [62] that proposes a comparative analysis among prediction techniques, used for developing a dynamic prediction based the resource allocation strategy. Among these techniques, the

Normalized Least Mean Square (NLMS) predictor is the one providing the best trade-off between complexity, accuracy, and responsiveness.

We customized general NLMS formulation for our problem (Eq. (14) to (20)). The general problem of prediction can be stated as follows:

Given a set of observations of a stochastic process $x(n)$, generate an estimation $\widehat{x}(n+k)$ of the value $x(n+k)$ that the process x will assume k steps ahead. Given a vector of ρ observations, $X = [x(n), x(n-1), x(n-\rho+1)]$, the predicted value $\widehat{x}$ is obtained by $\widehat{x} = \psi(X)$ where the function ψ is called the predictor. Various categories of predictors have been studied; however, considering the constraint on the complexity, the linear prediction category is the best suited for our aim. A linear prediction happens whenever the function $\psi(X)$ is linear. Putting it differently, the problem is to specify the impulse response $h(n)$ of the linear filter h such that:

$$\widehat{x}(n+k) = x(n) \otimes h(n) = \sum_{i=0}^{\rho-1} h(i)x(n-i) \tag{25}$$

The filters coefiňĄcients can be determined according to arbitrary optimality criteria. One of the widely adopted prediction technique is the so called Linear Minimum Mean Square Error (LMMSE) predictor, in which the values $h(n)$ are derived by minimizing the Mean Square Error of prediction:

$$\mathbb{E}[\epsilon^2(n)] = \mathbb{E}[(x(n+k) - \widehat{x}(n+k))^2] \tag{26}$$

The problem of this predictor is that the derivation of the LMMSE filter needs the knowledge of at least ρ values of the autocorrelation function of the stochastic process $x(n)$ and the inversion of a $\rho \times \rho$ matrix. These facts make LMMSE inappropriate for being used as on-line predictive method. Therefore, we consider the NLMS method, which is based on an adaptive mechanism. It does not require previous knowledge of the autocorrelation structure of the stochastic sequence [59]. The algorithm scheme is shown in Fig. 22.

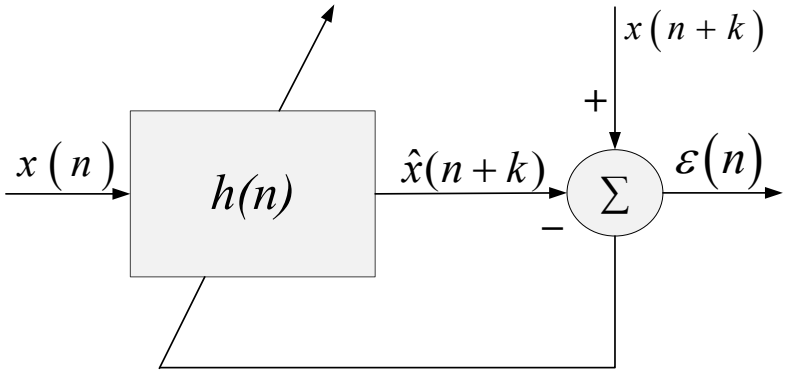


Figure 22. NLMS method

The filter coefiňĄcients are time varying and are tuned on the basis of the feedback information carried by the error $\epsilon(n)$. We define the vector of filter coefiňĄcients at time n with h_n. The values of h adapt dynamically

in order to decrease the Mean Square Error. Note that $\epsilon(n) = x(n+k) - \widehat{x}(n+k)$ and $X(n) = [x(n), x(n-1), ..., x(n-\rho+1)]$. The NLMS works as follows: Initialize the coefficient $h(0)$; and for each new data, update the filter $h(n)$ according to the recursive equation:

$$h(n+1) = h(n) + \mu \frac{\epsilon(n)X(n)}{\| X(n) \|^2} \tag{27}$$

where $\| x(n) \|^2 = X(n)X^T(n)$ and μ is a fixed parameter called step size. Pursuant to [59], NLMS converges so long as $0 < \mu < 2$. At time n, the values $x(n+k)$, hence $\epsilon(n)$, are not known. Therefore, the value $\epsilon(n-k)$ is used instead and the one step NLMS predictor update equation becomes:

$$h(n+1) = h(n) + \mu \frac{\varepsilon(n-1)X(n-1)}{\| X(n-1) \|^2} \tag{28}$$

Algorithm 3 presents the NLMS algorithm formulation for a general system, where $X(n)$ and $\widehat{x}(n+1)$ are the input and output of the predictor, respectively.

Algorithm 3: NLMS for Load Estimation

Parameters: ρ =filter order, μ =step size
Initialize: $h_0 = 0$
for $i = 1, 2, ...$ **do**
    $X(n) = [x(n), x(n-1), ..., x(n-\rho+1)]$
    $h(n+1) = h(n) + \mu \frac{\varepsilon(n-1)X(n-1)}{\|X(n-1)\|^2}$
    $\| X(n) \|^2 = X(n)X^T(n)$
    $\epsilon(n) = x(n+k) - \widehat{x}(n+k)$
    $\widehat{x}(n+1) = h(n)X^T(n)$
end

The NLMS predictor needs the configuration of two parameters: the filter order ρ and the step size μ. The filter order ρ has been chosen to be 30, since it produced a good performance based on the results analysis. In the case of μ, it is relevant to note that one of the important advantages of using NLMS is that it is less sensitive to the step size compared with other linear predictors. In our experiments, we chose the step value of 0.8 as a trade-off between fast convergence and responsiveness to input change. We run various tests to tune this parameter. For small value of μ, we apperceive that the prediction function slowly converges and is unable to follow sudden workload increase. For values of μ greater than 0.5, the prediction function is not sensitive to step size and results in a quicker response to workload changes.