Real-Time Optimized NFV Architecture for Internetworking WebRTC and IMS

Duong Tuan Nguyen, Kim Khoa Nguyen, Saida Khazri, Mohamed Cheriet
Ecole de technologie Superieure, University of Quebec
Montreal, Quebec, Canada, H3C 1K3
duong-tuan.nguyen.1@ens.etsmtl.ca,{knguyen,saida.khazri}@synchromedia.ca,mohamed.cheriet@etsmtl.ca

***Abstract*—Network Function Virtualization (NFV) technology has emerged as a promising solution to optimize the deployment of network elements in cloud computing environment, both in terms of user Quality of Service (QoS) and resource allocation. To deliver IP Multimedia Subsystem (IMS) advanced services across multiple access networks, a cloud-based model likely improves not only flexibility in network management but also in service invocation. This model is particularly suitable for the inter-network between the IMS and Web Real-Time Communication (WebRTC) domain, which is a natural combination to significantly expand potential end-points of multimedia sessions. However, optimally allocating resource for each component in the system to provide QoS is challenging, and has not been fully considered in prior work. In this paper, we investigate an NFV architecture to enable effective communication between IMS and WebRTC users supporting smart community services, and propose an optimization model to design and allocate resources for such system while ensuring desired QoS level. Experimental results show our algorithms effectively allocate required amounts of resources for virtual components according to real-time increase or decrease in user demand.**

I. Introduction

Nowadays, to accommodate significant traffic growth, network operators are facing many issues, from space to locate new physical servers or network equipment to increasing costs of energy. They have also taken into account the scarcity of skills necessary to design, operate and manage such complex hardware-based infrastructure. To deal with these issues, a novel paradigm, called Network Function Virtualization (NFV) [1] is introduced. It enables network services to be provisioned via software-based network functions and elements (i.e. bridge, router, load balancer, firewall, etc.) sitting on top of general-purpose servers instead of specialized hardware.

From Telco's perspective, existing IP Multimedia Subsystem (IMS) systems [2] suffer from limitations in deployment and performance. As the number of IMS subscribers increases, such complex systems require more resources, i.e. Call Session Control Function (CSCF) servers, media gateways, application servers. As a result, the optimal deployment and effective management of those systems impose new challenges. Moreover, to provide ubiquitous access to multimedia services on mobile and wireless access networks, an IMS system is typically integrated with a variety of new communication services such as Rich Communication Suite (RCS), Internet Protocol television (IPTV), wireless sensor network, peer-to-peer overlay network, and WebRTC [3]. Such a collection of hardware and software components that are coupled with each other becomes more resistant to any evolution from both application and network layers. An innovation in service layer requires operators to upgrade or change the underlying hardware. Similarly, updating software requires the enhancement of physical entities. These challenges make NFV technology become a promising solution for IMS providers.

We address in this paper the problem of WebRTC $\leftrightarrow$ IMS interworking to provide Telco-based smart community services. WebRTC, a new IETF and W3C standard [3], is recently an emerging solution for developing real-time communication in Web environment. It enables multimedia interaction capabilities in web browsers on any platform without installing a specific plug-in or using a third-party application. In addition, as mobile terminals increase in quantity and the ability of accessing the Internet almost anywhere, employing WebRTC may help IMS operators in dramatically expanding the number of potential end-points for multimedia sessions. In turn, WebRTC can benefit from IMS's layered architecture to grow independently on underlying network infrastructure. While the traditional model based on physical entities imposes technical challenges for the WebRTC $\leftrightarrow$ IMS integration, an NFV model requires new modeling and optimization techniques for optimizing design, resource allocation, and providing QoS.

Significant progress has been recently made in proposing WebRTC $\leftrightarrow$ IMS models, for example, the Third Generation Partnership Project (3GPP) effort on modifications of the IMS architecture enables WebRTC-based clients to access IMS [4]. The factor that distinguishes our work from previous literature is an entire interworking system based on NFV paradigm, and a new modeling and optimization algorithms that help design such system with minimal resource consumption.

The major contributions of this article include:

- An NFV architecture to bring WebRTC service to the IMS domain. In our design, the installation and deployment of both of IMS and WebRTC-related entities are achieved using the NFV technology.
- A multi-access communication architecture that allows WebRTC users using several network adapters (i.e. Ethernet, Wi-Fi, Cellular) alternatively to establish connection to remote IMS domains in an efficient way.
- An analytical model to calculate both communication performance and resource requirements of the system. We

also implement a prototype and measure performance parameters as inputs to the model and evaluate its accuracy.
- Two algorithms to optimally allocate virtual resources and communication channels in multi-access environment.

The rest of the paper is organized as follows. Section II reviews the NFV concept, WebRTC protocol and IMS system. Section III proposes an NFV-based interworking architecture and system interactions in login, calling and network access adaptation phases. System modeling for both compute and network resources and optimal resource allocation algorithms are discussed in section IV and V respectively. Section VI provides an experimental analysis of CPU usage as well as login and calling session delays. Finally, the conclusion is drawn, and we present our future work.

II. Background

This section presents the NFV technology and an overview of previous work related to the virtual IMS (vIMS), followed by a brief explanation of WebRTC $\leftrightarrow$ IMS and related studies.

A. Network Function Virtualization and virtual IMS

NFV was firstly introduced in [1] as a means to overcome the barriers of typically using real proprietary appliances. Accordingly, in a network virtualization environment, many kinds of network services are consolidated to co-exist over the same physically dedicated servers. Several effort in the NFV area has been proposed by organizations, i.e. ETSI [5], IRTF [6] and IETF [7].

Migrating the IMS system to an NFV-based cloud platform allows service providers to launch new services with agility and easily scale capacities to fulfill customer requirements. Such a vIMS solution enables both signaling and data transferring functions to run on a common virtualization platform built on commodity servers, while securing carrier-grade reliability and stability. As a result, IMS environment is optimized by flexibly deploying any combination of separated IMS functions so that resources can be aligned themselves with service demands.

Several studies have been proposed to adapt NFV to IMS framework. Three architectural concepts for the implementation of cloud-based IMS platform, namely virtualized-IMS (vIMS), split-IMS, and merge-IMS, along with a management architecture were explained in [8]. Another research [9] describes a high availability solution to avoid performance degradation through live migration of IMS virtual instances. In [10], a service-specific virtual approach is proposed to reduce a huge amount of the signaling processing load in both Evolved Packet Core (ePC) and IMS of 5G mobile communication system. The idea is to create virtual networks responsible for each service and manage these mappings via a service binding function. Deploying IMS onto virtual machines (VMs) is also taken into account in [11] with hardware-assisted virtualization technology supports. The test results show the virtualization outperforms a bare-metal solution in terms of response delays. In [12], the performance of a virtualized OpenIMS core with an integrated horizontally scalable HSS database is also evaluated with respect to processing delay of different types of messages and scenarios. While all of the mentioned works have dealt with performance metrics like CPU usage, memory usage, session delay, they have not considered the dynamicity of service requests and optimal resource allocation. Dynamic method in [13] aims at addressing the problem of resource allocation; however it uses a simple static usage threshold which is unable to handle the fluctuation of application needs. In fact, unlike the traditional environment where adding extra resource (i.e. CPU, RAM) is not easily done without service interruption, the virtualization paradigm is naturally scalable. On the other hand, dynamic allocation virtual resource generates new parameters which need to be optimized, i.e. the number of CPUs that can be provided in real-time, or the adaption of network access.

B. WebRTC and IMS Interworking

Designed to rapidly migrate multimedia services from desktop applications to Web environment, WebRTC has received a great attention from carriers as a means to attract more subscribers. Nevertheless, many previous attempts to bringing WebRTC to IMS world are still facing challenges. Firstly, IMS multimedia services are SIP-based whereas WebRTC does not specify any particular signaling model other than the offer/answer mechanism [14], and recently Session Description Protocol [15]. This makes the IMS system, which is inherently complicate with numerous nodes with extensive functionality, more complex as well as raises new scalability issues. Secondly, although WebRTC had a large number of standardized documents, its maturity is still not comparable to IMS. Thus, providers who plan to launch WebRTC service in its infant stage may experience poor resource estimation which likely results in over-provisioning or bottle-neck problem at the server side.

Several WebRTC $\leftrightarrow$ IMS interworking implementations have been done so far in academic community, i.e. [16]–[18]. To the best of our knowledge, most of these efforts have been devoted to implementation architecture. Stochastic models or dynamic resource allocation for such system are not fully considered. In particular, adopting NFV paradigm to design the WebRTC-IMS interworking system has never been presented. Such an NFV-based interworking system may accelerate the deployment of new WebRTC services for IMS providers. Moreover, the flexibility of virtual environment also helps validate system models and perform integration tests before going to production.

III. NFV-Based Internetworking Architecture

In this section, we describe an overview of the proposed NFV architecture allowing users to access multimedia service via an Internet connection and use WebRTC-capable browsers to communicate with IMS subscribers.

A. Overview of Interworking Architecture

As illustrated in Fig. 1, the system is composed of servers logically located in WebRTC and IMS domains. These servers

are VM instances deployed onto cloud data center. These VMs are created by a hypervisor on a physical host. The two domains may be managed by one or many operators. IMS subscribers (IMS UE) access vIMS system which is composed of Proxy-CSCF (P-CSCF), Interrogating-CSCF (I-CSCF), Serving-CSCF (S-CSCF), and Home subscriber server (HSS). WebRTC users (WUs) can connect to the virtual gateway (vGateway or vGW) and other servers such as Web server and STUN/TURN via the Internet. This architecture allows the provider to respond to new traffic demand by quickly adding capacity if required or easily and cleanly decommission it if unnecessary. Furthermore, innovations in WebRTC or IMS can be adopted independently on computing hardware, as a result, it may offer a better service to clients with a significant reduction in capital expenditure (CAPEX).

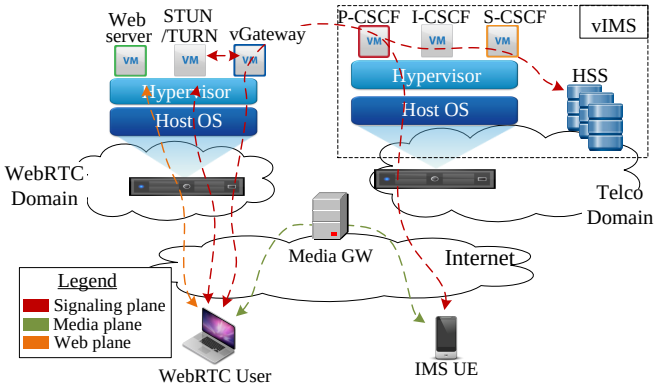

Fig. 1. The NFV architecture of WebRTC ↔ IMS interworking system

The multimedia communication service involves the IMS UE and the WUs. Signaling messages are transmitted from clients to vIMS or vGateway whereas media packets are relayed through intermediate entities between the two domains. This relay function can reside in the vGateway or in separated elements like media gateway (Media GW).

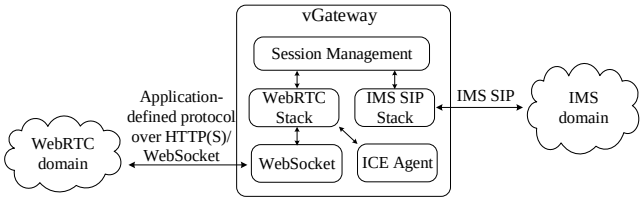

Fig. 2. Functional components of the vGateway

At the WebRTC side, the Web server processes normal HTTP requests while the STUN/TURN server is responsible to provide NAT traversal for connection between WU and IMS UE in accordance with Interactive Connectivity Establishment (ICE) procedure. The vGateway (Fig. 2) plays two roles, as a WebRTC endpoint and an IMS client. The integration of WebRTC and IMS-related protocols, namely WebRTC stack, ICE agent, WebSocket and IMS SIP stack in the vGateway bridges two domains. In addition, the Session Management component allows the vGateway to receive WebRTC messages from the WU, interpret them into corresponding messages before pushing them toward the vIMS. Similarly, vIMS components accept messages coming from the IMS UE, translate them into WebRTC signaling messages and transmit to the WU. There are several solutions to manage user information from two sides, depending on service business model. A detail discussion on this aspect is out of the scope of this paper.

B. Basic Messages Flow

This subsection discusses message flows for the WU to login and to start an interworking calling session with the IMS UE. The reverse direction is similar. A mechanism that allows WU to benefit from a device with multiple network interfaces is also explained.

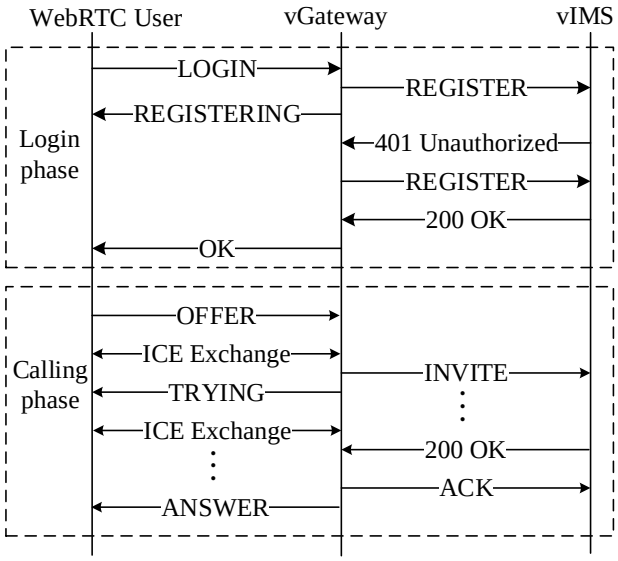

Fig. 3. Message flows of login and calling phases

1) Login phase: This phase Fig. 3 is similar to the registration phase in the IMS world: after logging in, a WU can launch other media-related sessions within a given period of time or it has to continuously send a keepalive message to maintain the session. In reality, for security purpose, additional messages (i.e. authorization, authentication) may be involved. However, such process is out of scope of this work and therefore not depicted in Fig. 3.

After succesfully establishing a two-way communication on WebSocket connection with the vGateway, the WU can submit a LOGIN request that contains required identification information (i.e. username, password, P-CSCF address, digest authentication scheme, etc.) to vGateway to authorize with vIMS system.

Upon reception of the LOGIN request from the WU, the vGateway will act as an IMS client that registers with the vIMS, and in parallel, notifies the WU about the pending status (after transmitting SIP REGISTRATION request). If the

registration is successful with a 200 OK message from P-CSCF, the vGateway will inform the WU that the login phase is complete. Otherwise, a failure response will be returned.

2) Calling phase: In this phase Fig. 3, an audio/video interaction between the WU and the IMS UE are set up. To establish the media plane, the WU and vGateway need to exchange OFFER and ANSWER messages as session descriptions that specify parameters to indicate what to transmit to the remote side, as well as how to handle the media that is received.

Firstly, the WU sends an initial OFFER which contains basic information, i.e. a list of media codecs, and IP addresses that it is willing to receive or able to process. By enabling trickle ICE [19], the WU then gathers and provides more candidates to the vGateway. Similarly to the login phase, the vGateway extracts the session description from the WU to generate a SIP INVITE message to vIMS in parallel with performing ICE procedure with the WU. It informs the WU about any change in session status by messages from P-CSCF (i.e. 100 TRYING, 180 RINGING) and keeps receiving ICE candidates from the WU until the call is accepted by the IMS UE.

This phase is completed when the vGateway responds SIP ACK to SIP 200 OK from P-CSCF and answers the WU with the final session description.

3) Network access adaptation phase: Scenarios in which WebRTC endpoint may have multiple available network interfaces available have been discussed in some drafts of RTCWeb Working Group, i.e. [20], [21], [22]. In essential, the keepalive message is leveraged as a tool to detect failures of connection to vGateway (and other servers) and to force the routing mechanism to automatically switch to alternative interfaces. By doing at the application level, our approach is transparent to any change in WebRTC specification. For this hand-over process, two states are taken into consideration, namely whether it is logged in and whether it is in calling session.

For the first state, when the connection is back from other interface, the WU automatically re-sends its information to the vGateway to renew its LOGIN state. In the IMS side, the vGateway is not required to make any update to current REGISTERED session.

The second state starts after the first state and includes an orchestration performed by both WU and vGateway to renew the signaling session and to create a new media plane. Firstly, the vGateway maintains connection with IMS UE within an interval of time until the path to the WU is restored. Once the login session is successfully re-created, the WU performs the calling session again with less information than the precedent since some are already cached at the vGateway. After that, the vGateway combines the new information with its cache data before carrying out another calling session with IMS UE.

IV. System Modeling

This section presents the modeling of the entire system with respect to computing servers and network based on queueing models and optimization techniques.

A. Server Model

We consider the workload of the system's control plane based on the queueing models. The entire interworking system is represented as an open feed-forward network as shown in Fig. 4. Each incoming message after being processed by a server, will join and hence be the input of a next server, which corresponds to the SIP message flow processing (Fig. 3). This flow continues until the vGateway responses the session's status (success or fail) to the WU.

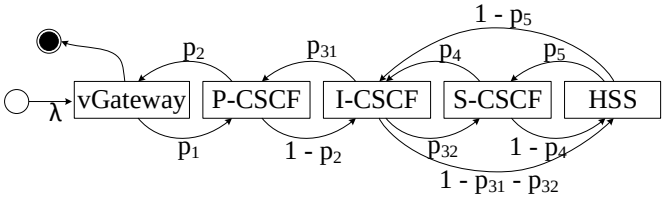

Fig. 4. Tandem queue model for interworking system

We consider two types of queues for each server of the system. Since most of modern computers can share its resources among the jobs by exploiting multiple-threaded mechanism to allow more than one job in service at the same time, a M/G/1 queue with a Processor Sharing service discipline is applied for this situation. M and G refer to message arrival according to a Poisson process with rate λ and the service distribution is arbitrary having rate μ and the 1 refers to a single server. The second queue is G/G/1 which is characterized by arbitrary arrival process and general distribution of service times. The primary purpose of this second model is to verify our experimental result in general case. In both queuing models, we assume network condition is stable, and has no direct impact on the message receiving rate of each server.

System performance is evaluated via two metrics: 1) the CPU utilization or traffic intensity ρ_i, $i = 1, \ldots, 5$ which is defined as the mean of arrivals per mean service time at servers vGateway, P-CSCF, I-CSCF, S-CSCF, and HSS respectively, and 2) the delay time that the WU has to wait until the successful completion of either login d_{login} (lasting from LOGIN message to the final OK message in Fig. 3) or calling session $d_{calling}$ (between OFFER message and TRYING message in Fig. 3). Their values are calculated as the weighted sum of the sojourn time of a message at each server W_i. More specifically, CPU utilization at i^{th} server and the total delay for a session are thus given by:

$$\rho_i = \frac{\lambda_i}{\mu_i} \tag{1}$$

$$d_{login} = \sum_{i=1}^{5} w_i W_i \tag{2}$$

$$d_{calling} = c w_1 W_1 \tag{3}$$

respectively, where w_i is the number of messages a server deals within a session, λ_i, and μ_i are the mean of total arrival rates and service rates. Note that μ_i (requests per millisecond) is also the server capacity to render service.

Let λ denote the initial arrival rate to the system; let $p_k, k \in \{1, 2, 31, 32, 4, 5\}$ denote the probability with which a message at a server is transited to another one. These transition probabilities vary depending on the session and enable the application of our model to both login and calling phases. To obtain the results in (1,2,3), W_i is calculated in terms of pre-defined parameters, i.e. λ, μ_i, or p_k.

1) M/G/1 based Processor-Sharing Server Model: According to [23], the departure process in an M/G/1 queueing system in which service discipline is processor-sharing is also a Poisson process with the same rate as that of the arrival process. Therefore, λ_i can be obtained as the solution of:

$$\lambda_1 = \lambda + p_2\lambda_2 \tag{4}$$
$$\lambda_2 = p_1\lambda_1 + p_{31}\lambda_3 \tag{5}$$
$$\lambda_3 = (1-p_2)\lambda_2 + p_4\lambda_4 + (1-p_5)\lambda_5 \tag{6}$$
$$\lambda_4 = p_{32}\lambda_3 + p_5\lambda_5 \tag{7}$$
$$\lambda_5 = (1-p_4)\lambda_4 + (1-p_{31}-p_{32})\lambda_3 \tag{8}$$

Solving in terms of λ yields:

$$\lambda_1 = \frac{\lambda}{1-p_1} \tag{9}$$
$$\lambda_2 = \frac{p_1\lambda}{p_2(1-p_1)} \tag{10}$$
$$\lambda_3 = \frac{p_1(1-p_2)\lambda}{p_2p_{31}(1-p_1)} \tag{11}$$
$$\lambda_4 = \frac{p_1(1-p_2)(p_{32}+p_5-p_5p_{31}-p_5p_{32})\lambda}{p_2p_{31}(1-p_1)(1-p_5+p_4p_5)} \tag{12}$$
$$\lambda_5 = \frac{p_1(1-p_2)(1-p_{31}-p_{32}p_4)\lambda}{p_2p_{31}(1-p_1)(1-p_5+p_4p_5)} \tag{13}$$

By substituting (9), (10), (11), (12), (13) into (1), we obtain the CPU utilization at each server.

The mean sojourn time a message spends in such M/G/1-PS server can obtained using prior work [24]:

$$W_i = \frac{1}{\mu_i - \lambda_i} \tag{14}$$

2) G/G/1 based Server Model: Let $A_i(.)$ and $\sigma^2_{A_i}$ denote the distribution function and variance of the inter-arrival times as i.i.d random variables. Similarly, let $B_i(.)$ and $\sigma^2_{B_i}$ the same notations for the service times also as i.i.d random variables. The boundaries of W_i for the G/G/1 system can be obtained from any previous standard work [25] as follows:

$$\frac{\rho_i^2 + \lambda_i^2\sigma_B^2 - 2\rho_i}{2\lambda_i(1-\rho_i)} \leqslant E[W_i] \leqslant \frac{\lambda_i(\sigma^2_{A_i} + \sigma^2_{B_i})}{2(1-\rho_i)} \tag{15}$$

B. Network Access Cost Model

In order to build an algorithm that optimally selects network access interface, we assume s, n as service type (e.g. calling, content sharing, live video streaming) and network interface respectively. Let $p^n_{s,i}$ denote the cost in the i^{th} parameter to carry out service s on the interface n. Let $w_{s,i}$ indicate the weight assigned to the i^{th} parameter to perform the service s. We consider the cost of each network interface. The definition of f^n_s for the service s and the interface n is given by following formula:

$$f^n_s = \sum_s \sum_i w_{s,i}\mathcal{N}(p^n_{s,i}) \tag{16}$$

where $\sum_i w_{s,i} = 1$. $\mathcal{N}(p^n_{s,i})$ is the normalized $p^n_{s,i}$ function. The optimal network interface is the one with the lowest value of f^n_s.

For example, if only monetary cost and bandwidth are considered with the normalized function of natural logarithm, the cost function can be calculated as:

$$f^n_s = w_b ln(\frac{1}{B_n}) + w_c ln(C_n)$$

where B_n and C_n represent the bandwidth and monetary cost for using network n.

Other parameters can be power consumption of device, session/packet delay, packet loss, jitter.

V. Optimal Resource Allocation Algorithm

This section introduces two algorithms to optimally allocate computing resource and network resource.

In the M/G/1-PS model, there are several parameters that need to be considered. We aim to determine not only the initial arrival rate λ and service rate μ but also the server's processing capacity in terms of the number of CPUs in real-time. From (14), the total average delay session is inversely proportional to server processing capacity. In other words, by adapting the resource which becomes accessible thanks to virtualization technology optimally allocating virtual resources, the system can elastically meet the dynamic requirements of traffic load.

Algorithm 1 optimizes compute resource allocation for WebRTC gateway and each of IMS servers by leveraging historical traffic data, the arrival rate and service rate, which are assigned to default values in the beginning, are continuously updated in real-time manner. The number of incoming messages starts at step 2. The loop at step 3 indicates that packets are incrementally collected in order to calculate inter-arrival time between two consecutive messages (step 4) and service time (step 5). When the number of received messages reaches to N (step 6), λ and μ are re-calculated. These parameters are updated in step 7 as the property of Poisson process and step 8 as the average of observed values. Then we compute the actual CPU utilization and the service time from λ, μ and the current number of CPUs. Note that the functions F and G can determined from (1) and (14). If either CPU usage or service time is greater or equal to the corresponding threshold which represent the QoS level of metrics, the number of CPUs will be increased (step 12), otherwise decreased (step 13) before resetting the counter (step 16).

The network selection procedure (Algorithm 2) returns an optimal network access with regards to bandwidth and monetary cost. Initially, there is no interface opted, thus the value $optimal\_access$ is NULL (step 2). At step 4, the total cost function of each network interface f^n is computed using (16).

Input : $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$: arrival rate and service time
MC: maximum of CPU usage,
MW: maximum of allowed total waiting time
N: the number at which $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are updated.

1. **begin**
2. INITIALIZE $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$;
3. START counting incoming packet with a vCPU;
4. **while** *receiving HTTP(S)/SIP-related packet* **do**
5. COMPUTE inter-arrival time between this packet and the precedent;
6. COMPUTE service time as the difference of this arrival time and departure time of next on-the-fly packet;
7. **if** *packet is the* N^{th} *one* **then**
8. UPDATE $\boldsymbol{\lambda}$ to the reciprocal of the average of inter-arrival times;
9. UPDATE $\boldsymbol{\mu}$ as the mean of all service times;
10. COMPUTE $CPU\_utilization$ as the function F of new $\boldsymbol{\lambda}$, new $\boldsymbol{\mu}$, and number of CPUs using (1);
11. COMPUTE mean $Service\_time$ as the function G of new $\boldsymbol{\lambda}$, new $\boldsymbol{\mu}$, using (14);
12. **if** $\boldsymbol{MC} \leqslant CPU\_utilization$ **or** $\boldsymbol{MW} \leqslant Service\_time$ **then**
13. INCREASE number of CPUs;
14. **else**
15. DECREASE number of CPUs;
16. **end**
17. RESET counter;
18. **end**
19. **end**
20. **end**

Algorithm 1: Optimal compute resource allocation for vGateway and CSCF servers

Input : S: set of service types s
N: set of network interfaces n
f_s^n: cost function for service s and interface n
Δ: timeout after which network access costs are re-computed

1. **begin**
2. **while** *TRUE* **do**
3. **foreach** $service\ s\ in\ S$ **do**
4. SET $optimal\_interface$ as NULL ;
5. **foreach** $interface\ n\ in\ N$ **do**
6. COMPUTE f_s^n using (16);
7. **if** $f_s^n \leqslant optimal\_cost\_function$ **or** $optimal\_interface$ *is NULL* **then**
8. SET n as $optimal\_interface$;
9. UPDATE $optimal\_cost\_function$ with interface n;
10. **end**
11. **end**
12. **end**
13. SLEEP Δ
14. **end**
15. **end**

Algorithm 2: Optimal network interface selection

Step 5 compares the costs of interface n and the last candidate to check whether n is better. If so, it is designated as the new candidate, and the minimal cost value is correspondingly updated. The network interface with lowest cost value will be selected (step 10).

VI. Evaluation

This section presents a performance analysis of the proposed NFV-based architecture. We compare experimental results with the theoretical values to evaluate the accuracy of the models, and to improve the algorithms.

A. Experiment Setup

Our testbed, based on an Ericsson Blade System consists of six VMs. Four VMs host basic components of a vIMS system (i.e. P-CSCF, I-CSCF, S-CSCF, and HSS) and two VMs host the vGateway and the Web server. Each VM is configured with a virtual CPU and 512MB of memory. All VMs run Ubuntu and are connected within a simple local network topology in bridging mode. Both vGateway and vIMS software support multi-thread mechanism for M/G/1 processor-sharing system. To minimize the deviation of performance evaluation, the WU and IMS UE are both in the same subnet. The Web server not only processes HTTP transactions but also acts as a log server that collects delay information from the WU. By putting the DNS server on the same VM with P-CSCF server, DNS queries can be instantly executed so that they are neglected from the evaluation.

Three PCs equipped with a core i7 CPU and 8GB of memory are used to simultaneously run separated instances of IMS clients or Web browsers for WUs. To perform login or calling procedure with minimal human intervention, we develop an automation testing tool for Web applications that can be applied to most of Web browsers. This tool provides the ability to open/close a browser, access to a given URL, set pre-defined values to Web controls, as well as click buttons.

The experiments are recurrently carried out with different numbers of WUs (i.e. 5, 10, 15, 20, 25, 30, 35, 40) and coming rates of requests (i.e. $1sec^{-1}, 3sec^{-1}, 5sec^{-1}, 7sec^{-1}, 9sec^{-1}$). The increasing number of WUs helps enlarge the number of trials and thus makes the average of the data closer to the expected value according to the law of large numbers. The PC used in the experiment may support at maximum 40 WUs in the same time. To generate requests as a Poisson process for login phase with the pre-defined λ, the automation tool triggers a click event to LOGIN or CALL button after each interval of time calculated using exponential random variable (in order of millisecond). Such interval periods can be retrieved according to [26]. The values of service rate are obtained as described at step 6 of Algorithm 1. We compute the service

rate of each server by performing 100 randomly login and calling sessions and the results are shown in Table I. At each server, we use *mpstat* tool to record the actual CPU usage in **.cpu* files, whereas the HTTP(S)/SIP-related messages as input for step 4 & 5 of Algorithm 1 are captured as **.pcap* files by *tcpdump* tool. The actual session delay (d_{actual_login}, $d_{actual\_calling}$) are reported by the WUs and sent to the Web server as **.del* files. These metrics are obtained similarly as methods used for Registration Request Delay (RRD) and successful Session Request Delay (SRD) metrics as explained in [27].

B. Performance Evaluation

We compare data collected from aforementioned experiments with the results obtained from the proposed model. For the probabilities p_i, we entirely rely on counting scheme on **.pcap* files in Section VI-A rather than involving specifications of IMS message flows. This is because both HTTP(S) and SIP messages are transmitted several times or split into multiple packets. Based on the number of messages that the vGateway sends to the WU (called n_1) and to the P-CSCF (called n_2), we have $p_1 = n_1/n_2$. Similarly, substituting the values of p_i, $i = 1, \ldots, 5$ back to (9), (10), (11), (12), (13) yields $\lambda_i, i = 1, \ldots, 5$ as illustrated in Table I.

TABLE I
EXPERIMENTAL ARRIVAL RATE AND SERVICE RATE RESULTS

Server	vGateway	P-CSCF	I-CSCF	S-CSCF	HSS
$\boldsymbol{\lambda_i}$	2.5λ	3.0λ	4.5λ	3.38λ	3.75λ
$\boldsymbol{\mu_i}$	0.26	0.23	0.20	0.30	0.28

The values of $w_i, i = 1, \ldots, 5$ are calculated as the number of messages each server manipulates as well. Hence, from (2,3), we have the values of $w_i, i = 1, \ldots, 5$ for each type of session delay (i.e. login and calling).

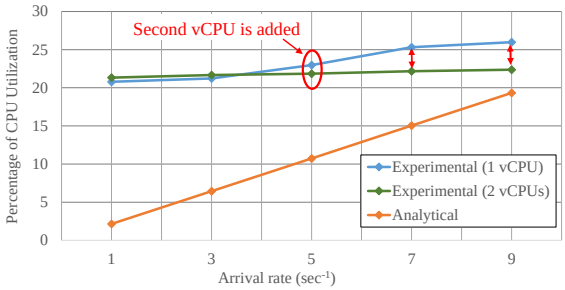

Fig. 5. Comparison of CPU Utilization

Fig. 5, Fig. 6 and Fig. 7 illustrate the difference between theoretical values and experimental results which are already normalized for a better visualization, in terms of CPU utilization during login phase, login/calling session delay and their low boundaries. In general, the values are higher in experimental situations. This mismatch is resulted from the assumptions on the model as well as from some overhead that has been discarded in the calculation (i.e. CPU used for OS, impact of hypervisor). The chart in Fig. 5 shows that the actual CPU usage with one vCPU is generally over 20% whereas the analytical model predicts a smaller value. As shown in the graph, the real values remains nearly to 21% at rate $1sec^{-1}$ and $3sec^{-1}$, and more than 25% at rate $9sec^{-1}$. There is a significant increase between rate $3sec^{-1}$ and $7sec^{-1}$. However, this growth is not as fast as it is in analytical model. To verify the resource allocation of the algorithm 1, we add additional CPU and performed the experiments with all arrival rates. The results from Fig. 5 demonstrate the improvement of proposed algorithm with the maximum value of CPU usage (MC as defined in Algorithm 1) at 25%. At rate $5sec^{-1}$ when the CPU usage (23%) is about to go up, by allocating more resource, the level of CPU usage stays under MC as seen at rate $7sec^{-1}$ and $9sec^{-1}$.

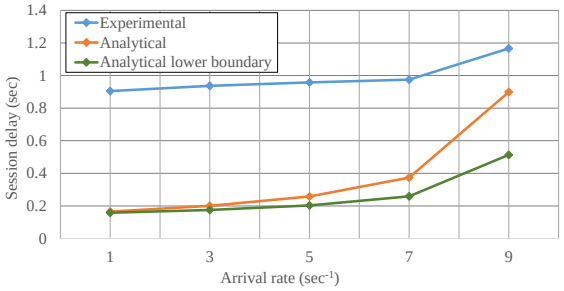

Fig. 6. Comparison of Login/Registration Session Delay

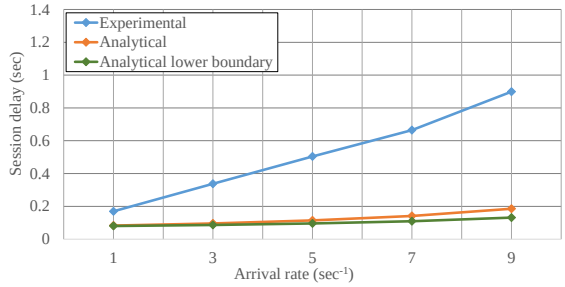

Fig. 7. Comparison of Calling Session Delay

Similarly, there are also gaps between normalized theoretical and practical session delay values in Fig. 6 and Fig. 7 due to model assumption and neglected overhead. The upper limits are not shown since it far exceeds the actual values at high λ. Interestingly in Fig. 7, over the next arrival rates, the actual $d_{actual\_calling}$ slightly differs from $d_{calling}$ around 0.12sec and then sharply rises. The trend in Fig. 6 is similar, however the gap is narrower at the highest arrival rate (sec^{-1}). This phenomenon can be explained by the involvement of servers in the formulation of login delay in (2) and calling delay in (3). More precisely, the calling session delay is merely at the vGateway where the variance of its analytical value at rate $1sec^{-1}$ (0.08sec) and $9sec^{-1}$ (0.18sec) is negligible in comparison with others, i.e. I-CSCF (0.07sec vs 0.32sec), S-CSCF (0.10sec vs 1.00sec) and HSS(0.10sec vs 1.5sec).

VII. Conclusion

In this paper, we have presented a new NFV architecture for interworking WebRTC and IMS users. We also propose two algorithms to tackle the problem of elastically allocating computing resources and selecting network access for the proposed NFV model. Using M/G/1-PS queueing model and experimental results, we have built models for service rate, the CPU usage, the session delay as well as the relationship between analytical values and practical results, which can be used to optimally allocate physical resources according to application requirements. To the best of our knowledge, this work is the first effort dealing with both stochastic modeling and resource optimization for NFV-based systems.

In the future, we plan to investigate other models (i.e. G/M/k) to predict the server performance. Furthermore, we will extend the algorithm for access selection with additional metrics, like reliability, availability, and energy efficiency.

Acknowledgment

The authors thank NSERC and Ericsson for funding the project CRDPJ 469977. This research also receives support from the Canada Research Chair, Tier 1, hold by Mohamed Cheriet.

References

[1] R. Guerzoni, *Network Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges and Call for action. Introductory white paper*, in SDN and OpenFlow World Congress, June, 2012.

[2] G. Camarillo, and M. A. Garcia-Martn, *The 3G IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds*, Wiley, 2008.

[3] H. Alvestrand, *Overview: Real Time Protocols for Browser-based Applications*, draft-ietf-rtcweb-overview-15, Internet Engineering Task Force (IETF), Jan. 2016.

[4] 3GPP TS 23.228, *IP Multimedia Subsystem*, v13.4.0, Sept. 2015.

[5] ETSI Industry Specification Group (ISG) Network Functions Virtualization (NFV), *Network Functions Virtualisation (NFV); Architectural Framework*, European Telecommunications Standards Institute, ETSI-GS-NFV-002, 2014. [Online]. Available: http://www.etsi.org. [Accessed: Mar. 22, 2016]

[6] N. Figueira, R. Krishnan, D. Lopez, S. Wright, and D. Krishnaswamy, *Policy Architecture and Framework for NFV Infrastructures*, draft-irtf-nfvrg-nfv-policy-arch-03, Internet Research Task Force, Mar. 2016.

[7] P. Quinn, and T. Nadeau, *Problem Statement for Service Function Chaning*, Informational RFC 7498, Internet Engineering Task Force (IETF), Apr. 2015.

[8] G. Carella, et al., *Cloudified IP Multimedia Subsystem (IMS) for Network Function Virtualization (NFV)-based Architectures*, IEEE Symposium on Computers and Communication (ISCC), June, 2014

[9] M. Fakhfakh, I. L. Bedhiaf, O. Cherkaoui, and M. Frikha, *High Availability in IMS Virtualized Network*, in Proceedings of the 1st International Conference on Communications and Networking, Tunisia, Nov. 2009.

[10] M. Ito, K. Nakauchi, Y. Shoji, N. Nishinaga, and Y. Kitatsuji, *Service-Specific Network Virtualization to Reduce Signaling Processing Loads in EPC/IMS*, Special Secion on 5G Wireless Technologies: Perspectives of the Next Generation Mobile Communications and Networking, vol. 2, pp. 1076-1084, Sept. 2014.

[11] L. Liao, V. C.M. Leung, and M. Chen, *Virtualizing IMS Core and Its Performance Analysis*, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 142, pp. 53-65, Mar. 2015.

[12] M. Umair, *Performance Evaluation and Elastic Scaling of an IP Multimedia Subsystem Implemented in a Cloud*, M.S. thesis, ICT, KTH, Stockholm, Sweden, 2013.

[13] F. Lu, H. Pan, X. Lei, X. Liao, and H. Jin, *A Virtualization-based Cloud Infrastructure for IMS Core Network*, in Proceedings of the International Conference on Cloud Computing Technology and Science, Bristol, UK, Dec. 2013.

[14] J. Uberti, C. Jenning, and E. Rescorla, *Javascript Session Establishment Protocol*, draft-ietf-rtcweb-jsep-14, Internet Engineering Task Force (IETF), Mar. 2016.

[15] S. Nandakumar, and C. Jennings, *SDP for the WebRTC*, draft-ietf-rtcweb-sdp-01, Internet Engineering Task Force (IETF), Mar. 2016.

[16] B.S. Cruz, and J.P. Barraca, *IMS Centric Communication Supporting WebRTC Endpoints*, in IEEE Symposium on Computers and Communication (ISCC), July 2015.

[17] A. Amirante, T. Castaldi, L. Miniero, and S.P. Romano, *Janus: a general purpose WebRTC gateway*, in Proceedings of the Conference on Principles, Systems, and Applications of IP Telecommunications, Chicago, USA, Sept-Oct. 2014.

[18] T. Bach, M. Maruschke, J. Zimmermann, K. Hänsge, and M. Baumgart, *Combination of IMS-based IPTV Services with WebRTC*, in Proceedings of the 9th Internation Multi-Conference on Computing in the Global Information Technology, Seville, Spain, June 2014.

[19] E. Ivov, E. Rescorla, J. Uberti, and P. Saint-Andre, *Trickle ICE: Incremental Provisioning of Candidates for the Interactive Connectivity Establishment (ICE) Protocol*, draft-ietf-ice-trickle-01, Internet Engineering Task Force (IETF), Dec. 2015.

[20] R. Jesup, S. Loreto, and M. Tuexen, *WebRTC Data Channels*, draft-ietf-rtcweb-data-channel-13, Internet Engineering Task Force (IETF), Jan. 2015.

[21] B. Schwartz, and J. Uberti, *Recursively Encapsulated TURN (RETURN) for Connectivity and Privacy in WebRTC*, draft-ietf-rtcweb-return-01, Internet Engineering Task Force (IETF), Jan. 2016.

[22] J. Uberti, and G. Shieh, *WebRTC IP Address Handling Recommendations*, draft-ietf-rtcweb-ip-handling-01, Internet Engineering Task Force (IETF), Mar. 2016.

[23] *Stochastic Processes: Modelling and Simulation*, 1st ed., Elsevier Science B.V., Amsterdam, The Netherlands, 2003, pp.557-572.

[24] M. Zukerman, *Introduction to Queueing Theory and Stochastic Teletraffic Models*, [Online]. Available: http://arxiv.org/pdf/1307.2968.pdf. [Accessed: Apr. 10, 2016]

[25] U.N. Bhat, *The General Queue G/G/1 and Approximations*, in *An Introduction to Queueing Theory: modeling and analysis in applications*, Birkhäuser Verlag, 2008.

[26] E.D. Knuth, *Random Numbers*, in *The Art of Computer Programming*, vol. 2, 2nd ed., Addison-Wesley, Reading Mass., 1981, p. 128.

[27] D. Malas, and A. Morton, *Basic Telephony SIP End-to-End Performance Metrics*, RFC 6076, Internet Engineering Task Force (IETF), Jan. 2011.