Design and Analysis of Deadline and Budget Constrained Autoscaling (DBCA) Algorithm for 5G Mobile Networks

Tuan Phung-Duc[†], Yi Ren[*], Jyh-Cheng Chen[*], *Fellow, IEEE*, and Zheng-Wei Yu[*]
[†]Faculty of Engineering, Information and Systems, University of Tsukuba, Ibaraki, Japan
[*]Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, R.O.C.
Emails:[†]tuan@sk.tsukuba.ac.jp, [*]{renyi, jcc, zwyu12260}@cs.nctu.edu.tw

***Abstract*—In cloud computing paradigm, virtual resource autoscaling approaches have been intensively studied recent years. Those approaches dynamically scale in/out virtual resources to adjust system performance for saving operation cost. However, designing the autoscaling algorithm for desired performance with limited budget, while considering the existing capacity of legacy network equipment, is not a trivial task. In this paper, we propose a Deadline and Budget Constrained Autoscaling (DBCA) algorithm for addressing the budget-performance tradeoff. We develop an analytical model to quantify the tradeoff and cross-validate the model by extensive simulations. The results show that the DBCA can significantly improve system performance given the budget upper-bound. In addition, the model provides a quick way to evaluate the budget-performance tradeoff and system design without wide deployment, saving on cost and time.**

***Index Terms*—Autoscaling Algorithm, Modeling and Analysis, Network Function Virtualization, 5G, Cloud Networks, Virtualized EPC**

I. Introduction

The emergence of Network Functions Virtualization (NFV) is changing the way of how mobile operators increase the capacities of their network infrastructures. NFV offers fine-grained on-demand adjustment of network capabilities. Virtualized Network Functions (VNFs) instances can be scaled-out/in (turn on/off) to adjust computing and network capabilities for saving energy and resources. A classic case is Animoto, an image-processing service provider, experienced a demand surging from 50 VM instances to 4,000 instances in three days, April 2008. After the peak, the demand fell sharply to an average level. Animoto only paid for 4,000 instances for the peak time [1].

Designing good *autoscaling* strategies for budget constraints while meeting performance requirements is challenging. In particular, operation cost is decreased by reducing the number of power-on VNF instances. On the other hand, resource under-provisioning may cause Service Level Agreements (SLAs) violations, leading to low Quality of user Experience (QoE). Therefore, the goal of desired autoscaling strategies is to meet the budget constraint while maintaining an acceptable level of performance. Then, a budget-performance tradeoff is formed: The system performance is improved by adding more VNF instances while operation cost is reduced by the opposite way.

Designing autoscaling strategies for 5G mobile networks is different from that for traditional cloud computing scenarios. Specifically, in previous cloud autoscaling schemes (e.g., [2]–[13]), only virtualized resources are considered. This is not suitable for typical cellular networks. Given widely deployed existing legacy network equipment, the desired solution should consider the capacities of both legacy network equipment and VNFs. For example, consider VNF only case that a VNF scaling-out from 1 VNF instance to 2 VNF instances increases 100% capacity. Whereas, its capacity only grows less than 1% if legacy network equipment (say 100 VNF instance capability) is counted. Current cloud autoscaling schemes usually ignore the non-constant issue.

In this paper, we investigate the budget-performance tradeoff in terms of deadline constraint, VM setup time, and the legacy equipment capacity. We improve our recent work [14] by further considering deadline constraint for incoming requests, i.e., a request will be dropped if a pre-specified timer is expired. This is a more practical assumption compared with that in [14], in which no deadline constraint is considered. To the best of our knowledge, this is the first work from this perspective. We then propose a Deadline and Budget Constrained Autoscaling (DBCA) algorithm for addressing the tradeoff. The DBCA considers available legacy equipment powered on all the time, while virtualized resources are divided into k VNF instances. Then the DBCA scales out/in (turns on/off) VNF instances depending on job arrivals. Here, a central issue is how to choose a suitable k for balancing the tradeoff. We then derive a detailed analytical model to answer this question. The analytical model quantifies the budget-performance tradeoff and cross-validates against extensive ns2 simulations. Furthermore, we propose a recursive approach to reduce the complexity of the computational procedure from $O(k^3K^3)$ to $O(kK)$ where K the system capacity. Our model provides mobile operators with guidelines to design optimal VNF autoscaling strategies by their management policies in a systematical way, and enable wide applicability in various scenarios, and therefore, have important theoretical significance.

The rest of this paper is organized as follows. Section II reviews the related work. Section III briefly introduces some background material on mobile networks and NFV architecture. Section IV presents the proposed optimal algorithm for VNF autoscaling applications. Section V addresses the analytical models, followed by numerical results illustrated in Section VI. Section VII offers conclusions.

II. Related Work

Recent years, autoscaling mechanisms have been intensively studied [2]–[19]. A straightforward and commonly used autoscaling approach is that autoscaling decisions are made based on resource utilization indicators (e.g., CPU, memory usage, etc). An example is the default autoscaling approaches offered by Amazon EC2 and Microsoft Azure. A scale-out request is sent right way if the current CPU usage exceeds a predefined threshold. However, specifying the threshold value is not easy while considering VM setup time. Indeed, the setup lag time could be as long as 10 min or more to start an instance in Microsoft Azure; and the lag time could be various from time to time [20]. Thus it may happen that the instance is too late to serve the VNF so that one needs to leave more redundant while setting the threshold. To handle the setup time, prediction/learning models are utilized to estimate the workload arrivals for autoscaling decision making, such as Exponential weighted Moving Average (EMA) [2], [3], Auto-Regressive Moving Average (ARMA) [4], [5], Auto-Regressive Integrated Moving Average (ARIMA) [6], [7], machine learning [8], [9], Markov model [10], [11], recursive renewal reward [12], and matrix analytic method [13]. However, *the mechanisms [2]–[13] only consider virtualized resource itself (cloud resource) while overlooking legacy (fixed) resources*, which are not suitable for typical cellular networks.

Perhaps the closest models to ours were studied in [14]–[19] that both the capacities of fixed legacy network equipment and dynamic autoscaling cloud servers are considered. The authors in [15], [16] consider setup time without defections [15] and with defections [16]. Our recent work [18] relaxes the assumption in [15], [16] that after a setup time, all the cloud servers in the block are active concurrently. We further consider a more realistic model that each server has an independent setup time. However, in [15], [16], [18], all the cloud servers were assumed as a whole block, which is not practical where each cloud server should be allowed to scale-out/in dynamically. Considering all cloud servers as a whole block was relaxed to sub-blocks in [17], [19]. However, either setup time is ignored [17], or fixed legacy network capacity is not considered [19]. Our recent work [14] fixes the research gap, whereas job deadline constraint is not considered.

III. Background

Mobile Core Network (CN) is one of the most important parts in mobile networks. The main target of NFV is to virtualize the functions in the CN. The most recent CN is the Evolved Packet Core (EPC) introduced in Long Term Evolution (LTE). Here, we use an example to explain EPC and virtualized EPC (vEPC) when NFV is deployed. Fig. 1 shows a simplified example of NFV enabled LTE architecture consisted of Radio Access Network (RAN), EPC, and external Packet Data Network (PDN). In particular, the EPC is composed of legacy EPC and vEPC. In the following, we brief introduce them respectively.

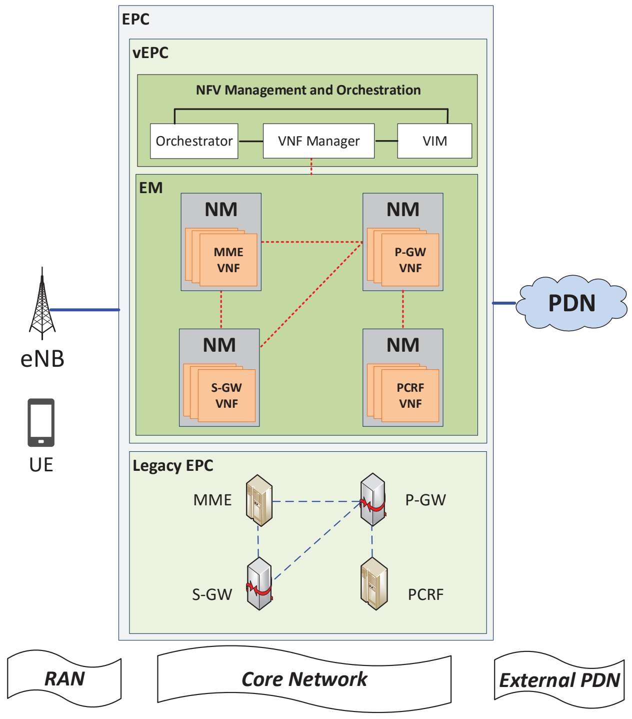


Fig. 1: A simplified example of NFV enabled LTE architecture.

A. Legacy EPC and vEPC

EPC is the CN of the LTE system. Here, we only show basic network functions, such as Serving Gateway (S-GW), PDN Gateway (P-GW), Mobility Management Entity (MME), and Policy and Charging Rules Function (PCRF) in the EPC.

To virtualize the above network functions, 3GPP introduces NFV management functions and solutions for vEPC based on ETSI NFV specification [21], as shown in Fig. 1. The network functions (e.g., MME, PCRF) are denoted as Network Elements (NE), which are virtualized as VNF instances. Network Manager (NM) provides end-user functions for network management of NEs. Element Manager (EM) is responsible for the management of a set of NMs. NFV management and orchestration controls VNF instance scaling procedure, which are detailed as follows.

- VNF scale-in/out: VNF scale-out adds additional VMs to support a VNF instance, adding more virtualized hardware resources (i.e., compute, network, and storage capability) into the VNF instance. In contrast, VNF scale-in removes existing VMs from a VNF instance.
- VNF scale-up/down: VNF scale-up allocates more hardware resources into a VM for supporting a VNF instance (e.g., replace a One-core with Dual-core CPU). Whereas, VNF scale-down releases hardware resources from a VNF instance.

IV. Proposed Deadline and Budget Constrained Autoscaling Algorithm

In this section, we propose Deadline and Budget Constraint Autoscaling (DBCA) algorithm to meet budget constraint while maintaining acceptable levels of performance. More running VNF instances reduce the possibility of SLAs violations. However, this may lead to redundant power-on VNF instances, resulting in extra operation cost. We refer the trade-off as budget-performance tradeoff. Section IV-A introduces the DBCA algorithm for balancing the tradeoff. Section IV-B further defines budget constraint and other three performance metrics for evaluating SLAs violations.

A. System Model and DBCA: Deadline and Budget Constraint Autoscaling

In general, we consider that a 5G EPC consists of legacy network entities (e.g., MME, PCRF) and VNFs [22], [23]. For a network entity, its capacity is supported by both legacy network equipment and VNF instances. Fig. 2 illustrates a simplified example of a network entity queueing model considering the capacities of both VNF instances and legacy network equipment. Specifically, the capacity of its legacy network equipment is assumed to be n_0 VNF instance capacities while k denotes the number of VNF instances for supporting the network entity. That is, the total capacity of the network entity is $k + n_0 = N$.

From the network entity's point of view, we assume that user requests arrive according to a Poisson process with rate λ. The capacity of a VNF instance is assumed to accept one job at a time and the service time is assumed to follow the exponential distribution with mean $1/\mu$. When a user request arrives, the job first enters a limited First-Come-First-Served (FCFS) queue waiting for processing. Each job has deadline constraint, which is a random variable following the exponential distribution with mean $1/\theta$. In other words, the job will quit the queue if its waiting time exceeds its deadline. Without loss of generality, the legacy network equipment is always on while VNF instances will be powered on (or off) according to the number of waiting jobs in the queue. Moreover, a VNF instance needs a setup time to be available to serve user requests, which is assumed to be an exponentially distributed random variable with mean value $1/\alpha$.

DBCA utilizes two thresholds, '`Up`' and '`Down`', or U_i and D_i to control the VNF instances $i = 1, 2, \cdots, k$. Further, let $n_1 = n_0 + 1$ and $n_i = n_{i-1} + 1$ ($i = 1, 2, \cdots, k$), i.e., $n_k = N$. In other words, DBCA sends orders to NFV management and orchestration to turn on/off VNF instances to adjust network capacities.

- *U_i, power up the i-th VNF instances:* If the i-th VNF instance is turned off and the number of requests in the system increases from $U_i - 1$ to U_i, then the VNF instance is powered up after a setup time to support the system. During the setup time, a VNF instance cannot serve user requests, but consumes power (or money for renting cloud services). Here, we specify $U_i = n_i$. It is equivalent to that when the number of requests increases from n_{i-1} to n_i, the i-th VNF instance is powered up.
- *D_i, power down the i-th VNF instances:* If the i-th VNF instance is operative, and the number of requests in the system drops from $D_i + 1$ to D_i, then the VNF instance is powered down instantaneously. In this paper, we choose $D_i = n_{i-1}$. It is equivalent to that when the number of requests drops from n_i to n_{i-1}, we turn off the i-th VNF instance.

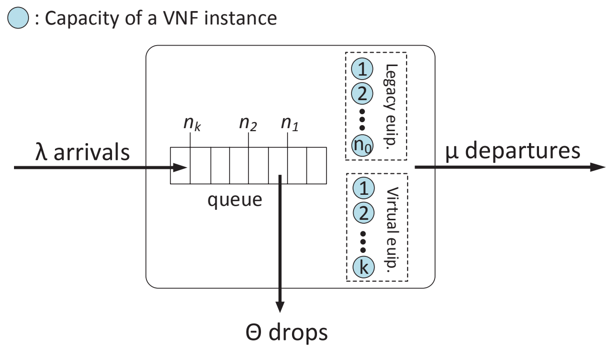


Fig. 2: A service center with reserve blocks.

B. Performance Metrics

The system performance is evaluated by four metrics: the average response time in the queue W_q, the average number of running VNF instance S, user request blocking probability P_b, and user request dropping probability P_d. We define them as follows.

- *The average response time in queue W_q* is defined as a job request's waiting time in queue. In other words, it reveals how long time a job request can be served.
- *The average number of running VNF instances S* addresses the operation cost of virtual equipment.
- *Dropping probability P_d* is the probability that a request's waiting time in queue exceeds its deadline constraint.
- *Blocking probability P_b* is the probability that a request is denied due to system busy.

The closed-form solutions of W_q, S, P_b, and P_d are given as (4), (5), (2), and (6) in Section V. Thus, the system performance P has the form

$$P = w_1 W_q + w_2 S + w_3 P_b + w_4 P_d, \tag{1}$$

where coefficients w_1, w_2, w_3, and w_4 denote the weight factors for W_q, S, P_b, and P_d, respectively. Increasing w_1 (or w_2, w_3, w_4) emphasizes more on W_q (or S, P_b, P_d). Here, we do not specify either w_1 or w_2 (w_3, w_4) due to the fact that such a value should be determined by a mobile operator and must take management policies into consideration.

V. Analytical Model

In this section, we propose the analytical model for DBCA. The goal of the analytical model is to cross-validate the accuracy of the simulation experiments and to analyze both the

TABLE I: List of Notations

Notation	Explanation
N	The total capacities of a network entity
K	The number of maximum jobs can be accommodated in the system
k	The number of VNF instances
P	System performance
W	Average response time
W_q	Average response time in queue
S	Average VM cost
P_b	Blocking probability
P_d	Dropping probability
w_1	Weight factor for W_q
w_2	Weight factor for S
w_3	Weight factor for P_b
w_4	Weight factor for P_d
n_0	The capacities of legacy network equipment
U_i	The `Up` threshold to control the VNF instances
D_i	The `Down` threshold to control the VNF instances
m_i	The i-th reserve sub-block ($i = 1, 2, \cdots k$).
λ	Job arrival rate
μ	Service rate for each server
α	Setup rate for each virtual server
θ	Abandonment rate of each job

operation cost and the system performance for DBCA. Given the analytical model, one can quickly obtain the operation cost and system performance for DBCA, without real deployment, saving on cost and time.

We model the system as a queueing model with N servers and a capacity of K, i.e., the maximum of K jobs can be accommodated in the system. Job arrivals follow the Poisson distribution with rate λ. A VNF instance (server) accepts one job at a time, and its service time follows the exponential distribution with mean $1/\mu$. There is a limited FCFS queue for those jobs that have to wait for processing.

In this system, a server is turned off immediately if it has no job to do. Upon arrival of a job, an OFF server is turned on if any and the job is placed in the buffer. However, a server needs some setup time to be active so as to serve waiting jobs. We assume that the setup time follows the exponential distribution with mean $1/\alpha$. Let j denotes the number of customers in the system and i denotes the number of active servers. The number of reserves (server) in setup process is $\min(j - n_i, N - n_i)$. Here, $n_i = n_{i-1} + m_i$, where $m_i = 1$ for all i (block size is one). Therefore, in this model a server in reserve blocks is in either BUSY or OFF or SETUP. We assume that waiting jobs are served according to an FCFS manner. We call this model an $M/M/N/K$ Setup queue.

Here, we present a recursive scheme to calculate the joint stationary distribution. Let $C(t)$ and $L(t)$ denote the number of active servers and the number of customers in the system, respectively. It is easy to see that $\{X(t) = (C(t), L(t)); t \geq 0\}$ forms a Markov chain on the state space:

$$\begin{aligned}\mathcal{S} =& \{(i,j); 1 \leq i \leq k, j = n_i, n_i + 1, \dots, K-1, K\} \\ & \cup \{(0,j); j = 0, 1, \dots, K-1, K\}.\end{aligned}$$

Fig. 3 shows the transition among states for the case where

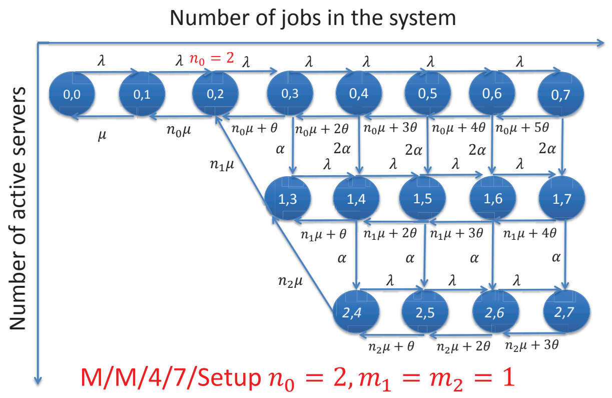


Fig. 3: Transition among states ($N = 4, n_0, = 2, m_1 = m_2 = 1$, and $K = 7$).

$N = 4, n_0 = 2, m_1 = m_2 = 1$, and $K = 7$. Let $\pi_{i,j} = \lim_{t\to\infty} \mathrm{P}(C(t) = i, L(t) = j)$ ($(i,j) \in \mathcal{S}$) denote the joint stationary distribution of $\{X(t)\}$. Here, we derive a recursion for calculating the joint stationary distribution $\pi_{i,j}$ ($(i,j) \in \mathcal{S}$). The balance equations for states with $i = 0$ read as follows.

$$\begin{aligned} \lambda\pi_{0,j-1} &= j\mu\pi_{0,j}, \\ & \quad \text{for } \ j = 0, 1, \dots, n_0, \\ \lambda\pi_{0,j-1} + n_0\mu\pi_{0,j+1} &= (\lambda + \min(j - n_0, N - n_0)\alpha + n_0\mu)\pi_{0,j}, \\ & \quad \text{for } \ j = n_0, n_0 + 1, \dots, K-1, \\ \lambda\pi_{0,K-1} &= (n_0\mu + (N - n_0)\alpha)\pi_{0,K}, \end{aligned}$$

leading to

$$\pi_{0,j} = b_j^{(0)}\pi_{0,j-1}, \qquad j = 1, 2, \dots, K.$$

The sequence, $\{b_j^{(0)}; j = 1, 2, \dots, K\}$ is given as follows.

$$b_j^{(0)} = \frac{\lambda}{j\mu}, \qquad j = 1, 2, \dots, n_0,$$

and

$$b_j^{(0)} = \frac{\lambda}{A1_j}, \qquad j = K-1, K-2, \dots, n_0 + 1,$$

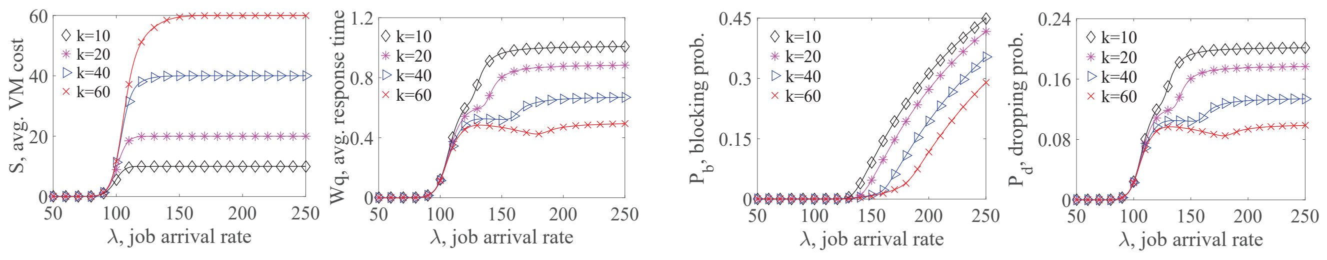
where $A1_j = \lambda + n_0\mu + \min(j - n_0, N - n_0)\alpha + (j - n_0)\theta - (n_0\mu + (j + 1 - n_0)\theta)b_{j+1}^{(0)}$ and

$$b_K^{(0)} = \frac{\lambda}{n_0\mu + (N - n_0)\alpha + (K - n_0)\theta}.$$

Furthermore, it should be noted that $\pi_{1,1}$ is calculated using the local balance equation in and out the set $\{(0,j); j = 0, 1, \dots, K\}$ as follows.

$$n_1\mu\pi_{1,1} = \sum_{j=n_1}^{K} \min(j, N - n_0)\alpha\pi_{0,j}.$$

Remark. *We have expressed $\pi_{0,j}$ ($j = 1, 2, \dots, K$) and $\pi_{1,1}$ in terms of $\pi_{0,0}$.*



(a) Impacts on S. (b) Impacts on Wq. (c) Impacts on P_b. (d) Impacts on P_d.

Fig. 4: Impacts of k on the performance metrics ($n_0 = 100$). The *lines* denote analytical results, and the *points* represent simulation results.

Similarly, we can express $\pi_{1,j}$ ($j = 2, 3, \ldots, K$) in terms of $\pi_{1,1}$. Then $\pi_{2,2}$ is obtained according to $\pi_{1,j}$ ($j = 2, 3, \ldots, K$). Next, using similar recursive formulae, we can express all $\pi_{i,j}$ ($(i,j) \in \mathcal{S}$) in terms of $\pi_{0,0}$ which is uniquely determined using the normalization condition:

$$\sum_{(i,j)\in\mathcal{S}} \pi_{i,j} = 1.$$

It is worth to mention that the complexity of the computational procedure is of order $O(k \times K)$ instead of $O(k^3 \times K^3)$ if we directly solve the system of balance equations by a general method.

Due to the page limitation, we only show the final derivation results as follows. Interested reader may refer to [24] for detailed mathematical analysis.

Let $\mathrm{E}[L]$ denote the mean number of jobs in the system. We have

$$\mathrm{E}[L] = \sum_{(i,j)\in\mathcal{S}} \pi_{i,j} j = \sum_{i=0}^{n_0-1} \pi_{0,j} j + \sum_{i=0}^{k} \sum_{j=n_i}^{K} \pi_{i,j} j.$$

Let P_b denote the blocking probability. We have

$$P_b = \sum_{i=0}^{k} \pi_{i,K}. \tag{2}$$

It follows from Little's law that

$$W = \frac{\mathrm{E}[L]}{\lambda(1-P_b)} = \frac{\sum_{i=0}^{n_0-1} \pi_{0,j} j + \sum_{i=0}^{k} \sum_{j=n_i}^{K} \pi_{i,j} j}{\lambda(1 - \sum_{i=0}^{k} \pi_{i,K})}. \tag{3}$$

We obtain

$$W_q = W - \frac{1}{\mu}. \tag{4}$$

The mean number of VNF instances is given by

$$S = \sum_{(i,j)\in\mathcal{S}} \pi_{i,j}(n_i - n_0) + \sum_{i=0}^{k} \sum_{j=n_i}^{K} \pi_{i,j} \min(j - n_i, N - n_i), \tag{5}$$

where the first term is the number of VNF instances that are already active while the second term is the mean number of VNF instances in setup mode.

Let $\mathrm{E}[Q]$ denote the mean number of waiting jobs in the system. We have

$$\mathrm{E}[Q] = \sum_{i=0}^{k} \sum_{j=n_i}^{K} \pi_{i,j}(j - i).$$

Let P_d denote the reneging probability that a waiting job abandons from the system. We have

$$P_d = \frac{\mathrm{E}[Q]\theta}{\lambda(1-P_b)} = \frac{\sum_{i=0}^{k} \sum_{j=n_i}^{K} \pi_{i,j}(j-i)\theta}{\lambda(1-P_b)}, \tag{6}$$

where the numerator and the denominator are the abandonment rate and the arrival rate of accepted jobs, respectively.

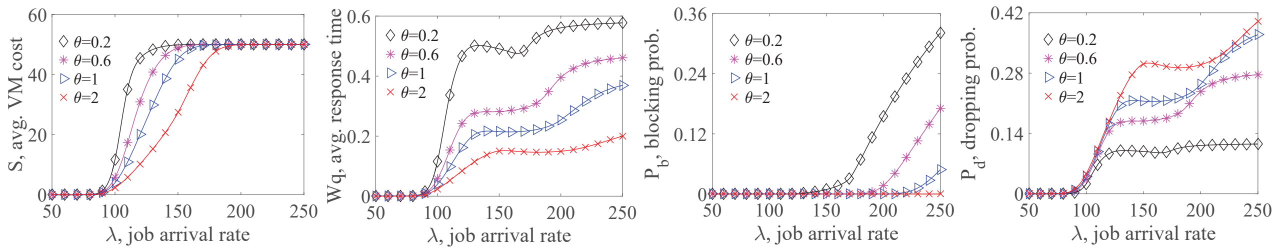
Again, based on the above derived performance metrics W_q, S, P_b, and P_d, mobile operators can easily design network optimization strategies according to (1).

VI. Simulation and Numerical Results

This section provides both simulation and numerical results for the analytical model addressed in Section V. The analytical model is cross-validated by extensive simulations by using ns2, version 2.35 [25] with real measurement results for parameter configuration[1]: λ by Facebook data center traffic [26], μ by the base service rate of a Amazon EC2 VM [27], and α by the average VM startup time [28]. If not further specified, the following parameters are set as the default values for performance comparison: $n_0 = 110$, $\mu = 1$, $\alpha = 0.005$, $K = 250$, $\lambda = 50 \sim 250$ (see Table 1 for details). The results are based on exponential distribution for job request inter arrival time and VNF instance service time with mean $1/\lambda$ and $1/\mu$. The simulation time is 300,000 seconds. And $15 \sim 75$ millions job requests were generated during the extensive simulations.

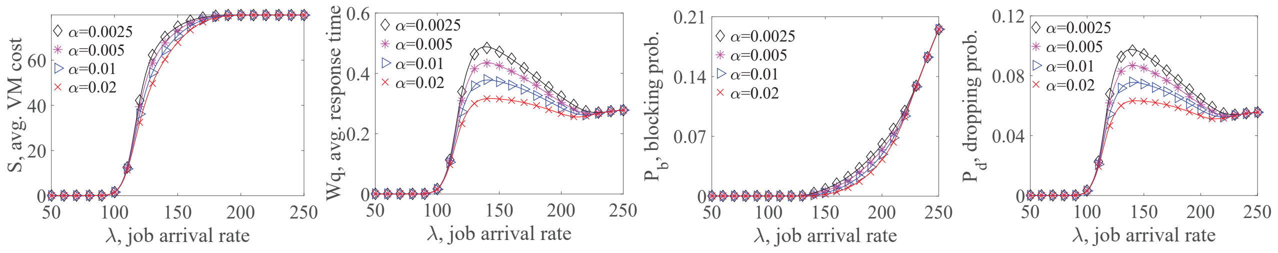
Figs. 4, 5, 6, 7, 8 not only demonstrate the correctness of our analytical model, but also illustrate the impacts of λ, k, θ, α, n_0, K on the performance metrics: average VM cost S, average response time in queue W_q, blocking probability P_b, and dropping probability P_d, respectively. *In the figures, the lines denote analytical results, and the points represent simulation results.* Each simulation result in the figures is

[1]Due to simulation time limitation, λ and μ are scaled down accordingly with the same ratio λ/μ.



(a) Impacts on S. (b) Impacts on Wq. (c) Impacts on P_b. (d) Impacts on P_d.

Fig. 5: Impacts of θ on the performance metrics ($n_0 = 100$, $k = 50$). The *lines* denote analytical results, and the *points* represent simulation results.



(a) Impacts on S. (b) Impacts on Wq. (c) Impacts on P_b. (d) Impacts on P_d.

Fig. 6: Impacts of α on the performance metrics ($k = 80$). The *lines* denote analytical results, and the *points* represent simulation results.

the mean value of the results in 300,000 seconds with 95% confidence level.

A. Impacts of Arrival Rate λ

We first look into the impacts of job request arrival rate λ. Mobile operators cannot adjust λ but are able to monitor it and configure network parameters k, θ, α, n_0, and K for network optimization accordingly.

Figs. 4(a), 5(a), 6(a), 7(a), 8(a) depict the impacts of λ on S. In general, one can see that S initiates at 0 at the beginning and then starts to raise sharply when λ passes $n_o\mu$. The reason is that the incoming job requests are served by the legacy equipment when $\lambda < n_0\mu$. No VMs are powered on. Then DBCA starts to turn on VMs to handle job requests as λ is increasing. Later, S reaches at a bound even if λ continues growing. This is because all the k VMs are turned on so that S is bounded as k VM costs.

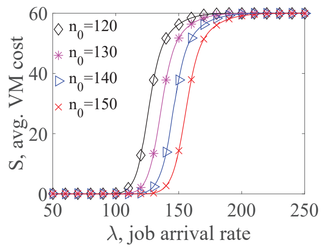
Figs. 4(b), 5(b), 6(b), 7(b), 8(b) show the impacts of λ on W_q. Interestingly, the trend of the curves can generally be divided into four phases: zero phase, ascent phase, descent phase, and saturation phase. In the zero phase, W_q is zero because incoming jobs are served immediately by available capacities. In the ascent phase, W_q raises sharply due to the setup time of VMs. Specifically, when λ approaches to $n_0\mu$ and then larger than $n_0\mu$, VMs start to be powered on and to serve jobs. In doing so, however, W_q still grows sharply because jobs have to wait for turning on processes of VMs. Later, W_q starts to decrease due to new running VMs as shown as the third (descent) phase. In the forth (ascent) phase, W_q starts to grow again and then saturates at a bound. The reason of ascent is that the system is not able to serve the coming jobs when $\lambda \geq (n_0 + k)\mu$. Finally, the curves reach to saturation because the capacity of the system is too full to handle the jobs and the value of W_q is limited by the total system capacity K.

In Figs. 4(c), 5(c), 6(c), 7(c), and 8(c), we study the impacts of λ on P_b. The trends of the curves are relatively simple compared with the above two metrics. Generally, the curves are growing as λ increases. In particular, P_b initiates at 0 and starts to increase when $\lambda > (n_0 + k)\mu$. The reason is that the system starts to reject jobs when the queue is full.

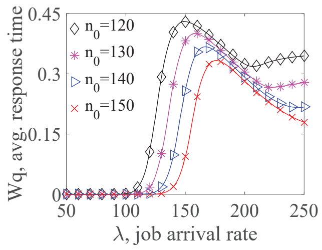
Figs. 4(d), 5(d), 6(d), 7(d), 8(d) illustrate the impacts of λ on P_d. One can see that the trends of the curves are similar with that of W_q. Note that job requests start to quit the queue if the waiting time exceeds their deadline constraints. So P_d is highly related to W_q. If W_q is large then jobs are dropped with high probability. This also explains why the trends are similar. Please refer to the above discussion of W_q for P_d.

B. Impacts of the Number of VNF Instances k

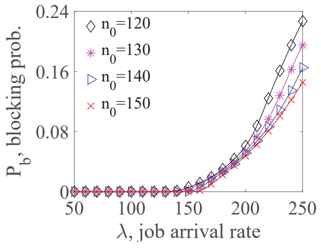
The figures in Fig. 4 depict the impacts of k on performance metrics S, W_q, P_b, and P_d, respectively. We can see that increasing k from 10 to 60 leads to the gains of S while decreasing W_q, P_b, and P_d accordingly. A larger k means that more VMs could be used to handle the growing job requests so W_q, P_b, and P_d are improved. If a operator wants to adjust



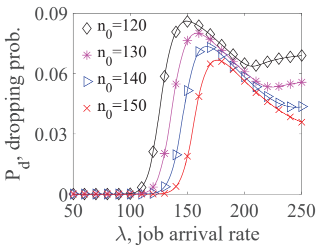
(a) Impacts on S.



(b) Impacts on Wq.

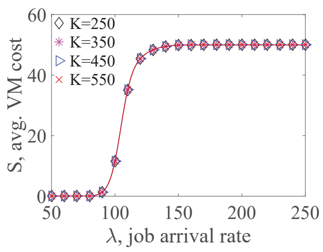


(c) Impacts on P_b.

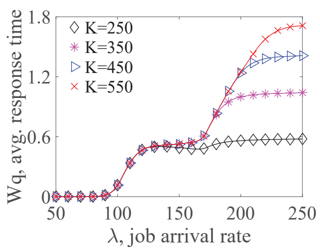


(d) Impacts on P_d.

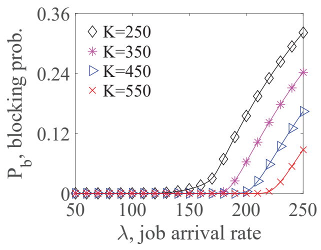
Fig. 7: Impacts of n_0 on the performance metrics ($k = 60$). The *lines* denote analytical results, and the *points* represent simulation results.



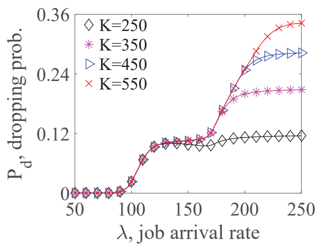
(a) Impacts on S.



(b) Impacts on Wq.



(c) Impacts on P_b.



(d) Impacts on P_d.

Fig. 8: Impacts of K on the performance metrics ($k = 50$). The *lines* denote analytical results, and the *points* represent simulation results.

budget constraint S, the operator can specify a suitable k based on (5).

C. *Impacts of Abandon Rate θ*

In Figs. 5(a), 5(b), 5(c), and 5(d), we study the impacts of abandon rate θ on S, W_q, P_b, and P_d, respectively. Recall that a job request is assumed to have a deadline constraint with mean $1/\theta$, meaning that the job will stop waiting in the queue if the waiting time exceeds its deadline. We observe that increasing θ decreases S, W_q, and P_b while enlarging P_d. Specifically, as shown in Fig. 5(a), θ has no impacts on S when $\lambda < n_0\mu$ or $\lambda > (n_0 + k)\mu$. The reason is that S only depends on the number of running VMs. Whereas, when $n_0\mu < \lambda < (n_0 + k)\mu$, a larger θ leads to less S because more jobs are dropped from the system. In addition, the impacts of θ on W_q is illustrated in Fig. 5(b). A larger θ makes a smaller W_q. The reason is that when more jobs quit from the queue, the rest of the jobs need to wait less time. Fig. 5(c) shows that increasing θ leads to less P_b. The reason is straightforward. More jobs quitting from the queue means that the system has more available capacities to handle the incoming jobs. In Fig. 5(d), we observe that a larger θ means more P_d. It coincides with the definition of P_d.

D. *Impacts of VM Setup Rate α*

Figs. 6(a), 6(b), 6(c), 6(d) illustrate the impacts of α on S, W_q, P_b, and P_d, respectively. Recall that VMs are assumed to have a setup time with mean value $1/\alpha$. To reduce the setup time, NFV Management and Orchestration can perform scale-up procedure to add resources (e.g., CPU, memory) to make VMs more powerful. We observe that less setup time decreases S, W_q, P_b, and P_d. The reason is that short setup time leads to that VMs can be quicker to be available for handling the jobs, resulting in less operation cost (see Fig. 6(a)), lower waiting time for jobs (see Fig. 6(b)), smaller blocking probability (see Fig. 6(d)), and reduced dropping probability as shown in Fig. 6(d).

E. *Impacts of Capacities of Legacy Equipment n_0*

Figs. 7(a), 7(b), 7(c), 7(d) show the impacts of n_0 on S, W_q, P_b, and P_d, respectively. We observe that the curves initiate at 0 then stay at 0 for a period and start to grow up as λ increases. n_0 decides the length of the period when the curves start to ascend. The reason is that the legacy equipment can handle incoming jobs within its capacity. When λ exceeds the capacity of the legacy equipment, the performance metrics S, W_q, P_b, and P_d start to grow up.

F. *Impacts of System Capacity K*

In Figs. 8(a), 8(b), 8(c), and 8(d), we investigate the impacts of K on S, W_q, P_b, and P_d, respectively. As shown in Fig. 8(a), we observe that K has limited impacts on S. As we discussed in Section VI-B, S is mainly decided by k. Figs. 8(b), 8(c), and 8(d) show that K has significant impacts on W_q, P_b as well as P_d. Different K makes huge gaps between the curves. Moreover, a large K leads to a larger W_q as well as P_d but makes P_b smaller. The reason is that it enables more jobs waiting in the queue rather than dropping them.

VII. Conclusions

In this paper, we have proposed DBCA for addressing the tradeoff between operation budget constraint S and system performance which is evaluated by three performance metrics: the average job response time W_q, blocking probability P_b, and dropping probability P_d. Our work addresses the research gap by considering both VM setup time and the capacity of legacy equipment in NFV enabled EPC scenarios. Compared with our previous work [14], the model quantifies a more practical case. Our results show that the analytical model provides a quick way to help mobile operators to plan and design network optimization strategies without wide deployment, saving on cost and time. Moreover, based on our analytical model, mobile operators can easily estimate operation budget given desired system performance, vice versa.

As our future work, one extension is to generalize the VM setup time and the arrival and service time. Right now there is no literature to support that they are exponential random variables. These results could be generalized by using orthogonal polynomial approaches [29]. Also, we plan to relax the assumption of VM scaling in/out capability, i.e., from one VNF instance per time to arbitrary instances per time. We plan to complete these works in follow-up papers.

Acknowledgements

The research of Tuan Phung-Duc is supported in part by JSPS KAKENHI Grant Numbers 2673001. Yi Ren's research is co-sponsored by MOST 105-2221-E-009-101-MY3, MOST 104-2221-E-009-100-MY3, MOST 104-3115-E-009-006, and MOST 104-3115-E-009-002.

References

[1] Animoto's facebook scale-up. [Online]. Available: http://blog.rightscale.com/2008/04/23/animoto-facebook-scale-up/.

[2] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Trans. Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1107–1117, 2013.

[3] F. Jokhio, A. Ashraf, S. Lafond, I. Porres, and J. Lilius, "Prediction-based dynamic resource allocation for video transcoding in cloud computing," in *Proc. Euromicro Int'l Conf. Parallel, Distributed and Network-Based Processing (PDP)*, Feb. 2013.

[4] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *Proc. IEEE Int'l Conf. Cloud Computing (CLOUD)*, Jul. 2011.

[5] J. M. Tirado, D. Higuero, F. Isaila, and J. Carretero, "Predictive data grouping and placement for cloud-based elastic server infrastructures," in *Proc. IEEE/ACM Int'l Symp. Cluster, Cloud and Grid Computing (CCGrid)*, May 2011.

[6] D. Niu, H. Xu, B. Li, and S. Zhao, "Quality-assured cloud bandwidth auto-scaling for video-on-demand applications," in *Proc. IEEE INFOCOM*, Mar. 2012.

[7] R. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using ARIMA model and its impact on cloud applications' QoS," *IEEE Trans. Cloud Computing*, vol. 3, no. 4, pp. 449 – 458, Aug. 2014.

[8] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 155–162, 2012.

[9] A. A. Bankole and S. A. Ajila, "Cloud client prediction models for cloud resource provisioning in a multitier web application environment," in *Proc. IEEE Int'l Symp. Service Oriented System Engineering (SOSE)*, Mar. 2013.

[10] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "Cloudscale: elastic resource scaling for multi-tenant cloud systems," in *Proc. ACM Symp. on Cloud Computing (SoCC)*, Oct. 2011.

[11] A. Khan, X. Yan, S. Tao, and N. Anerousis, "Workload characterization and prediction in the cloud: A multiple time series approach," in *Proc. IEEE NOMS*, Apr. 2012.

[12] A. Gandhi, S. Doroudi, M. Harchol-Balter, and A. Scheller-Wolf, "Exact analysis of the M/M/k/setup class of Markov chains via recursive renewal reward," *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 1, pp. 153–166, 2013.

[13] T. Phung-Duc, "Exact solutions for M/M/c/setup queues," *Springer Telecommunication Systems*, pp. 1–16, May 2016.

[14] Y. Ren, T. Phung-Duc, J.-C. Chen, and Z.-W. Yu, "Dynamic Auto Scaling Algorithm (DASA) for 5G mobile networks," in *Proc. IEEE GLOBECOM '16*.

[15] I. Mitrani, "Managing performance and power consumption in a server farm," *Annals of Operations Research*, vol. 202, no. 1, pp. 121–134, 2013.

[16] ——, "Service center trade-offs between customer impatience and power consumption," *Elsevier Performance Evaluation*, vol. 68, no. 11, pp. 1222 – 1231, Nov. 2011.

[17] ——, "Trading power consumption against performance by reserving blocks of servers," *Springer Computer Performance Engineering, LNCS*, vol. 7587, pp. 1–15, 2013.

[18] J. Hu and T. Phung-Duc, "Power consumption analysis for data centers with independent setup times and threshold controls," in *Proc. Int'l Conf. Numerical Analysis And Applied Mathematics (ICNAAM)*, Sep. 2014.

[19] T. Phung-Duc, "Multiserver queues with finite capacity and setup time," in *Proc. Int'l Conf. Analytical and Stochastic Modelling Techniques and Applications (ASMTA)*, May 2015.

[20] Z. Hill, J. Li, M. Mao, A. Ruiz-Alvarez, and M. Humphrey, "Early observations on the performance of Windows Azure," in *Proc. ACM HPDC*, Jun. 2010.

[21] 3GPP TR 32.842 V13.1.0, "Telecommunication management; Study on network management of virtualized networks (Release 13)," Tech. Rep., Dec. 2015.

[22] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal, "NFV: state of the art, challenges, and implementation in next generation mobile networks (vEPC)," *IEEE Network*, vol. 28, no. 6, pp. 18–26, Nov. 2014.

[23] I. Chih-Lin, C. Rowell, S. Han, Z. Xu, G. Li, and Z. Pan, "Toward green and soft: a 5G perspective," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 66–73, Feb. 2014.

[24] "Design and Analysis of Deadline and Budget Constrained Autoscaling (DBCA) Algorithm for 5G mobile networks," National Chiao Tung University, Tech. Rep., 2016. [Online]. Available: https://arxiv.org/abs/1609.09368

[25] "The network simulator - ns-2." Available: http://www.isi.edu/nsnam/ns/.

[26] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *Proc. ACM SIGCOMM*, 2015.

[27] M. Gilani, C. Inibhunu, and Q. H. Mahmoud, "Application and network performance of Amazon elastic compute cloud instances," in *Proc. IEEE Int'l Conf. Cloud Networking (CloudNet)*, 2015.

[28] M. Mao and M. Humphrey, "A performance study on the VM startup time in the cloud," in *IEEE Int'l Conf. Cloud Computing (CLOUD),*, 2012.

[29] J. Pender, "Gram charlier expansion for time varying multiserver queues with abandonment," *SIAM Journal on Applied Mathematics*, vol. 74, no. 4, pp. 1238–1265, 2014.