Design and Analysis of Dynamic Auto Scaling Algorithm (DASA) for 5G Mobile Networks

Yi Ren, Member, IEEE, Tuan Phung-Duc, Zheng-Wei Yu, and Jyh-Cheng Chen, Fellow, IEEE

Abstract

Network Function Virtualization (NFV) enables mobile operators to virtualize their network entities as Virtualized Network Functions (VNFs), offering fine-grained on-demand network capabilities. VNFs can be dynamically scale-in/out to meet the performance desire and other dynamic behaviors. However, designing the auto-scaling algorithm for desired characteristics with low operation cost and low latency, while considering the existing capacity of legacy network equipment, is not a trivial task. In this paper, we propose a VNF Dynamic Auto Scaling Algorithm (DASA) considering the tradeoff between performance and operation cost. We develop an analytical model to quantify the tradeoff and validate the analysis through extensive simulations. The NFV enabled Evolved Packet Core (EPC) is modeled as queueing model while legacy network equipment are considered as reserved a block of servers; and VNF instances are powered on and off according to the number of job requests present. The results show that the DASA can significantly reduce operation cost given the latency upper-bound. Moreover, the models provide a quick way to evaluate the cost-performance tradeoff and system design without wide deployment, which can save cost and time.

Index Terms

Auto Scaling Algorithm, Modeling and Analysis, Network Function Virtualization, 5G, Cloud Networks, Virtualized EPC

I. INTRODUCTION

Long Term Evolution-Advanced (LTE-A) has become a commonly used communication technology worldwide and is continuously expanding and evolving. A recent report states that
95 operators have commercially launched LTE-A networks in 48 countries and that the total smartphone traffic is expected to rise 11 times between 2015 and 2021 [1]. Accordingly, operators are improving their network infrastructure to increase capacity and to meet the demand for fast-growing data traffic.

The emergence of Network Functions Virtualization (NFV) enables mobile operators to manage their network equipment in a fine-grained and efficient way [2]. Indeed, legacy network infrastructure equipment suffers from the nature of user experience where data traffic usually have peaks during a day while having relative low utilization in the rest of time (e.g., in the midnight). To guarantee the Quality of user Experience (QoE), operators usually leave spare capacities for tackling the peak traffic while deploying network equipment. Accordingly, the network equipment are under low utilization during the non-busy period. NFV enables operators virtualize hardware resources and makes special-purpose network equipment toward software solutions, i.e., Virtualization Network Function (VNF) instances running on off-the-shelf servers. A VNF instance running on Virtual Machine (VMs) can scale-out/in (turn on/off) to adjust the VNF’s computing and networking capabilities, saving on both energy and resources. A classic case is Animoto, an image-processing service provider, experienced a demand surging from 50 VM instances to 4000 instances (Amazon EC2 instance) in three days, April 2008; after the peak, the demand fell sharply to an average level [3]. Animoto only paid for 4000 instances for the peak time.

Given the fact that auto-scaling VNF instance can decrease operation cost while meeting the demand for VNF data service, designing good strategies for allocating VNF instance adaptive to on-demand data services is of importance, which, however, is not a trivial task. Specifically, the operation cost is reduced by decreasing the number of power-on VNF instances. On the other hand, resource under-provisioning may cause Service Level Agreements (SLAs) violations. Therefore, the goal of desired strategies is to reduce operation cost while maintaining acceptable levels of performance such as the average response time. Then, a cost-performance tradeoff is formed: The VNF performance is improved by scaling-out the number of VNF instances while operation cost is reduced by the opposite way.

In this paper, we study the cost-performance tradeoff while considering both VM setup time and the legacy equipment capacity. To the best of our knowledge, this is the first work from this perspective. We propose a dynamic auto-scaling approach for addressing the cost-performance tradeoff, namely Dynamic Auto Scaling Algorithm (DASA). DASA considers available legacy
network equipment as a block and powered on all the time, while virtualized resources are divided into $k$ VNF instances. The VNF instances are scaled in and out depending on the number of jobs in the system. A central issue is how to specify a suitable $k$ for the tradeoff. We derive detailed analytical models to answer this question. The cost-performance tradeoff is quantified as an operation cost metric and a performance metric of which closed-form solutions are provided and validated against extensive ns-2 simulations. Moreover, we develop a recursive method that reduces the complexity of the computational procedure from $O(k^3K^3)$ to $O(kK)$, where $K$ is the system capacity. The models enable wide applicability in various scenarios, and therefore, have important theoretical significance. Furthermore, this work offers network operators guidelines to design optimal VNF auto-scaling strategy by their management policies in a systematical way.

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

II. 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. I 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.

A. Legacy EPC

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.

The P-GW is a gateway providing connectivity between User Equipment terminals (UEs) and an external PDN. An “S2c” interface is defined [4] to support the communication between a P-GW and a UE. The S-GW is responsible for user data functions enabling routing and packet
Fig. 1: A simplified example of NFV enabled LTE architecture.

Fig. 2: Mapping of 3GPP management reference model and ETSI NFV management and organization.

forwarding to the P-GW. MME handles UE mobility, for instance, idle mode UE paging and tagging procedure. PCRF is a policy and charging control element for policy enforcement, flow-based charging, and service data flow detection.
B. vEPC

Generally, the vEPC can be divided into two main components: NFV management and orchestration and 3GPP management reference model [5].

1) NFV Management and Orchestration: NFV management and orchestration controls the lifecycles of VNFs. That is, it defines if a VNF should scale-out/in/up/down, allocating hardware resources to the VNF. Additionally, it manages both hardware and software resources to support VNFs. In other words, it can be considered as a bridge between resources and VNFs. The actions of VNF scaling 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, in the sense that virtualized hardware resources are freed and no longer needed.
- 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.

2) 3GPP Management Reference Model: 3GPP introduces NFV management functions and solutions for mobile core networks based on ETSI NFV specification [5]. Fig. 2 illustrates the mapping between 3GPP management reference model and ETSI NFV management and organization. We detail each reference points as follows.

- Network Element (NE) is a discrete LTE-A entity (e.g., S-GW, MME), containing virtualized or non-virtualized network function.
- Network Manager (NM) provides end-user functions for network management.
- Element Manager (EM) is responsible for the management of a set of NMs.
- Domain Manager (DM) manages element and domain in a sub-network.

C. VNF Instance Scaling Procedures

VNF manager allocates resources by using two scaling procedures: VNF instance expansion procedure for adding resources to a VNF, and VNF instance contraction procedure for the resource release from a VNF.

1) VNF instance expansion procedure: Fig. 3 illustrates a simplified VNF instance expansion procedure. VNF capability expansion includes two actions: (1) scale-out: add new a VM, and (2) scale-up: configuration changes to the VM (e.g., add CPU or memory)
Fig. 3: VNF instance expansion procedure.

- Step 1: NM/EM/NM (via NFV Orchestrator, NFVO) sends capability expansion request to the VNF Manager using the operation Scale VNF of the VNF Lifecycle Management interface, see 1(a), 1(b), and 1(c).
- Step 2: The VNF Manager sends lifecycle change notification to EM and NFVO about the start of the scaling operation.
- Step 3-14: The VNF Manager sends a request to the NFVO for the VNF expansion. Then the NFVO checks if free resources are available and send ACK/NACK to the VNF Manager for VNF expansion.
- Step 15: EM configures the VNF with application specific parameters.
Step 16: EM notifies the newly updated and configured additional capacity to NM.

2) VNF instance contraction procedure: Similar with the expansion procedure, the contraction procedure also has two actions: (1) scale-in: shut down and remove VMs, and (2) scale-down: free virtualized resource from existing VMs. Please refer to [5] for the detailed procedure.

III. PROPOSED VNF INSTANCE AUTO-SCALING ALGORITHM

The goal of VNF instance auto-scaling algorithm is to reduce operation cost while providing acceptable levels of performance. Here, the performance is evaluated by average response time per user request. More power-on VNF instances reduce the possibility of SLAs violations. However, this may incur redundant power-on VNF instances, leading to more operation cost. We refer the tradeoff as cost-performance tradeoff. To balance the tradeoff, we first propose a VNF instance Dynamic Auto-Scaling Algorithm (DASA), and then present the optimal DASA.

A. System Model and DASA: Dynamic Auto-Scaling Algorithm

Consider a 5G EPC comprised of both legacy network entities (e.g., MME, PCRF) and VNFs. A VNF, consisting of $k$ VNF instances, is used to add more capacities to its corresponding legacy network entity (see Fig. 4 as an example). We assume that the capacity of the legacy network entity equals to $n_0$ VNF instance capacities. That is, the total capacity of the network entity is $k + n_0 = N$. We assume that $n_1 = n_0 + 1$ and $n_i = n_{i-1} + 1$ ($i = 1, 2, \cdots k$). It should be noted that $n_k = N$. User request arrives with rate $\lambda$. A VNF instance accepts one job at a time with service rate $\mu$. There is a limited First-Come-First-Served (FCFS) queue for those requests that have to wait for processing. The legacy network equipment is always on while its VNF instances will be added (or removed) according to the number of waiting user requests in the buffer. It is
worth to mention that the VNF instances need some setup time to be available so as to process waiting requests.

Two thresholds, 'up' and 'down', or \( U_i \) and \( D_i \), denote the control of the VNF instances \( i = 1, 2, \cdots, k \).

- \( U_i \), 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 drop from \( n_i \) to \( n_{i-1} \), we turn off the \( i \)-th VNF instance.

The system performance is evaluated by two metrics: the average response time in the queue per request, \( W_q \), and the average number of VNF instance consuming power, \( S \). The closed-form solutions of \( W_q \) and \( S \) are given as (10) and (11) in Section IV. Thus, the system performance \( P \) has the form

\[
P = w_1 W_q + w_2 S,
\]

where coefficients \( w_1 \) and \( w_2 \) denote the weight factors for \( W_q \) and \( S \), respectively. Increasing \( w_1 \) (or \( w_2 \)) emphasizes more on \( W_q \) (or \( S \)). Here, we do not specify either \( w_1 \) or \( w_2 \) due to the fact that such a value should be determined by a mobile operator and must take management policies into consideration. Next, we provide an example of specifying \( k \) based on different weights accordingly to a operator’s management policies.

### B. Optimal DASA

We propose Algorithm 1 for operators to specify \( k \) based on the weight factors \( w_1 \) and \( w_2 \). The Algorithm 1 takes input \( S \), \( W_q \), \( \delta = \frac{w_2}{w_1} \) and outputs an optimal value \( k_{opt} \). We denote \( S \) and \( W_q \) as the maximum values of \( S \) and \( W_q \) in the system. Note that \( S \) and \( W_q \) can be measured and defined by operators for their system. Initially, we set \( k \) to 0 and \( k \) is bound by \( K - n_0 \).
Algorithm 1 Selecting an optimal $k$ value

**Input:** $S$, $W_q$, $\delta$, $K$

**Output:** $k_{op}$

1. Initialize $k$ as 0
2. while $k \leq K - n_0$ do
3.     $S' = S/S$
4.     $W_q' = W_q/W_q$
5.     if $S'/W_q' < \delta$ then
6.         $k = k + 1$
7.     else
8.         return $k$
9.     end if
10. end while

Fig. 5: Selection of the optimal $k$ for a given $\delta$.

As $k$ starts from 0, the ratio of $S'/W_q'$ increases every loop accordingly. The loop does not stop until it finds the lowest $k$ value $k_{op}$.

Fig. 5 illustrates a graphical plot of $S$ and $W_q$, where each point in the curve is depicted from paired $S$ and $W_q$ associated with $k$. Two curves with $n_0 = 100$ and $n_0 = 120$ are demonstrated and three lines with example weighting parameters ($\delta = \frac{2}{5}, 1, \frac{5}{3}$) are depicted. The intersection of the dotted line and the curve is the optimal $k$ given weight factor coefficient $\delta$. Take the red cycle line ($n_0 = 120$) as an example, the optimal $k$ is 32 and 16 for $\delta = 1$ and $\delta = \frac{2}{5}$, respectively.

IV. **Analytical Model**

In our previous paper [6], we evaluated the performance of the proposed DASA by extensive simulations. To further understand the performance of DASA, we propose a detailed mathematical analysis model in this section. The goal of the analytical model is to cross-validate the accuracy
| Notation | Explanation |
|----------|-------------|
| $N$      | The number of servers in server center |
| $K$      | The number of maximum jobs can be accommodated in the system |
| $k$      | The number of VNF instances |
| $P$      | System performance |
| $W$      | Average response time per job |
| $W_q$    | Average response time in the queue per job |
| $S$      | Average VM cost |
| $w_1$    | Weight factor for $W_q$ |
| $w_2$    | Weight factor for $S$ |
| $n_0$    | The number of permanently operative servers |
| $U_i$    | The up threshold to control the reserve sub-blocks |
| $D_i$    | The down threshold to control the reserve sub-blocks |
| $m_i$    | The $i$-th reserve sub-block ($i = 1, 2, \cdots k$). |
| $\lambda$ | Job arrival rate |
| $\mu$    | Service rate for each server |
| $\alpha$ | Setup rate for each virtual server |

TABLE I: List of Notations

The simulation experiments and to analyze both the operation cost and the system performance for DASA. Given the analytical model, one can quickly obtain the operation cost and system performance, without real deployment, saving on cost and time. For instance, even in our simplified simulation settings (less arrival rate) in Section V, it is still very time consumption to get simulation results, e.g., tens of hours per simulation.

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 Poisson distribution with rate $\lambda$. A VNF instance (server) accepts one job at a time, and its service rate follows the exponential distribution with rate $\mu$. 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 an 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 VNF instances 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:

$$S = \{(i, j); 1 \leq i \leq k, j = n_i, n_i + 1, \ldots, K - 1, K\}$$

$$\cup \{(0, j); j = 0, 1, \ldots, K - 1, K\}.$$

Fig. 6 shows the transition among states for the case where $N = 4, n_0 = 2, m_1 = m_2 = 1$ and $K = 7$. Let $\pi_{i,j} = \lim_{t \to \infty} P(C(t) = i, L(t) = j)$ $((i, j) \in S)$ denote the joint stationary distribution of $\{X(t)\}$. Here, we derive a recursion for calculating the joint stationary distribution $\pi_{i,j}$ $((i, j) \in S)$. The balance equations for states with $i = 0$ read as follows.

$$\lambda \pi_{0,j-1} = j \mu \pi_{0,j}, \quad \text{for } j = 0, 1, \ldots, n_0,$$

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

$$\lambda \pi_{0,K-1} = (n_0 \mu + (N - n_0) \alpha) \pi_{0,K},$$

leading to

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

\[
b_j^{(0)} = \frac{\lambda}{j\mu}, \quad j = 1, 2, \ldots, n_0,
\]

and

\[
b_j^{(0)} = \frac{\lambda}{\lambda + n_0\mu + \min(j - n_0, N - n_0)\alpha - n_0\mu b_{j+1}^{(0)}}, \quad j = K - 1, K - 2, \ldots, n_0 + 1,
\]

where

\[
b_K^{(0)} = \frac{\lambda}{n_0\mu + (N - n_0)\alpha}.
\]

Furthermore, it should be noted that \( \pi_{1,1} \) is calculated using the local balance equation in and out the set \( \{(0, j); j = 0, 1, \ldots, K\} \) 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 \( \pi_{0,j} \) \( j = 1, 2, \ldots, K \) and \( \pi_{1,1} \) in terms of \( \pi_{0,0} \).

Next, we consider the case \( i = 1 \).

**Lemma 1.** We have

\[
\pi_{1,j} = a_j^{(1)} + b_j^{(1)}\pi_{1,j-1}, \quad j = 2, 3, \ldots, K - 1, K,
\]

where

\[
a_j^{(1)} = \frac{n_1\mu a_j^{(1)} + \min(j - n_0, N - n_0)\alpha\pi_{0,j}}{n_1\mu + \lambda + \min(j - n_1, N - n_1)\alpha - n_1\mu b_j^{(1)}}, \quad (1)
\]

\[
b_j^{(1)} = \frac{\lambda}{n_1\mu + \lambda + \min(j - n_1, N - n_1)\alpha - n_1\mu b_{j+1}^{(1)}}, \quad (2)
\]

for \( j = K - 1, K - 2, \ldots, 2 \) and

\[
a_K^{(1)} = \frac{(N - n_0)\alpha\pi_{0,K}}{n_1\mu + (N - n_1)\alpha}, \quad b_K^{(1)} = \frac{\lambda}{n_1\mu + (N - n_1)\alpha}.
\]
Proof. We prove using mathematical induction. Balance equations are given as follows.

\[(\lambda + n_1 \mu + \min(j - n_1, N - n_1) \alpha) \pi_{1,j} = \lambda \pi_{1,j-1} + n_1 \mu \pi_{1,j+1} + \min(j - n_0, N - n_0) \alpha \pi_{0,j},\]
for \(2 \leq j \leq K - 1\),

\[(\mu + \min(K - n_1, N - n_1) \alpha) \pi_{1,K} = \lambda \pi_{1,K-1} + (N - n_0) \alpha \pi_{0,K}.\]

(4)

It follows from (4) that

\[\pi_{1,K} = a^{(1)}_K + b^{(1)}_K \pi_{1,K-1},\]

leading to the fact that Lemma 1 is true for \(j = K\). Assuming that Lemma 1 is true for \(j + 1\), i.e., \(\pi_{1,j+1} = a^{(1)}_{j+1} + b^{(1)}_{j+1} \pi_{1,j}\). It then follows from (3) that Lemma 1 is also true for \(j\), i.e., \(\pi_{1,j} = a^{(1)}_j + b^{(1)}_j \pi_{1,j-1}\).

Theorem 1. We have the following bound.

\[a^{(1)}_j \geq 0, \quad 0 \leq b^{(1)}_j \leq \frac{\lambda}{n_1 \mu + \min(j - n_1, N - n_1) \alpha},\]

for \(j = 2, 3, \ldots, K - 1, K\).

Proof. We use mathematical induction. It is easy to see that the theorem is true for \(j = K\). Assuming that the theorem is true for \(j + 1\), i.e.,

\[0 \leq a^{(1)}_{j+1}, \quad 0 \leq b^{(1)}_{j+1} \leq \frac{\lambda}{n_1 \mu + \min(j - n_1, N - n_1) \alpha},\]

where \(j = 1, 2, \ldots, K - 1\).

Thus, we have \(\mu b^{(1)}_{j+1} < \lambda\). From this inequality, (1) and (2), we obtain

\[b^{(1)}_j \leq \frac{\lambda}{n_1 \mu + \min(j - n_1, N - n_1) \alpha},\]
and \(a^{(1)}_j \geq 0\). \(\square\)

It should be noted that \(\pi_{2,2}\) can be calculated using the local balance between the flows in
and out the set of states \( \{(i, j); i = 0, 1, j = i, i + 1, \ldots, K\} \) as follows.

\[
n_2 \mu \pi_{2,n_2} = \sum_{j=n_2}^{K} \min(j-n_1, N-n_1) \alpha \pi_{1,j}.
\]

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

We consider the general case where \( 2 \leq i \leq k - 1 \). Similar to the case \( i = 1 \), we can prove the following result by mathematical induction.

**Lemma 2.** We have

\[
\pi_{i,j} = a_j^{(i)} + b_j^{(i)} \pi_{i,j-1}, \quad j = i + 1, i + 2, \ldots, K - 1, K,
\]

where

\[
a_j^{(i)} = \frac{n_i \mu a_{j+1}^{(i-1)} + \min(N - n_{i-1}, j - n_{i-1}) \alpha \pi_{i-1,j}}{\lambda + \min(N - n_i, j - n_i) \alpha + n_i \mu - n_i \mu b_{j+1}^{(i-1)}}, \quad (5)
b_j^{(i)} = \frac{\lambda}{\lambda + \min(N - n_i, j - n_i) \alpha + n_i \mu - n_i \mu b_{j+1}^{(i-1)}}, \quad (6)
\]

and

\[
a_K^{(i)} = \frac{(N - n_{i-1}) \alpha \pi_{i-1,K}}{(N - n_i) \alpha + n_i \mu}, \quad b_K^{(i)} = \frac{\lambda}{(N - n_i) \alpha + n_i \mu}.
\]

**Proof.** The balance equation for state \((i, K)\) is given as follows.

\[
((N - n_i) \alpha + n_i \mu) \pi_{i,K} = \lambda \pi_{i,K-1} + (c - n_{i-1}) \alpha \pi_{i-1,K},
\]

leading to the fact that Lemma 2 is true for \( j = K \). Assuming that

\[
\pi_{i,j+1} = a_{j+1}^{(i)} + b_{j+1}^{(i)} \pi_{i,j}, \quad j = i + 1, i + 2, \ldots, K - 1.
\]

It then follows from

\[
(\lambda + \min(N - n_i, j - n_i) \alpha + n_i \mu) \pi_{i,j} = \lambda \pi_{i,j-1} + n_i \mu \pi_{i,j+1} + \min(N - n_{i-1}, j - n_{i-1}) \alpha \pi_{i-1,j},
\]

\[
j = K - 1, K - 2, \ldots, i + 1,
\]

that

\[
\pi_{i,j} = a_j^{(i)} + b_j^{(i)} \pi_{i,j-1}.
\]
Theorem 2. We have the following bound.

\[ a_j^{(i)} > 0, \quad 0 < b_j^{(i)} < \frac{\lambda}{n_i \mu + \min(j - i, N - n_i) \alpha}, \]

for \( j = n_i + 1, n_i + 2, \ldots, K, i = 1, 2, \ldots, k - 1. \)

Proof. We also prove using mathematical induction. It is clear that Theorem 2 is true for \( j = K. \)
Assuming that Theorem 2 is true for \( j + 1, \) i.e.,

\[ a_j^{(i+1)} > 0, \quad 0 < b_j^{(i+1)} < \frac{\lambda}{n_i \mu + \min(j + 1 - n_i, N - n_i) \alpha}, \]

for \( j = i + 1, i + 2, \ldots, K - 1, i = 1, 2, \ldots, c - 1. \) It follows from the second inequality that \( i \mu b_j^{(i+1)} < \lambda. \) This together with formulae (5) and (6) yield the desired result.

It should be noted that \( \pi_{i+1,i+1} \) is calculated using the following local balance equation in and out the set of states:

\[ \{ (k, j); k = 0, 1, \ldots, i; j = k, k + 1, \ldots, K \} \]

as follows.

\[ n_{i+1} \mu \pi_{i+1,i+1} = \sum_{j=n_{i+1}}^{K} \min(j - n_i, N - n_i) \alpha \pi_{i,j}. \]

Remark. We have expressed \( \pi_{i,j} \) \((i = 0, 1, \ldots, c - 1, j = i, i + 1, \ldots, K)\) and \( \pi_{i+1,i+1} \) in terms of \( \pi_{0,0}. \)

Finally, we consider the case \( i = c. \) Balance equation for state \((c, K)\) yields,

Lemma 3. We have

\[ \pi_{k,j} = a_j^{(k)} + b_j^{(k)} \pi_{k,j-1}, \quad j = n_k + 1, n_k + 2, \ldots, K, \]

where

\[ a_j^{(k)} = \frac{n_k \mu a_{j+1}^{(k)} + (N - n_k - 1) \alpha \pi_{k-1,j}}{\lambda + n_k \mu - n_k \mu b_{j+1}^{(k)}}, \quad j = K - 1, K - 2, \ldots, n_k + 1, \quad (7) \]

\[ b_j^{(k)} = \frac{\lambda}{\lambda + n_k \mu - n_k \mu b_{j+1}^{(k)}}, \quad j = K - 1, K - 2, \ldots, n_k + 1, \quad (8) \]
and
\[ a^{(k)}_K = \frac{\alpha \pi_{k-1,K}}{n_{k\mu}}, \quad b^{(k)}_K = \frac{\lambda}{n_{k\mu}}. \]

Proof. The global balance equation at state \((k, K)\) is given by
\[ n_{k\mu}\pi_{k,K} = (N - n_{k-1})\alpha \pi_{k-1,K} + \lambda \pi_{k,K-1}, \]
leading to
\[ \pi_{k,K} = a^{(k)}_K + b^{(k)}_K \pi_{k,K-1}. \]

Assuming that \(\pi_{k,j+1} = a^{(k)}_{j+1} + b^{(k)}_{j+1} \pi_{k,j}\), it follows from the global balance equation at state \((k, j)\),
\[ (\lambda + n_{k\mu})\pi_{k,j} = \lambda \pi_{k,j-1} + n_{k\mu} \pi_{n_{k,j+1}} + (N - n_{k-1})\alpha \pi_{k-1,j}, \]
\[ j = n_k + 1, n_k + 2, \ldots, K - 1, \]
that \(\pi_{k,j} = a^{(k)}_j + b^{(k)}_j \pi_{k,j-1}\) for \(j = n_k + 1, n_k + 2, \ldots, K\).

Theorem 3. We have the following bound.
\[ a^{(k)}_j > 0, \quad 0 < b^{(k)}_j < \frac{\lambda}{n_{k\mu}}, \quad j = n_k + 1, n_k + 2, \ldots, K - 1. \]

Proof. We also prove using mathematical induction. It is clear that Theorem 3 is true for \(j = K\). Assuming that Theorem 3 is true for \(j + 1\), i.e.,
\[ a^{(k)}_{j+1} > 0, \quad 0 < b^{(k)}_{j+1} < \frac{\lambda}{n_{k\mu}}, \]
\[ j = n_k + 1, n_k + 2, \ldots, K - 1, \]
It follows from the second inequality that \(n_{k\mu}b^{(k)}_{j+1} < \lambda\). This together with formulae (7) and (8) yield the desired result.

We have expressed all the probability \(\pi_{i,j}\) \(((i, j) \in S)\) in terms of \(\pi_{0,0}\) which is uniquely determined by the normalizing condition.
\[ \sum_{(i,j) \in S} \pi_{i,j} = 1. \]
Let $E[L]$ denote the mean number of jobs in the system. We have

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

Let $P_b$ denote the blocking probability. We have

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

It follows from Little’s law that

$$W = \frac{E[L]}{\lambda(1 - P_b)} = \frac{\sum_{i=0}^{n_0-1} \pi_{0,j} j + \sum_{i=0}^{K} \sum_{j=n_i}^{n} \pi_{i,j} j}{\lambda(1 - \sum_{i=0}^{k} \pi_{i,K})}. \quad (9)$$

We obtain

$$W_q = W - \frac{1}{\mu}. \quad (10)$$

The mean number of VNF instances is given by

$$S = \sum_{(i,j) \in S} \pi_{i,j} (n_i - n_0) + \sum_{i=0}^{K} \sum_{j=n_i}^{n} \pi_{i,j} \min(j - n_i, N - n_i), \quad (11)$$

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.

We have solved a system of $n_0 + \sum_{i=0}^{k} (K - n_i) = O(kK)$ unknown variables. The computational cost by a conventional method is $O(k^3K^3)$. It is easy to see that the computational complexity of our recursive scheme is only $O(kK)$. Furthermore, our method is numerically stable since it manipulates positive numbers.

V. SIMULATION AND NUMERICAL RESULTS

The analytical results in Section IV are validated by extensive simulations by using ns-2, version 2.35 [7]. Here, we used real measurement results for parameter configuration: $\lambda$ by Facebook data center traffic [8], $\mu$ by the base service rate of a Amazon EC2 VM [9], and $\alpha$ by the average VM startup time [10]. If not further specified, the following parameters are set as the default values for performance comparison: $n_0 = 110$, $\mu = 1$, $\alpha = 0.005$, $K = 250$, $\lambda/\mu$.

\footnote{Due to the simulation time limitation, $\lambda$ and $\mu$ are scaled down accordingly with the same ratio $\lambda/\mu$.}
Fig. 7: Impacts on $S$ while $\frac{1}{\mu}$, $\frac{1}{\lambda}$, and $\frac{1}{\alpha}$ are exponential distribution.

$\lambda = 50 \sim 250$ (see Table 1 for details). The simulation time is 300,000 seconds. And $15 \sim 75$ millions job requests were generated during the extensive simulations.

Figs. 7-8 illustrate both the simulation and analytical results in terms of the performance metrics: average VM cost $S$ and average response time in a queue per job $W_q$, respectively. In the figures, the lines denote analytical results, and the points represent simulation results. Each simulation result in the figures is the mean value of the results in 300,000 seconds with 95% confidence level. In the following sections, we show the impacts of $\lambda$, $k$, $K$, $n_0$, $\alpha$ on the performance metrics $S$ and $W_q$, respectively.

A. Impacts of arrival rate $\lambda$

Figs. 7(a)-7(d) show the impacts of $\lambda$ on $S$. Generally, one can see that $S$ is 0 at the beginning, then grows sharply, later raises smoothly and reach at a bound as $\lambda$ increases. The reasons are as follows. When $\lambda < n_0\mu$, the incoming jobs are handled by the legacy equipment. No VMs
(a) Impacts of $k$ on $W_q$ ($n_0 = 100$).

(b) Impacts of $\alpha$ on $W_q$ ($k = 80$).

(c) Impacts of $K$ on $W_q$ ($k = 50$).

(d) Impacts of $n_0$ on $W_q$ ($k = 60$).

Fig. 8: Impacts on $W_q$ while $1/\mu$, $1/\lambda$, and $1/\alpha$ are exponential distribution.

are turned on. Later, VMs are turning on as $\lambda$ approaches to $(n_0 + k)\mu$. Accordingly, the server cost $S$ increases as $\lambda$ grows. Then $S$ stops growing when $\lambda > (n_0 + k)\mu$. Because all the $k$ VMs are turned on so that $S$ is bounded as $k$ VM costs.

Figs. 8(a)-8(d) illustrate the impacts of $\lambda$ on $W_q$. Interestingly, the trend of the curves can generally be divided into three phases: ascent phase, descent phase, and saturation phase. In the first phase, $W_q$ grows sharply due to the setup time of VMs. Specifically, when $\lambda < n_0\mu$, $W_q$ is almost 0 as all jobs are handled by legacy equipment. As $\lambda$ approaches to $n_0\mu$ and then larger than $n_0\mu$, VMs start to be turned on. However, in this phase $W_q$ still raises due to the setup time of VMs. The reason is that VMs just start to be turned on and do not reach their full capacities.

In the second phase, we can see that $W_q$ starts to descend because the VMs start serving jobs.

\[2\text{In Figs. 8(b) and 8(d) only two phases are displayed due to the range of } \lambda. \text{ Given a larger } \lambda, \text{ all the three phases will be shown.}\]
In the third phase, however, $W_q$ starts to ascend again and then saturate at a bound. The reason of ascent is that when $\lambda \geq (n_0 + k)\mu$, the system is not able to handle the coming jobs. Finally, the curves go to saturation because the capacity of the system is too full to handle the jobs and the value of $W_q$ is limited by $K$.

**B. Impacts of the number of VNF instances $k$**

Fig. 7(a) shows the impacts of $k$ on $S$. The impacts of $k$ is shown as the ascend phase before the bound or the gap between the initial point and the bound. A larger $k$ leads to a longer ascend phase (or bigger gap). Moreover, as $\lambda$ grows, a larger $k$ means that more VMs could be used to handle the growing job requests. So $S$ increases accordingly. If an operator wants to bound VM budget $S$, the operator can specify a suitable $k$ based on (11). We can also see that the gaps are the same in Figs. 7(b)-7(d) due to the same $k$ in these figures.

Fig. 7(a) shows the impacts of $k$ on $W_q$. The impacts of $k$ are shown as the length of the second phase as discussed in Sec. V-A. The length of the second phase prolongs as $k$ increases. Because a larger $k$ gives the system more capability to handle the raising job requests. That is, it delays the time that the system capacity reaches its bound. If an operator wants to bound job request response time $W_q$, the operator can choose a suitable $k$ based on (10).

**C. Impacts of VM setup rate $\alpha$**

Recall that $\alpha$ is the setup rate of VMs. To change setup rate, one can adjust resources (e.g., CPU, memory) for VMs. Fig. 7(b) shows the impacts of $\alpha$ on $S$. The impacts of $\alpha$ are shown as the slope of the curves. A larger $\alpha$ means smaller slope, but $\alpha$ has no effects at the beginning and the end of the curves. The reasons are as follows. A larger $\alpha$ means smaller VM setup time. A smaller VM setup time helps VM faster to be turned on and to handle jobs so that the system is more efficient than the VM with large setup time.

Fig. 8(b) illustrates the impacts of $\alpha$ on $W_q$. Again, the impacts of $\alpha$ are shown as the slope of curves. In contrast, a larger $\alpha$ leads to smaller slopes. Also, $\alpha$ decides the maximum value of $W_q$. The reason is that smaller setup time enables VM to handle jobs faster.

**D. Impacts of system capacity $K$**

Fig. 7(c) and Fig. 8(c) depict the impacts of $K$ on $S$ and $W_q$, respectively. Based on our observation on Fig. 7(c), $K$ has limited impacts on $S$. As we discussed in Sec. V-A, $S$ is mainly
decided by $k$. As shown in Fig. 8(c), the impacts are significant on $W_q$. Different $K$ makes huge gaps between the curves. The curves also form as three phases. A large $K$ leads to a larger $W_q$. The reason is that it enables more jobs waiting in the queue rather than dropping them.

\[ E. \text{ Impacts of legacy equipment capacity } n_0 \]

Fig. 7(d) and Fig. 8(d) illustrate the impacts of $n_0$ on $S$ and $W_q$, respectively. We observe that the curves initiate at 0 then fix at 0 for a period and start to grow up as $\lambda$ increases. $n_0$ decides the length of the period when the curves start to grow up. The reason is that the legacy equipment can handle jobs within its capacity. When $\lambda$ exceeds the capacity of the legacy equipment, both $S$ and $W_q$ start to grow up.

Overall, Figs. 7-8 not only demonstrate the correctness of our analysis model, but also show the impacts of $\lambda$, $k$, $K$, $\mu$, $n_0$, $\alpha$ on the performance metrics $S$ and $W_q$. Moreover, although service time $1/\mu$ is assumed to be exponential distribution, the proposed analytical model is also compatible for service time with deterministic, normal, Uniform, Erlang, Gamma distribution. Accordingly, the analytical model enables wide applicability in various scenarios, and therefore, have important theoretical significance. Due to page limitation, more simulation results in terms of service time with various distributions are given in the Appendix.

\[ \text{VI. RELATED WORK} \]

VNF instance (or VM) auto-scaling has been intensive studied [11]–[28]. To handle VM setup lag time, researchers proposed various prediction approaches to predict the VM load in order to boot VMs before existing VMs are overloaded, such as Exponential weighted Moving Average (EMA) [11], [12], Auto-Regressive Moving Average (ARMA) [13], [14], Auto-Regressive Integrated Moving Average (ARIMA) [15]–[18], machine learning [19]–[21], Markov model [22], [23], and queueing model [24]–[28].

The basic idea of EMA, ARMA, and ARIM is moving average, where the most recent input data constrained in a moving window are used to predict the next input data. Specifically, in [11], the authors proposed an EMA-based scheme to predict the CPU loading. The scheme was implemented in Domain Name System (DNS) server and evaluation results showed that the capacities of servers are well utilized. The authors in [12] introduced a novel prediction-based dynamic resource allocation algorithm to scale video transcoding service in the cloud. They used a two-step load prediction method, resulting in a reduced number of required VMs.
ARMA model adds autoregressive (AR) into moving average. A resource allocation algorithm based on ARMA model was addressed in [13], where empirical results show significant benefits both to cloud users and cloud service providers. In [14], the authors addressed a load forecasting model based on ARMA, achieving around 6% prediction error rate and saving up to 44% servers compared with random content-based distribution policy.

Unlike ARMA, ARIMA performs differencing for input data. The authors in [15] proposed a predictive and elastic cloud bandwidth auto-scaling system considering multiple data centers. This is the first work for linear scaling from multiple cloud service providers. The work [16] takes the VM migration overhead into account when designing their auto-scaling scheme, where extensive experiments were conducted to certify the good performance. In [17], a new problem of dynamic workload fluctuations of each VM and the resource conflict handling was addressed. The authors further proposed an ARIMA-based server state predictor to adaptive resource allocation for VMs. Experiments showed that the state predictor achieved excellent prediction results. Another ARIMA-based workload prediction scheme was proposed in [18], where real traces of requests to web servers from the Wikimedia Foundation was used to evaluate its prediction accuracy. The results show that the model achieves up to 91% accuracy.

Machine learning approaches are also used for the design of cloud auto-scaling schemes [19]–[21]. The authors in [19] proposed a neural network and linear regression based auto-scaling scheme. The work [20] implements a Bayesian Network based cloud auto-scaling scheme. In [21], the authors evaluated three machine learning approaches, linear regression, neural network, and Support Vector Machine (SVM), and their results show that SVM-based scheme outperforms the other two.

Markov model also has been widely used in cloud auto-scaling schemes [22], [23]. The authors in [22] developed CloudScale, an automatic elastic resource scaling system for multiple cloud service providers, saving 8-10% total energy consumption and 39-71% workload energy consumption with little impact to the application performance. In [23], the authors proposed a novel multiple time series approach based on Hidden Markov Model (HMM). The scheme well characterizes the temporal correlations in the discovered VM clusters and to predict variations of workload patterns.

However, the mechanisms [11]–[23] either ignore VM setup time or only consider virtualized resource itself while overlooking legacy (fixed) resources. This is not practical in typical cellular networks. Although a scale-out request can be sent right way, a VNF instance cannot be available...
immediately. The 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 [29]. It could happen that the instance is too late to serve the VNF if the lag time is not taken into consideration. The capacity of legacy network equipment is also an issue worth careful consideration. For example, a network operator deployed legacy network equipment wants to increase network capacities by using NFV technique. The desired solution should consider the capacities of both legacy network equipment and VNFs. 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 auto-scaling schemes usually ignore the non-constant issue.

Perhaps the closest models to ours were studied in [24]–[28] that both the capacities of fixed legacy network equipment and dynamic auto-scaling cloud servers are considered. The authors in [24], [25] consider setup time without defections [24] and with defections [25]. Our recent work [27] relaxes the assumption in [24], [25] 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 [24], [25], [27], 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 [26], [28]. However, either setup time is ignored [26], or fixed legacy network capacity is not considered [28].

VII. CONCLUSIONS

In this paper, we have proposed DASA for addressing the tradeoff between performance and operation cost. We developed analytical and simulation models to study the average job response time $W_q$ and operation cost $S$. Our model fills the research gap by taking both VM setup time and the capacity of legacy equipment into consideration in NFV enable EPC scenarios. Our study provides mobile operators with guidelines to bound their operation cost and configure job request response delay. Based on our performance study, the operators can further design optimization strategies without wide deployment, saving on cost and time.

As our future work, one extension is to generalize the VM setup time and the arrival and service time of data. Right now there is no literature to support that they are exponential random variables. These results could be generalized by Markovian Arrival Processes [30].
or approximated by using orthogonal polynomial approaches \cite{31}. 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. Moreover, another extension is to consider impatient customer. That is, waiting jobs have time-limit after that they will abandon from the system. We plan to complete these works in follow up papers.

REFERENCES

[1] Ericsson, “Mobility report on the pulse of the network society,” Ericsson, Tech. Rep., Nov. 2015.
[2] 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./Dec. 2014.
[3] Animoto’s facebook scale-up. [Online]. Available: http://blog.rightscale.com/2008/04/23/animoto-facebook-scale-up/.
[4] 3GPP TS 23.261 V12.0.0, IP flow mobility and seamless Wireless Local Area Network (WLAN) offload; Stage 2 (Release 12), Std., Sep. 2014.
[5] 3GPP TR 32.842 V13.1.0, “Telecommunication management; Study on network management of virtualized networks (Release 13),” Tech. Rep., Dec. 2015.
[6] Y. Ren, T. Phung-Due, Z.-W. Yu, and J.-C. Chen, “Dynamic Auto Scaling Algorithm (DASA) for 5G mobile networks,” submitted to IEEE GLOBECOM ’16.
[7] ”The network simulator - ns-2.” Available: http://www.isi.edu/nsnam/ns/.
[8] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, “Inside the social network’s (datacenter) network,” in Proc. ACM SIGCOMM, 2015.
[9] M. Gilani, C. Inibhunu, and Q. H. Mahmoud, “Application and network performance of Amazon elastic compute cloud instances,” in Proc. IEEE 4th Int’l Conf. Cloud Networking (CloudNet), 2015, pp. 315–318.
[10] M. Mao and M. Humphrey, “A performance study on the VM startup time in the cloud,” in IEEE 5th Int’l Conf. Cloud Computing (CLOUD), 2012, pp. 423–430.
[11] 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.
[12] F. Jokhio, A. Ashraf, S. Lafond, I. Porres, and J. Lilius, “Prediction-based dynamic resource allocation for video transcoding in cloud computing,” in Proc. 21st Euromicro Int’l Conf. Parallel, Distributed and Network-Based Processing (PDP), 2013, pp. 254–261.
[13] 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), 2011, pp. 500–507.
[14] J. M. Tirado, D. Higuero, F. Isaila, and J. Carretero, “Predictive data grouping and placement for cloud-based elastic server infrastructures,” in Proc. 11th IEEE/ACM Int’l Symp. Cluster, Cloud and Grid Computing (CCGrid), 2011, pp. 285–294.
[15] D. Niu, H. Xu, B. Li, and S. Zhao, “Quality-assured cloud bandwidth auto-scaling for video-on-demand applications,” in Proc. IEEE INFOCOM, 2012, pp. 460–468.
[16] Q. Huang, S. Su, S. Xu, J. Li, P. Xu, and K. Shuang, “Migration-based elastic consolidation scheduling in cloud data center,” in Proc. IEEE 33rd Int’l Conf. Distributed Computing Systems Workshops (ICDCSW), 2013, pp. 93–97.
[17] Q. Huang, K. Shuang, P. Xu, J. Li, X. Liu, and S. Su, “Prediction-based dynamic resource scheduling for virtualized cloud systems,” Journal of Networks, vol. 9, no. 2, pp. 375–383, 2014.
In this section, extensive simulation results in terms of mean service time $1/\mu$ with various distributions are introduced. Since there is no empirical evidence for the distribution of the mean service time $1/\mu$, in the proposed analytical model it is assumed to be exponential distribution with mean $1/\mu$ for the sake of simplicity. According to the results, the proposed analytical model is compatible for service time with deterministic (Figs. 9 and 10), normal (Figs. 11 and 12), Uniform (Figs. 13 and 14), Erlang (Figs. 15, 17, 16, and 18), Gamma distribution (Figs. 19, 21, 20, and 22). The above results demonstrate that the proposed model can be used for modeling the system with the aforementioned service time distribution.
Fig. 9: Impacts on $S$ while service time is fixed.
Fig. 10: Impacts on $W_q$ while service time is fixed.
Fig. 11: Impacts on $S$ while service time is normal distribution with $\sigma = 0.1$. 

(a) Impacts of $k$ on $S$ ($n_0 = 100$).

(b) Impacts of $\alpha$ on $S$ ($k = 80$).

(c) Impacts of $K$ on $S$ ($k = 50$).

(d) Impacts of $n_0$ on $S$ ($k = 60$).
Fig. 12: Impacts on $W_q$ while service time is normal distribution with $\sigma = 0.1$. 

(a) Impacts of $k$ on $W_q$ ($n_0 = 100$).

(b) Impacts of $\alpha$ on $W_q$ ($k = 80$).

(c) Impacts of $K$ on $W_q$ ($k = 50$).

(d) Impacts of $n_0$ on $W_q$ ($k = 60$).
(a) Impacts of $k$ on $S$ ($n_0 = 100$).

(b) Impacts of $\alpha$ on $S$ ($k = 80$).

(c) Impacts of $K$ on $S$ ($k = 50$).

(d) Impacts of $n_0$ on $S$ ($k = 60$).

Fig. 13: Impacts on $S$ while service time is uniform distribution with \textit{min} = 0, \textit{max} = 2.
Fig. 14: Impacts on $W_q$ while service time is uniform distribution with $min = 0$, $max = 2$. 

(a) Impacts of $k$ on $W_q$ ($n_0 = 100$).

(b) Impacts of $\alpha$ on $W_q$ ($k = 80$).

(c) Impacts of $K$ on $W_q$ ($k = 50$).

(d) Impacts of $n_0$ on $W_q$ ($k = 60$).
Fig. 15: Impacts on $S$ while service time is Erlang distribution with shape parameter 5.
(a) Impacts of $k$ on $W_q$ ($n_0 = 100$).

(b) Impacts of $\alpha$ on $W_q$ ($k = 80$).

(c) Impacts of $K$ on $W_q$ ($k = 50$).

(d) Impacts of $n_0$ on $W_q$ ($k = 60$).

Fig. 16: Impacts on $W_q$ while service time is Erlang distribution with shape parameter 5.
Fig. 17: Impacts on $S$ while service time is Erlang distribution with shape parameter 10.
(a) Impacts of $k$ on $W_q$ ($n_0 = 100$).

(b) Impacts of $\alpha$ on $W_q$ ($k = 80$).

(c) Impacts of $K$ on $W_q$ ($k = 50$).

(d) Impacts of $n_0$ on $W_q$ ($k = 60$).

Fig. 18: Impacts on $W_q$ while service time is Erlang distribution with shape parameter 10.
Fig. 19: Impacts on $S$ while service time is Gamma distribution with shape parameter 5.5 and scale parameter 0.18.
Impacts of $k$ on $W_q$ ($n_0 = 100$).

Impacts of $\alpha$ on $W_q$ ($k = 80$).

Impacts of $K$ on $W_q$ ($k = 50$).

Impacts of $n_0$ on $W_q$ ($k = 60$).

Fig. 20: Impacts on $W_q$ while service time is Gamma distribution with shape parameter 5.5 and scale parameter 0.18.
Fig. 21: Impacts on $S$ while service time is Gamma distribution with shape parameter 10.5 and scale parameter 0.0952.
(a) Impacts of $k$ on $W_q$ ($n_0 = 100$).

(b) Impacts of $\alpha$ on $W_q$ ($k = 80$).

(c) Impacts of $K$ on $W_q$ ($k = 50$).

(d) Impacts of $n_0$ on $W_q$ ($k = 60$).

Fig. 22: Impacts on $W_q$ while service time is Gamma distribution with shape parameter 10.5 and scale parameter 0.0952.