This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3169034, IEEE Transactions on Mobile Computing

Design and Analysis of Dynamic Block-setup Reservation Algorithm for 5G Network Slicing

Cheng-Ying Hsieh, Tuan Phung-Duc, Yi Ren, Jyh-Cheng Chen, *Fellow, IEEE*

Abstract—In 5G, network functions can be scaled out/in dynamically to adjust the capacity for network slices. The scale-out/-in procedure, namely autoscaling, enhances performance by scaling out instances and reduces operational costs by scaling in instances. However, the autoscaling problems in 5G networks are different from those in traditional cloud computing. The 5G network functions must be considered the simultaneous deployment of multiple instances; moreover, the deployment of 5G network functions is more frequent than that of traditional cloud computing. Both the number and timing of deployment will substantially affect the cost-effectiveness of the system. In this paper, we first identify the autoscaling issues specifically based on the 3GPP standards. We develop a low-complexity analytical queuing model to formulate the problem and quantify a set of performance metrics with closed-form solutions. The proposed analytical model and closed-form solutions are cross-validated by extensive simulations. The analytical model offers design insights and theoretical guidelines, helping us study the effectiveness of reservations. We proposed a dynamic block-setup reservation algorithm (DBRA) to find the optimal reserved number and threshold value of network slices. Therefore, mobile operators can balance the system's cost-effectiveness without large-scaled testing and real deployment, saving cost on time and money.

Index Terms—Network Slicing, Reservation, Block Setup, 5G, Performance Analysis, 3GPP Standards

1 Introduction

From 1st generation (1G) to 4G, broadband networks are primarily designed for humans to use. Starting from 3GPP Release 15 (R15) [1], the 5G system is being designed not only for humans but also for all kinds of things, including machines. In addition, the network functions in the 5G core networks will be deployed not only by network operators. Enterprises, factories, or government agencies will also be able to build their own private mobile networks to increase performance and enhance security. When users deploy a private mobile network, the network functions of the *control plane* could be still provided by operators. The users own only the *user plane* functions according to their requirements.

In 3GPP R14, which is still referred to as 4G, the serving gateway (SGW) and PDN gateway (PGW) are responsible for forwarding data. However, both of them deal with the functions in both the control plane and user plane. This design causes redundant overhead when the SGW and PGW are deployed. Therefore, according to 3GPP TS 29.244 [2], *control and user plane separation (CUPS)* separates the SGW and PGW into SGW-c, SGW-u and PGW-c, PGW-u for the control plane and user plane, respectively. The SGW-u and PGW-u have evolved together, eventually yielding the *user plane function (UPF)*, which specifically focuses on the tasks of data forwarding in 5G. In this way, the deployment of UPFs has become more flexible and reliable.

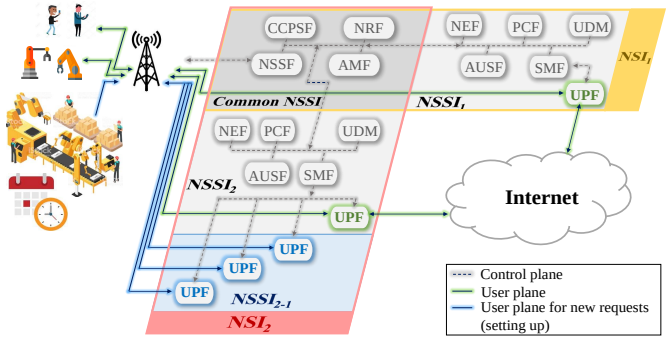


Fig. 1: Network slicing in 3GPP Release 15 (R15)

Network function virtualization (NFV), which virtualizes physical network nodes into virtualized network functions (VNFs), has become a key technology for 5G networks. Through NFV, operators can dynamically deploy virtual resources to achieve optimal cost-effectiveness. As defined in 3GPP TS 28.530 [3], a *network slice* consists of several *network slice instances (NSIs)*, which can be scaled out/in (turned on/off) to dynamically adjust the capacity of the core network functions.

As shown in Fig. 1, a network slice consists of multiple NSIs, such as NSI_1 and NSI_2, where one NSI can be composed of multiple *network slice subnet instances (NSSIs)* or a NSI is composed of a single NSSI. For example, NSI_1

- *C.-Y. Hsieh and J.-C. Chen are with the Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu 300, Taiwan, E-mail: ingyaya36.me03g@nctu.edu.tw, jcc@nycu.edu.tw.*
- *T. Phung-Duc is with the the Faculty of Engineering, Information and Systems, University of Tsukuba, Tsukuba, Japan. E-mail: tuan@sk.tsukuba.ac.jp*
- *Y. Ren is with the School of Computing Sciences, University of East Anglia, Norwich, NR4 7TJ, U.K. E-mail: e.ren@uea.ac.uk*

Manuscript received June 20, 2021; revised Dec. 21, 2021 and Mar. 29, 2022; accepted Apr. 5, 2022

consists of the shared NSSI and $NSSI_1$. An NSSI consists of multiple *network function instances (NFIs)*, which essentially are VNFs. For example, NEF, PCF, UDM, AUSF, SMF, and UPF are all VNFs inside $NSSI_1$. In addition, NSIs can share the control plane functions on one NSSI (e.g., the shared NSSI shown in Fig. 1 is shared between NSI_1 and NSI_2). The shared NSSI mainly consists of control-plane network functions. Since the performance bottleneck is on the user plane, control-plane network functions that are not deployed frequently are usually integrated and shared by NSSIs, such as $NSSI_1$ and $NSSI_2$. When establishing a connection, a user will first send an initial authentication request to the control plane. After setting up the session successfully, the data can be transmitted through the UPFs without going through the control plane. However, when a user suddenly sends out a large amount of data, the operator could dynamically set up more UPFs to mitigate system congestion.

In 5G, UPF is a user-plane function, which is used to transmit data. Specifically, UPF is responsible for maintaining a routing table and establishing GTP tunnels to forward packets. Before the transmission, UPF should decode packets' headers and search the routing rules in its dataset. In 3GPP R15, a session management function (SMF) may control multiple UPFs. However, for operators, how to set up a large number of UPFs with the lowest cost while also meeting user requirements is a challenge. In this paper, we propose a systematic way to reach this goal.

Similar to cloud autoscaling strategies, autoscaling in NSSIs can provide flexibility, reduce operational costs, and meet the performance requirements of users. However, the autoscaling of NSSIs in 3GPP is different from that in traditional cloud autoscaling due to the following:

(1) Block setup: As aforementioned, a network slice in 5G has three different layers: NSI, NSSI, and NFI. To deploy network slices, we typically need to deal with an NSSI, which not only consists of several NFIs but also may be combined with other NSSIs to form a new NSI. That is, the network slicing defined in the 3GPP standards needs to account for the composition and combination of each layer. Furthermore, according to 3GPP R15, 5G network functions are highly dependent on one another. To deploy UPFs, for example, the network must take into account the system capacity of the SMF. If the required number of UPFs exceeds the load of the SMF, additional SMFs may also be deployed. Therefore, unlike the traditional cloud computing mechanisms, 5G network slicing should not only consider a layered architecture but also analyze the impact of scaling *multiple NFIs*, which is called a *block* in this paper.

(2) Reservation: In ETSI GS NFV-IFA 010 [4] and 3GPP 28.531 [5], the standards bodies propose the *reservation* concept, where the system turns on network instances in advance to prevent sudden traffic demands. Although *reservation* can alleviate congestion and reduce the number of times NFIs are set up, the resources are wasted if the system deploys an excessive number of reservations. Therefore, there is a trade-off: too few reservations will downgrade the performance, while too many reservations will increase the cost. In standards, only the idea of *reservation* is proposed. The detailed algorithm is left for vendors and operators.

In this paper, we specifically study the effect of the reservation concept and build a system model of the block setup for future 5G networks. One of the major challenges for the problems discussed in the last section is how to set up the parameters for the block setup and reservation to find the optimal solution. In this paper, we propose the *dynamic block-setup reservation algorithm (DBRA)*, which can find the best parameters for any given block size. In this section, we summarize the contributions of this paper:

- *Proposed reservation algorithm with block setup:* As aforementioned, 5G network functions are turned on/off more frequently than traditional cloud computing. Because the instances in the setup process cannot provide service immediately, the costs are caused by the time of the setup process. Frequent scaling not only deteriorates system performance but also leads to significant costs. Therefore, in this paper, we consider the impacts of reservation and propose a DBRA algorithm, which can help mobile operators find an optimal reserved number and a suitable threshold value. With the proposed DBRA, mobile operators can significantly reduce the frequency of deployment and optimize the cost-effectiveness of the overall system.
- *Model network slicing with block setup:* Existing studies [6]–[11] have considered how to set up a bulk of VNFs at the same time. However, these studies focus on how to turn on/off all VNFs inside the bulk *together*, which does not fit with 5G systems where the autoscaling strategy should not only consider multiple instances in an NSSI but also deploy multiple NSSIs in a network slice. To this end, we model *multiple VNFs as a block* and take the deployment of multiple blocks into consideration. Our mathematical model is based on 3GPP standards and enables operators to tune the parameters (e.g., block size, system scale, arrival rate, service rate, etc.) to investigate their impacts on system performance. With these theoretical guidelines, mobile operators can estimate the system performance and corresponding costs without real deployment, which saves money and time.
- *Low-complexity analytical solution:* To analyze the system cost-effectiveness, we use a queuing model formulated by a two-dimensional Markov chain. In general, the computational complexity for finding the steady-state probabilities of a Markov chain with M states is of order $O(M^3)$. In this paper, we can obtain the steady-state probabilities with a simple recursive solution with a complexity of $O(M)$, which is the lowest possible order. Because our Markov chain is two-dimensional, the number of states M is large, and thus, the complexity reduction is significant. Hence, we can obtain three system metrics, i.e., response time W, total cost C, and cost response-time product (CRP), in a quick manner. We also consider various system operational costs. When operators want to set up multiple NFIs at a time, they can analyze the best timing and the best number of blocks.

The rest of the paper is organized as follows. We first

survey related work in Section 2. We then present the proposed DBRA in Section 3. In Section 4, we discuss the performance evaluation. Finally, Section 5 summarizes this paper.

2 Related Work

Recently, dynamic resource allocation has been extensively studied in the 5G NFV community. The currently well-known studies about resource allocation of 5G network slicing can be founded in [12]–[18]. The related studies providing theoretical insight generally falls into the following three categories: setup cost [6], [19], [20], threshold [21]–[24], and bulk/block setup [6]–[11]. In the following paragraphs, we compare our work with current studies in terms of network slicing, setup-cost strategies, Threshold-based approaches, and Bulk-/Block-setup mechanism.

(1) Current network slicing studies: For the resource allocation of 5G network slicing, the authors of [12] propose a multi-domain architecture, which simplifies the operations by dividing the managerial center into four strata, and this significantly raises the scalability and flexibility of the original architecture. However, the cost-effectiveness of handling the multi-domain architecture was not discussed in this paper. In contrast, the authors of [13] study the joint optimal deployment of VNFs by taking both the characteristics of cloud architecture and computing resource allocation into consideration. Nevertheless, the most significant difference between [13] and our work is that the authors of [13] give attention to arranging arrival tasks within limited computing resources, while our work seeks to turn on/off the computing resources with the change of tasks' arrival rate. In addition, the authors of [14], [15] discuss the dynamic resource allocation of network slicing with a probabilistic isolation guarantee, in which the authors formulate their question with an integer programming problem. Although the results show that the proposed solution outperforms traditional modeling resource utilization, more detailed performance metrics (e.g., tasks' response time, setup costs, idle costs, bus costs, etc.) have not been discussed. In addition, the authors of [16]–[18] collect many well-known studies to discuss the resource allocation of network slicing. However, even though many dynamic auto-scaling models have been proposed, they are still not flexible enough to formulate the behaviors of network slices that consist of a layered structure. Moreover, the other prominent difference between our work and current studies is resources' reservation that significantly reduces the loss of performance and costs in such a frequent deployment of 5G network slicing.

(2) Setup-cost strategies: To optimize the setup cost, the authors of [19] use Markov decision processes to derive structural properties that can be used to analyze the optimal policies. Later, the same authors extend their previous work to further discuss the bulk-setup queuing model in terms of the setup cost, staggered thresholds, and optimal ratio between static and dynamic servers [6]. Specifically, the authors treat all dynamic servers as a bulk. When the workload reaches a threshold, the system sets up the whole bulk of dynamic servers. To ensure cost-effectiveness, the system must adjust the threshold and the proportion of dynamic servers to optimize resource utilization. The other study subdivides the cost of the setup process into the instant cost and processing cost [20]. Specifically, it reveals the trade-off for the M/M/c/Setup queuing models, turning the servers on or off according to the traffic load. However, the setup costs discussed in [6], [19], [20] are the result of earlier studies and thus are not close enough to the actual network slicing defined in recent 3GPP standards, leading to a research gap for this topic. In this paper, we investigate the effect of the reservation concept and design the block setup.

(3) Threshold-based approaches: Recently, threshold-based approaches have been intensively studied (e.g., [21]–[24]). The basic idea is to use a threshold to determine the timing to switch servers on/off. For the single-threshold mechanism, the authors of [21] propose an energy-efficient resource allocation algorithm that considers the states of various server operations and the cost incurred in each status. Reference [22] uses a single threshold strategy to discuss the batch service mechanism. The arrival jobs have an unknown probability to balk from the queue. However, although the single-threshold manner is efficient, it is insufficient in some deployment cases that take many factors into consideration. To this end, the authors in [23], [24] propose a novel dual-threshold method to address the shortcomings of single threshold approaches. Two thresholds are used to deactivate/activate computing resources separately. Specifically, the authors of [23] offer a threshold-oriented operation model for reducing the power consumption in a data center. Their model analyses the Pareto optimization problem under varying parameters and requirements. The authors of [24] take dynamic deployment and reservation into consideration, which enhances both performance and efficiency significantly. However, [23], [24] do not consider multiple-instance deployment and layered structure in 5G network slicing scenarios. From a mathematical point of view, the model in [24] is an infinite buffer model for which the solution is totally different. The Markov chain in [24] is two-dimensional but one dimension has only three states. Thus, solving such a Markov chain by generating function is relatively easy. In our work, we take advantage of the single-/dual-threshold manners to design the block-setup mechanism, in which we optimize the setup timing and the number of instances by controlling a threshold value and determine the turn-off timing and the number of instances based on operators' needs. Because we have multiple thresholds, setup time, and finite buffer, our Markov chain is more complex and needs a larger computational complexity than a conventional method. Here, we successfully derive a simple recursion method that has the smallest computational complexity, i.e., the same order as the number of states.

(4) Bulk-/Block-setup mechanism: The bulk-setup algorithms run in the threshold-type queuing models. When the number of arriving jobs reaches the threshold, the models set up a bulk of servers at a time. Maccio et al. [6] discuss the problems of the bulk size and the threshold. They discuss how to achieve better cost-effectiveness by adjusting both the bulk size and threshold values. In addition to the bulk setup, bulk services are considered in a finite-capacity single-server model, which explores the effect of the general bulk service rule according to a Markovian

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3169034, IEEE Transactions on Mobile Computing

arrival process [7]–[11].

In our previous works, we proposed DBCA [25], DASA [26], and ASA [27] to analyze the system performance of auto-scaling mechanisms. However, our past papers focus on single-instance provisioning, which is insufficient to analyze 5G network slicing. In 5G, there are several NFIs in an NSSI, and multiple NSSIs are deployed to form a network slice. Even though the bulk schemes can consider the behaviors of multiple servers/jobs, they are still not flexible and efficient enough to discuss how to set up multiple instances in more subtle strategies. In this paper, we design a new queuing system that is capable of analyzing multiple NFIs deployed inside multiple NSSIs. We propose a block-setup mechanism that consists of multiple blocks, each of which contains k instances. In this way, we can help operators determine strategies to deploy network slices according to any given block size k.

3 Proposed Dynamic Block-Setup and Reservation Algorithm (DBRA)

As shown in Fig. 2, our system model consists of the following components:

- *Input Parameters*: We integrate system information, which includes the current traffic, system capacity, total number of instances, block size, setup rate, service rate, and cost coefficients.
- *System Modeling*: By plugging parameters into the system model, we can describe system behaviors in a mathematical way. With the system model, we can test various parameter combinations without actual deployment.
- *Performance Metrics*: To evaluate system performance, we derive closed-form solutions for both response time and costs. We also design a comprehensive metric, CRP, in which the convex function can help operators evaluate cost-effectiveness.
- *DBRA*: Based on the system model, we design this algorithm with gradient descent, in which we can obtain the best number of reserved instances and threshold value.
- *Outputs*: According to the results analyzed by DBRA, operators can deploy instances with the optimal reservations and threshold value, optimizing the costs-effectiveness of the overall system.

As shown in Fig. 2, in procedure (a), we establish a mathematical model to formulate system behaviors based on the given parameters. Next, in procedures (b)-(d), we calculate system performance with closed-form solutions in terms of cost, response time, and CRP. In procedure (e), based on the analysis of the performance metrics, we then find the optimal solutions for both the number of reserved instances (n_0) and the threshold value (m) with gradient descent. In procedures (f) and (g), the iterations of the DBRA update the system performance with the change of n_0 and m. To update the performance metrics, the procedure goes from (h) back to procedure (c), which forms the iteration: (c), (d), (e), (f), (g), (h), (c), and so on. Finally, when the CRP of the overall system reaches the global minimum, the optimal solutions of n_0 and m can be obtained. In the following

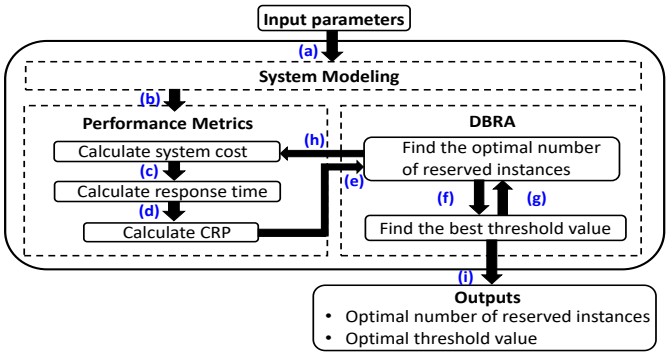


Fig. 2: Input–process–output (IPO) model

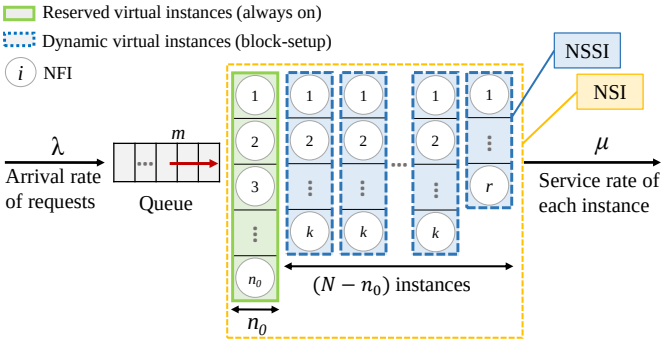


Fig. 3: System model of the proposed DBRA

sections, we discuss the details of each steps shown in Fig. 2 and list the notations in Table 1.

3.1 Input parameters

To conduct dynamic deployment, we collect system information that consists of the current traffic (λ), system capacity (K), total number of instances (N), block size (k), setup rate (α), service rate (μ), and cost coefficients (c_1, c_2, and c_3). Since the model built in this paper applies to various cases of 5G deployment, the process of information collection can be conducted in different managerial entities. For example, according to 3GPP 29.244 [2], to increase the efficiency of data transmission, UPFs are deployed by SMF, which synchronize users' data by using packet forwarding control protocol (PFCP). During the synchronization, the above information can be wrapped in the usage reporting rule (URR) sent from UPF to SMF as input parameters.

3.2 System Modeling

The system model shown in Fig. 3 depicts the behaviors of the dynamic deployment for 5G network slicing in a mathematical way. Through this model, we can optimize cost-effectiveness of the system by analyzing the reservation and threshold value. As aforementioned in Section 1, different from traditional cloud computing, network slices are deployed with a layered structure in which multiple NFIs are set up to form an NSSI, and multiple NSSIs are deployed to make an NSI. To this end, we consider k instances as *a block* to describe the relationship between NFIs and an NSSI, and we set up the blocks separately to describe the relationship between NSSIs and an NSI. In addition, because frequent

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3169034, IEEE Transactions on Mobile Computing

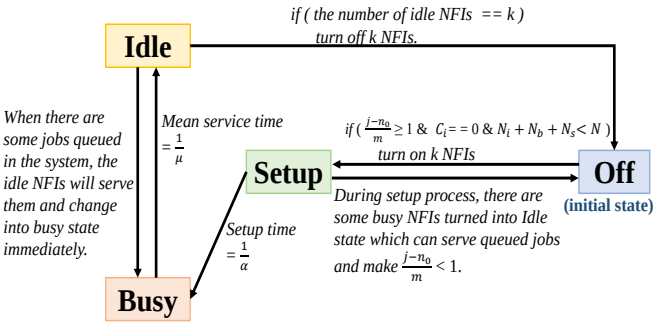


Fig. 4: Rules of the state transition in the block setup

deployment will substantially degrade system performance, we take n_0 reserved instances into consideration and set a threshold value (m) to control the setup timing. We denote the arrival rate and service rate as λ and μ, respectively. Through the closed-form solutions derived from the 2D-Markov chain, the system's performance metrics such as the response time and corresponding costs (setup, busy, and idle) can be obtained. With the derived performance metrics, the optimal solutions of the reserved instances (n_0) and threshold value (m) can be analyzed. More details regarding the analysis of optimization are expounded in Section 3.5. In this section, we first introduce three important variables that are the number of reservations (n_0), the threshold value (m), and the block size (k), and we then explain how to derive the closed-form solutions of the system performance metrics from our system model.

- *The number of reservations (n_0):* In DBRA, we reserve n_0 instances (NFIs), which are always ready to serve users. That is, n_0 instances are always on. There are $N - n_0$ instances that are set up dynamically, where N is the total number of instances that the system can accommodate. By changing the ratio between n_0 and $N - n_0$, we can help operators analyze the optimal reserved number of instances to achieve the best cost-effectiveness.
- *Block size (k):* Every time we want to set up instances, we set up a block of k instances. As shown in Fig. 3, an instance corresponds to an NFI, and a block of NFIs corresponds to an NSSI. By setting up multiple NSSIs, we can set up an NSI dynamically with various numbers of NSSIs. In our system model, we use threshold value (m) to determine *when* and *how many* instances need to be set up according to operators' cost-effective strategies. For example, when an operator wants to enhance performance, we recommend a smaller m and larger k. In contrast, when an operator wants to reduce costs, we recommend a larger m and smaller k.
- *Threshold value (m):* In the model, we set up a block of k instances when the number of service requests waiting in the queue reaches the threshold m. Moreover, if the number of service requests is two times the threshold value m, we set up two blocks of instances, and so on. The value of the threshold m controls the setup timing and number of blocks. In addition, we use the block size (k) specified by the operator as the other threshold. Specifically, to avoid over switching, the instances that have been set up should be sustained for a while even if they are idle. Therefore, the block size (k) can be applied as the deactivation threshold. That is, we turn off a block of instances if there are (k) instances idle and turn off two blocks of instances if there are ($2k$) instances idle, and so on. In this way, if operators want to save costs, they can choose a small block size, and the advantages are that the instances can be deployed more subtly and turned off faster. On the contrary, if operators want to enhance performance, they can choose a large block size, and the advantages are that the instances can be deployed faster and sustained longer to avoid over switching.

Based on the model shown in Fig. 3, the challenge now is how to set up the parameters to find the optimal solution. Fig. 2 illustrates the system flow of the proposed block-setup mechanism. Operators first determine k, the number of NFIs in an NSSI. Based on k and λ, the arrival rate of service requests, the proposed DBRA can find the CRP, which is presented in Section 3.3. Based on the CRP, DBRA then finds the optimal solution for the reservation by using the algorithms described in Section 3.5. With the proposed mathematical model and algorithms, we provide a systematic way to solve the problem.

As shown in Fig. 3, the service discipline is first come first served (FCFS), and each service request needs one available instance. We assume that the system capacity is finite with a size K. Thus, a newly arrived service request cannot enter the queue when there are already K requests in the system. Furthermore, since the traffic in 5G is different from that of traditional cloud computing, transient optimization on dynamic deployment is not suited for realistic 5G scenarios. Specifically, the traffic in 5G changes in milliseconds, while switching on/off instances takes minutes. For the uncertainty of the system, we consider the arrival rate (λ) of service requests with a Poisson distribution and model both the service rate (μ) and setup rate (α) with an exponential distribution. The mathematical results analyzed by our system model are convergent from widely random variables. The main advantage is that the system will not be affected by sudden changes of traffic. Specifically, if the operator immediately turns on instances when the traffic suddenly rises, the instantaneous decrease in traffic will make the instances switched too often, leading to significant costs. Thus, by using our system model, operators only need to update required inputs (Section 3.1) as the mean values of the parameters change. To further ensure that our system model is compatible with various scenarios, we use an ns-2 simulation to cross-validate the correctness for the change of arrival rate in many kinds of probability distributions. For more details about the experimental results, please refer to Section 4.2. The notations used in this paper are listed in Table 1.

In this study, we turn on and off ($N - n_0$) instances dynamically on a block basis. Specifically, we set up a block of k instances altogether when the number of service requests waiting in the queue reaches the threshold m. Additionally, we shut down a block of k instances when all k instances are

TABLE 1: List of Notations

Notation	*Definition*
λ	Arrival rate of service requests
K	System capacity (maximum number of requests)
N	Total number of instances that system accommodates
n_0	Number of reserved instances
k	Number of instances in a block (block size)
r	Size of the block comprising the remaining instances
α	Setup rate
μ	Service rate
m	Threshold
W	Average response time
C	Average total cost
C_b	Average busy cost
C_i	Average idle cost
C_s	Average setup cost
N_b	Number of busy instances
N_i	Number of idle instances
N_s	Number of setup instances
$c1$	Weight factor for the cost of idle instances
$c2$	Weight factor for the cost of the setup at one instance
$c3$	Weight factor for the cost of the setup process
CRP	Cost response-time product
$\hat{V}$	Current CRP value during the iteration
β_m	Learning rate of the threshold
β_n	Learning rate of the reserved number of virtual instances
A'	Maximum number of blocks
S'	Number of blocks in the setup state
M	Number of states in the Markov chain

idle. In each block, there are k instances except for the last block, which may have r instances, where ($r < k$). Because our model is designed to use multiple blocks to set up the instances, it is necessary to consider the case that the number of instances cannot be divided evenly. For example, there are a total of 100 instances in the system, where we reserved 30 instances as always-on and dynamically deploy the rest of the instances with the block size $k = 9$. That is, there are (100-30)/9 blocks with size $k = 9$ and the remaining of 7 instances form another block. We denote this remainder number of the instances as r. Thus, r is less than k ($r < k$).

Furthermore, we consider the setup time, which follows an exponential distribution with a rate α. After completing the setup process, instances can immediately provide a service when a request arrives. The service time of a request follows an exponential distribution with a rate μ. An instance becomes idle as soon as it finishes its service. Fig. 4 shows the state transition diagram of an instance. The initial state is off, in which an instance cannot provide any service. When $\left\lfloor \frac{j-n_0}{m} \right\rfloor \geq 1$, where j is the number of requests in the system, there are no idle instances. When the number of instances (on or during setup) is less than the system capacity, a block of instances goes to the setup state. That is, $N_i + N_b + N_s < N$, where N is the number of total instances, and N_i, N_b, and N_s are the number of instances in the idle, busy, and setup states, respectively. During the setup process, the instances are set up completely after an exponentially distributed time with mean $1/\alpha$. However, the setting up of instances may still be "turned off" because there may be some instances that have just finished their services and transit from the busy state to the idle state. Those instances in the idle state can serve waiting requests

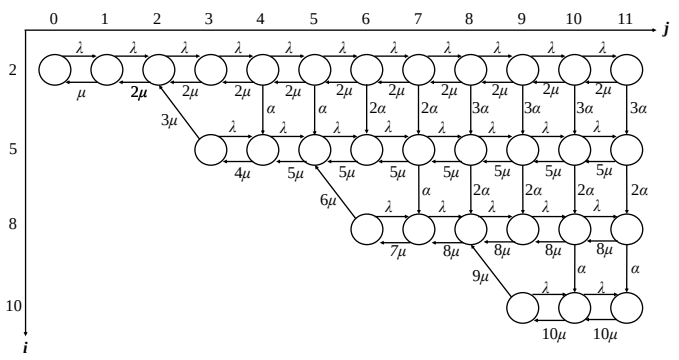


Fig. 5: State transition diagram with $n_0 = 2$; $m = 2$; $k = 3$; $N = 10$; and $K = 11$.

immediately. Therefore, the condition of $\left\lfloor \frac{j-n_0}{m} \right\rfloor < 1$ occurs, and the setting up of instances is interrupted. On the other hand, if the system sets up a block of instances successfully, these instances will initially be in the busy state and then switch to the idle state after an exponentially distributed time with mean $1/\mu$. Finally, in the idle state, an instance can switch to the busy state immediately if there is a request waiting in the queue. However, it may return to the off state when the number of idle instances reaches k.

Let (i, j) denote the state, where i instances are initiated (i.e., the instances are in the busy or idle state) while j service requests remain in the system. Therefore, $min(i, j)$ instances are busy, $max(0, i - j)$ instances are idle and $max(0, j - i)$ requests are waiting for service in the queue. Let $\mu_{i,j}$ denote the service rate at state (i, j) and $\alpha_{i,j}$ denote the setup rate at state (i, j). We then have:

$$\begin{aligned} \mu_{i,j} &= \min(i\mu, j\mu) \\ \alpha_{i,j} &= \begin{cases} 0, & i \geq j \\ \min\left(\left\lfloor \frac{j-i}{m} \right\rfloor \alpha, \left\lceil \frac{N-i}{k} \right\rceil \alpha\right), & i < j \end{cases} \end{aligned} \tag{1}$$

where

$$\begin{aligned} i \in I = \{n_0, n_0 + k, \cdots, n_0 + \left\lceil \frac{N - n_0}{k} - 2 \right\rceil k, \\ n_0 + \left\lceil \frac{N - n_0}{k} - 1 \right\rceil k, N\} \\ j \in J = \{0, 1, \cdots, K\}. \end{aligned} \tag{2}$$

Let $i(t)$ denote the number of busy or idle instances at time t, and let $j(t)$ denote the number of service requests in the queue at time t. Furthermore, we define the state space S as:

$$S = \{(i, j) \mid i \in I, j \in J, j \geq f(i)\} \tag{3}$$

where

$$f(i) = \begin{cases} 0, & i = n_0 \\ i - k + 1, & n_0 < i < N \\ i - r + 1, & i = N \end{cases} \tag{4}$$

Thus, $(i(t), j(t)); t \geq 0)$ forms a continuous-time Markov chain in the state space S. The state transition diagram of slicing instances is illustrated in Fig. 5. The derivation of the stationary state probability is shown in Section 3.3.1, in which one can see that the computational

complexity of our recursive algorithm is on the order of $O(|S|)$, where $|S|$ is the number of states in our Markov chain, as shown in Fig. 5. Please note that in general, the computational complexity for finding the steady-state probabilities of a Markov chain with $|S|$ states is on the order of $O(|S|^3)$. The complexity reduction in our model is significant.

3.3 Performance metrics

To evaluate the system performance and the cost, we define the following metrics and provide their closed-form formulas in this section.

- Mean response time (W): The response time we consider is end-to-end. It is the round-trip time for a user equipment (UE) to connect to the Internet through a 5G base station and the 5G core network.
- Mean total cost (C): As shown in Fig. 4, an instance may be in the setup state or idle state due to our proposed block setup. The cost of an instance in the busy, setup, and idle states is defined as C.
- Cost response-time product (CRP): The energy response-time product (ERP) is widely used as a metric to capture the energy performance trade-offs [28]. In our model, we are more interested in the system cost rather than in the energy. Thus, we quantify the cost and model the CRP.

Based on the results we derive in section 3.3.1, the closed forms of the metrics are derived as follows:

$$W = \frac{\sum_{(i,j)\in S} j\pi_{i,j}}{\lambda(1-\sum_{i\in I}\pi_{i,K})} \quad (5)$$

$$C = c_1 C_i + (c_2 + c_3\alpha)C_s + C_b \quad (6)$$

$$CRP = W \times C, \quad (7)$$

where

$$C_i = \sum_{(i,j)\in S} \max(i-j,0)\pi_{i,j} \quad (8)$$

$$C_s = \sum_{i\in I\setminus\{N\}} \sum_{j=i}^{K} \min \left(\left\lfloor \frac{j-i}{m} \right\rfloor k, (\lceil \frac{N-i}{k} \rceil - 1)k + r \right) \pi_{i,j} \quad (9)$$

$$C_b = \sum_{(i,j)\in S} \min(i,j)\pi_{i,j}. \quad (10)$$

C_i, C_s, and C_b are the costs caused by the system idle, setup, and busy states, respectively. c_1 is the weight factor for idle instances. c_2 and c_3 are the weight factors for the instant cost and processing cost of the entire setting-up process, respectively. The most critical advantage of this design is that operators can consider the costs of various physical resources (e.g., spectrum, electricity consumption, external heat dissipation, etc.) without limitation, and we do not assume the types of physical resources that operators consider in advance.

In 2010, Gandhi [28] proposed the energy response-time product (ERP), which has been widely used as a metric to capture the energy performance trade-offs. In our model, we are more interested in the system cost rather than in the energy. Thus, we quantify the cost and model the CRP. The most critical purpose of the CRP is to form a trade-off problem convex function, where the minimum value of the CRP is regarded as a suggested solution. However, two points should be noticed. First, the value of the CRP is only a suggested metric. Operators can still evaluate system performance with other metrics. To balance the cost-effectiveness, however, CRP is a simple indicator that has been comprehensively proved and widely used in dynamic autoscaling studies. Second, the CRP is not always a convex function. Only the variable that makes the CRP form a convex function exists in the optimal solution. For example, CRP, the function of the reserved number of instances (n_0) and the threshold value (m) form a convex function. Thus, we can analyze the optimal solutions for these two factors. More details about the proof for the convex function can be found in Section 3.5.

3.3.1 Derivation of the Stationary State Probability

In this section, we derive the stationary distribution of the Markov chain. According to Fig. 5, we define $\pi_{i,j}$ as the stationary state probability of state (i,j). Using the balance equations, we can obtain all the stationary state probabilities. In the following paragraphs, we show the balance equations in three parts: ($i = n_0$), ($i = n_0 + k,\ n_0 + 2k, ..., N - r$), and ($i = N$).

(I) $\pi_{i,j}$ with $i = n_0$:

$$\lambda\pi_{n_0,j} = \mu_{n_0,j+1}\pi_{n_0,j+1}, \quad j = 0, \quad (11)$$

$$(\lambda + \mu_{n_0,j})\,\pi_{n_0,j} = \lambda\pi_{n_0,j-1} + \mu_{n_0,j+1}\pi_{n_0,j+1}, \quad 0 < j < n_0, \quad (12)$$

$$(\lambda + \mu_{n_0,j})\,\pi_{n_0,j} = \lambda\pi_{n_0,j-1} + \mu_{n_0,j+1}\pi_{n_0,j+1} + \mu_{n_0,j+1}\pi_{n_0+k,j+1},\ j = n_0, \quad (13)$$

$$(\lambda + \mu_{n_0,j} + \alpha_{n_0,j})\,\pi_{n_0,j} = \lambda\pi_{n_0,j-1} + \mu_{n_0,j+1}\pi_{n_0,j+1},\ n_0 < j < K, \quad (14)$$

$$(\mu_{n_0,j} + \alpha_{n_0,j})\,\pi_{n_0,j} = \lambda\pi_{n_0,j-1}, \quad j = K. \quad (15)$$

When $n_0 = 0$, we obtain (16) instead of (11), (12) and (13):

$$\lambda\pi_{n_0,j} = \mu_{n_0+k,j+1}\pi_{n_0+k,j+1}, \quad j = 0, \quad (16)$$

Furthermore, we use the following equations to derive $\pi_{n_0,j}$:

$$\pi_{n_0,j} = \frac{1}{j!}\left(\frac{\lambda}{\mu}\right)^j \pi_{n_0,0}, \quad 0 < j \le n_0,$$
$$\pi_{n_0,j} = b_{n_0,j}\pi_{n_0,j-1}, \quad n_0 < j \le K, \quad (17)$$

where

$$b_{n_0,j} = \begin{cases} \frac{\lambda}{\lambda+\mu_{n_0,j}+\alpha_{n_0,j}-n_0\mu b_{n_0,j+1}}, n_0 < j < K, \\ \frac{\lambda}{n_0\mu+\alpha_{n_0,K}}, j = K. \end{cases} \quad (18)$$

(II) $\pi_{i,j}$ with $i = n_0 + k, n_0 + 2k, \cdots, N - r$:
We calculate $\pi_{i,i-k+1}$ by using the following balance equation:

$$(i-k+1)\mu\pi_{i,i-k+1} = \sum_{j=i-k+1}^{K} \alpha_{i-k,j}\pi_{i-k,j}, \quad (19)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3169034, IEEE Transactions on Mobile Computing

and we obtain the following balance equations:

$$(\lambda+\mu_{i,j})\,\pi_{i,j}=\mu_{i,j+1}\pi_{i,j+1}+\alpha_{i-k,j}\pi_{i-k,j},\quad j=i-k+1, \tag{20}$$

$$(\lambda+\mu_{i,j})\,\pi_{i,j}=\lambda\pi_{i,j-1}+\mu_{i,j+1}\pi_{i,j+1}+\alpha_{i-k,j}\pi_{i-k,j},\quad i-k+1<j<i, \tag{21}$$

$$(\lambda+\mu_{i,j})\,\pi_{i,j}=\lambda\pi_{i,j-1}+\mu_{i,j+1}\pi_{i,j+1}+\alpha_{i-k,j}\pi_{i-k,j}+\mu_{\min(i+k,N),j+1}\pi_{\min(i+k,N),j+1},\quad j=i, \tag{22}$$

$$(\lambda+\mu_{i,j}+\alpha_{i,j})\,\pi_{i,j}=\lambda\pi_{i,j-1}+\mu_{i,j+1}\pi_{i,j+1}+\alpha_{i-k,j}\pi_{i-k,j},\quad i<j<K, \tag{23}$$

$$(\mu_{i,j}+\alpha_{i,j})\,\pi_{i,j}=\lambda\pi_{i,j-1}+\alpha_{i-k,j}\pi_{i-k,j},\quad j=K. \tag{24}$$

However, when $k=1$, we obtain (25) instead of (20), (21) and (22):

$$(\lambda+i\mu)\,\pi_{i,i}=(i+1)\mu\pi_{\min(i+k,N),i+1}+i\mu\pi_{i,i+1}+\alpha_{i-k,i}\pi_{i-k,i},\quad j=i. \tag{25}$$

Furthermore, we derive $\pi_{i,j}$ with $i=n_0+k,n_0+2k,\cdots,N-r$ as follows:

$$\begin{aligned}\pi_{i,j}&=a_{i,j}+b_{i,j}\pi_{i,j-1},\qquad j=i-k+2,\\ \pi_{i,j}&=a_{i,j}+b_{i,j}\pi_{i,j-1}+c_{i,j}\pi_{i,j-2},\\ &\qquad i-k+2<j\le i,\\ \pi_{i,j}&=a_{i,j}+b_{i,j}\pi_{i,j-1},\qquad i<j\le K,\end{aligned} \tag{26}$$

where

$$a_{i,j}=\begin{cases}-\frac{\alpha_{i-k,j-1}\pi_{i-k,j-1}}{\mu_{i,j}}, i-k+2\le j\le i,\\ \frac{\mu_{i,j+1}a_{i,j+1}+\alpha_{i-k,j}\pi_{i-k,j}}{\lambda+\mu_{i,j}+\alpha_{i,j}-\mu_{i,j+1}b_{i,j+1}}, i<j<K,\\ \frac{\alpha_{i-k,K}\pi_{i-k,K}}{\mu_{i,K}+\alpha_{i,K}}, j=K,\end{cases}$$

$$b_{i,j}=\begin{cases}\frac{\lambda+\mu_{i,j-1}}{\mu_{i,j}}, i-k+2\le j\le i,\\ \frac{\lambda}{\lambda+\mu_{i,j}+\alpha_{i,j}-\mu_{i,j+1}b_{i,j+1}}, i<j<K,\\ \frac{\lambda}{\mu_{i,K}+\alpha_{i,K}}, j=K,\end{cases} \tag{27}$$

$$c_{i,j}=\left\{-\frac{\lambda}{\mu_{i,j}}, i-k+2<j\le i.\right.$$

Please note that $\pi_{i,i-k+1},\pi_{i,i-k+2}$ must be calculated first to calculate (26) recursively.

(III) $\pi_{i,j}$ with $i=N$:
We can calculate $\pi_{N,N-k+1}$ by using the balance equation:

$$(N-r+1)\mu\pi_{N,N-r+1}=\sum_{j=N-r+1}^{K}\alpha_{N-r,j}\pi_{N-r,j}. \tag{28}$$

Then, we obtain the following balance equations:

$$(\lambda+\mu_{N,j})\,\pi_{N,j}=\mu_{N,j+1}\pi_{N,j+1}+\alpha_{N-r,j}\pi_{N-r,j},\quad j=N-r+1, \tag{29}$$

$$(\lambda+\mu_{N,j})\,\pi_{N,j}=\lambda\pi_{N,j-1}+\mu_{N,j+1}\pi_{N,j+1}+\alpha_{N-r,j}\pi_{N-r,j},\quad N-r+1<j<K, \tag{30}$$

$$\mu_{N,j}\pi_{N,j}=\lambda\pi_{N,j-1}+\alpha_{N-r,j}\pi_{N-r,j},\quad j=K. \tag{31}$$

Furthermore, we derive $\pi_{N,j}$ as follows.

$$\pi_{N,j}=a_{N,j}+b_{N,j}\pi_{N,j-1},\qquad N-r+2\le j\le K, \tag{32}$$

where

$$a_{N,j}=\begin{cases}\frac{\mu_{N,j+1}a_{N,j+1}+\alpha_{N-r,j}\pi_{N-r,j}}{\lambda+\mu_{N,j}-\mu_{N,j+1}b_{N,j+1}}, N-r+2\le j<K,\\ \frac{\alpha_{N-r,K}\pi_{N-r,K}}{\mu_{N,K}}, j=K,\end{cases} \tag{33}$$

$$b_{N,j}=\begin{cases}\frac{\lambda}{\lambda+\mu_{N,j}-\mu_{N,j+1}b_{N,j+1}}, N-r+2\le j<K,\\ \frac{\lambda}{\mu_{N,K}}, j=K.\end{cases} \tag{34}$$

So far, we have derived the stationary state probabilities for states in S. The sum of these probabilities must be 1 due to the following property:

$$\sum_{(i,j)\in S}\pi_{i,j}=1. \tag{35}$$

This yields $\pi_{n_0,0}$ and all steady-state probabilities. The computational complexity of our recursive algorithm is on the order of $O(|S|)$, where $|S|$ is the number of states in our Markov chain, as shown in Fig. 5. However, in general, the computational complexity for finding the steady-state probabilities of a Markov chain with $|S|$ states is on the order of $O(|S|^3)$. Thus, the complexity reduction in our model is significant.

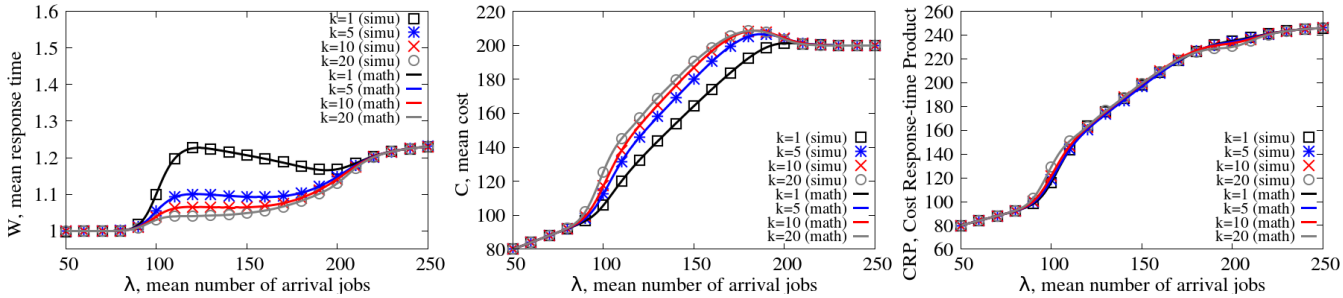
In this paper, we conduct extensive simulations by ns-2 [29] to cross-validate our analytical model. The simulation results reveal two findings. (1) The simulation results match the numerical consequence of the proposed analytical model, which verifies the correctness of the analytical model. (2) The analytical model provides theoretical insights and guidelines while designing network slicing strategies without real implementation. It can save time and reduce costs for operators.

3.4 Characteristics of the block-setup mechanism

One of our major contributions is the establishment of a dynamic-deployment queuing model, which paves the way for the analysis of reservations (n_0) and the threshold value (m). As shown in Fig. 6 and Fig. 7, we show the features of the block-setup mechanism with the change in arrival rate λ on the performance metrics W, C, and CRP under the following parameter settings: $\mu=1$, $\alpha=0.5$, $n_0=100$, $N=200$, $K=250$, $c_1=0.6$, $c_2=1.0$ and $c_3=3.0$. Here, we set different thresholds $m=5$ (a relatively small value) and $m=30$ (a relatively large value) for Fig. 6 and Fig. 7, respectively, to study the impacts of the threshold m. Moreover, we set block sizes $k=1,5,10,20$ to see the impacts of k, which are illustrated as four curves in these figures. Regarding these curves, the numerical results of the proposed analytical model are depicted by the lines, while those of the ns-2 simulation are illustrated as spots.

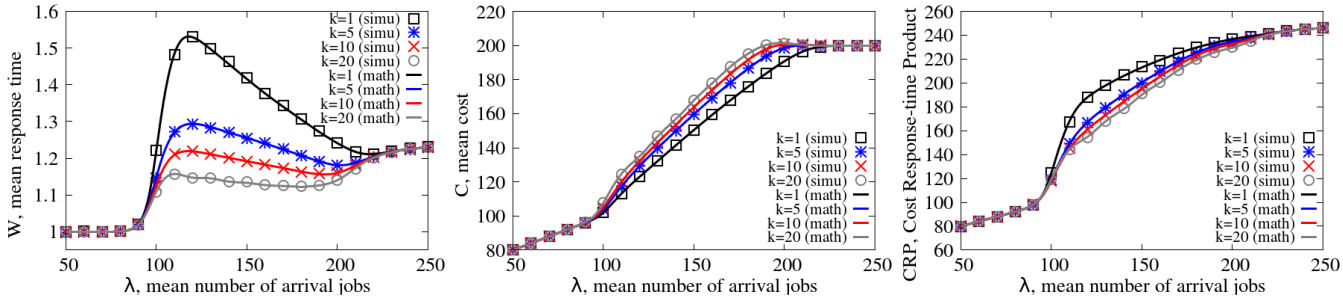
(1) Impact on the response time: W is the mean response time of the system, corresponding to the y-axis in Fig. 6(a) and Fig. 7(a). Specifically, we discuss the results in three parts in terms of the arrival rate λ, corresponding to the x-axis. (i) ($\lambda<n_0$): When the arrival rate λ gradually approaches the number of reserved instances ($n_0=100$), the system capability is insufficient to handle the workloads. Thus, virtual instances begin to be set up, which causes the response time to increase. (ii) ($n_0\le\lambda<N$): The mean response time stops rising and reduces slightly because the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3169034, IEEE Transactions on Mobile Computing



(a) Average response time (W) (b) Average total cost (C) (c) Cost response-time product (CRP)

Fig. 6: Impact of the block size ($k = 1, 5, 10, 20$) and threshold ($m = 5$) on the cost, response time, and CRP



(a) Average response time (W) (b) Average total cost (C) (c) Cost response-time product (CRP)

Fig. 7: Impact of the block size ($k = 1, 5, 10, 20$) and threshold ($m = 30$) on the cost, response time, and CRP

virtual instances are set up as blocks. However, if a system is deployed with a smaller block size k or larger threshold m, its mean response time will be longer because a small block size results in fewer instances set up at once, while a large threshold leads to a longer setup period. (iii) $(N \leq \lambda)$: No matter how large the block size k that is used, the mean response time will be convergent because all instances are already in the busy state.

(2) Impact on cost: C is the total cost of the system and takes the idle, setup, and busy states into consideration. The impacts on C are depicted in Fig. 6(b) and Fig. 7(b), which we discuss in three parts. (i) $(\lambda < n_0)$: As the arrival rate λ increases, the number of busy instances increases, which causes the total cost to increase slightly. (ii) $(n_0 \leq \lambda < N)$: As the arrival rate exceeds the capability of the reserved resources, the system starts to turn on more instances and increases the cost. Specifically, instances set up with a smaller block size and larger threshold value lead to lower costs because a block with a smaller size can decrease the setup cost, while a larger threshold can reduce the setup frequency. (iii) $(N \leq \lambda)$: When the arrival rate is close to the limit of the system capability, the cost is slightly reduced because all instances are in the busy state; hence, no more instances can be set up, eventually resulting in system cost convergence.

(3) Impact on the CRP: Since a trade-off exists between the cost and the response time in terms of the threshold value m and block size k, we use the CRP to reflect the cost-effectiveness. Fig. 6(c) and Fig. 7(c) show the impacts, which we discuss in three parts: (i) $(\lambda < n_0)$: We can see that as the arrival rate increases, the CRP increases because both the response time and the cost increase. We also observe that the block size k has no impact on the CRP because the reserved instances are still able to handle the workload. (ii) $(n_0 \leq \lambda < N)$: In this interval, the results of the CRP are mainly affected by the threshold value m. As shown in Fig. 6(c), when the threshold is small ($m = 5$), the instance can be set up immediately, which reduces the response time significantly while incurring only a slightly higher cost. On the other hand, a large threshold (e.g., $m = 30$) reduces the cost but leads to a higher response time. (iii) $(N \leq \lambda)$: Similar to the results of both the response time W and cost C, the CRP achieves convergence when the traffic exceeds the system capability.

We have demonstrated that the block size k and threshold value m have significant impacts on the response time, cost, and CRP. Because the block size and arrival rate are variables, obtaining an optimal threshold value is difficult. The reason is that setting up multiple instances at a single time causes oscillation on the CRP distributions, which makes the global minimum of the CRP difficult to analyze. To this end, in Section 3.5, we propose an optimal algorithm, DBRA, which introduces the impact of reservations and derives the optimal solutions for both the threshold value m and the number of reservations n_0.

3.5 Dynamic Block-setup and Reservation Algorithm (DBRA)

In this section, we introduce the proposed DBRA in three parts. Firstly, we prove the convexity of CRP-n_0 and CRP-m in 3.5.1. Next, we introduce the processing procedure of DBRA in section 3.5.2. Finally, we discuss the time complexity of DBRA in section 3.5.3.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3169034, IEEE Transactions on Mobile Computing

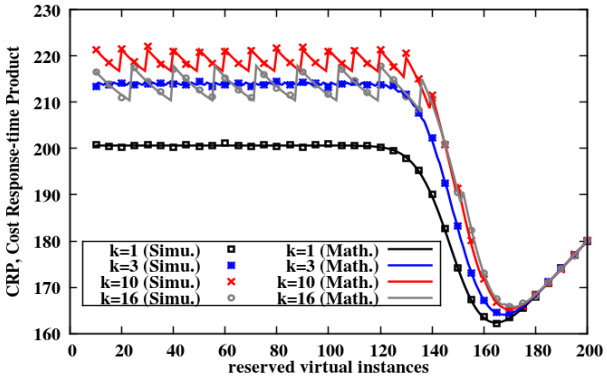


Fig. 8: Finding the best n_0 for the minimum CRP ($\lambda = 150$)

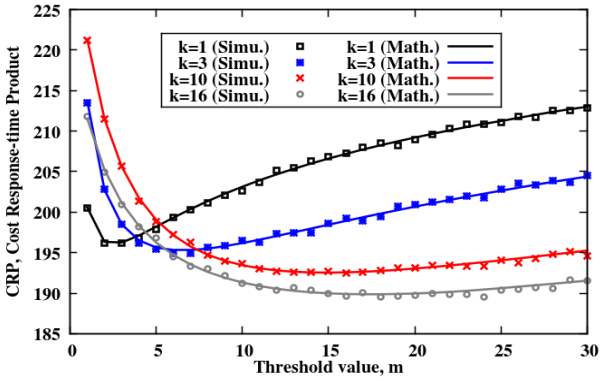


Fig. 9: Finding the best m for the minimum CRP ($\lambda = 150$)

3.5.1 The convexity of CRP-n_0 and CRP-m

Since the convexity of CRP-n_0 can be observed easily from a physical perspective, we use Fig. 8 to illustrate that the convexity of CRP-n_0 is definite. To the contrary, because the convexity of CRP-m cannot be observed intuitively, we prove it with a second-order derivative function and show the result in Fig. 9. In addition, since the queuing model derived in this paper is mainly based on Poisson distribution, the convexity of CRP-n$_0$ and CRP-m for other inter-arrival distributions are shown in the technical report [30]. More details about the convexity of CRP-n_0 and CRP-m are introduced in the following paragraphs.

(I) The convexity of CRP-n_0:
In Fig. 8, we illustrate the convexity of CRP-n_0 with the most extreme case ($m = 1$) because a small threshold value easily leads to the oscillation, which makes the optimization more difficult than other cases. In Fig. 8, the curves can be divided into three segments as follows. (1) *Oscillation segment:* Because the number of reserved instances is not sufficient to provide service immediately (i.e., n_0 is small), the operator has to dynamically turn on more virtual instances to enhance the performance. However, with the change in the number of reserved instances, different remainders of block size will lead to different performance results. Specifically, there are a total of N instances in the system, which consists of n_0 reserved instances and $(N-n_0)$ dynamic instances. In the dynamic instances, we consider k instances deployed at a time. Thus, there are $\lfloor \frac{N-n_0}{k} \rfloor$ blocks with block size k and a remaining block with block size r. Therefore, in an oscillation period, there are k different results with the change in n_0 because the number of remainders ($r = (N - n_0) \ \% \ k$) is within the range $r = [0 : k-1]$. (2) *Descending segment:* When the number of reserved instances (n_0) increases, the system gradually relies on the reserved instances instead of the dynamic ones. Both the setup cost (C_s) and the response time (W) can therefore be decreased. Eventually, the system reaches the minimum CRP with a proper n_0. (3) *Ascending segment:* When n_0 continues to increase, however, the response time cannot be further reduced, but costs will keep increasing because of too many idle reserved instances. Finally, the trend of CRP-n_0 is formed as a convex function in which the local minimums are distributed regularly, and the global minimum can be found by *gradient descent*.

(II) The convexity of CRP-m:
We confirm the convexity of CRP-m by using a second-order derivative function. That is, CRP-m is a convex function if $\frac{\partial^2 CRP}{\partial m^2} > 0$. As shown in (7), $\frac{\partial^2 CRP}{\partial m^2}$ can be obtained as follows:

$$\frac{\partial^2 CRP}{\partial m^2} = \frac{\partial^2 W}{\partial m^2} C + 2\frac{\partial W}{\partial m}\frac{\partial C}{\partial m} + W\frac{\partial^2 C}{\partial m^2}, \tag{36}$$

where $\frac{\partial W}{\partial m}$, $\frac{\partial C}{\partial m}$, $\frac{\partial^2 W}{\partial m^2}$, and $\frac{\partial^2 C}{\partial m^2}$ can be found in (37), (38), (39), and (40), respectively.

$$\frac{\partial W}{\partial m} = \frac{[\sum_{(i,j)\in S} j\frac{\partial \pi_{i,j}}{\partial m}]\lambda(1-\sum_{i\in I}\pi_{i,K})}{[\lambda(1-\sum_{i\in I}\pi_{i,K})]^2} + \frac{\lambda(\sum_{(i,j)\in S} j\pi_{i,j})\sum_{i\in I}\frac{\partial \pi_{i,K}}{\partial m}}{[\lambda(1-\sum_{i\in I}\pi_{i,K})]^2}, \tag{37}$$

$$\frac{\partial C}{\partial m} = \{\sum_{(i,j)\in S} c_1[max(i-j,0)\frac{\partial \pi_{i,j}}{\partial m}] + min(i,j)\frac{\partial \pi_{i,j}}{\partial m}\} + (c_2+c_3\alpha)\sum_{i\in I}\sum_{j=i}^{K}[-\lfloor\frac{(j-i)k}{m^2}\rfloor\pi_{i,j} + \lfloor\frac{(j-i)k}{m}\rfloor\frac{\partial \pi_{i,j}}{\partial m}] \tag{38}$$

$$\frac{\partial^2 W}{\partial m^2} = \{\sum_{(i,j)\in S}\lambda j[(1-\sum_{i\in I}\pi_{i,K})\frac{\partial^2 \pi_{i,j}}{\partial m^2} + \pi_{i,j}\sum_{i\in I}\frac{\partial^2 \pi_{i,K}}{\partial m^2}]\} + [\sum_{(i,j)\in S} j\frac{\partial \pi_{i,j}}{\partial m} + \frac{\pi_{i,j}\sum_{i\in I}\frac{\partial \pi_{i,k}}{\partial m}}{\lambda(1-\sum_{i\in I}\pi_{i,K})}](2\lambda\sum_{i\in I}\frac{\partial \pi_{i,K}}{\partial m}), \tag{39}$$

$$\frac{\partial^2 C}{\partial m^2} = [\sum_{(i,j)\in S} max(i-j,0)c_1\frac{\partial^2 \pi_{i,j}}{\partial m^2} + min(i,j)\frac{\partial^2 \pi_{i,j}}{\partial m^2}] + (c_2+c_3\alpha)\sum_{i\in I}\sum_{j=i}^{K}\frac{(j-i)k}{m}[\frac{2\pi_{i,j}}{m^2} - \frac{2}{m}\frac{\partial \pi_{i,j}}{\partial m} + \frac{\partial^2 \pi_{i,j}}{\partial m^2}]. \tag{40}$$

As shown in Fig. 5, setting up a block of instances should take a setup time of $(1/\alpha_{i,j})$, in which $\alpha_{i,j}$ is defined as (1). Apparently, with the increase in the threshold value m, the setup rate ($\alpha_{i,j}$, $i > n_0$) decreases, which makes the probability of the states ($\pi_{i,j}$, $i > n_0$) reduced. That is, we can confirm that $\frac{\partial \pi_{i,j}}{\partial m} < 0$, which thereby leads to $\frac{\partial W}{\partial m} < 0$ and $\frac{\partial C}{\partial m} < 0$. In addition, the balance equations

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3169034, IEEE Transactions on Mobile Computing

Algorithm 1: DBRA

Input: λ, k
Output: optimal (CRP, m, n_0)
Initialize m and n_0 arbitrarily;
$(CRP, m, n_0) = Gradient\ Descent\ (m, n_0)$;
/* *Gradient Descent* is presented in Func. 1 */
$n_0 \leftarrow n_0 + k$;
while $Local\ Search\ (m, n_0) = True$ **do**
  /* *Local Search* is presented in Func. 2 */
  $n_0 \leftarrow n_0 + k$;
end
$(CRP, m, n_0) = Gradient\ Descent\ (m, n_0)$;
return optimal (CRP, m, n_0);

Function 1: Gradient Descent

Input: m, n_v
Output: Local minimum (CRP, m, n_0)
Compute $CRP(m, n_0)$;
$V \leftarrow CRP(m, n_0)$;
Set learning rate β_m, β_n;
while ΔV *not converge* **do**
  $m \leftarrow m$ - $\beta_m \frac{\partial CRP}{\partial m}$ and $n_0 \leftarrow n_0$ - $\beta_n \frac{\partial CRP}{\partial n_0}$;
  Compute $CRP(m, n_0)$;
  $\hat{V} \leftarrow CRP(m, n_0)$;
  $\Delta V \leftarrow \left|\hat{V} - V\right|$;
  $V \leftarrow \hat{V}$;
end
return (V, m, n_0);

Function 2: Local Search

Input: m, n_0
Output: $True$ or $False$
Compute $CRP(m, n_0)$;
Compute $CRP(m, n_0 + 1)$ and $CRP(m, n_0 - 1)$;
Set learning rate β_m, β_n;
if $CRP(m, n_0) < CRP(m, n_0 + 1)$ **then**
  if $CRP(m, n_0) < CRP(m, n_0 - 1)$ **then**
    return $True$;
  else
    return $False$;
  end
else
  return $False$;
end

of $\pi_{i,j}$ shown in Section 3.3.1 (II) indicate that the states ($\pi_{i,j}$, $i = n_0 + k, n_0 + 2k, \cdots, N - r$) are directly affected by the setup rate ($\alpha_{i,j}$). Therefore, we can infer that $\frac{\partial^2 W}{\partial m^2} > 0$ and $\frac{\partial^2 C}{\partial m^2} > 0$ because $\frac{\partial^2 \alpha_{i,j}}{\partial m^2} > 0$. Finally, by plugging the results of (37), (38), (39), and (40) back into (36), we can confirm that $\frac{\partial^2 CRP}{\partial m^2} > 0$, which proves the convexity of CRP-m. As shown in Fig. 9, the trend of CRP-m is a convex function in which the global minimum can easily be found by *gradient descent*.

3.5.2 The procedure of DBRA

As shown in Algorithm 1, operators can arbitrarily choose a set of initial parameters (m, n_0) and find the adjacent local minimum after the first *Gradient Descent*, in which the learning rate of both the threshold (β_m) and the reserved number (β_n) are determined by the operators. Next, according to the convexity of CRP-n_0, we take advantage of the fact that the oscillation period is a block size k. By this way, we can increase n_0 by k to move to the next local minimum. After reaching the next local minimum, we use the Function 2 to determine whether it is currently a local minimum or not. If yes, we continue to increase n_0 by k. Otherwise, it means that the indicator has already jumped out of the oscillation period. In the last step of Algorithm 1, we use the last *Gradient Descent* to find the point of the global minimum with the best n_0 and m for the current λ and k.

3.5.3 Time complexity of DBRA

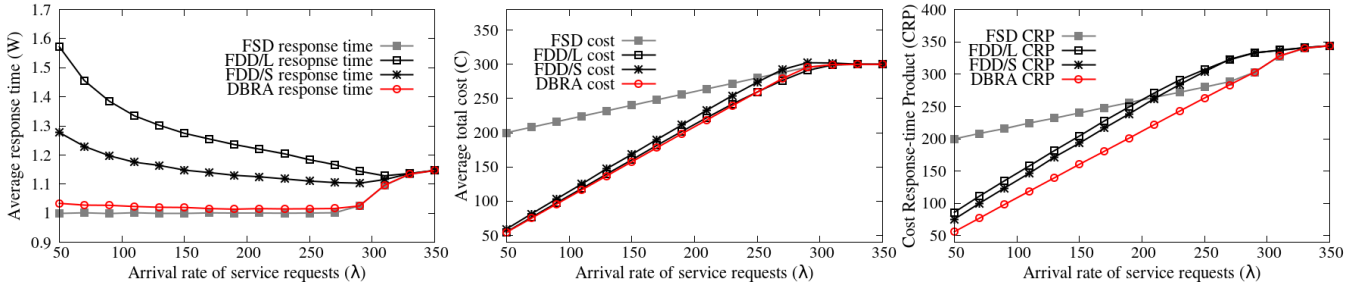
As the discussion of Section 3.5.2, the largest time complexity of CRP-n_0 is at $k = 1$ because there is no oscillation that can move n_0 forward by k in each iteration. The required iterations are at most $\lceil \frac{N}{\beta_n} \rceil$, where β_n is the learning rate of n_0. In contrast, for the time complexity of CRP-m, the required steps of iterations are at most $\lceil \frac{K-N}{\beta_m} \rceil$, in which β_m is the learning rate of m, and $(K-N)$ is the maximum queue length. Since the iterations of CRP-m are conducted within each iteration of CRP-n_0, the overall steps of DBRA are at most $\lceil \frac{N(K-N)}{\beta_n \beta_m} \rceil$. For the most complicated case, both β_n and β_m are set to 1, which makes the maximum steps of DBRA become $N(K-N)$. Finally, in a real-world 5G scenario, most arrival requests are buffered at a base station (gNB) before being dispatched to UPFs. Thus, we consider $K >> N$ and estimate the time complexity of DBRA as $O(NK)$. In addition, the deployment information will be collected at a central managerial unit. For example, in 5G, the session management function (SMF) synchronizes user data with UPFs by using packet forwarding control protocol (PFCP), which consists of five packet processing rules, namesly, the packet detection rule (PDR), forwarding action rule (FAR), QoS enhancement rule (QER), usage reporting rule (URR), and buffering action rule (BAR). Based on the collective information, operators can conduct our DBRA on SMF to deploy UPFs with the optimal reserved number and the threshold value. Since the above collective information is sent in the control plane which will not cause significant traffic and will hardly affect system performance, we temporarily ignore the impact of control-plane messages and leave this issue for our future works.

4 Performance Evaluation

As discussed in Sec. 2, previous studies are not close enough to the actual network slicing defined in the most recent 3GPP standards. Although we want to compare our proposed DBRA with other algorithms, there are none based on 3GPP R15. Thus, we compare the DBRA with two baseline algorithms:

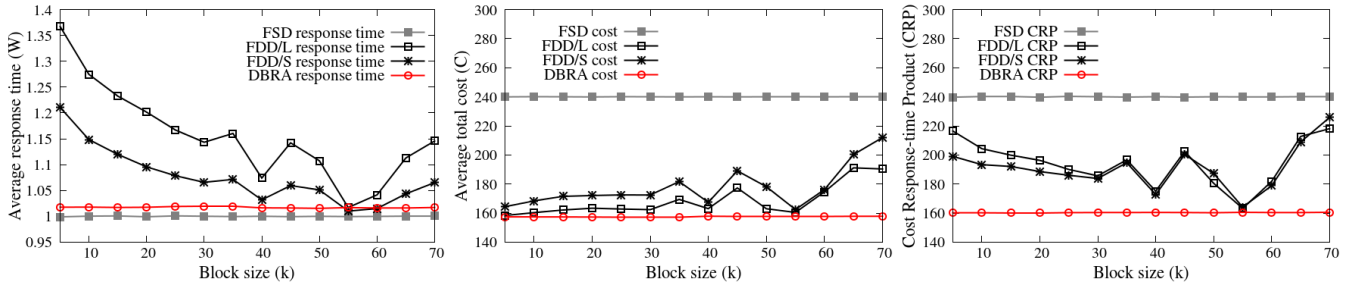
1) *Fully Static Deployment (FSD):* In FSD, all NFIs are always turned on. That is, n_0 in Fig. 3 equals N. This strategy reduces the setup cost and decreases

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3169034, IEEE Transactions on Mobile Computing



(a) Average response time (W) (b) Average total cost (C) (c) Cost response-time product (CRP)

Fig. 10: Comparison with the baseline algorithms (FSD, FDD/L, FDD/S) regarding the impact of the request arrival rate (λ) $(k = 10)$



(a) Average response time (W) (b) Average total cost (C) (c) Cost response-time product (CRP)

Fig. 11: Comparison with the baseline algorithms (FSD, FDD/L, FDD/S) regarding the impact of the block size (k) $(\lambda = 150)$

the response time. However, many instances may be idle, which may lead to high costs. Essentially, FSD will not set up virtual instances dynamically, which is typical in legacy systems such as 4G.

2) *Fully Dynamic Deployment (FDD):* In FDD, n_0 in Fig. 3 equals 0. Thus, FDD can reduce costs by scaling in/out the virtual instances dynamically. However, the response time may increase. The actual performance of FDD highly depends on the threshold m. Thus, we further categorize FDD into FDD/L and FDD/S: (1) In FDD/L, the threshold value m is tuned to the maximum $(K - N)$, and the system is more conservative in turning on NFIs. (2) In FDD/S, where m is half of the maximum, and the system is more dynamic.

4.1 Experimental Results

According to the rules of the AWS EC2 [31], the number of instances that a user can rent is at most 20 in each region. Since AWS divides its entire service region into 25 regions, the number of instances that a user can use is up to 500. In addition, according to the rules of Microsoft Azure [32], each user can rent up to 800 instances. To evaluate the performance of DBRA in a rational range, we consider the median value from both of these two famous cloud providers. Since the median values of the limits for EC2 and Azure are 500/2 and 800/2, respectively, we set $N = 300$ (which can be other values too). We also set $K = 350$ to discuss fully-loaded cases. Please note other reasonable values can also be chosen. We did choose other values to evaluate the performance. Furthermore, in Fig. 10, we show the difference between each method with the change of the arrival rate. To observe the system from an available state to an overloading state, we set the range of the arrival rate within $\lambda = [50 : 350]$. In addition, to observe the impact of the block-setup deployment, we set at least 5 blocks in the system. Thus, in Fig. 11, the range of the block size that we set is $k = [1 : 70]$. Since block-setup deployment is a multi-dimensional problem, we observe the trend of each method with the change of the arrival rate and block size by fixing one to each other. Specifically, in Fig. 10, to purely discuss the difference between each method with the change of the arrival rate, we fix the block size at $k = 10$ because too extreme cases $(k = 1; k = 70)$ may make the discussion not general enough. Identically, in Fig. 11, to purely discuss the impact of the block size, we fix the arrival rate with the value $\lambda = 150$. Although other parameters result in different figures, the conclusions and insights obtained from them are the same.

For the weight factors (c_1, c_2, c_3), we refer to [33]–[35] to set the values in a rational range. In [33], the authors show extensive testing results to emphasize how significant workloads should be burdened while multiple instances are set up. In addition, the authors of [34], [35] discuss the setup costs of multiple instances in many different environments such as Amazon EC2, Window Azure, Rackspace, etc. Based on the testing results of the above studies, the costs caused by each state of instances are ordered as $setup > busy > idle$. Since the costs estimated by operators are different from one another, we tentatively set the weight of the idle cost as $c_1 = 0.6$, and the setup cost as $c_2 = 1, c_3 = 3$. We set the weight of the busy cost as 1. Although other parameters result in different figures, the conclusions and insights obtained from them are the same.

In Fig. 10(a)–Fig. 10(c), all algorithms converge together when the system is fully loaded. Thus, there is no difference between the algorithms when λ is high. On the other hand, as shown in Fig. 11(a)–Fig. 11(c), we discuss the impact of the block size (k) at $\lambda = 150$. With the change in block size k, the performance of our proposed DBRA is as stable as that of FSD, while that of FDD fluctuates. Through a comparison of Fig. 10 and Fig. 11, we can verify the significant influence of the DBRA on the reservations.

As shown in Fig. 10(a) and Fig. 11(a), among all of the strategies, FSD has the lowest response time because it keeps all NFIs running all the time. As shown in Fig. 10(b) and Fig. 11(b), however, FSD results in high costs because a very large number of NFIs may be idle. In terms of the CRP, FSD is not flexible enough, leading to higher values of the CRP than those of other algorithms when λ is small.

FDD/L uses not only a dynamic strategy to reduce the number of idle NFIs but also a large threshold ($m = 50$) to constrain the number of block setups. Although FDD/L can reduce costs significantly compared with FSD, as illustrated in Fig. 10(b) and Fig. 11(b), it delays in turning on the NFIs, resulting in a higher response time, as shown in Fig. 10(a) and Fig. 11(a). Finally, for the overall comprehensive indicator CRP, the performance of FDD/L is better than that of FSD due to its flexibility.

In addition, FDD/S is fully dynamic. Unlike FDD/L, FDD/S uses a smaller threshold ($m = 25$). Thus, FDD/S is more aggressive in the deployment of NFIs. Therefore, as shown in Fig. 10(a) and Fig. 11(a), the response time can be further reduced compared with that of FDD/L. Although the setup cost of FDD/S is slightly higher than that of FDD/L, the CRP of FDD/S is still lower than that of FDD/L when λ is low.

Because a slice with a large block size can launch more NFIs at a time, when the block size increases, the response time decreases and the cost increases. Particularly, in Fig. 11, FDD is close to the DBRA in terms of W, C, and CRP when the block size $k = 40$ and 55. This result occurs because the best number of reservations (n_0) we analyzed is 163, and FDD with a block size $k = 40$ and $k = 55$ can result in 160 and 165 NFIs, respectively, when $\lambda = 150$. Thus, the performance of FDD can be close to that of the DBRA when $k = 40$ and 55. Nevertheless, we do not intervene in the determination of the block size (k). We pay more attention to developing a flexible model for the analysis of reservations (n_0) and the threshold (m).

As discussed in Sec. 3, the proposed DBRA can find the optimal CRP. It can effectively determine the best number of reserved NFIs (n_0). Thus, it allows service requests to be served as soon as they enter the system. As shown in Fig. 10(a) and Fig. 11(a), the DBRA is only 1.43% worse than FSD in terms of the response time. In addition, the DBRA adjusts the threshold m dynamically. That is, the system can adjust the setup time and number of NFIs dynamically. Thus, the number of idle NFIs is reduced. As shown in Fig. 10(b) and Fig. 11(b), the cost of the DBRA is the smallest. Finally, Fig. 10(c) and Fig. 11(c) show that the DBRA has the smallest CRP. In addition to Poisson distribution, other types of arrival rates have similar results, which are presented in Section 4.2.

4.2 Other distributions of arrival rates

In this section, we show that our queuing model is accurate in terms of the arrival rate with other probability distributions.

Due to the nature of telecommunication systems, to implement algorithms of a real-life scenario is expensive in terms of the time and cost. In this paper, we used a queuing model to quantify the behaviors of autoscaling network slices, where the Poisson process was used to approximate the arrival traffic for the network. We understand that the Poisson process cannot capture all the properties of the arrival traffic with general distributions. Here, we conducted extensive simulations by taking more probability distributions into consideration. Arrival traffic with a normal distribution, uniform distribution, and Pareto distribution were used as input for our simulation models.

Our simulation results demonstrate that our queuing model did a very good job in approximating the arrival traffic with the above distributions, as shown in Table 2. Specifically, for the normal distribution, we set various values for the mean value and standard deviation for the arrival rate distribution. Specifically, for the normal distribution, we set various values for the mean value and standard deviation for the arrival rate distribution. The values of the inter-arrivals in normal distribution are strictly positive because the probability of the negative inter-arrival is very small in our settings, and thus the impact on our results is limited. The results show that the similarity between the mathematical and experimental results is 94%, 96%, and 97% in terms of CRP, cost, and response time, respectively. This demonstrates the good accuracy of our queuing model. Similarly, the accuracy of the worst case we tested was still approximately 91%, 93%, and 97% for the uniform distribution and 97%, 98%, and 98% for the Pareto distribution.

5 Conclusions

In this paper, to analyze the network slicing strategies for 5G networks, we specifically model the network slicing behaviors based on 3GPP R15. We study the effects of the *reservation* and *block-setup* concepts. Although the reservation concept may lead to additional idle costs, it can reduce the response time. We specifically study the effect of reservations and build a system block-setup model for future 5G networks. Therefore, operators can find the best parameters (m, n_0) to control reservations and optimize the cost-effectiveness for any given block size k. The proposed DBRA has low computational complexity. We also compare the proposed DBRA with two baseline algorithms. The results show that DBRA has the smallest CRP.

Acknowledgements

Cheng-Ying Hsieh and Jyh-Cheng Chen's work was supported in part by the Ministry of Science and Technology of Taiwan under grant numbers MOST 110-2224-E-A49-002 and MOST 108-2221-E-009-042-MY3. The research of Tuan Phung-Duc was supported in part by JSPS KAKENHI Grant Numbers 18K18006 and 21K11765. The research of Yi Ren was supported in part by EPSRC EP/T022566/1, EP/T024593/1, and the Royal Society IEC\R3\213100. The

TABLE 2: Comparison of the experimental results with the mathematical model derived under the Poisson process

Probability distributions of the arrival rate	$m = 30$			$m = 5$		
	CRP	Cost	Response time	CRP	Cost	Response time
Poisson (avg=[50,60,...,250])	99.48%	99.81%	99.62%	99.58%	99.70%	99.80%
Normal (avg=[50,60,...,250], std=avg$\times$0.18)	96.63%	98.38%	98.15%	94.61%	96.16%	98.32%
Normal (avg=[50,60,...,250], std=avg$\times$0.15)	95.93%	98.10%	97.73%	94.51%	96.14%	98.24%
Normal (avg=[50,60,...,250], std=avg$\times$0.1)	95.97%	98.10%	97.68%	94.42%	96.09%	98.23%
Normal (avg=[50,60,...,250], std=avg$\times$0.05)	95.78%	98.05%	97.63%	94.36%	96.40%	98.18%
Uniform (min=0, max=[50,60,...,250]$\times$1.8)	94.78%	95.67%	97.30%	95.90%	96.60%	97.99%
Uniform (min=0, max=[50,60,...,250]$\times$2.0)	97.56%	98.82%	98.69%	96.67%	97.74%	98.86%
Uniform (min=0, max=[50,60,...,250]$\times$2.2)	91.88%	93.86%	97.82%	91.27%	93.00%	97.72%
Pareto (avg=[50,60,...,250], shape=2)	97.32%	98.64%	98.70%	98.00%	98.80%	99.20%
Pareto (avg=[50,60,...,250], shape=3)	97.19%	98.59%	98.54%	96.14%	97.37%	98.69%
Pareto (avg=[50,60,...,250], shape=4)	96.02%	98.02%	97.89%	94.84%	96.43%	98.29%

authors would like to thank Mr. Shinto Hideyama, a former student of the second author at University of Tsukuba, for his help in the derivation of the stationary distribution and some initial numerical examples. The authors are very grateful to the editor and all reviewers for their valuable comments to improve this article.

REFERENCES

[1] 3GPP, 3rd Generation Partnership Project "The Mobile Broadband Standard", Available: https://www.3gpp.org/.

[2] 3GPP, "Technical Specification Group Core Network and Terminals; Interface between the Control Plane and the User Plane Nodes; Stage 3," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 29.244, 06 2019, version 15.6.0.

[3] 3GPP, "Technical Specification Group Services and System Aspects; Management and orchestration; Concepts, use cases and requirements," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 28.530, 12 2018, version 15.1.0.

[4] ETSI, "Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Functional requirements specification MANO Functional Rqmts Spec," European Telecommunications Standards Institute (ETSI), Group Specification (GS) GS NFV-EVE 010, 02 2018, version 2.4.1.

[5] 3GPP, "Technical Specification Group Services and System Aspects; Management and orchestration; Provisioning," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 28.531, 9 2018, version 15.0.0.

[6] V. J. Maccio and D. G. Down, "Exact analysis of energy-aware multiserver queueing systems with setup times," *in Proc. of IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pp. 11–20, Sept. 2016.

[7] S. K. Bar-Lev, M. Parlar, D. Perry, W. Stadje, and F. A. V. der Duyn Schouten, "Applications of bulk queues to group testing models with incomplete identification," *European Journal of Operational Research*, vol. 183, no. 1, pp. 226 – 237, 2007.

[8] A. Banerjee and U. Gupta, "Reducing congestion in bulk-service finite-buffer queueing system using batch-size-dependent service," *Performance Evaluation*, vol. 69, no. 1, pp. 53 – 70, 2012.

[9] A. Banerjee, U. Gupta, and S. Chakravarthy, "Analysis of a finite-buffer bulk-service queue under markovian arrival process with batch-size-dependent service," *Computers and Operations Research*, vol. 60, pp. 138 – 149, 2015.

[10] A. Maity and U. C. Gupta, "Analysis and optimal control of a queue with infinite buffer under batch-size dependent versatile bulk-service rule," *OPSEARCH*, pp. 472–489, 2015.

[11] B. Kar, E. H. Wu, and Y. Lin, "Energy cost optimization in dynamic placement of virtualized network function chains," *IEEE Transactions on Network and Service Management*, 2018.

[12] T. Taleb, I. Afolabi, K. Samdanis, and F. Z. Yousaf, "On multi-domain network slicing orchestration architecture and federated resource control," *IEEE Network*, 2019.

[13] A. De Domenico, Y.-F. Liu, and W. Yu, "Optimal virtual network function deployment for 5g network slicing in a hybrid cloud infrastructure," *IEEE Transactions on Wireless Communications*, 2020.

[14] J. Li, J. Liu, T. Huang, and Y. Liu, "Dra-ig: The balance of performance isolation and resource utilization efficiency in network slicing," *IEEE International Conference on Communications (ICC)*, 2020.

[15] Q.-T. Luu, S. Kerboeuf, and M. Kieffer, "Uncertainty-aware resource provisioning for network slicing," *IEEE Transactions on Network and Service Management*, 2021.

[16] R. Su, D. Zhang, R. Venkatesan, Z. Gong, C. Li, F. Ding, F. Jiang, and Z. Zhu, "Resource allocation for network slicing in 5g telecommunication networks: A survey of principles and models," *IEEE Network*, 2019.

[17] F. Debbabi, R. Jmal, L. C. Fourati, and A. Ksentini, "Algorithmics and modeling aspects of network slicing in 5g and beyonds network: Survey," *IEEE Access*, 2020.

[18] A. Banchs, G. de Veciana, V. Sciancalepore, and X. Costa-Perez, "Resource allocation for network slicing in mobile networks," *IEEE Access*, 2020.

[19] V. J. Maccio and D. G. Down, "On optimal control for energy-aware queueing systems," *27th International Teletraffic Congress*, pp. 98–106, Sep. 2015.

[20] T. Phung-Duc, "Multiserver queues with finite capacity and setup time," *Analytical and Stochastic Modelling Techniques and Applications*, pp. 173–187, 2015.

[21] Hao, Yaqian, Wang, Jinting, Yang, Mingyu, Wang, and Ruoyu, "Equilibrium analysis of the m/m/1 queues with setup times under n-policy," *in Proc. of Queueing Theory and Network Applications*, 2017.

[22] O. Bountali and A. Economou, "Equilibrium threshold joining strategies in partially observable batch service queueing systems," *Annals of Operations Research*, Aug 2017.

[23] C. Schwartz, R. Pries, and P. Tran-Gia, "A queuing analysis of an energy-saving mechanism in data centers," *in Proc. of The International Conference on Information Network*, pp. 70–75, Feb. 2012.

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

[25] T. Phung-Duc, Y. Ren, J. Chen, and Z. Yu, "Design and Analysis of Deadline and Budget Constrained Autoscaling (DBCA) Algorithm for 5G Mobile Networks," *in Proc. of IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 94–101, Dec. 2016.

[26] Y. Ren, T. Phung-Duc, J. Chen, and Z. Yu, "Dynamic auto scaling algorithm (dasa) for 5g mobile networks," *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2016.

[27] Y. Ren, T. Phung-Duc, Y.-K. Liu, J.-C. Chen, and Y.-H. Lin, "Asa: Adaptive vnf scaling algorithm for 5g mobile networks," *2018 IEEE 7th International Conference on Cloud Networking (CloudNet)*, 2018.

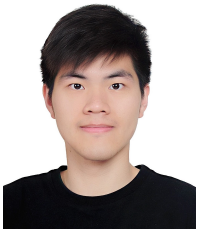
[28] A. Gandhi, V. Gupta, M. Harchol-Balter, and M. A. Kozuch, "Optimality analysis of energy-performance trade-off for server farm management," *Perform. Eval.*, pp. 1155–1171, Nov. 2010.

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

[30] "Technical report for the inter-arrivals with different probability distributions" Available: http://wire.cs.nctu.edu.tw/er/TechnicalReport.pdf.

[31] "Cluster configuration guidelines and best practices" Available:

https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-plan-instances-guidelines.html.
[32] "Resources not limited to 800 instances per resource group" Available: https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/resources-without-resource-group-limit.
[33] T. L. Nguyen and A. Lebre, "Virtual machine boot time model," *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pp. 430–437, 2017.
[34] Razavi, Kaveh, Razorea, L. Mihai, Kielmann, and Thilo, "Reducing vm startup time and storage costs by vm image content consolidation," *Euro-Par 2013: Parallel Processing Workshops*, pp. 75–84, 2014.
[35] M. Mao and M. Humphrey, "A performance study on the vm startup time in the cloud," *2012 IEEE Fifth International Conference on Cloud Computing*, pp. 423–430, 2012.



Cheng-Ying Hsieh received his B.S. degree in mechanical engineering from National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 2014. He is now a Ph.D. candidate in the Department of Computer Science at National Yang Ming Chiao Tung University (NYCU), formerly NCTU. His research interests include 5G mobility networks, queueing analysis, deep learning.

Tuan Phung-Duc is an Associate Professor at Faculty of Engineering, Information and Systems, University of Tsukuba. He received a Ph.D. in Informatics from Kyoto University in 2011. He is in the Editorial Board of the KSII Transactions on Internet and Information Systems and two other international journals. He served a Guest Editor of the special issue of Annals of Operations Research on Retrial Queues and Related Models and currently is serving as a Guest Editor of the Special Issue of the same journal on Queueing Theory and Network Applications. He was the Chairman of 10th International Workshop on Retrial Queues (WRQ'2014) and the TPC co-chair of 23rd International Conference on Analytical, and Stochastic Modelling Techniques and Applications (ASMTA'16), TPC co-chair of The 13th and 14th International Conference on Queueing Theory and Network Applications (QTNA2018, QTNA2019), General co-chair of EAI VALUETOOLS 2020 - 13th EAI International Conference on Performance Evaluation Methodologies and Tools, and General chair of ASMTA/EPEW : The 26th International Conference on Analytical & Stochastic Modelling Techniques & Applications / 17th European Performance Engineering Workshop. Dr. Phung Duc received the Research Encourage Award from The Operations Research Society of Japan in 2013. His research interests include Applied Probability, Stochastic Models and their Applications in Performance Analysis of Telecommunication and Service Systems.

Yi Ren received a Ph.D. degree in information communication and technology from the University of Agder, Grimstad, Norway, in 2012. He was with the Department of Computer Science, National Chiao Tung University (NCTU), Hsinchu, Taiwan, as a Postdoctoral Fellow, an Assistant Research Fellow, and an Adjunct Assistant Professor from 2012 to 2017. He is currently a Lecturer with the School of Computing Science, University of East Anglia (UEA), Norwich, U.K. His current research interests include Internet of Things and 5G mobile technology: security, performance analysis, protocol design, radio resource allocation, mobile edge computing, WiFi and Bluetooth Technology, 3GPP, LTE, software-defined networking, network function virtualization, etc. He was a recipient of the Best Paper Award in IEEE MDM 2012.

Jyh-Cheng Chen (S'96-M'99-SM'04-F'12) received the Ph.D. degree from the State University of New York at Buffalo in 1998. He was a Research Scientist with Bellcore/Telcordia Technologies, Morristown, NJ, USA, from 1998 to 2001, and a Senior Scientist with Telcordia Technologies, Piscataway, NJ, USA, from 2008 to 2010. He was with the Department of Computer Science, National Tsing Hua University (NTHU), Hsinchu, Taiwan, as an Assistant Professor, an Associate Professor, and a Full Professor from 2001 to 2008. He has been a Faculty Member with National Yang Ming Chiao Tung University (NYCU), formerly National Chiao Tung University (NCTU), since 2010. He is currently the Chair Professor with the Department of Computer Science, and the Dean of the College of Computer Science, NYCU. Dr. Chen is a Distinguished Member of the Association for Computing Machinery (ACM). He was a member of the Fellows Evaluation Committee, IEEE Computer Society. He has received numerous awards, including the Outstanding Teaching Award from both NCTU and NTHU, the Outstanding I.T. Elite Award, Taiwan, the Mentor of Merit Award from NCTU, the Medal of Honor and the K. T. Li Breakthrough Award from the Institute of Information and Computing Machinery, the Outstanding Engineering Professor Award from the Chinese Institute of Engineers, the Outstanding Research Award from the Ministry of Science and Technology, the Best Paper Award for Young Scholars from the IEEE Communications Society Taipei and Tainan Chapters, and the IEEE Information Theory Society Taipei Chapter, and the Telcordia CEO Award.