Optimization-Based Resource Management Strategies for 5G C-RAN Slicing Capabilities

Frank Yeong-Sung Lin[1], Chiu-Han Hsiao[1], Yean-Fu Wen[2], and Ya-Syuan Wu[1]

1. Department of Information Management
National Taiwan University
Taipei, Taiwan (R.O.C.)
yeongsunglin@gmail.com
{d98725001; r06725016}@ntu.edu.tw

2. Graduate Institute of Information Management
National Taipei University
New Taipei City, Taiwan (R.O.C.)
yeanfu@mail.ntpu.edu.tw

***Abstract*—In an emerging paradigm in 5G networks, the computations of various types of mobile applications are offloaded to cloud environments. Edge and core clouds provide computing, storage, and networking resources to serve as a generic computing platform. Network slicing techniques offer an effective way to boost various types of services, such as delay-sensitive or computationally intensive application, that are deployed on-demand in a shared resource infrastructure. This study proposes a resource management mechanism to optimize the quality of experience (QoE) of users in terms of the delay gap tolerance. The Min-Fit algorithm is a heuristic-based solution that flexibly chooses a server that has the maximum remaining CPU resources for satisfying user requirements. A mathematical model is formulated for 5G slicing capabilities to optimize the delay gap using a resource management approach to achieve QoE. Some computational experiments are demonstrated as performance evaluations for verifying the suitability of our proposed approach in terms of slicing capabilities. The results show that our approach outperforms other algorithms, which have larger delay gaps.**

Keywords—5G; Edge computing; Quality of experience (QoE); Resource management, Slicing

I. Introduction

Various technologies are being changed to provide a common connected platform for various 5G applications. One major change is the decomposition of typical base stations into remote radio heads (RRHs) and baseband units (BBUs) that are installed in the fronthaul and backhaul of the cloud radio access network (C-RAN), respectively. The traditional centralized architecture of the core network has evolved into a cloud-based architecture, which separates parts of the control plane from the user plane and thereby reduces control signaling and data transmission delays. Virtualized network functions can be created by using software-defined networking (SDN) and network function virtualization (NFV) techniques to assign servers to core and edge clouds appropriately. Corresponding virtual machines (VMs) are distributed in the core and edge clouds to execute virtualized network functions. SDN is a promising technology that helps to simplify network design and management. NFV creates a logical network of VMs over several physical servers. The network service chaining model or network slicing has now emerged as a novel 5G network architecture [1].

A. Overview of Network Slicing Techniques

Network slicing techniques can boost various types of on-demand services in a shared resource infrastructure bounded by edge and core cloud environments [2]. Centralized computing relies on software-based programming rather than hardware-based configuration to control all virtualized network functionalities. Each network slice enables software reconfiguration for upgrading network topologies for achieving resource management efficiency and performance improvement [3]. Two types of cloud networks are set in the backhaul, as shown in Fig. 1.

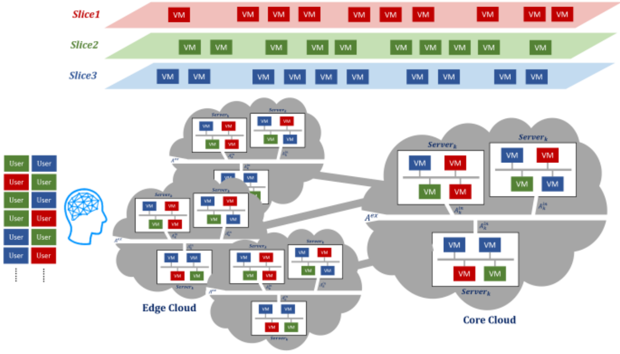

Fig. 1. Network slicing.

C-RAN servers are located separately in the edge and core clouds to form a centralized pool of virtualized functionalities. For user plane functions, functions in the packet data network gateway (P-GW) and serving gateway (S-GW) are shifted to the edge cloud to provide low-latency services to reduce the burden on the backhaul. For control plane functions, BBUs perform data forwarding between RRHs and the C-RAN; these are mainly located in the edge cloud. Mobility management, virtualized resource management, and interference management functions are performed in the core cloud. Mobile edge computing platforms are also deployed in the edge cloud along with data forwarding and content storage servers that can collaboratively store, compute, and transmit large amounts of data efficiently in real-time [4].

978-1-5386-4646-5/18/$31.00 ©2018 IEEE

B. Motivation

This study proposes a resource allocation scheme that is tailored for various quality of experience (QoE) requirements in network slicing. For instance, ultrareliable and low-latency communication (uRLLC), Internet of things, and enhanced mobile broadband (eMBB) slices are the three fundamental types of network slicing in 5G systems. uRLLC slices are used for video streaming and online gaming, in which communication services are more sensitive to delay and have delay tolerance. The QoE is defined by performance metrics such as the difference in delay tolerance and delay in the slice [4]. This study addresses resource or task assignment as the research objective function to maximize the minimal QoE for each request appropriately in the slice.

C. Paper Organization

The remainder of this paper is organized as follows. Section II presents a literature review of current ideas and mechanisms in emerging 5G technologies. Section III describes the problem definitions of resource management and presents the formulation of a mathematical programming problem. In Section IV, several proposed solution processes contained in heuristics are developed to find an optimal solution. Section V presents various computational experiments and discusses and validates the results. Finally, Section VI discusses the conclusions and future work in this area.

II. Literature Review

Network slicing can be implemented using the virtualization, SDN, and NFV of the system architecture. An end-to-end network slice is a specific collection of network modules and functions that can be compared with other network slices [5]. A network slice is a tailored and connected set or chain of network functions formed through logical linking in a virtual network. Each slice satisfies the specific requirements of a service, such as those for bandwidth, delay, delay tolerance, and the business model [6]. Many virtualized units (e.g., VMs) are required by the computation and communication requests that are allocated in a resource pool to perform service slicing in diverse QoE requirements [7]. A user-centric service slicing strategy for various delay and transmission bit-rate requirements was proposed and a genetic algorithm was devised to optimize virtualized radio resource management based on resource pooling in [8]. In [9], a network slicing mechanism was introduced for network edge nodes to offer low-latency services to users, in which the centralized core network entities and related applications are shifted to the network edge to reduce delay and burden in the backhaul.

In [10], an auction-based revenue optimization method was proposed for resource management in each slice to satisfy user requirements and increase network revenue. The auction mechanism comprised a price competition model and an auction mechanism for network slicing [5]. Furthermore, in [11], admission and allocation algorithms were developed for maximizing system revenue by solving a problem modeled as a semi-Markov decision process. Thus far, few studies have investigated the analytics of QoE models. Thus, the aim of present study is to propose a proof-of-concept system that demonstrates a cloud-based network slicing approach in a 5G network.

III. Mathematical Formulation

A. Problem Description

A resource management problem with limited resources is formulated from the perspective of network service providers. We choose QoE for the performance measurement and use it as the objective function; we also consider users' strong focus on latency. Based on the network slicing concept, VMs in our model are requested by different users and require resources to serve the users. The model supports network resource allocation to network slices. When facing a batch of requests, we assume that a resource management strategy is executed simultaneously to determine which server a slice should be assigned to.

B. System Architecture and Problem Formulation

Tables I and II show the given parameters and decision variables of our system model, respectively.

TABLE I. Summary of Given Parameters

Notation	Description
S	Index set of physical servers {1, 2, 3,…, $\|S\|$} in the cloud computing system.
I	Index set of users {1, 2, 3,…, $\|I\|$} in the cloud computing system.
W_i	Index set of slicing types of VMs {1, 2, 3,…, $\|W_i\|$} required by user $i \in I$. (Note that w_i is also referred to as the total number of VMs required by user $i \in I$.)
A_k^{in}	Internal communication bandwidth rate of server $k \in S$.
A^{ex}	External communication bandwidth rate of whole cloud computing system.
P_k	Total number of CPU cores in server $k \in S$.
Π_k	Processing capability for each CPU core in server $k \in S$.
M_k	Total RAM capacity in server $k \in S$.
D_k	Total storage capacity in server $k \in S$.
N_k	Maximum number of VMs available on server $k \in S$.
C_{ij}	Total CPU processing capability required by user $i \in I$ on VM $j \in W_i$.
R_{ij}	Total RAM capability required by user $i \in I$ on VM $j \in W_i$.
H_{ij}	Total storage capability required by user $i \in I$ on VM $j \in W_i$.
B_{ij}	Total bandwidth rate required by user $i \in I$ on VM $j \in W_i$.
L_{ij}	Delay tolerance of user $i \in I$ on VM $j \in W_i$.
O	Minimum number of servers that are switched on at any time.
β^p	Weight for adjusting processing delay.
β^{in}	Weight for adjusting internal transmission delay.
β^{ex}	Weight for adjusting external transmission delay.
γ^p	Bias for adjusting processing delay.
γ^{in}	Bias for adjusting internal transmission delay.
γ^{ex}	Bias for adjusting external transmission delay.

TABLE II. Summary of Decision Variables

Notation	Description
d_{ij}	Aggregated delay for user $i \in I$ on VM $j \in W_i$.
d_k^p	Processing delay on server $k \in S$.

d_k^{in}	Internal transmission delay on server $k \in S$.
d^{ex}	External transmission delay.
y_{ijk}	1 if VM $j \in W_i$ of user $i \in I$ is served on server $k \in S$ and 0 otherwise.
x_k	1 if server $k \in S$ is open and 0 otherwise.
p_{ijk}	Number of CPU cores allocated to VM *j* of user *i* on server *k*; $p_{ijk} \geq 0, \forall i \in I, j \in W_i, k \in S$.

The delay gap is defined as the difference between the delay tolerance and the actual time delay to achieve QoE. The objective function aims to maximize the minimum delay gap among all VMs requested by users.

Objective function:

$$\max_{i \in I, j \in W_i} \min \left(L_{ij} - d_{ij} \right) \qquad \text{(IP)}$$

Subject to:

1) Assignment-Related Constraints:

Equation (1) indicates that each VM required by a user can only be served on one server; this means that these VMs are inseparable. Equation (2) indicates that the number of VMs assigned to server N_k should not exceed the maximum number of VMs available on that server.

$$\sum_{k \in S} y_{ijk} \leq 1 \qquad \forall i \in I, j \in W_i \qquad (1)$$

$$\sum_{i \in I} \sum_{j \in W_i} y_{ijk} \leq N_k \qquad \forall k \in S \qquad (2)$$

2) Resource Constraints:

We define resources offered by a server as a set of five factors: number of CPU cores, processing capability rate of each CPU core, RAM capability rate, hard disk capability rate, and internal bandwidth rate. The total resources required by VMs for each server cannot exceed its available resources, which are formulated in (3) to (7). The external bandwidth rate for the whole cloud computing system is also given. Equation (8) shows that total bandwidth required by all VMs accepted by the cloud system should not exceed the external bandwidth. Moreover, (9) implies that the number of power-on servers must exceed a given default setting.

$$\sum_{i \in I} \sum_{j \in W_i} C_{ij} y_{ijk} \leq \sum_{i \in I} \sum_{j \in W_i} \Pi_k p_{ijk} \qquad \forall k \in S \qquad (3)$$

$$\sum_{i \in I} \sum_{j \in W_i} p_{ijk} \leq P_k \qquad \forall k \in S \qquad (4)$$

$$\sum_{i \in I} \sum_{j \in W_i} R_{ij} y_{ijk} \leq M_k \qquad \forall k \in S \qquad (5)$$

$$\sum_{i \in I} \sum_{j \in W_i} H_{ij} y_{ijk} \leq D_k \qquad \forall k \in S \qquad (6)$$

$$\sum_{i \in I} \sum_{j \in W_i} B_{ij} y_{ijk} \leq A_k^{in} \qquad \forall k \in S \qquad (7)$$

$$\sum_{k \in S} \sum_{i \in I} \sum_{j \in W_i} B_{ij} y_{ijk} \leq A^{ex} \qquad (8)$$

$$O \leq \sum_{k \in K} x_k \leq |K| \qquad (9)$$

3) Delay Constraints:

We defined the delay within a VM as the sum of the processing delay, internal transmission delay, and external transmission delay, as shown in (10). These delays are expressed in (11), (12), and (13), respectively. The processing delay of a VM is directly proportional to the CPU usage rate of its server. The internal transmission delay of a VM is proportional to the traffic of the internal bandwidth of the server. The external transmission delay of the whole cloud computing system is proportional to the traffic of the overall required bandwidth.

$$d_{ij} = \sum_{k \in S} d_{ijk}^{p} + \sum_{k \in S} d_{ijk}^{in} + d_{ij}^{ex} \qquad \forall i \in I, j \in W_i \qquad (10)$$

$$d_{ijk}^{p} = \frac{\sum_{i \in I} \sum_{j \in W_i} C_{ij} y_{ijk}}{P_k \Pi_k} \beta^{p} + \gamma^{p} \qquad \forall i \in I, j \in W_i, k \in S \qquad (11)$$

$$d_{ijk}^{in} = \frac{\sum_{i \in I} \sum_{j \in W_i} C_{ij} y_{ijk}}{A_k^{in}} \beta_k^{in} + \gamma_k^{in} \qquad \forall i \in I, j \in W_i, k \in S \qquad (12)$$

$$d_{ij}^{ex} = \frac{\sum_{i \in I} \sum_{j \in W_i} \sum_{k \in S} C_{ij} y_{ijk}}{A^{ex}} \beta^{ex} + \gamma^{ex} \qquad \forall i \in I, j \in W_i \qquad (13)$$

4) Delay-Related Constraints:

Equation (14) implies that all VMs must be served within the tolerance delay. Thus, for all VMs required by users, the delay cannot exceed the delay tolerance.

$$L_{ij} - d_{ij} \geq 0 \qquad \forall i \in I, j \in W_i \qquad (14)$$

IV. Solution Approach

This section describes our proposed resource management approach. The Min-Fit algorithm is a heuristic-based allocation mechanism. For optimizing a delay gap, it searches for the server that has the lowest CPU processing capability usage rate every time a VM is requested. However, even the least-occupied server may be too busy to serve the VM. In this case, a new server is switched on if a powered-off server is available. Because this approach always chooses the server that has the minimum computing resources used, we call it the Min-Fit algorithm.

Min-Fit Algorithm

```
for each user
    for each VM
        for each open server
            get amount of used computing resources
        end
        select server k with least computing resources used
        if server k can serve VM then
            VM is assigned to server k
            update computing resources used for server k
        end
        else
```

```
            if number of open servers is less than total
            number of servers then
                open a new server
                assign VM to new server
                update computing resources used for server
            end
        end
    end
end
```

V. Computational Experiments

This section provides a performance comparison of our algorithm with three other common methods. First-Fit always scans from the beginning of the server list and assigns a VM to the first available server. Next-Fit scans the server list and assigns a VM to the first server that has sufficient capacity. Round-Robin chooses a server according to the forward-and-backward order of the server list. As in our proposed Min-Fit algorithm, these three methods turn on a server if required as long as one is available.

A. Environments

Two types of servers are designed in the experiment. The number of servers in the edge cloud is larger than that in the core cloud. The edge servers have smaller capacity than do the core servers. Six types of VMs are generated randomly, and each type of VM requires a different amount of resources. Table III shows the values of the experimental parameters.

TABLE III. Values of Experimental Parameters

Parameter	Value
Number of CPU cores in server (P_k)	2
Processing capability of each CPU core (Π_k), total RAM capacity (M_k), total storage capacity (D_k) and internal bandwidth rate (A_k^{in}) of edge server	150
Processing capability of each CPU core (Π_k), total RAM capacity (M_k), total storage capacity (D_k) and internal bandwidth rate (A_k^{in}) of core server	300
Maximum number of VMs available on edge server (N_k)	12
Maximum number of VMs available on core server (N_k)	24
External communication bandwidth rate of whole cloud computing system (A^{ex})	30,000
Weight for adjusting processing delay, internal transmission delay, and external transmission delay (β^p , β^{in} , and β^{ex} , respectively)	3
Bias for adjusting processing delay, internal transmission delay, and external transmission delay (γ^p , γ^{in} , and γ^{ex} , respectively)	0.5
Total RAM capacity, storage capacity, and bandwidth required by a VM requested by a user (R_{ij} , H_{ij} , and B_{ij} , respectively)	Random from 20 to 40
Total CPU processing capacity required by a VM requested by a user (C_{ij})	Random from 40 to 80
Delay tolerance of a VM requested by a user (L_{ij})	Random from 7 to 11
Number of slicing types of VMs required by a user (W_i)	Random from 1 to 6

B. Performance Evaluation

Three experimental cases combined with algorithms and our approach are proposed to evaluate the performance. Furthermore, the results are compared in terms of four parameters: minimum delay gap, average delay gap, maximum delay, and average delay.

1) Case A: Increasing the Number of Users

Case A deals with the condition in which the number of users increases. As a result, the total number of requested VMs increases. Fig. 2 shows that Min-Fit has the maximum value of minimum delay gap for all VMs, making it a suitable solution for the system model mentioned in Section III. Furthermore, Fig. 3 shows that Min-Fit has the minimum value of maximum delay gap. Therefore, it outperforms the other three methods.

TABLE IV. Experimental Environment for Case A

Parameter	Value
Maximum number of total servers	120
Number of edge servers	60
Number of core servers	15-60
Delay of switching on a server	0.5
Number of users	100-240

Fig. 4 and 5 show that the Min-Fit method outperforms the First-Fit and Next-Fit methods when the number of users is small. As the number of users increases, the results of Min-Fit are slightly higher than those of First-Fit and Next-Fit. Round-Robin seems to provide the best result in terms of the average delay gap and average delay. However, this method is prone to failure in generating feasible solutions with limited resources, because it always requires more servers (in this case, 4 on average) than other methods.

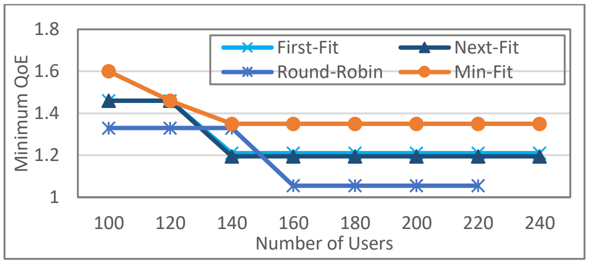

Fig. 2. Minimum delay gap with different number of users.

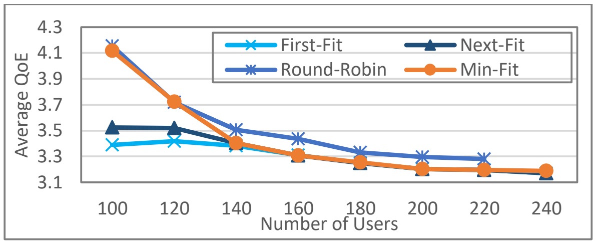

Fig. 3. Average delay gap with different number of users.

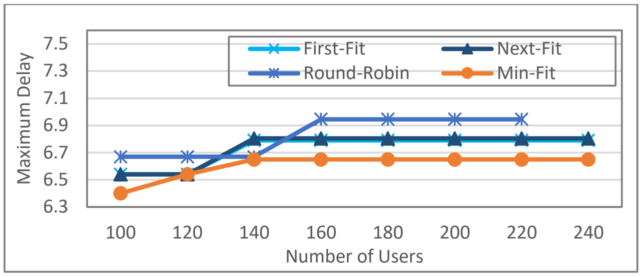

Fig. 4. Maximum delay with different number of users.

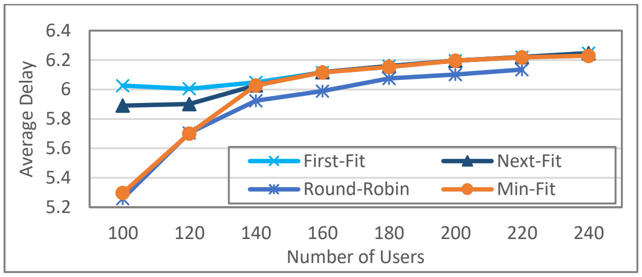

Fig. 5. Average delay with different number of users.

2) *Case B: Increasing the Number of Edge Servers*

Case B deals with the condition in which the number of edge servers increases for a fixed number of core servers. All algorithms are unable to provide solutions when less than 70 edge servers are available. Fig. 6 and 7 suggest that the First-Fit and Next-Fit methods provide the better solutions of minimum QoE and the maximum delay compared with the Round-Robin and Min-Fit methods. However, in terms of overall performance in average, the Round-Robin and Min-Fit methods outperform the First-Fit and Min-Fit methods in Fig. 8 and 9. Most importantly, the gaps between their overall results increase when there are more available resources. Min-Fit, is more practical than Round-Robin because it requires fewer servers for serving all VMs, thus making it easier to obtain feasible solutions.

TABLE V. EXPERIMENTAL ENVIRONMENT FOR CASE B

Parameter	**Value**
Number of edge servers	60-90
Number of core servers	30
Number of users	180

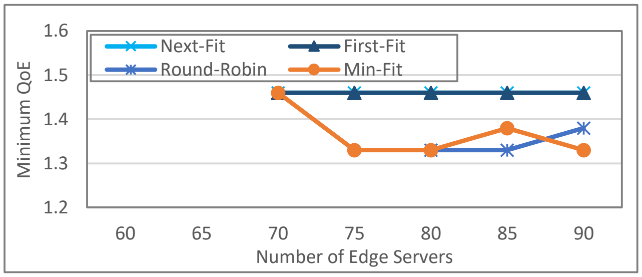

Fig. 6. Minimum QoE with different number of edge servers.

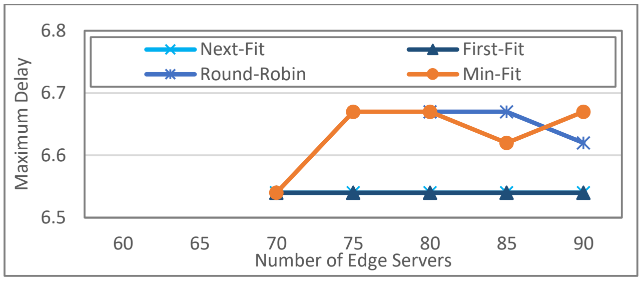

Fig. 7. Maximum delay with different number of edge servers.

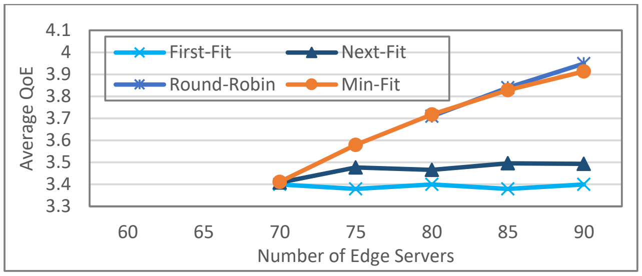

Fig. 8. Average delay gap with different number of edge servers.

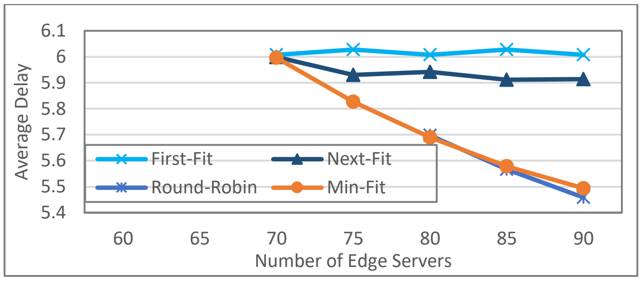

Fig. 9. Average delay with different number of edge servers.

3) *Case C: Increasing the Number of Core Servers*

Case C deals with the condition in which the number of core servers increases for a fixed number of edge servers. All methods are unable to provide feasible solutions when less than 30 core servers are available. Fig. 10 and 11 suggest that the First-Fit and Next-Fit methods provide the better solutions of minimum QoE and the maximum delay compared with the Round-Robin and Min-Fit methods. The Round-Robin method always requires a higher number of servers than other methods, making it inappropriate. Furthermore, the results of *Case C* in Fig. 12 and 13 suggest that the Min-Fit and Round-Robin methods provide larger average delay gap and less average delay than the First-Fit and Next-Fit methods do, respectively. The difference in average delay gap and average delay between the proposed method and other methods increases with the amount of resources.

TABLE VI. EXPERIMENTAL ENVIRONMENT FOR CASE C

Parameter	**Value**
Number of edge servers	70
Number of core servers	24-42
Number of users	180

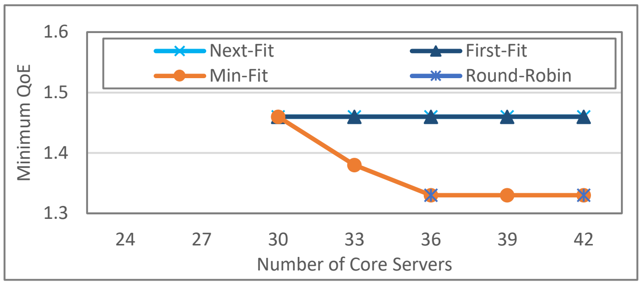

Fig. 10. Minimum delay gap with different number of core servers

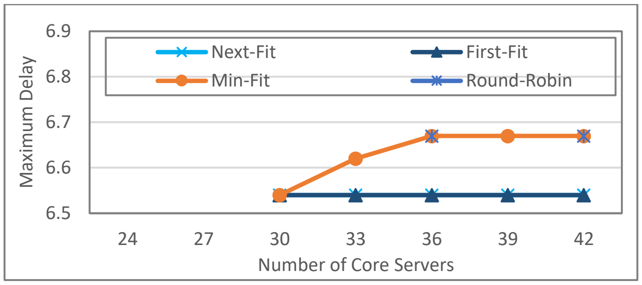

Fig. 11. Maximum delay with different number of core servers.

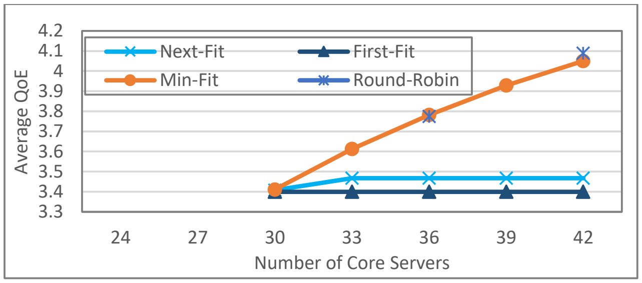

Fig. 12. Average delay gap with different number of core servers.

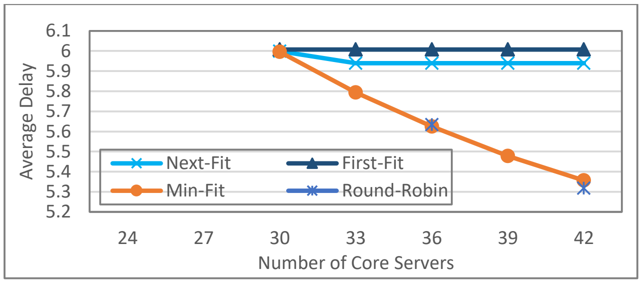

Fig. 13. Average delay with different number of core servers.

VI. Conclusions

The slicing concept was used to increase the flexibility of network infrastructure to realize the requirements of 5G networks and to satisfy diverse user demands. In this study, we formulate a system model with a slicing scheme and use the delay gap as the objective function. We present a simple and promising resource management approach and prove its effectiveness through three different experimental cases. More complicated conditions such as arriving users and their requirements will be considered in future works. Algorithms will be designed to solve problems, and their performance will be compared with those of existing methods.

Acknowledgment

This work is supported in part by the Ministry of Science and Technology (MOST), Taiwan, under grant numbers 106-2221-E-002-032 and 106-2221-E-305-004.

References

[1] R. Chaudhary, N. Kumar, and S. Zeadally, "Network Service Chaining in Fog and Cloud Computing for the 5G Environment: Data Management and Security Challenges," *IEEE Communications Magazine*, vol. 55, no. 11, pp. 114-122, November 2017.

[2] V. Sciancalepore, K. Samdanis, X. Costa-Perez, D. Bega, M. Gramaglia, and A. Banchs, "Mobile traffic forecasting for maximizing 5G network slicing resource utilization," in *the IEEE International Conference on Computer Communications* (INFOCOM 2017), Atlanta, GA, USA, pp. 1–9, May 2017.

[3] G. S. Aujla, R. Chaudhary, N. Kumar, J. J. P. C. Rodrigues, and A. Vinel, "Data Offloading in 5G-Enabled Software-Defined Vehicular Networks: A Stackelberg-Game-Based Approach," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 100–108, August 2017.

[4] H. Zhang, N. Liu, X. Chu, K. Long, A. H. Aghvami, and V. C. M. Leung, "Network Slicing Based 5G and Future Mobile Networks: Mobility, Resource Management, and Challenges," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 138–145, August 2017.

[5] M. Jiang, M. Condoluci, and T. Mahmoodi, "Network Slicing Management & Prioritization in 5G Mobile Systems," in *the Euro. Wireless 2016*, Oulu, Finland, pp. 1–6, May 2016.

[6] S. Sharma, R. Miller, and A. Francini, "A Cloud-Native Approach to 5G Network Slicing," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 120-127, August 2017.

[7] M. Peng, C. Wang, V. Lau, and H. V. Poor, "Fronthaul-constrained Cloud Radio Access Networks: Insights and Challenges," *IEEE Wireless Communications*, vol. 22, no. 2, pp. 152-160, April 2015.

[8] X. Xu, H. Zhang, X. Dai, Y. Hou, X. Tao, and P. Zhang, "SDN Based Next Generation Mobile Network with Service Slicing and Trials," in *China Communications*, vol. 11, no. 2, pp. 65-77, February 2014.

[9] J. Heinonen, P. Korja, T. Partti, H. Flinck and P. Pöyhönen, "Mobility Management Enhancements for 5G Low Latency Services," in *the 2016 IEEE International Conference on Communications Workshops* (*ICC 2016*), Kuala Lumpur, Malaysia, pp. 68-73, May 2016,.

[10] H. Zhang, C. Jiang, N.C. Beaulieu, X. Chu, X. Wang, and T.Q.S. Quek, "Resource Allocation for Cognitive Small Cell Networks: A Cooperative Bargaining Game Theoretic Approach," *IEEE Transactions on Wireless Communications*, vol. 14, no. 6, pp. 3481–3493, June 2015.

[11] D. Bega, M. Gramaglia, A. Banchs, V. Sciancalepore, K. Samdanis, and X. Costa-Perez, "Optimising 5G Infrastructure Markets: The Business of Network Slicing," in *the IEEE Conference on Computer Communications* (*INFOCOM 2017*), pp. 1-9, Atlanta, GA, USA, May 2017.