applied sciences

MDPI

Article

Reinforcement Learning Based Resource Management for Network Slicing

Yohan Kim, Sunyong Kim and Hyuk Lim *

School of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology (GIST), 123, Cheomdangwagi-ro, Buk-gu, Gwangju 61005, Korea; yohankim@gist.ac.kr (Y.K.); sunyongkim@gist.ac.kr (S.K.)

* Correspondence: hlim@gist.ac.kr; Tel.: +82-62-715-2229

Received: 3 May 2019; Accepted: 3 June 2019; Published: 9 June 2019

Abstract: Network slicing to create multiple virtual networks, called *network slice*, is a promising technology to enable networking resource sharing among multiple tenants for the 5th generation (5G) networks. By offering a network slice to slice tenants, network slicing supports parallel services to meet the service level agreement (SLA). In legacy networks, every tenant pays a fixed and roughly estimated monthly or annual fee for shared resources according to a contract signed with a provider. However, such a fixed resource allocation mechanism may result in low resource utilization or violation of user quality of service (QoS) due to fluctuations in the network demand. To address this issue, we introduce a resource management system for network slicing and propose a dynamic resource adjustment algorithm based on reinforcement learning approach from each tenant's point of view. First, the resource management for network slicing is modeled as a Markov Decision Process (MDP) with the state space, action space, and reward function. Then, we propose a Q-learning-based dynamic resource adjustment algorithm that aims at maximizing the profit of tenants while ensuring the QoS requirements of end-users. The numerical simulation results demonstrate that the proposed algorithm can significantly increase the profit of tenants compared to existing fixed resource allocation methods while satisfying the QoS requirements of end-users.

Keywords: network slicing; dynamic resource adjustment; Q-learning

1. Introduction

5G networks are expected to provide a wide variety of services to individual users and industry customers who may have different quality of service (QoS) requirements. The explosive data traffic demand in 5G has stressed the need for the network capacity increase. Thus, 5G should be more flexible and scalable enough to meet a variety of requirements of the traffic. However, traditional network upgrade methods for increasing the network capacity such as vertical and horizontal scaling create excessive operating and capital expenses of network operators. The concept of network slicing through resource sharing has been presented as a promising solution to achieve efficient and affordable 5G. Network slicing divides a single physical network into multiple virtual networks to be configured to provide specific QoS as required by different applications and traffic classes. From the network management point of view, the mechanism for resource allocation across slices is the major research direction for network slicing and enables each slice's own set of allocated resources to be separately operated and maintained

Network slicing provides new business opportunities for service providers and vertical industries, including healthcare, IoT, smart cities, and mobile broadband services. Since network slices will be used by traffic engineering businesses, network slicing is a matter of business and economic model as well as a simple resource allocation mechanism. Entities involved in the network slicing business can

be roughly categorized into slice providers and slice tenants. The owner of each network slice is known as a slice tenant [1]. The slice provider provides the customized network to the slice tenants [2]. Each entity independently pursues a profit-earning business model. The slice tenants have the discretion to decide how much amount of resource should be purchased according to the traffic and market conditions. The slice provider can establish a service level agreement (SLA) contract according to resource requests from slice tenants. In addition, users, who consume services offered by each tenant, pay tenants a fee for using the services.

Despite the resource sharing concept, which is a key idea of network slicing, has recently attracted considerable attention, an efficient use of resources is still a challenge. In legacy networks, every tenant pays a fixed and roughly estimated monthly or annual fee for shared resources according to a contract signed with a provider in an exclusive and excessive manner [3]. The 3rd Generation Partnership Project (3GPP) suggests that static resource allocation based on fixed network sharing can be one of the approaches for resource management in network slicing. However, such a static allocation mechanism may lead to low efficiency. For example, suppose that an amount of bandwidth enough to support the peak traffic demand is statically allocated to a network slice. In this case, the bandwidth utilization will be low on average due to continuous fluctuations in the network demand, even leading to a very high tenant's cost. Obviously, exclusive and excessive resource allocation, which cannot adapt to the change of network demand, is not a cost-effective method.

A dynamic resource trading mechanism can be a solution to the inefficient resource allocation. With dynamic trading of shared resources, a network operator can allocate the necessary resources as traffic changes. However, it is difficult to negotiate and optimize resource allocation, because the resource is not managed by a single management entity, and several business entities independently participate in resource trading for their own benefit. This dynamic trading involves a very high computational cost and can make it impractical for real-scenarios. As a solution, Q-learning can be a reasonable approach; it can choose an optimal action in real-time without considering the future information at upcoming slots, which may represent a definite advantage in real-world scenarios.

On the 5G standardization front, various approaches to resource management in network slicing are currently specified by the 3GPP. The slice tenant and provider are the entities that provide various services; thus, they should negotiate the resource according to the information such as the type of service and traffic fluctuations. In this paper, we propose a resource management mechanism based on variations of the traffic mix using Q-learning algorithm. Our approach is based on the ability of tenants to manage resource allocation and negotiate their resource with the provider. Since we focus on resource management from each tenant's point of view, the Q-learning algorithm operates on each tenant's resource management controller. A tenant has two business interfaces: one interface toward the provider, and the other toward the end-user. The former interface is used for acquiring network resources upon the requirements and paying for the resources to the provider. In other words, each independent tenant trades resources with the provider, and there is no direct resource transaction between the tenants. The tenant interacts with end-users using the latter interface to provide the resources to them. Under such a Q-learning-based dynamic resource trading environment, each tenant exhibits a strategic behavior to maximize its own profit. Meanwhile, since the tenants are independent business entities, they want to purchase as much amount of resource as they need, and, in this case, the physical resource may become scarce. Therefore, the provider manages the limited resource by increasing the resource price as the resource demand increases. This basic knowledge of the market is that the price increases if the demand increases, and this mechanism keeps the trading market in balance [4]. Our contributions are summarized as follows.

- We model a network slicing resource trading process between the slice provider and multiple tenants in a network slicing environment using a Markov Decision Process (MDP).
- We propose taking a ratio of the number of inelastic flows to the total number of flows in each network slice as an MDP state parameter to characterize different QoS satisfaction properties of each slice to the change of slice resource allocation.

- We propose a Q-learning-based dynamic resource management strategy to maximize tenant's profit while satisfying QoS of end-users in each slice from each tenant's point of view.

The rest of the paper is organized as follows. In Section 2, we briefly review the related work on resource management systems and business models in network slicing. In Section 3, we describe the service model and resource trading model for network slicing. In Section 4, we outline the design of the proposed resource management system using an MDP. In Section 5, we propose a dynamic resource adjustment algorithm based on the Q-learning approach. We present the performance evaluation results in Section 6. Finally, we conclude this paper in Section 7.

2. Related Work

The topic of network slicing has received substantial attention among the research community. The end-to-end network slicing allocates slice components spread across the entire network to the tenants to satisfy the diverse requirements with flexibility and scalability [5]. From the current technology's point of view, although network slicing is still a theoretical step, many researchers are looking forward to making a commercial product out of the network slicing concept for 5G systems because of the flexibility and scalability provided by network slicing. In fact, business opportunities of network slicing paradigm can promote an economic advantage for 5G. In [6], Habiba and Hossain conducted a survey of the economic aspects of network slicing virtualization in terms of network resource and pricing based on various auction mechanisms. They presented the fundamentals of auction theory in designing the business model and simultaneously attempted to optimize resource utilization and profit using a pricing rule. The auction theory is able to model interactions among the provider and multiple tenants. In [7], Luong et al. reviewed resource pricing approaches for efficient resource management to maximize the benefits of the provider and tenants. They emphasized that an efficient mechanism of resource allocation is required for the high resource utilization of the provider and QoS for end-users of tenants.

There has been a significant amount of research work on the management and orchestration of network slicing. In [8], Alforabi et al. comprehensively summarized principal concepts, enabling technologies and solutions as well as the current standardization efforts such as the software-defined networking (SDN) and network function virtualization (NFV). By presenting relevant research, they described how end-to-end network slicing can be achieved by designing, architecting, deploying, and managing network components in 5G networks. An important issue of network slicing is the resource allocation to satisfy a number of heterogeneous service requirements by taking the tenant's point of view. Under such a resource allocation model, every tenant has the ability to allocate network resources to their own slice according to their strategic behavior. In [9,10], Caballero et al. focused on resource allocation based on a game-theoretical model for tenants' strategic behavior and analyzed parameters to reduce cost while maintaining the QoS. The Fisher market mechanism has been used to share the resources of each tenant in a fair manner according to different requirements of end-users who are served by each tenant. The tenants indicate their preferences to the infrastructure and then receive resources of the base station according to their preferences. On the other hand, there is a strategy of the resource allocation and admission control focusing on how the mobile network operator (MNO) adjust slice parameters to maximize the overall revenue. When a tenant issues a service request to create a new slice, the MNO considers the admission of the slice. If MNO determines approval of the request, a slice is created and granted to the tenant. In [11], Bega et al. proposed a mechanism that MNO has the authority of admission control of tenants to maximize the revenue of a network. In [3], Han et al. proposed a genetic optimization for resource management across the slices using an MNO's binary decision.

However, the majority of such studies have not considered dynamically changing resource requirements over time. Thus, most of their business models were quite simple in that a fixed and roughly estimated monthly or annual fee was paid for the resource sharing. The operation in such a framework can often incur unnecessary expenditure due to unused resources or resource

scarcity. To address this issue, there are some studies that have considered resource trading in the context of network slicing. In [12], Xie et al. proposed a dynamic resource pooling and trading mechanism in network virtualization using Stackelberg game to achieve more efficient utilization of network resources. In [13,14], Akguel et al. proposed an algorithm for trading resources among tenants. However, their approach involves a high computational cost that can make it impractical for real-scenarios. In contrast, we consider Q-learning-based approach to determine how much amount of resource a tenant needs to purchase in advance using only the current information while considering the future rewards at upcoming slots. This reinforcement learning based approach can solve the resource trading problem for network slicing with reasonably low complexity, and thus is applicable to real-world applications in practice.

3. System Model

3.1. Overall System Architecture

In this section, we first describe the overall system architecture depicted in Figure 1. Table 1 summarizes the notation adopted in this work. Consider a network slicing infrastructure consisting of a set of network slices denoted as $\mathbb{N} = \{1, 2, \cdots, N\}$, where N is the number of slices. These multiple slices are built upon a physical network comprised of a radio access network and core network. There are two different types of resources spread across the entire physical network: virtual network function (VNF) resources and transmission resources. Network flows in a slice $n \in \mathbb{N}$, denoted as f, traverse a sequence of data centers executing a specific VNF and a number of physical transmission links and nodes before reaching the destination. The flows consume computing resources when traversing network servers, and occupy transmission resources on physical links and nodes for transmission. We treat the computing resources for network services, analytics and security in the data center as a single type of integrated VNF resources. Similarly, integrated transmission resources indicate wired and wireless bandwidth. We simply consider that such types of resources are proportional to the data rate. We assume that time takes discrete values denoted as $t = 1, 2, 3, \cdots$, and the achievable rate for a network slice at a time slot t is denoted as $R_{n,t}$. In this paper, bandwidth is used as an example of resources unless otherwise stated.

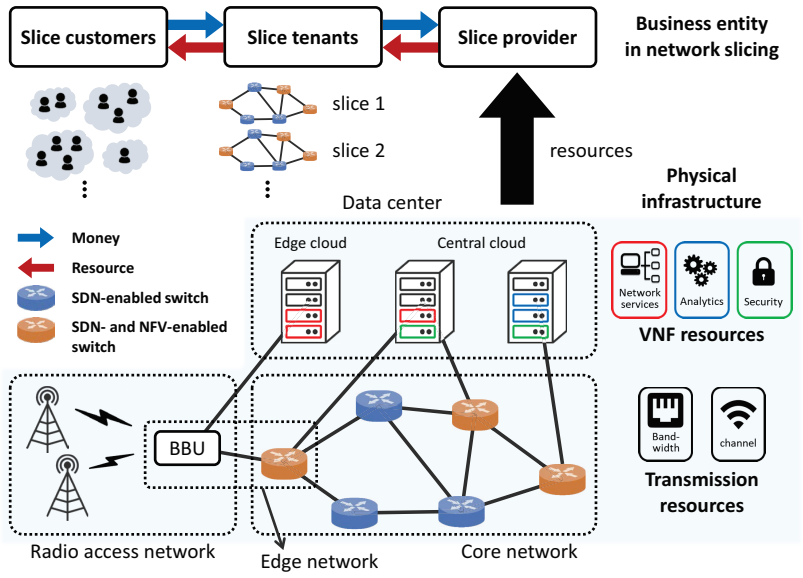


Figure 1. Business entities and shared resources in network slicing infrastructure.

Table 1. Parameters.

Parameter	Meaning
$n \in \mathbb{N}$	Set of network slices
$f \in F$	Set of flows
$i \in I$	Set of inelastic flows
$e \in E$	Set of elastic flows
t	Time slot
T	Lifetime of slice
$\hat{t}$	Time zone index
τ	Number of time zones
h	Interval of resource trading
$R_{n,t}$	Achievable rate of slice n at t
$R_{n,t}^{buy}$	Total purchased bandwidth of slice n at t
$R_{n,t}^{sell}$	Total selling bandwidth of slice n at t
$R_{n,t}^{violate}$	Total bandwidth that violates QoS of slice n at t
φ	Flow ratio
d_i	Instantaneous bandwidth demand of inelastic flow i
d_e	Average bandwidth demand of elastic flow e
$D_{n,t}$	Total bandwidth demand of slice n at t
r_t^i	Data rate consumed by inelastic flow i at t
r_t^e	Data rate consumed by elastic flow e at t
$s_t \in S$	Set of states
$a_t \in A$	Set of Actions
σ	Resource trading unit
$v(s_t, a_t)$	Reward function
V_t^π	Total discounted reward
p^{buy}	Unit buying price
p^{sell}	Unit selling price
ρ	Price ratio
α	Learning rate
γ	Discount factor
ϵ	ϵ-greedy parameter

In our system, three stakeholders, namely the slice provider, slice tenant, and end-user, interact to realize the end-to-end communication service. As the slice provider and a number of slice tenants are independent business entities, they act independently to generate their own profits. Each slice tenant makes requests for the creation of new network slice instances and manages own network slice. From the business point of view, the tenants try to purchase resources for slice creation from the provider upon receiving requirements of the end-users in their slices. The provider possesses a physical resource pool and can sell resources to the tenants on demand. The total amount of the resource pool is R^{phy}. The initial resource R^{ini} is first bundled into slices and then allocated to the slices. The tenants monitor the resource demand and utilization at every discrete time slot t. The resource amount allocated to each slice n at time slot t through resource trading is $R_{n,t}^{buy}$. The total resource amount allocated to all slices cannot exceed the amount of the available physical resource, (i.e., $R_{n,t} = \sum_{n \in \mathbb{N}} R_{n,t}^{buy} \leq R^{phy}$). Since there may exist unnecessary resources waste or resource scarcity due to demand fluctuations of end-users, the tenants should be able to trade unused resources and acquire additional resources from the provider. Therefore, each tenant should determine whether to increase or decrease the amount of resource being bought from the provider at each trading interval h. However, this adjustment may cause an overhead, especially for short periods of trading. Because an excessive resource adjustment rate can be a burden on a resource controller, the operator should set the trading interval h appropriately depending on the dynamics of the demand change.

Let p^{buy} and p^{sell} denote a unit buying price and a unit selling price of the bandwidth, respectively. Therefore, the tenants pay a fee of $p^{buy} \cdot R_{n,t}^{buy}$ for the purchased resources to the provider every time slot t. If the total resource demand of the slices and the amount of resource allocated to the slices

change, the provider also should change the resource price, which is the buying price. This leads tenants to reduce the amount of resource purchase by raising the price when the resource becomes scarce. We simply assume that the unit buying price is proportional to the total allocated resource amount of all slices as follows:

$$p^{buy} = \nu \cdot \sum_{n \in \mathbb{N}} R^{buy}_{n,t}, \quad (1)$$

where ν is a constant value. Similarly, the end-users pay a fee of $p^{sell} \cdot R^{sell}_{n,t}$ for their resource consumption to each tenant, where $R^{sell}_{n,t}$ denotes the resource amount that each tenant sells to the end-users. It is assumed that each tenant determines a fixed unit selling price when entering into the initial contract with the end-users.

3.2. Service and Traffic Model

In general, network slices may support different services (e.g., streaming, P2P, and web) consisting of a single type or multiple types of the network traffic. Thus, we assume that slices consist of different types of traffic depending on their services. Internet traffic is mainly categorized into two types of traffic: elastic traffic and inelastic traffic [15]. Since elastic traffic is less sensitive to the delay and bandwidth constraints, it does not instantly require minimum bandwidth requirements but usually prefers high throughput. The file transfer service is an example of the elastic traffic. On the other hand, inelastic traffic usually requires strict bandwidth to support real-time traffic such as streaming services and online gaming. For this reason, it always needs to receive instantaneous bandwidth at every time slot t. We denote $I = \{1, \ldots, |I|\}$ as the set of inelastic flows, and $E = \{1, \ldots, |E|\}$ as the set of elastic flows. Each inelastic flow $i \in I$ always requires an instantaneous bandwidth d_i at every time slot t, whereas each elastic flow $e \in E$ needs an average bandwidth d_e over a long time scale. In our system model, it is assumed that all slices are characterized by the ratio of elastic traffic to inelastic traffic. Our framework has the flexibility to accommodate various types of service models. For example, if the tenant is a streaming service operator, it may have a slice with almost inelastic flows and others with almost elastic flows. On the other hand, in the case the tenant is a mobile network operator, the flows of a slice can have different service requirements such as an elastic and inelastic traffic mix scenario.

To evaluate the QoS directly depending on the resources consumed by end-users, we describe how the respective QoS function of each of the traffic types can be obtained from the general utility function used in [16]. Figure 2 shows that the QoS function reflects the performance according to the bandwidth requirements of each traffic type. Typically, elastic traffic does not need to achieve the minimum rate, i.e., $d^{min}_e = 0$. Instead of the minimum rate, elastic traffic requires the average bandwidth, denoted as d^{avg}_e, because this traffic does not have strict delay and bandwidth constraints as mentioned above. It also does not require maximum bandwidth because high throughput is usually desirable. Therefore, its QoS gradually increases as the received bandwidth increases. Unlike elastic traffic, inelastic traffic always needs to achieve the minimum rate, i.e., $d^{min}_i > 0$. When the received bandwidth of each flow is lower than d^{min}_i, the QoS is the minimum value. QoS gradually increases as the received bandwidth increases between d^{min}_i and d^{max}_i similar to elastic traffic, which reflects the sensitivity of variations in QoS (e.g., video quality). Because improving the rate beyond what is required for the highest QoS is not useful, there exists a saturation region, i.e., the maximum QoS denoted as q^{max}. To easily analyze the performance, we assume in this study that QoS of each flow is q^{max} if the bandwidth is larger than d^{min}_i.

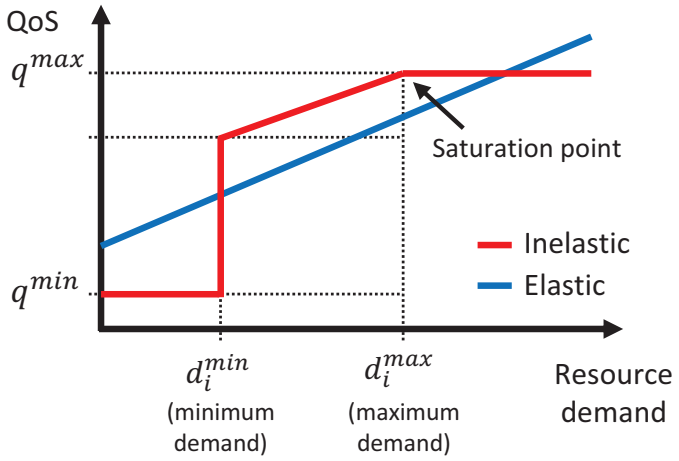


Figure 2. QoS with respect to the resource demand.

According to the system model explained above, the bandwidth is provided to inelastic flows at time slot t first. Thus, sufficient bandwidth is provided to inelastic flows to guarantee the minimum demand, while elastic flows equally receive the share of the remaining bandwidth. Therefore, inelastic flows are first provided with sufficient bandwidth as follows:

$$r_t^i = \min(\frac{R_{n,t}}{|I|}, d_i^{min}), \tag{2}$$

where r_t^i is the data rate consumed by inelastic flow i at t. If the number of inelastic flows is too large (i.e., $\sum_{i \in I} d_i^{min} / R_{n,t} > 1$), not all inelastic flows can receive enough bandwidth. In this case, we meet the demands of as many inelastic flows as possible and provide the remaining bandwidth to elastic flows. The remaining resources are equally shared among the elastic flows as follows:

$$r_t^e = \frac{R_{n,t} - \sum_{i=1}^{|I|} r_t^i}{|E|}, \tag{3}$$

where r_t^e denotes the data rate consumed by elastic flow e at t. Both inelastic and elastic traffic occur following a Poisson process, and the flows' lifetime follows exponential distribution in our system model.

4. Resource Trading System Using a Markov Decision Process

In this section, we outline the design of our dynamic resource trading system that aims at maximizing the profit of a slice tenant using an MDP. The MDP is defined by a 4-tuple as $M = \langle S, A, V, P \rangle$, where S, A, V, and P denote the state space, action space, reward function, and transition probability, respectively. In any state $s_t \in S$ at time slot t, one of the available actions $a_t \in A$ can be selected. After the selected action has been taken, the state s_t may change to the next state s_{t+1} with a state transition probability of $P(s_{t+1}|s_t, a_t)$. $V(s_t, a_t)$ represents an immediate reward that can be obtained by taking action a_t in state s_t.

4.1. State Space

In our system, the tenant plays a role as an agent that interacts with the environment and exhibits strategic behavior to maximize its reward. The state space at time slot t can be defined as

$$s_t = [\varphi, D_{n,t}, \rho, \hat{t}] \in S, \tag{4}$$

where φ, $D_{n,t}$, ρ, and $\hat{t}$ are the flow ratio of the slice, total resource demand, current market price ratio, and time zone index of the current time slot, respectively. The information of these four state elements is considered to be known to the tenant at each time slot t.

To characterize the type of services supported by each slice, we define the flow ratio, denoted as φ, which is computed by dividing the number of inelastic flows by the total number of flows as follows:

$$\varphi = \frac{|I|}{|I| + |E|}, \tag{5}$$

where I and E denote the set of inelastic and elastic flows, respectively. We expect φ to be used as a parameter to determine how much bandwidth the tenant purchases in advance. It is a very important approach for our dynamic resource trading system because the system will violate QoS and get a penalty if there is not enough immediate bandwidth. The effect of φ on the system performance is analyzed in detail in Section 6.

Let $D^i_{n,t}$ be the sum of the minimum demands of all inelastic flows belonging to slice n at time slot t and $D^e_{n,t}$ denote the sum of the average demands of all elastic flows served by slice n at time slot t, respectively, as follows:

$$D^i_{n,t} = \sum_{i \in I} d^{min}_i, \; D^e_{n,t} = \sum_{e \in E} d^{avg}_e. \tag{6}$$

Then, we can define the total resource demand at time slot t, which is the sum of the minimum demands of all inelastic flows and average demands of all elastic flows, as follows:

$$D_{n,t} = D^i_{n,t} + D^e_{n,t}. \tag{7}$$

If the total bandwidth demand exceeds the total purchased bandwidth at time slot t, i.e., $D_{n,t} > R_{n,t}$, resource utilization is 1 because the used bandwidth cannot exceed the total purchased bandwidth.

The price ratio, denoted as ρ, represents the ratio of the unit price of the bandwidth that tenants purchase from the provider to the unit price of the bandwidth that they sell to the end-user, i.e., $\rho = p^{buy}/p^{sell}$. ρ is included in the state since it highly affects the profit of the slice tenant. We assume that p^{buy} is determined by the provider according to the amount of the total resource remained in the physical network. For example, if the amount of remaining resource in the physical network is decreased, the provider increases p^{buy}. This implies that p^{buy} can play an important role in the resource balancing in the physical network through regulating the amount of resource being purchased by adjusting p^{buy}.

Time slot needs to be included in the state because the tenant may act differently at different time slots. As an effort to avoid a situation that the number of states goes infinitely high, we apply the concept of the time zone; i.e., we use the index of the time zone rather than the index of the actual time slot. Let τ and T denote the number of time zones in the slice and the lifetime of the slice, respectively. Then, the time zone index of the current time slot can be defined as follows:

$$\hat{t} = \left\lceil \frac{t \cdot \tau}{T} \right\rceil. \tag{8}$$

4.2. Action Space

In a given state s_t, the tenant selects an action a_t from the set of possible actions defined as

$$A = [+\sigma, +2\sigma, +4\sigma, +8\sigma, -\sigma, -2\sigma, -4\sigma, -8\sigma, 0], \tag{9}$$

where σ represents the resource trading unit. For instance, if a positive/negative action is taken, the agent increases/decreases the amount of resource being bought from the provider. $a_t = 0$ means that the agent keeps maintaining the amount of resource. We define a possible action set $\bar{A}$ in the action space A. In each trading interval h, $\bar{A}$ is constrained by the amount of the allocated resource.

For example, if a slice does not have enough resource to satisfy the requirements, i.e., $R_{n,t} < D_{n,t}$, it certainly has to purchase more bandwidth, whereas if $R_{n,t} = D_{n,t}$, one more action, $a_t = 0$, is also available. Otherwise, all actions are available. As a result, $\bar{A}$ can be determined as follows:

$$\bar{A} = \begin{cases} [+\sigma, +2\sigma, +4\sigma, +8\sigma], & \text{if } R_{n,t} < D_{n,t}, \\ [+\sigma, +2\sigma, +4\sigma, +8\sigma, 0], & \text{if } R_{n,t} = D_{n,t}, \\ A, & \text{otherwise.} \end{cases} \tag{10}$$

4.3. Reward Function

Let $v(s_t, a_t)$ denote the reward function that returns a reward indicating the profit of the tenant. When action a_t is taken at state s_t, $v(s_t, a_t)$ can be defined as

$$v(s_t, a_t) = p^{sell} \cdot (R_{n,t}^{sell} - \omega \cdot R_{n,t}^{violate}) - p^{buy} \cdot R_{n,t}^{buy}, \tag{11}$$

where ω is the QoS-weight factor determining the importance of the QoS compared to the profit. In Equation (11), $R_{n,t}^{violate}$ is an amount of bandwidth that violates QoS when the total resource demand $D_{n,t}$ exceeds the achievable rate $R_{n,t}$ at time slot t (i.e., $R_{n,t}^{violate} = R_{n,t} - D_{n,t}$). In other words, if the amount of bandwidth is insufficient compared to the resource demand, the QoS of some flows may be violated. In this case, a penalty will be imposed by $\omega \cdot R_{n,t}^{violate}$. To let the system meet the QoS more strictly, we can increase ω to impose a larger penalty for the QoS violations. Using the reward $v(s_t, a_t)$, we define the total discounted reward V_t^{π} to take into account future rewards as follows:

$$V_t^{\pi} = v(s_t, a_t) + \sum_{i=1}^{\infty} \gamma^i \cdot v(s_{t+1}, a_{t+1}), \tag{12}$$

where $0 \leq \gamma \leq 1$ is a discount factor weighting future rewards relative to current rewards. Our ultimate goal is to find an optimal policy that can maximize V_t^{π}. In the next section, a dynamic resource adjustment algorithm using Q-learning is proposed to find the optimal policy.

5. Dynamic Resource Adjustment Algorithm

Based on the model presented in the previous section, we propose a dynamic resource adjustment algorithm that aims at maximizing slice tenant's profit by learning with the reward resulting from the previous action while satisfying the requirements of flows. Our adjustment algorithm is based on the Q-learning approach. Q-learning is one of the most widely used reinforcement learning techniques, which finds and selects an optimal action based on Q-function that defines state–action pairs. A Q-value represents the expected long-term reward of an agent under the assumption that it chooses the best actions from a specific state. First, the agent perceives the current states of the system and takes an action according to policy π defined as

$$\pi = \max_{a \in A} Q(s_t, a_t), \tag{13}$$

where $Q(s_t, a_t)$ is the Q-value, and the system keeps a memory for all state–action pairs. If the action a_t that has the maximum Q-value at the current state is selected by the policy, the reward is calculated according to Equation (11), leading to updating the Q-value. If the agent infinitely visits all state–action pairs, Q-values converge to an optimal value.

However, always taking the best action with the maximum Q-value at every state (i.e., exploitation) may result in a wrong decision after the completion of learning. The agent should be exposed to as many states as possible to learn how to deal optimally with all of them. Thus, the algorithm also needs to explore all possible (s_t, a_t) options, including those that do not maximize the Q-value (i.e., exploration). The exploration strategy involves performing random actions to experience

environmental changes to preclude from focusing on immediate rewards, whereas the exploitation strategy makes use of only what the agent knows already. We adopt the ϵ-greedy method to establish the exploration and exploitation balance. It means that the agent exploits the action that minimizes the immediate reward $(1-\epsilon)$ and explores random actions with a probability ϵ. Therefore, the probability of selecting action a_t in state s_t under policy π, denoted as $P_\pi(s_t, a_t)$, can be represented as

$$\forall a_t \in A,\ P_\pi(s_t, a_t) = \begin{cases} 1-\epsilon+\dfrac{\epsilon}{|A|}, & \text{if } a_t = A^*, \\ \dfrac{\epsilon}{|A|}, & \text{if } a_t \neq A^*. \end{cases} \tag{14}$$

To maximize the overall reward of the agent, the algorithm constantly learns the characteristics of the system by exploring it and taking decisions many times. The action to be taken in the current state depends on the Q-value $Q(s_t, a_t)$. The learning process continuously updates Q-values until the system can take the best action for all states. The agent updates the action-value function according to the following rule:

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha\left\{v(s_t, a_t) + \gamma \max_{a \in A} Q(s_{t+1}, a)\right\}, \tag{15}$$

where $0 \leq \alpha \leq 1$ denotes the learning rate that determines how fast the learning happens, whereas the discount factor $0 \leq \gamma \leq 1$ weighs the importance of future rewards in making the current decision. When α is small, it will take a long time for the algorithm to converge, whereas, with a large α, the algorithm might not converge. Setting γ to a small value results in the Q-value being optimized by immediate rewards, whereas, with large γ values, the algorithm heavily depends on future rewards. An appropriate selection of α and γ can allow achieving effective control of the Q-learning process.

The pseudo-code of the proposed resource adjustment algorithm is shown in Algorithm 1. The algorithm maintains the Q-values that are being iteratively updated when new events happen. First, the parameters for Q-learning are initialized. In particular, the Q-values are initialized to zeros (Lines 1–2). At each time slot t, the agent monitors the states (i.e., the flow ratio of the slice, total resource demand, current market price ratio and time zone index) (Line 4). The possible action set $\bar{A}$ considering the current resource state is determined for each trading interval (Line 6). An action a_t with the highest Q-value or a random action with a probability of ϵ following the ϵ-greedy policy can be taken from $\bar{A}$ (Line 7). Based on the selected action, the environment changes the state and provides the agent with a reward (Line 8). The algorithm calculates the Q-value using the learning rate α (Line 9).

Algorithm 1 Resource adjustment algorithm using Q-learning.

1: Initialize $Q(s, a) = 0, \forall s \in S$ and $\forall a \in A$.
2: Initialize learning parameters α, γ and ϵ
3: **for** each time slot t **do**
4: Monitor states $s_t = [\varphi, D_{n,t}, \rho, \hat{t}]$
5: **for** each trading interval h **do**
6: Determine a possible action set $\bar{A}$ by (10)
7: Select action a_t from $\bar{A}$ using policy π derived from ϵ-greedy by (13)
8: Take action a_t and observe $v(s_t, a_t)$ and s_{t+1}
9: Update $Q(s_t, a_t)$ by (15)
10: **end for**
11: $t \leftarrow t+1$ and $s_t \leftarrow s_{t+1}$
12: **end for**

6. Performance Evaluation

This section describes the MATLAB simulation results to evaluate the performance and effectiveness of the proposed Q-learning-based resource adjustment algorithm.

6.1. Setup of the Simulation Environment

We implemented a network slicing simulation environment to simulate behaviors of a slice tenant and evaluate its profit and QoS. The tenant can trade resources according to the strategies defined in the simulation. Once the slice is created, the initial resource R^{ini} is allocated to the slice, and the slice has to keep the minimum resource ($\min_t R_{n,t} = R^{ini}$). We considered a scenario with 200 flows, where the ratio φ between inelastic and elastic flows is randomly chosen to be between 0 and 1. We set the number of time zones τ to 10 and the lifetime of the slice to 100 time slots. Unless otherwise stated, the unit buying price and unit selling price of the bandwidth were set to \$40 and \$200, respectively, and the QoS weight ω was set to 1.6. We assumed that requests for flow creation are issued following a Poisson process of rate λ, and every flow has a random lifetime that follows an exponential random variable of rate μ. The initial resource R^{ini} is first bundled into slices and then allocated to slices.

6.2. Q-learning Algorithm Convergence

To demonstrate the effectiveness of our Q-learning algorithm, we observed the average reward of Q-learning with respect to trading episodes. The Q-values are stored in the Q-table and updated during the training process of the algorithm. The Q-table is initialized with zeros. We considered that the Q-learning algorithm is implemented by the tenant with $\alpha = 0.1$ and $\gamma = 0.9$. To speed up the convergence rate, α can be set closer to 1; however, a higher value of α would lead to local optimization. We also employed the ϵ-greedy strategy to make sure that all state–action pairs are explored enough before converging to a particular action. Since the value of exploration was high at the beginning, we defined ϵ as a linear function of the episode. In other words, we set the initial value of ϵ to 1, and ϵ approached 0 as the episode number increased. This was only done for the first 50% of the episodes, whereas ϵ was 0 for the rest of the episodes. This was to ensure that every action was performed a sufficient number of times before converging to a particular action.

Figure 3 illustrates the convergence speed of the Q-learning algorithm with the average reward and standard deviation (red region) obtained per episode during the learning process. We assumed that the system had reached convergence when the mean of all Q-values did not change over time. It was observed that Q-values started from a low initial reward level but quickly converged to a competitive trading strategy with a satisfying reward performance within hundreds of episodes. This result shows that Q-values are successfully learned after a certain number of episodes and can achieve better resource trading than that without any prior training.

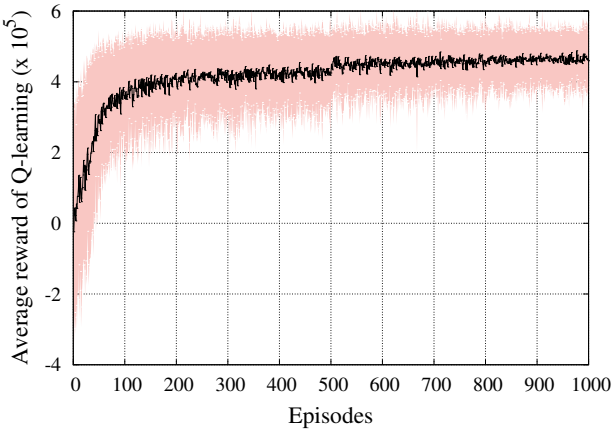


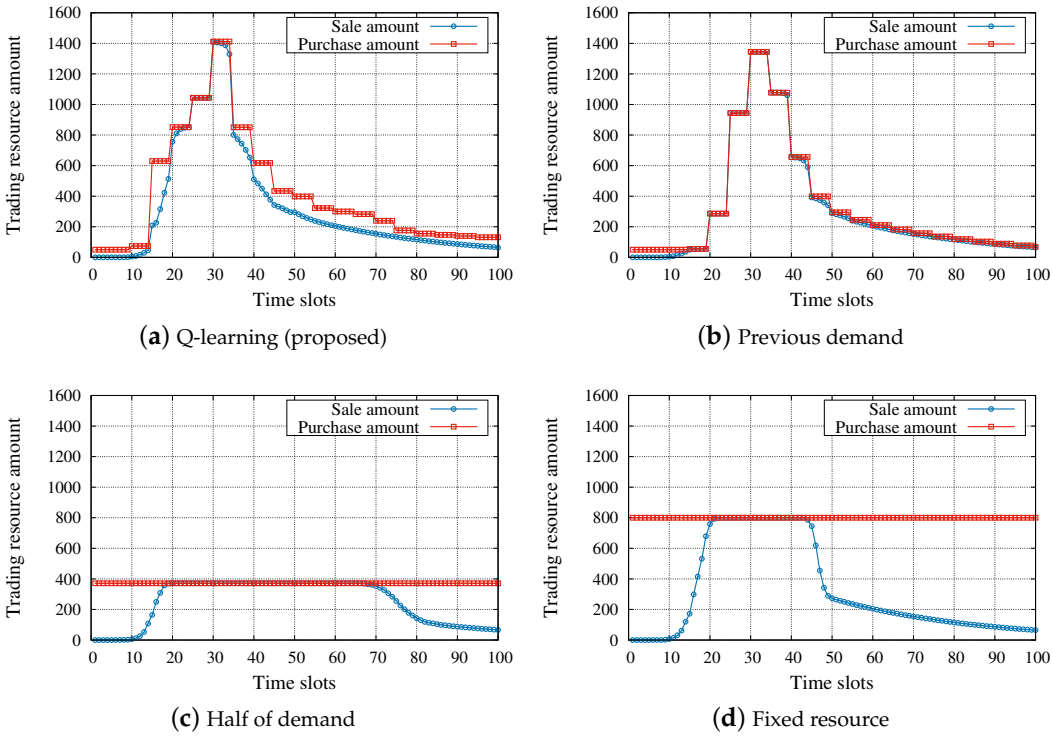
Figure 3. The average reward over episodes using Q-learning with the parameters, $\alpha = 0.1$, $\gamma = 0.9$.

6.3. Competitiveness

To verify the competitiveness of our algorithm, we next evaluated the performance of our algorithm by comparing it with the following strategies:

- *Fixed-resource strategy* allocates enough resources in an exclusive and excessive manner. This strategy is the most straightforward algorithm.
- *Half-of-demand strategy* allocates resources equal to a half of the entire resource demand of the previous episode.
- *Previous-slot-demand strategy* dynamically allocates resources corresponding to the resource demand of the previous time slot.

Among these strategies, *fixed resource* and *half of demand* are static resource allocation algorithms, whereas *previous demand* and our Q-learning-based algorithm are dynamic trading strategies. Figure 4 shows the purchase and sale amounts obtained by each of the strategies. The same simulation parameters were used in all four simulations as mentioned above, where the flow ratio was chosen randomly. In the static resource allocations, available resources are distributed among tenants based on a predefined algorithm regardless of current states of slices. Note that we set the fixed purchase amount of *fixed-resource* method so that the QoS satisfaction rate is less than 0.01 (i.e., $R_{n,t}^{buy} = 800$). As expected, the *fixed-resource* method, which is the basic strategy of a legacy network, does not violate the QoS but results in a very low resource utilization. Although the *half-of-demand* strategy tries to overcome the low resource utilization problem of the *fixed-resource* strategy, the QoS can be violated to a greater extent when increasing the proportion of inelastic flows in the slice. In the dynamic resource adjustment algorithms, the slices' states are used to effectively allocate existing resources. The two dynamic strategies adopted a trading interval of 5. We can observe in Figure 4a that our algorithm performed very close to the optimal policy (i.e., the sale amount equals the purchase amount at every time slot). In particular, unlike the *previous-slot-demand* strategy, our algorithm pre-purchases resources to avoid the QoS violation when resources are abruptly needed.



(**a**) Q-learning (proposed) (**b**) Previous demand

(**c**) Half of demand (**d**) Fixed resource

Figure 4. Change of the trading resource amount with respect to the time slot.

6.4. Impact of the Trading Interval on the Algorithm Performance

Generally, setting a short trading interval results in an increase of the computational cost. The short trading interval may affect controller overhead in the case when tenants request a very dynamic resource demand. Because the computational resource is unlimited, the trading interval should be long enough. Otherwise, the result of the resource trading may fluctuate severely, making a negative impact on the resource allocation performance. For these reasons, we conducted a simulation to assess how the profit and QoS change for different values of the trading interval h.

Figure 5 illustrates the comparison of the tenant profit achieved by the five algorithms. The static algorithms, i.e., *half-of-demand* and *fixed-resource*, are not affected by the changes in the trading interval. The tenant has to pay for the resources that end-users do not actually use since the requested resources are statically provisioned. Therefore, there is no way to make an extra income from the unused resources resulting in a generally low profit. The dynamic strategies do not need to exclusively reserve resources every time, and they can trade the unused or additional resources at the next trading interval. As the trading interval increased, the *previous-slot-demand* algorithm demonstrated a slightly decreasing profit because it did not consider future information, whereas the proposed algorithm was scarcely affected by the changes in the trading interval and earned the highest profit compared to the other algorithms.

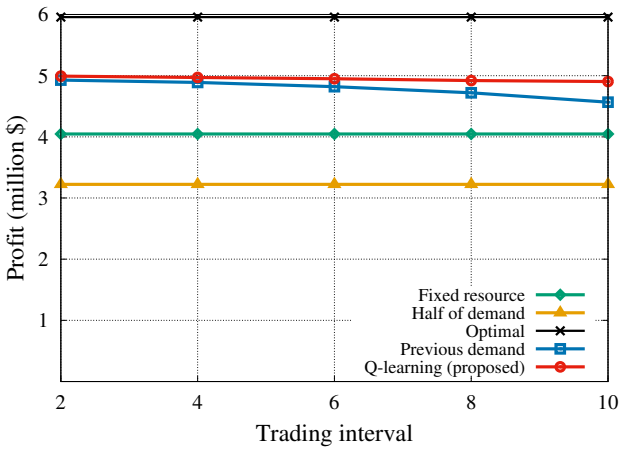


Figure 5. Profit with respect to the trading interval.

Figure 6 demonstrates that the proposed algorithm kept a very high QoS satisfaction rate because it allowed purchasing the required bandwidth in advance. This is the result of taking optimal actions according to the states for every t on the basis of the state–action pairs, i.e., the Q-values. On the other hand, the QoS satisfaction rate achieved by *previous-slot-demand* strategy decreased with the increasing trading interval. Similar to the previously stated result for this strategy, this indicates that the algorithm only focuses on the demand at the previous trading interval without considering the upcoming demand. The obtained results show that our algorithm works well even with a large trading interval to avoid the control overhead due to dynamic resource trading.

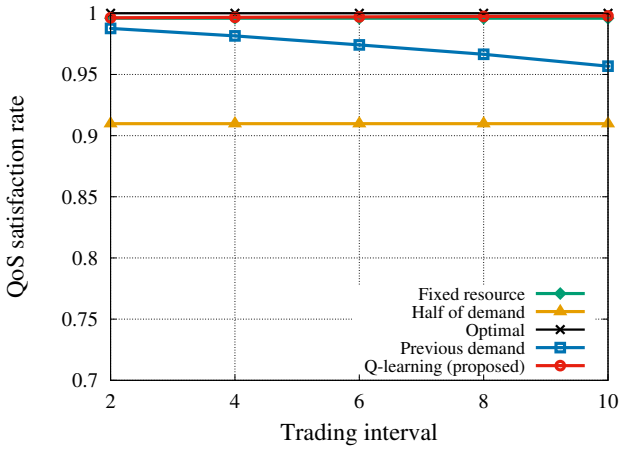


Figure 6. QoS satisfaction rate with respect to the trading interval.

6.5. QoS Violation WITH Flow Ratio Change

To study the impact of the flow ratio on the algorithm performance, we plotted the QoS satisfaction rate of each algorithm according to the flow ratio φ in Equation (5). Note that we set the trading interval to 5 unless otherwise noted. Since inelastic flows usually require strict bandwidth, if the proportion of inelastic flows become large, the number of QoS violations may increase.

Indeed, the result illustrated in Figure 7 demonstrates that the QoS satisfaction rate of all algorithms decreased as φ increased in general. In particular, even though the *previous-slot-demand* strategy is a dynamic adjustment algorithm, it depends upon the preceding information before trading interval h. Since this algorithm did not prepare for sudden network traffic increase, QoS satisfaction rate decreased as the proportion of inelastic flows that require instantaneous bandwidth increased. Our algorithm guaranteed the QoS regardless of the inelastic flow ratio except in the case of the overwhelming number of inelastic traffic (of course pretty good performance). The result shows that our algorithm could adaptively determine how much amount of resource a tenant purchases in advance according to the inelastic flow ratio.

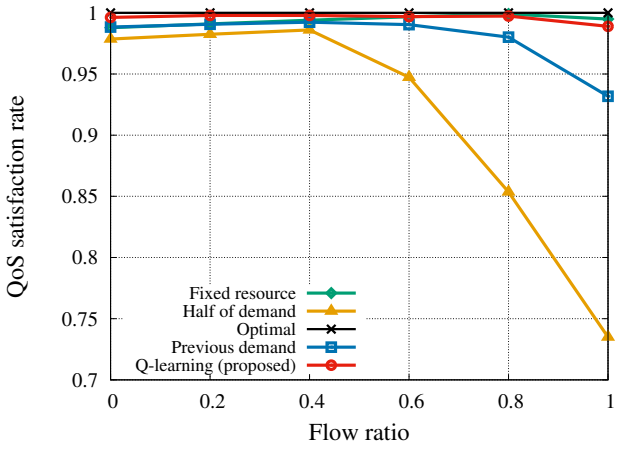


Figure 7. QoS satisfaction rate with respect to the flow ratio.

6.6. QoS Weight

We investigated the effect of penalty setting on the safety pursuit of the resource trading in Q-learning. In Equation (11), the penalty is a function of the QoS weight ω. The tenant putting a high value on the QoS satisfaction rate of end-users may set the QoS weight large for the Q-learning process. In this case, the tenant purchases more resources than it needs because it tends to pursue safety against QoS violations.

Figure 8 shows the total amount of the resource that the tenant trades during the lifetime of the slice. When a high QoS weight was applied, the tenant unnecessarily purchased more resources. This implies that a higher QoS weight can decrease the number of QoS violations but also lead to a lower profit.

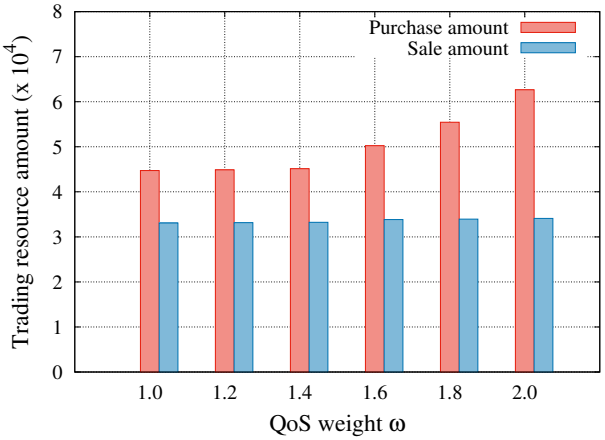


Figure 8. Trading resource amount with respect to the QoS weight

6.7. Adaptive Resource Trading Based on the Market Price.

Finally, we investigated the effects of the pricing adjusted by the provider on the resource trading. We assumed that the buying price p^{buy} is relative to the total allocated resource amount of all slices in Equation (1). This is a consideration of the following two concerns in our business model. One concern is in terms of tenants: what should be done if sudden physical resource shortages arise due to increased traffic and increased resource requirements. It is reasonable to allow tenants to determine their resource amounts before a resource shortage occurs. The other concern is in terms of a provider, who pursues its profits as an independent business entity. As can be noticed in Figure 9, only the proposed algorithm adjusted the purchase amount of the resource according to the buying price change. The change of the buying price may be caused by an increase in the total traffic volume of all slices. Since we set the selling price to \$200, purchasing resources at a buying price larger than \$200 would lead to a serious profit loss. Considering this fact, we can observe that our algorithm minimized the damage by reducing the amount of purchased resource when the unit buying price exceeded \$200. This result also indicates that the provider is able to manage and balance the allocation of limited resources through the price adjustments.

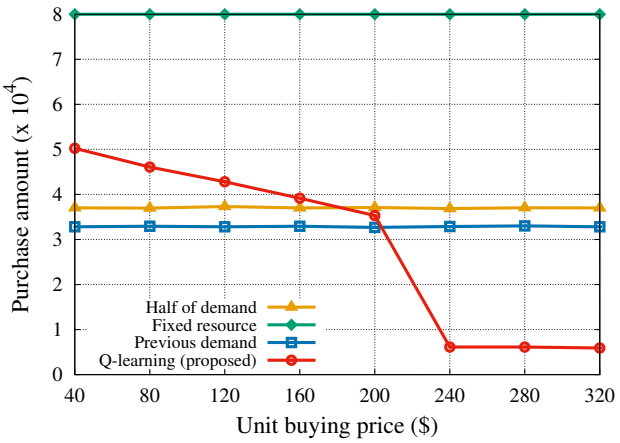


Figure 9. Purchase amount with respect to unit buying price.

7. Conclusions

In this study, we considered the dynamic resource trading in network slicing to maximize the profit of tenants while ensuring the QoS requirements of end-users. We demonstrated that our proposed Q-learning-based algorithm can dynamically trade resources with the provider considering both short and long trading intervals based on several states of the tenant. The obtained numerical simulation results demonstrate that the proposed algorithm can significantly increase the profit of the tenant compared to the existing fixed resource allocation methods.

Author Contributions: Conceptualization, Y.K.; methodology, Y.K. and S.K.; investigation, Y.K.; formal analysis, Y.K.; validation, Y.K.; writing—original draft preparation, Y.K.; writing—review and editing, S.K. and H.L.; and supervision, H.L.

Funding: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2017R1A2B2010478).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Garcés, P.C.; Pérez, X.C.; Samdanis, K.; Banchs, A. RMSC: A Cell Slicing Controller for Virtualized Multi-tenant Mobile Networks. In Proceedings of the IEEE Vehicular Technology Conference (VTC), Glasgow, UK, 11–14 May 2015; pp. 1–6.
2. Zhou, X.; Li, R.; Chen, T.; Zhang, H. Network Slicing as a Service: Enabling Enterprises' Own Software-Defined Cellular Networks. *IEEE Commun. Mag.* **2016**, *54*, 146–153. [CrossRef]
3. Han, B.; Lianghai, J.; Schotten, H.D. Slice as an Evolutionary Service: Genetic Optimization for Inter-Slice Resource Management in 5G Networks. *IEEE Access* **2018**, *6*, 33137–33147. [CrossRef]
4. Whelan, J.; Msefer, K.; Chung, C.V. Economic Supply & Demand. In *MIT System Dynamics in Education Project*; MIT: Cambridge, MA, USA, 2001.
5. Rost, P.; Mannweiler, C.; Michalopoulos, D.S.; Sartori, C.; Sciancalepore, V.; Sastry, N.; Holland, O.; Tayade, S.; Han, B.; Bega, D.; et al. Network Slicing to Enable Scalability and Flexibility in 5G Mobile Networks. *IEEE Commun. Mag.* **2017**, *55*, 72–79 [CrossRef]
6. Habiba, U.; Hossain, E. Auction Mechanisms for Virtualization in 5G Cellular Networks: Basics, Trends, and Open Challenges. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2264–2293. [CrossRef]
7. Luong, N.C.; Wang, P.; Niyato, D.; Wen, Y.; Han, Z. Resource Management in Cloud Networking using Economic Analysis and Pricing Models: A Survey. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 954–1001. [CrossRef]
8. Afolabi, I.; Taleb, T.; Samdanis, K.; Ksentini, A.; Flinck, H. Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies and Solutions. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2429–2453. [CrossRef]
9. Caballero, P.; Banchs, A.; de Veciana, G.; Costa-Pérez, X. Network Slicing Games: Enabling Customization in Multi-tenant Networks. In Proceedings of the IEEE International Conference on Computer Communications (INFOCOM), Atlanta, GA, USA, 1–4 May 2017; pp. 1–3.
10. Caballero, P.; Banchs, A.; de Veciana, G.; Costa-Pérez, X.; Azcorra, A. Network Slicing for Guaranteed Rate Services: Admission Control and Resource Allocation Games. *IEEE Trans. Wirel. Commun.* **2018**, *17*, 6419–6432. [CrossRef]
11. Bega, D.; Gramaglia, M.; Banchs, A.; Sciancalepore, V.; Samdanis, K.; Costa-Pérez, X. Optimising 5G Infrastructure Markets: The Business of Network Slicing. In Proceedings of the IEEE International Conference on Computer Communications (INFOCOM), Atlanta, GA, USA, 1–4 May 2017; pp. 1–9.
12. Xie, W.; Zhu, J.; Huang, C.; Luo, M.; Chou, W. Network Virtualization with Dynamic Resource Pooling and Trading Mechanism. In Proceedings of the IEEE Global Communications Conference (GLOBECOM), Austin, TX, USA, 8–12 December 2014; pp. 1829–1835.
13. Akguel, O.U.; Malanchini, I.; Suryaprakash, V.; Capone, A. Service-Aware Network Slice Trading in a Shared Multi-Tenant Infrastructure. In Proceedings of the IEEE Global Communications Conference (GLOBECOM), Singapore, 4–8 December 2017; pp. 1–7.
14. Akguel, O.U.; Malanchini, I.; Capone, A. Dynamic Resource Trading in Sliced Mobile Networks. *IEEE Trans. Netw. Serv. Manag.* **2019**, *16*, 220–233. [CrossRef]

15. Cheung, M.H.; Mohsenian-Rad, A.H.; Wong, V.W.; Schober, R. Random Access for Elastic and Inelastic Traffic in WLANs. *IEEE Trans. Wirel. Commun.* **2010**, *9*, 1861–1866. [CrossRef]
16. Jin, J.; Sridharan, A.; Krishnamachari, B.; Palaniswami, M. Handling Inelastic Traffic in Wireless Sensor Networks. *IEEE J. Sel. Areas Commun.* **2010**, *28*, 1105–1115. [CrossRef]