Dynamic Network Slicing for Fog Radio Access Networks

Almuthanna Nassar, and Yasin Yilmaz, *Member, IEEE*
Electrical Engineering Department, University of South Florida, Tampa, FL 33620, USA
E-mails: {atnassar@mail.usf.edu; yasiny@usf.edu}

***Abstract*—Fog radio access network (F-RAN) has been recently proposed to satisfy the quality-of-service (QoS) requirements of the ultra-reliable-low-latency-communication (URLLC) IoT applications, hence fog nodes are empowered with computing and storage resources to independently deliver network functionalities at the edge of network without referring the users to the cloud. However, due to their limited resources, fog nodes should utilize their resources intelligently for low latency IoT applications to leverage the complementarity with cloud computing. We consider the problem of sequentially allocating fog node's limited resources to various IoT applications with heterogeneous latency needs. We formulate the problem as a finite-horizon Markov Decision Process (MDP), and present the optimal solution, known as the optimal policy, through dynamic programming. The fog node learns the optimal policy through interaction with the IoT environment, which enables adaptive resource allocation in different IoT environments. Comprehensive simulation results for various IoT environments corroborate the theoretical basis of the proposed MDP method.**

***Index Terms*—IoT communications, 5G cellular networks, Low-latency communications, Resource allocation, Markov decision process.**

I. Introduction

There is an ever-growing demand for wireless communication technologies to cope with the growing number of IoT devices and the increasing amount of traffic. For better user satisfaction, cloud radio access network (C-RAN) architecture is suggested for 5G, in which a powerful cloud controller with a pool of baseband units (BBU) and a storage pool supports a large number of distributed remote radio units (RRU) through high capacity fronthaul links [1]. However, C-RAN structure places a huge burden on the centralized cloud controller and its fronthaul, which causes more delay due to the limited fronthaul capacity and busy cloud servers in addition to the large transmission delays [2]. The latency issue in C-RAN becomes critical for IoT applications that cannot tolerate such delays. And this is why fog radio access network (F-RAN) is introduced for 5G, where fog nodes (FNs) are empowered with caching, signal processing and computing resources to independently deliver network functionalities to end users at the edge [3]. IoT applications have various latency requirements. Hence, especially in a heterogeneous IoT environment, FN must allocate its limited and valuable resources in a smart way. In this work, we present a novel framework for resource allocation in F-RAN to guarantee the efficient utilization of limited FN resources while satisfying the low-latency requirements of IoT applications [4].

Recently, a good number of works in the literature considered network slicing to achieve low latency for IoT applications in F-RAN. A comprehensive study of network slicing in 5G systems is considered in [5], [6]. Radio resource allocation for different network slices is exploited in [7]–[9] to support various quality-of-service (QoS) requirements and minimize the queuing delay for low latency requests, in which network is logically partitioned into a high-transmission-rate slice for mobile broadband (MBB) applications, and a low-latency slice which supports ultra-reliable low-latency communication (URLLC) applications. However, the network slicing literature deals with one-shot, i.e., static, resource allocation among various network slices and layers. In this work, we focus on the natural next step of static network slicing: dynamically optimizing the allocated limited resources to FNs to guarantee their efficient utilization. We compare the performance and adaptivity of the proposed dynamic network slicing method to the static network slicing approach from the recent literature. With the motivation of satisfying the low-latency requirements of heterogeneous IoT applications through F-RAN, we propose a novel framework based on finite-horizon Markov Decision Process (MDP). We also provide extensive simulation results in various IoT environments of heterogeneous latency requirements to evaluate the performance and adaptivity of the proposed dynamic network slicing method and compare it to the static network slicing approach with various slicing thresholds.

The remainder of the paper is organized as follows. Section II introduces the system model. The proposed MDP formulation for the considered resource allocation problem is given in Section III. Optimal policy and the related algorithm are discussed in Section IV. Simulation results are presented in Section V. Finally, we conclude the paper in Section VI.

II. System Model

We consider the F-RAN structure shown in Fig. 1, in which FNs are connected through the fronthaul to the cloud controller, where a massive computing capability, centralized baseband units (BBUs) and cloud storage pooling are available. To overcome the challenge of increasing number of IoT devices and low-latency applications, and to ease the burden on the fronthaul and the cloud, FNs are empowered to deliver network functionalities at the edge. Hence, they are equipped with caching capacity, computing and signal processing capabilities. These resources are limited, and therefore need to

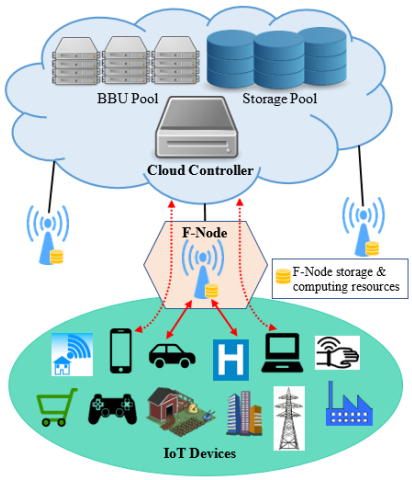

Fig. 1. Fog-RAN System Model. The FN serves heterogeneous latency needs in IoT environment, and is connected to the cloud through the fronthaul links. Solid red arrows represent local service by FN in the fog slice to satisfy low-latency requirements, and dashed arrows represent referral to the cloud slice to save FNs limited resources

be utilized efficiently. IoT applications have various levels of latency requirement; while some users are more delay-tolerant such as a smart phone used for video streaming, others can be classified as low-latency users, e.g., an autonomous car. So, it is sensible for the FN to give higher priority to the request from an autonomous car since it has higher utility to the fog network. Hence, we define utility to be equal to the priority level of a user, which is directly proportional to its level of latency requirement.

An end user attempts to access the network by sending a request to the nearest FN. The FN checks the priority level and then takes a decision whether to serve the user locally at the edge using its own computing and processing resources or refer it to the cloud. We consider the FN's computing and processing capacity is limited to N slots. We assume that the time required to fill the slots is much shorter than the average user serving time, and thus consider a single filling period of slots with no slot becoming available in the meantime. FNs should be smart to learn how to decide (serve/refer to the cloud) for each user (i.e., how to allocate its limited resources), in a way to achieve the conflicting objectives of maximizing the average total utility of served users over time and minimizing its idle (i.e., no-service) time.

One approach to deal with this resource allocation problem is to apply static network slicing [5], [6] based on the user utility, in which the network is logically partitioned into two slices [7]–[9], a fog slice handling high-utility IoT requests of low-latency demand, and cloud slice handling low-utility users. Hence a network slicing threshold on the user's priority level is required. For instance, if we consider ten different priorities $\{1, 2, 3, ..., 10\}$ of IoT applications where 10 represents the highest priority user and 1 is for the lowest priority, then we can define a slicing threshold rule "serve if priority greater than 5". However, such a slicing policy is sub-optimum since the FN will always be waiting for a user to satisfy the threshold, which will increase the idle time. The main drawback of this policy is that it cannot adapt to the dynamic IoT environment to achieve the objective. For instance, when the user priorities are almost uniformly distributed, a very selective policy with a high threshold will stay idle most of the time, whereas an impatient slicing policy with a low slicing threshold will in general obtain a low average served utility. A mild slicing policy with threshold 5 may in general perform better than the extreme policies, yet it will not be able adapt to a dynamic IoT environment in which the density of high priority users, i.e., low-latency users, changes over time. A better solution for the F-RAN resource allocation problem is to use MDP techniques which can continuously learn the environment and adapt the network slicing rule accordingly.

III. Finite-Horizon MDP Formulation

We formulate the F-RAN resource allocation problem in the form of finite-horizon Markov decision process (MDP). In finite-horizon MDP, there is a hard constraint on the FN in terms of the value of time as it must terminate within a limited time T_f regardless whether the N slots are filled or not. This means that the MDP will terminate either at the termination time T_f or before if all slots are filled earlier. The utility of a user, denoted with u, has the same value as user's priority level, e.g., $u \in \{1, 2, 3, ..., 10\}$. We consider that the FN has N slots of serving resources. The state of FN is denoted by $S \in \{S_{t,n}\}$, where t and n represent the time and the number of occupied slots, respectively. We start at $t = 0$ with all slots available, thus the initial state is $S_{0,0}$. The finite-horizon MDP has multiple terminal states. All the FN states $\{S_{T_f,n} : n = 0, 1, ..., N\}$ and $\{S_{t,N} : t = N, N+1, ..., T_f\}$ are terminal states. For instance, consider the example of an FN with $N = 2$ and a strict constraint to terminate within $T_f = 3$. The state transition graph for this finite-horizon MDP problem is shown in Fig. 2. There are four terminal states, $S_{3,0}$, $S_{3,1}$, $S_{3,2}$, and $S_{2,2}$. For a request from a user with utility u_t, at time t, if the FN decides to take the action $a_t = serve$, which means to serve the user at the edge, then it will gain its utility value as a reward $r_{t+1} = u_t$, and one slot of the FN's resources will be occupied. Otherwise, for the action $a_t = wait$, which means to refer the current user to the cloud and wait for a better future utility, the FN will maintain its resources but it will get a reward $r_{t+1} = -\eta$, where η is the penalty of waiting, whose role is to encourage less idle time. We define the state S of the FN at any time as the number of available slots at that time, where future state is independent of past states given the current state, i.e., Markov state. At every time step t, the FN receives a request from a user of utility u_t, and FN takes an action $a_t \in \{serve, wait\}$. Based on its decision, the FN receives an immediate reward $r_{t+1} \in \{u_t, -\eta\}$, and moves to a successor state S'.

We define the return G_t as the total discounted reward received from time t till the termination time which is limited by T_f, $G_t = \sum_{j=0}^{T-1} \gamma^j r_{t+j+1}$, where $\gamma \in [0, 1]$ is the discount factor which represents the weight or importance of future rewards with respect to the immediate reward, and

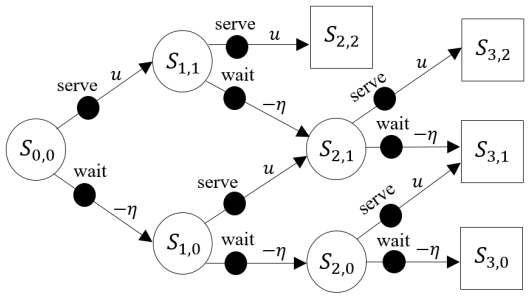

Fig. 2. State transition graph for FH MDP with $N = 2$ and $T = 3$, states are labeled by $S_{t,n}$ where t is the time and n is the number of filled slots.

after termination $r = 0$. $\gamma = 0$ ignores future rewards, whereas $\gamma = 1$ means that future rewards are of the same importance as the immediate rewards. For instance, if the FN terminates at $S_{3,2}$, then the return from initial state is given by $G_0 = r_1 + \gamma r_2 + \gamma^2 r_3$. The FN may have two different episodes from $S_{0,0}$ to $S_{3,2}$ depending on its actions. One episode is $S_{0,0} \rightarrow S_{1,1} \rightarrow S_{2,1} \rightarrow S_{3,2}$, with a corresponding return $G_0 = u_1 + \gamma(\gamma u_3 - \eta)$. And the other episode is $S_{0,0} \rightarrow S_{1,0} \rightarrow S_{2,1} \rightarrow S_{3,2}$ with a return $G_0 = \gamma(u_2 + \gamma u_3) - \eta$. Note that u_2 is referred to the cloud in the first episode, and u_1 is referred to the cloud in the second episode.

Starting at the initial state $S = S_{0,0}$, the objective is to find the optimal decision policy which maximizes the expected initial return $\mathbb{E}[G_0]$. The state-value function $V(S_{0,0})$, where $V(S)$ is shown in (1), is equal to the objective function $\mathbb{E}[G_0]$. $V(S)$ represents the long-term value of being in the state S in terms of the expected return which can be collected starting from this state onward till termination. Based on the definition a terminal state has a zero value since no reward can be collected from that state. The state value can be viewed also in two parts: the immediate reward from the action taken and the discounted value of the successor state where we end in. Similarly, we define the action-value function $Q(S, a)$ as the expected return that can be achieved after taking the action a at state S, as shown in (2). The action value function tells how good it is to take a particular action at a given state. The expressions in (1) and (2) are known as the Bellman expectation equations for state value and action value, respectively [10],

$$V(S) = \mathbb{E}[G_t|S] = \mathbb{E}[r_{t+1} + \gamma V(S')|S], \quad (1)$$

$$Q(S, a) = \mathbb{E}[G_t|S, a] = \mathbb{E}[r_{t+1} + \gamma Q(S', a')|S, a], \quad (2)$$

where a' denotes the successor action at the successor state S'. Note that, it is not straightforward to judge whether early or late termination necessarily maximizes the return.

The value of state $S_{0,0}$ is the expected return considering all dynamics and episodes, i.e., $V(S_{0,0} = \mathbb{E}[G_0]$. The objective of the FN is to utilize the N resource slots for high priority IoT applications in a timely manner. This can be done through maximizing the value of initial state $V(S_{0.0})$. To achieve this objective an optimal decision policy is required, which is discussed in the following section.

IV. Optimal Policy

A decision-making policy π is a way of selecting actions. It can be defined as the set of probabilities of taking a particular action given the state, i.e., $\pi = \{P(a|S)\}$ for all possible state-action pairs. The policy π is said to be optimal if it maximizes the value of all states, i.e., $\pi^* = \arg\max_{\pi} V_\pi(S), \forall S$. Hence, to solve the considered MDP problem, the FN needs to find the optimal policy through finding the optimal state-value function $V^*(S) = \max_{\pi} V_\pi(S)$, which is similar to finding the optimal action-value function $Q^*(S, a) = \max_{\pi} Q_\pi(S, a)$ for all state-action pairs. From (1) and (2), we can write the optimal state-value function as,

$$V^*(S) = \max_a Q^*(S, a) = \max_a \mathbb{E}[r_{t+1} + \gamma V^*(S')|S, a]. \quad (3)$$

The notion of optimal state-value function $V^*(S)$ greatly simplifies the search for optimal policy. Since the goal of maximizing the expected future rewards is already taken care of by the optimal value of the successor state, $V^*(S')$ can be taken out of the expectation in (3). Hence, the optimal policy is given by the best local action at each state,

$$a^* = \arg\max_a \mathbb{E}[r_{t+1}|S, a] + \gamma V^*(S'|S, a). \quad (4)$$

This solution approach is known as Dynamic Programming.

In our problem, first the user arrives, then we make a decision to serve or wait (refer to the cloud), meaning that the reward u for serving and the reward $-\eta$ for waiting are known at the time of decision making. Thus, from (4), the optimal action at state $S_{t,n}$ is given by

$$a^*_{t,n} = \begin{cases} Serve \text{ if } u > h_{t,n}, \\ Wait \text{ otherwise}, \end{cases} \quad (5)$$

where $h_{t,n} = \gamma[V^*(S_{t+1,n}) - V^*(S_{t+1,n+1})] - \eta$. Since the number of states is finite in the finite-horizon case, we can use the backward induction technique to compute the optimal thresholds $\{h_{t,n}\}$ assuming some training data $\{u_t\}$ is available to learn some key statistics of the IoT environment. Starting with the terminal states, which have zero value, we can compute the optimal state values and consequently the optimal thresholds for all states by moving backwards. Actually, we only need to compute the optimal thresholds for a subset of all states. Firstly, note that not all states $\{S_{t,n}\}$ are accessible for all t and n. Even if one slot is filled at each t, the states with $n > t$ are not accessible, as shown in Fig. 3.

Secondly, note that there are $T_f + 1$ terminal states with zero value ($T_f - N$ from early stopping with $T < T_f$ and $N + 1$ from $T = T_f$, $n = 0, 1, ..., N$), which do not require threshold, as shown with dark gray in Fig. 3. Next, note that for all the non-terminal states at time $T_f - 1$, both *serve* and *wait* actions result in a terminal state with zero value, thus the decision is made based on only the immediate rewards (u vs. $-\eta$). That is, at those states the optimal action is always *serve*, hence the threshold on u is zero and the state value is $\mathbb{E}[u]$, as shown in Fig. 3. Similarly, for $t = T_f - N, ..., T_f - 2$, there is a number of non-terminal states for which both actions yield the

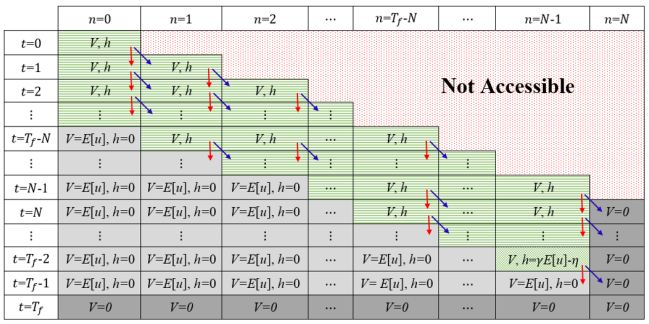

Fig. 3. FH state values and thresholds that need to be computed via backward induction (green diagonal band). Start with the farthest state $S_{T_f-2,N-1}$ (checked green box) and traverse backwards the diagonal band until the initial state $S_{0,0}$ (see Algorithm 1). The terminal states (dark gray), the trivial states whose optimal action is always serve (light gray), and the not accessible states (red dotted) are also shown.

same future value, hence have zero threshold and value $\mathbb{E}[u]$. Specifically, the states $\{S_{t,n} : t = T_f - l, n = 0, ..., N - l, l = 1, ..., N\}$ have state value $\mathbb{E}[u]$ and threshold 0, as shown with light gray in Fig. 3.

Finally, for the $(T_f - N)N$ remaining states in a diagonal band, shown with green in Fig. 3, the state values and the corresponding thresholds need to be computed backwards starting with the farthest state $S_{T_f-2,N-1}$ from the initial state $S_{0,0}$. The total reward is u if served, whereas it is $\gamma\mathbb{E}[u]-\eta$ if waited, giving the threshold $h_{T_f-2,N-1} = \gamma\mathbb{E}[u]-\eta$, as shown by the checked green box in Fig. 3. Then, its state value is written as

$$V(S_{T_f-2,N-1}) = P(u > h_{T_f-2,N-1})\mathbb{E}[u|u > h_{T_f-2,N-1}] + \{1 - P(u > h_{T_f-2,N-1})\}\{\gamma\mathbb{E}[u] - \eta\}, \quad (6)$$

where the first and second terms correspond to the serve and wait actions, respectively. Note that the probability $P(u > h_{T_f-2,N-1})$ and the expectation $\mathbb{E}[u|u > h_{T_f-2,N-1}]$ can be computed through some observations $\{u\}$ from the IoT environment. With $V(S_{T_f-2,N-1})$ computed, we can now find the threshold for the two undiscovered neighboring states above it, namely $S_{T_f-3,N-2}$ and $S_{T_f-3,N-1}$ using

$$h_{t,n} = -\eta + \gamma[V^*(S_{t+1,n}) - V^*(S_{t+1,n+1})], \quad (7)$$

from (5). Then, using the thresholds the state values are computed similarly to (6) as follows

$$V(S_{t,n}) = P(u > h_{t,n})\{\mathbb{E}[u|u > h_{t,n}] + \gamma V(S_{t+1,n+1})\} + \{1 - P(u > h_{t,n})\}\{\gamma V(S_{t+1,n}) - \eta\}. \quad (8)$$

In the same way, by computing first the threshold and then the state value via (7) and (8), respectively, the remaining states in the diagonal band are traversed backwards until the initial state $S_{0,0}$. The key statistics $P(u > h_{t,n})$ and $\mathbb{E}[u|u > h_{t,n}]$ are to be found from the IoT environment. The procedure for finding the optimal policy is summarized in Algorithm 1, where by default the terminal states have zero value and $x : -1 : y$ denotes the decrement by 1 from x to y. For notational simplicity, the trivial states are also included in the loops at lines 2 and 3. The range for the loops can be modified to exclude the trivial states.

Algorithm 1 Learning Optimum Policy for FH MDP

1: Select: $\gamma \in [0, 1]$, $\eta \in \mathbb{R}$;
2: **for** $i = N - 1 : -1 : 0$ **do**
3: **for** $j = T_f - 1 : -1 : i$ **do**
4: $h_{j,i} = \gamma[V(S_{j+1,i}) - V(S_{j+1,i+1})] - \eta$;
5: Compute $a \leftarrow \mathbb{E}[u|u > h_{j,i}]$ and $p \leftarrow P(u > h_{j,i})$;
6: $V(S_{j,i}) = p\{a + \gamma V(S_{j+1,i+1})\}$ $+ (1 - p)\{\gamma V(S_{j+1,i}) - \eta\}$;
7: **end for**
8: **end for**
9: Return $\{h_{j,i}\}$.

Recall that the FN objective is to maximize the expected total served utility and minimize the expected termination time. Hence, to compare the performance of dynamic network slicing provided in Algorithm 1 with the performance of a threshold-based static network slicing, which does not learn from the interactions with environment, we define an objective performance metric R as

$$R = \mathbb{E}\left[\sum_{m=1}^{M} u_m - \theta(T - M)\right], \quad (9)$$

where a served utility is denoted with u_m, the number of served IoT requests in an episode is denoted with M, $(T-M)$ represents the total idle time for RBs, and θ is a penalty for being idle.

V. Simulations

We next provide simulation results to compare the performance of the FN when implementing the MDP-based method, given in Algorithm 1, with the FN performance when a threshold-based slicing is employed. We consider that the FN is empowered with computing and storage resources of five slots, i.e., $N = 5$. We evaluate the performances in various IoT environments with different compositions of latency requirements. Specifically, we consider 10 utility classes with different latency requirements to exemplify the variety of IoT applications in an F-RAN setting. By changing the composition of utility classes, we generate 19 scenarios, 6 of which are summarized in Table I. Higher percentages of high-utility users make the IoT environment richer. Denoting an IoT environment of particular statistics with φ, in Table I we show the statistics of φ_1, φ_4, φ_7, φ_{10}, φ_{15}, and φ_{19}. The last two rows in Table I show the probability ρ of utility being greater than 5, and the expected value of u, respectively. The first 10 rows in the table provide detailed information given by the probability of each utility value in an IoT environment. In the considered 19 scenarios, ρ increases by 0.05 from 5% to 95% for $\varphi_1, \varphi_2, ..., \varphi_{19}$ respectively. The remaining 13 scenarios have statistics proportional to their ρ values. We started with a general scenario given by φ_7 for the following IoT applications: smart farming, smart retail,

TABLE I
UTILITY DISTRIBUTIONS CORRESPONDING TO A VARIETY OF LATENCY REQUIREMENTS OF IOT APPLICATIONS IN VARIOUS ENVIRONMENTS

	φ_1	φ_4	φ_7	φ_{10}	φ_{15}	φ_{19}
$P(u=1)$	0.015	0.012	0.01	0.008	0.004	0.001
$P(u=2)$	0.073	0.062	0.05	0.038	0.019	0.004
$P(u=3)$	0.365	0.308	0.25	0.192	0.096	0.019
$P(u=4)$	0.292	0.246	0.2	0.154	0.077	0.015
$P(u=5)$	0.205	0.172	0.14	0.108	0.054	0.011
$P(u=6)$	0.014	0.057	0.1	0.142	0.214	0.271
$P(u=7)$	0.013	0.051	0.09	0.129	0.193	0.244
$P(u=8)$	0.011	0.046	0.08	0.114	0.171	0.217
$P(u=9)$	0.009	0.034	0.06	0.086	0.129	0.163
$P(u=10)$	0.003	0.012	0.02	0.029	0.043	0.055
$\rho = P(u>5)$	5%	20%	35%	50%	75%	95%
$\mathbb{E}[u]$	3.82	4.4	4.97	5.55	6.5	7.27

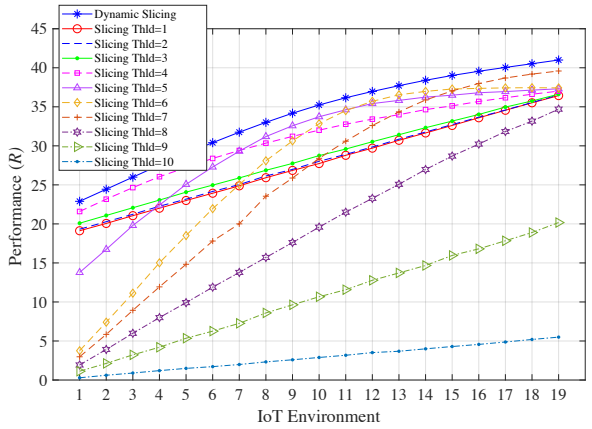

Fig. 4. Performance of FN with $N = 5$ and $T_f = 10$ in various IoT environments when applying Algorithm 1 with $\eta = 0$, $\gamma = 1$, and the threshold-based network slicing algorithm with different slicing thresholds.

smart home, wearables, entertainment, smart grid, smart city, industrial Internet, autonomous vehicles, and connected health, which correspond to the utility values $1, 2, ..., 10$, respectively. Then, we changed ρ to obtain the other scenarios.

We consider the objective performance metric given in (9) to compare the performance of the FN with $N = 5$ when applying the proposed dynamic programming algorithm (Algorithm 1) with the threshold-based slicing algorithm. We consider a finite horizon of $T_f = 10$, which serves as a strict termination time. Since we already have a time constraint on the FN which represents the value of time, we consider $\eta = \theta = 0$. As shown in Figs. 4 and 5, the dynamic network slicing algorithm exhibits the best performance as it adaptively learns how to balance early termination with higher utilities. It never terminates too early or too late ($T \approx 7.6$ for all environments as seen in Fig. 5), as opposed to the threshold-based slicing algorithm which is not adaptive to the environment. Dynamic slicing algorithm adaptively learns how to achieve the objective for all IoT environments under a strict termination time constraint.

VI. CONCLUSIONS

We proposed a finite-horizon Markov Decision Process (MDP) formulation for the resource allocation problem in Fog RAN for IoT services with heterogeneous latency requirements. We provided the optimum solution (decision policy) for the MDP problem using dynamic programming. Various IoT environments with different latency compositions were considered in the simulations to evaluate the performance of the proposed dynamic network slicing approach. The numerical results corroborated the fact that MDP methods adapt to the environment by learning the optimum policy from experience. We showed that the dynamic MDP-based slicing method always dominates the static threshold-based slicing method, which does not learn from the environment.

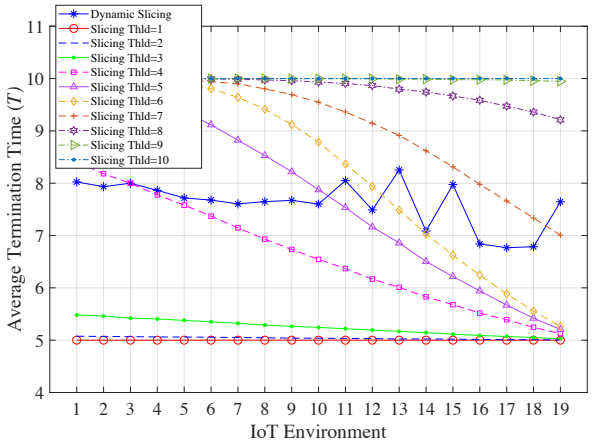

Fig. 5. The average termination time for FN with $N = 5$ and $T_f = 10$ in various IoT environments when applying Algorithm 1 with $\eta = 0, \gamma = 1$, and the threshold-based network slicing algorithm with different slicing thresholds.

REFERENCES

[1] S.-H. Park, O. Simeone, and S. Shamai, "Joint optimization of cloud and edge processing for fog radio access networks," in *Information Theory (ISIT), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 315–319.

[2] W. Wang, V. K. Lau, and M. Peng, "Delay-aware uplink fronthaul allocation in cloud radio access networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 7, pp. 4275–4287, 2017.

[3] Y.-Y. Shih, W.-H. Chung, A.-C. Pang, T.-C. Chiu, and H.-Y. Wei, "Enabling low-latency applications in fog-radio access networks," *IEEE network*, vol. 31, no. 1, pp. 52–58, 2017.

[4] G. P. Fettweis, "The tactile internet: Applications and challenges," *IEEE Vehicular Technology Magazine*, vol. 9, no. 1, pp. 64–70, 2014.

[5] H. Xiang, W. Zhou, M. Daneshmand, and M. Peng, "Network slicing in fog radio access networks: Issues and challenges," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 110–116, 2017.

[6] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2429–2453, 2018.

[7] T. Dang and M. Peng, "Delay-aware radio resource allocation optimization for network slicing in fog radio access networks," in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2018, pp. 1–6.

[8] L. Tang, X. Zhang, H. Xiang, Y. Sun, and M. Peng, "Joint resource allocation and caching placement for network slicing in fog radio access networks," in *2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 2017, pp. 1–6.

[9] Y. Sun, M. Peng, S. Mao, and S. Yan, "Hierarchical radio resource allocation for network slicing in fog radio access networks," *IEEE Transactions on Vehicular Technology*, 2019.

[10] R. Sutton, and A. BartoMack, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.