Jamming Attacks on NextG Radio Access Network Slicing with Reinforcement Learning

Yi Shi[1], Yalin E. Sagduyu[2], Tugba Erpek[2], and M. Cenk Gursoy[3]

[1]Commonwealth Cyber Initiative, Virginia Tech, Arlington, VA, USA
[2]National Security Institute, Virginia Tech, Arlington, VA, USA
[3]Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY, USA

***Abstract*—This paper studies how to launch an attack on reinforcement learning for network slicing in NextG radio access network (RAN). An adversarial machine learning approach is pursued to construct an over-the-air attack that manipulates the reinforcement learning algorithm and disrupts resource allocation of NextG RAN slicing. Resource blocks are allocated by the base station (gNodeB) to the requests of user equipments and reinforcement learning is applied to maximize the total reward of accepted requests over time. In the meantime, the jammer builds its surrogate model with its own reinforcement learning algorithm by observing the spectrum. This surrogate model is used to select which resource blocks to jam subject to an energy budget. The jammer's goal is to maximize the number of failed network slicing requests. For that purpose, the jammer jams the resource blocks and reduces the reinforcement learning algorithm's reward that is used as the input to update the reinforcement learning algorithm for network slicing. As result, the network slicing performance does not recover for a while even after the jammer stops jamming. The recovery time and the loss in the reward are evaluated for this attack. Results demonstrate the effectiveness of this attack compared to benchmark (random and myopic) jamming attacks, and indicate vulnerabilities of NextG RAN slicing to smart jammers.**

I. Introduction

A. Machine Learning for 5G Radio Access Network Slicing

As a key component of the *radio access network* (RAN), *network slicing* is considered to be instrumental in meeting the quality of experience (QoE) requirements in NextG communications systems. Network slicing divides communication resources into virtual resource blocks and allocates them to the requests of user equipments (UEs). These resource blocks can be allocated dynamically, e.g., via near-real time RAN Intelligent Controller (Near-RT RIC), to support different types of user applications such as Mobile Broadband (eMBB), massive machine-type communications (mMTC), and ultra-reliable low-latency communications [1]–[5].

Machine learning is considered as a key enabler for NextG communication systems to optimize decision making for complex tasks by learning from the rich representations of spectrum data. Applications of machine learning in NextG communications range from waveform design and spectrum situational awareness to security [6]. Machine learning also plays a major role in the design of network slicing solutions. Relevant examples of machine learning applications in network slicing include identifying applications and devices, classifying traffic [7] and managing load efficiency and availability of the network [8]. In NextG communication systems, training data may not be readily available to apply supervised learning solutions. Instead, *reinforcement learning* can be utilized as a model-free solution for network slicing in the RAN [9]–[17]. Reinforcement learning can effectively learn from the RAN performance and update its decisions to allocate resources to network slicing requests in the RAN.

This effort is supported by the U.S. Army Research Office under contract W911NF-17-C-0090. The content of the information does not necessarily reflect the position or the policy of the U.S. Government, and no official endorsement should be inferred.

In this paper, we study a *jamming attack* on network slicing. In this setting, the NextG base station, namely the gNodeB, is the victim system. Each UE requests network slices from the gNodeB. These requests arrive with the QOE requirements of user-centric priority (weight), throughput and latency (deadline). A reinforcement learning algorithm (e.g., the *Q-learning*) is run at the gNodeB to dynamically allocate resources to network slices for downlink communications from the gNodeB to the UEs.

B. Adversarial Machine Learning based Attacks on NextG Radio Access Network Slicing with reinforcement learning

A jammer can disrupt network slicing by jamming the resource blocks in the RAN. In this context, the reinforcement learning algorithm for network slicing is vulnerable to jamming attacks. The vulnerabilities of machine learning to various attacks in training and test times have been studied under adversarial machine learning. These attacks have been considered for wireless applications including spectrum sensing and signal classification [18], [19]. In particular, attacks on reinforcement learning algorithms have been studied in different wireless applications such as channel access in [20], [21], where a jammer can jam one channel over one time block only. It is also possible to attack network slicing in the RAN by generating fake requests in form of a flooding attack so that there may not be resources left to serve real network slicing requests [27].

This paper follows a separate attack approach and considers an attack on potentially multiple resource blocks that can be allocated to different users over a time horizon. The jamming attack on a resource block that is assigned to a network slicing request prevents the corresponding UE from achieving the required QoE. Therefore, the reward of this request drops to zero leading to performance loss. In addition, the reinforcement learning algorithm of the gNodeB takes this reward as the input (along with the state) to update itself. This reward feedback will make the reinforcement learning algorithm get confused and foot it into predicting the existence of jamming attacks even if there is no attack. As a result, a jamming attack can affect the current performance of network slicing as well its immediate future performance even after the jamming attack ends.

As time passes, network slicing starts recovering from the attack by updating the reinforcement learning algorithm with the correct feedback that arrives after the attack stops. We measure the performance of this attack by computing the *recovery time*, namely the duration of time from when jamming ends until the gNodeB's performance reaches the level before the attack starts. In addition, we measure the maximum and total loss in the reward during the recovery time.

We assume that the jammer cannot jam all resource blocks at the same time due to its *energy budget*. Under this constraint, the jammer needs to carefully select which resource blocks to jam. The jammer builds a surrogate model with its own reinforcement learning algorithm to learn how network slicing decisions are made. This approach is similar to building a deep neural network based surrogate model to jam data transmissions such as studied in [22], [23] or to launch adversarial attacks [24], [25] such as studied in [26]. However, the jammer in this paper builds a reinforcement learning algorithm as the surrogate model. The jammer uses this surrogate model to make jamming decisions in order to manipulate the victim's reinforcement learning algorithm for network slicing and maximize the number of failed network slicing requests.

This attack carefully exploits the following vulnerabilities: (i) the victim's reinforcement learning algorithm is affected by the manipulated rewards and (ii) it takes a while for the victim's reinforcement learning algorithm to recover even after the attack stops compared to conventional attacks [28], [29] that jam data transmissions and remain effective only during the jamming period.

We compare this attack with two benchmark jamming attacks, namely the *myopic jamming attack* that jams the resource blocks to maximize the jammer's instantaneous reward and the *random jamming* attack that jams randomly selected resource blocks). Our results demonstrate that the proposed attack is much more effective in reducing the reward of network slicing (under attack and after attack) and prolonging the recovery time from the attack.

C. Contributions and Paper Organization

The contributions of this paper are summarized as follows. For a reinforcement learning based NextG RAN slicing algorithm, we present a novel reinforcement learning based attack built upon adversarial machine learning that selectively jams the available resource blocks so that the victim's reinforcement learning algorithm receives incorrect reward (feedback) and updates itself in a wrong way, thereby leading to a significant performance loss of resource allocation for network slicing. We show that the attack is more successful than benchmark attack schemes in terms of increasing the recovery time and the loss in the reward for network slicing.

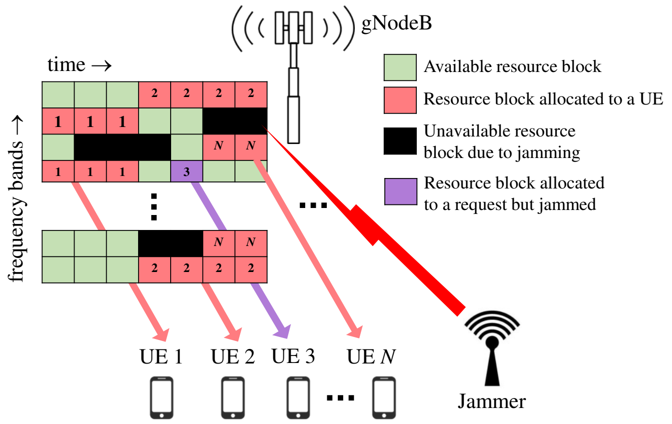
gNodeB
time →
frequency bands →
Available resource block
Resource block allocated to a UE
Unavailable resource block due to jamming
Resource block allocated to a request but jammed
UE 1
UE 2
UE 3
UE N
Jammer

Fig. 1: System model for NextG network slicing in the presence of a jammer.

The rest of the paper is organized as follows. Section II presents how the resources are allocated to network slices via a reinforcement learning algorithm. Section III introduces the reinforcement learning based jamming attack. Section IV provides the performance evaluation of the proposed attack. Section V concludes this paper.

II. The Victim System under Attack: Reinforcement Learning based Resource Allocation for NextG RAN Slicing

We start with describing the victim system that is attacked by the jammer, namely the reinforcement learning algorithm that is used for network slicing in the RAN. We consider the reinforcement learning formulation from [14] for allocation of resources to network slices. Note that the proposed attack can be also launched against other reinforcement learning based RAN slicing settings (e.g., [9], [30]). The system model is depicted in Fig. 1. We consider a general scenario in which multiple UEs send their network slicing requests over time with different QoE requirements including rate, latency (deadline), and lifetime demands, and weights. The gNodeB allocates the resource blocks to the selected requests. The objective of this resource allocation problem is to maximize the total weight of served requests over a time period. Any request that cannot be granted is kept in a waiting list until its deadline expires. The scenario also includes a jammer that is described in Section III.

Let $A(t)$ denote the set of active requests (that have just arrived or are waiting in the waiting lists) at time slot t. The rate requirement of UE i's request j is expressed as

$$D_{ij} \geq d_{ij}, \quad (i,j) \in A(t), \tag{1}$$

where D_{ij} and d_{ij} denote the achieved downlink data rate and the minimum required rate, respectively. The required bandwidth F_{ij} follows from this demanded rate [14].

We have the following constraints on resource assignments to network slices:

$$\sum_{(i,j) \in A(t)} F_{ij} \, x_{ij}(t) \leq F(t), \tag{2}$$

where $x_{ij}(t)$ is the binary indicator determining whether UE i's request j is satisfied at time t and $F(t)$ denotes the gNodeB's available communication resources (resource blocks) at time t (note that resources that have been assigned to requests and are not terminated yet become temporarily unavailable).

The update of resources from time $t-1$ to time t is given by

$$F(t) \quad = \quad F(t-1) + F_r(t-1) - F_a(t-1), \tag{3}$$

where $F_r(t-1)$ and $F_a(t-1)$ are the released and allocated resources at time $t-1$, respectively. If UE i's request j is satisfied and the service starts at time slot t, this request will end at the end of time slot $t + l_{ij} - 1$, where l_{ij} is the lifetime of UE i's request j. Then, the released and allocated resources at time t are given by

$$F_r(t) \quad = \sum_{(i,j) \in R(t)} F_{ij}, \tag{4}$$

$$F_a(t) \quad = \sum_{(i,j) \in A(t)} x_{ij}(t) F_{ij}, \tag{5}$$

where $R(t)$ is the set of requests ending (completed) at time t. Let w_{ij} denote the weight for UE i's request j to reflect its priority. The corresponding optimization problem can be formulated as

$$\max_{x_{ij}(t)} \sum_t \sum_{(i,j) \in A(t)} w_{ij} \, x_{ij}(t), \tag{6}$$

subject to (1)–(5).

We use Q-learning to learn the policy that decides on which action (resource assignment) to take by the gNodeB for a given state that includes available resources and active requests. The function $Q : S \times A \to \mathbb{R}$ is computed and maintained as a Q-table to evaluate the reward R for an action A taken at state S. At any time t, an action a_t is taken, the state transitions from s_t to s_{t+1} (depending on current state s_t and action a_t), a reward r_t is received, and Q is updated. Q function is initialized as a random matrix and updated by the weighted average of the old value and the new information as follows:

$$\begin{aligned} Q(s_t, a_t) \leftarrow & \; Q(s_t, a_t) \\ & + \alpha \cdot \left(r_t + \gamma \cdot \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right), \end{aligned} \tag{7}$$

where α is the learning rate within in $[0, 1]$ and γ is the discount factor in $[0, 1]$ for rewards over time. When the state size grows, it is better to approximate the Q-function by a deep neural network (leading to a deep Q-network formulation), which is computationally more efficient [15].

If UE i's request j is satisfied at time t, i.e., $x_{ij}(t) = 1$, the reward is w_{ij}. Otherwise, the reward is 0. This reward represents the satisfied QoE demands of network slices and thus indirectly helps quantify the QoE performance regarding throughput and delay. An action assigns resources to a network slicing request at a given time. Note that it is possible to take multiple actions at the same time. At any time t, the states are F binary variables that determine the availability of F resource blocks and (F_{ij}, w_{ij}) for a request under consideration. The state at time t transitions depending on the allocation of resources for requests granted at time t and the release of resources when lifetimes of active services expire at time t. The state transitions are given by (3)-(5).

III. Attack on Reinforcement Learning for NextG RAN Slicing

We consider a jammer that attacks the reinforcement learning algorithm used for NextG RAN slicing, e.g., the one discussed in Section II. Other example victim systems including the reinforcement learning based network slicing schemes in [9], [30] can be attacked similarly.

A. Reinforcement Learning based Attack

We consider *two unique properties* of reinforcement learning that we leverage to formulate and evaluate the attack.

1) The jammer can change the state or the reward by jamming the resource block, and consequently affect how the reinforcement learning algorithm operates.
2) After the jammer stops attacking, the reinforcement learning algorithm recovers over time.

We exploit the first property to design the attack on the reinforcement learning algorithm used for NextG RAN slicing. Note that its takes a while for the reinforcement learning algorithm to recover after the jamming attack stops. We measure the effect of the attack due to the second property in Section IV.

The jammer can affect the reward of the reinforcement learning algorithm if it jams a resource block to be allocated to a request. Then, the request fails even if the reinforcement learning algorithm allocates resources and there is no reward achieved.

We assume that the jammer has *limited jamming capability* (typically due to limited energy budget) and can jam at most B resource blocks at any given time. Therefore, the jammer needs to carefully select and jam the resource blocks that are likely to be allocated to network slicing requests.

In the ideal scenario, the jammer builds a *surrogate model* as another reinforcement learning algorithm to predict which resource blocks will be allocated. Then, the jammer can decide which resource blocks to jam. This scenario is not practical. First, the jammer does not have access to the state of the victim's reinforcement learning algorithm. Second, the jammer does not know the achieved reward which is given by the request's weight. Instead, the jammer needs to use a different reinforcement learning algorithm as an approximate surrogate

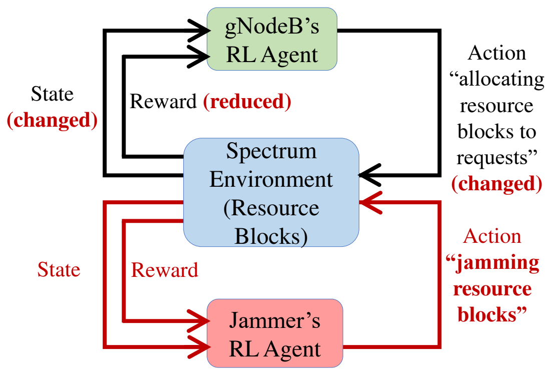
gNodeB's RL Agent
Action "allocating resource blocks to requests" (changed)
State (changed)
Reward (reduced)
Spectrum Environment (Resource Blocks)
State
Reward
Action "jamming resource blocks"
Jammer's RL Agent

Fig. 2: The interactions of the jammer and the victim system.

model. Different state, action, and reward properties of the jammer are given as follows.

- The *state* is the set of binary variables that determine whether resource blocks are available, or not. For that purpose, the jammer passively senses the resource blocks. Note that this corresponds to a *black-box attack*.
- An *action* selects which set of $\min\{B, F(t)\}$ resource blocks from $F(t)$ available resource blocks to jam, or decides not jam any resource block. Let $C^B_{F(t)}$ denote the number of B-combinations from a set of $F(t)$ elements, i.e., the number of possibilities in picking B out of $F(t)$). Then, the number of possible actions is $C^B_{F(t)} + 1$ if $F(t) > B$, or 2 (jam or not) if $F(t) \leq B$.
- The *reward* is determined by the number of requests that are jammed at a given time. If the transmission is not successful, the corresponding UE transmits a negative acknowledgment (NACK) at the end of a time slot. A retransmission is possible later subject to its deadline for reliable communications. The reward on a channel is set as 1 provided that the jammer jams a resource block and later observes the NACK. The jammer only needs to identify if a NACK is transmitted without any need to decode it. For that purpose, the jammer distinguishes the NACK from data transmissions by leveraging the properties that the NACK is shorter than data portion and appears between network slicing requests and data transmissions.

The jammer applies reinforcement learning to learn the effect of its attack and to update its Q-table by (7). The jammer selects which resource blocks to jam in order to maximize its expected jamming reward. This corresponds to an *over-the-air attack* that indirectly manipulates the reward of the reinforcement learning algorithm by jamming the resource blocks. Fig. 2 shows how the victim reinforcement learning algorithm of the gNodeB and the surrogate reinforcement learning algorithm of the jammer interact with each other.

B. Performance Metrics and Benchmark Attack Schemes

The performance loss due to the attack is computed by comparing it with the case of no attack. This loss is due to the failure of some requests due to jamming of resource blocks such that the reward does not include their weights. Another way of measuring the effect of the attack is to compute the recovery time of the reinforcement algorithm after the jamming attack ends. By jamming resource blocks, the jammer can change some rewards. Then, the reinforcement learning algorithm of the gNodeB updates itself with these changed rewards. Therefore, the attack also impacts how the reinforcement learning algorithm operates. Hence, even if the jamming attack ends, it takes a while for the performance of network slicing to return to the levels before the attack stops. The reason is that the gNodeB can only collect sufficient data over some time to update and correct its reinforcement learning algorithm and only after some time, its performance can reach to the level before the attack.

We measure the impact of the jammer by computing the following metrics.

- *Recovery time*: Recovery time is the time it takes (after the jamming attack ends) for the reward of the victim's reinforcement learning algorithm to reach back the level before the attack. If this recovery time is long, the jammer does not need to jam for a long time period. This way, the jammer can jam only for a short time period and resume jamming later before the recovery time. This helps the jammer operate in a stealth mode without continuous jamming and save energy.
- *Maximum performance loss*: Maximum performance loss is the maximum performance gap during the recovery time compared to the level before the attack. We measure the performance as the reward with running average. The maximum performance loss represents the maximum impact during the recovery time.
- *Total performance loss*: Total performance loss is the accumulated performance gap during the recovery time compared to the level before the attack. The total performance loss is a more robust metric than the above two, since it is not affected by a small performance loss (compared with recovery time) or a single extreme point (compared with the maximum performance loss).

We compare this attack with the case of no attack. In addition, we consider two benchmark attacks:

- *Random attack*: The jammer uniformly randomly selects some resource blocks from all resource blocks and jams them subject to its jamming budget.
- *Myopic attack*: The jammer selects which resource blocks to jam (subject to its jamming budget) in order to maximize its instantaneous reward. The myopic attack does not consider future rewards.

Note that the proposed attack takes some time to improve its attack actions as the jammer's reinforcement learning algorithm learns how to make jamming decisions over time. On the other hand, random and myopic attacks do not involve gradual improvement. For fair comparison, we measure the recovery time, maximum and total performance loss over the same period of time (including the warm-up time) for all these

TABLE I: Performance comparison of Q-learning and other attacks.

Attack scheme	Maximum number of jammed resource blocks	Recovery time	Maximum loss in reward	Total loss in reward
Q-learning	1	1038	1.447	736.216
	2	1191	1.801	911.604
	3	1548	1.957	1006.174
	4	2086	2.014	1038.988
	5	2038	2.714	1410.069
Myopic	1	1035	1.343	670.071
	2	1060	1.587	788.289
	3	1028	1.684	836.998
	4	1207	1.775	894.721
	5	1365	1.772	889.113
Random	1	1197	1.000	506.947
	2	1233	1.489	750.976
	3	1170	1.813	907.546
	4	1180	2.061	1032.088
	5	1202	2.273	1141.359

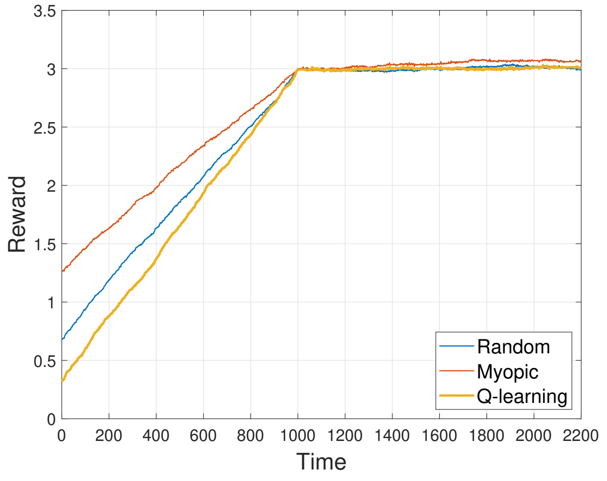

Fig. 3: The reward of reinforcement learning algorithm for NextG RAN slicing after the attack stops.

attacks. The evaluation of the attack performance is reported in Section IV.

IV. Performance Evaluation

We simulated a scenario that consists of one gNodeB and 30 UEs. The gNodeB may receive requests from all the UEs. Requests arrive with a rate of 0.05 per slot for each UE. Note that each time block (0.23 ms long with 60 kHz subcarrier spacing) is considered a slot. The weight is chosen uniformly randomly in $[1, 5]$ for each request. Similarly, the lifetime is assigned randomly in $[1, 10]$ slots, and the deadline is assigned randomly in $[1, 20]$ slots. The maximum received SNR is selected randomly from $[1.5, 3]$. We split the total bandwidth of 10 MHz to 11 bands corresponding to 11 resource blocks. We repeat this scenario over 1000 time slots to evaluate the impact of the attacks. We set the discount factor as $\gamma = 0.95$ and the learning rate as $\alpha = 0.1$ for Q-learning.

The benchmark of no attack case is run over 10000 slots. The jammer also launches its attack over 10000 slots. The

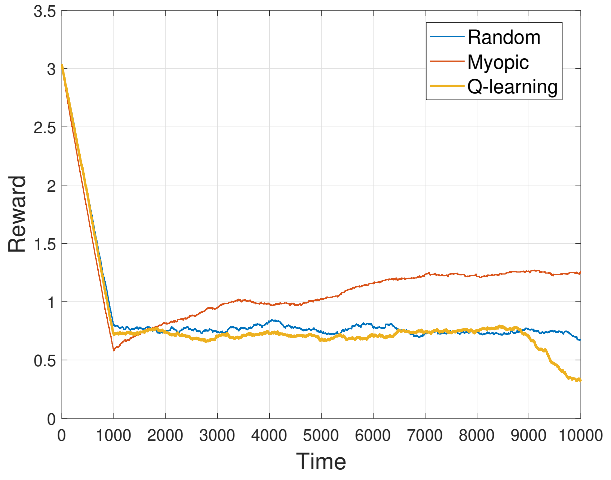

Fig. 4: The reward of the reinforcement learning algorithm for NextG RAN slicing under the attack.

reward is 3.032 over the first 1000 slots. This result is used as the benchmark for recovery. Then, we measure the average reward over the past 1000 slots after the jamming attack ends and once this average reward reaches 3.032, namely when the victim system recovers from the attack and the network slicing reaches the level of performance before the attack. In addition, we compute the (maximum and total) performance gap to the benchmark during the recovery time.

We also consider random and myopic jamming attacks for comparison purposes. The results for all these attacks are shown in Table I. Note that the results for random jamming are averaged over 20 runs. The reinforcement learning based attack that was introduced in Section III-A has longer and larger impact on the network slicing performance than the other two attacks, indicating that the proposed attack is more effective. In particular, the recovery time is increased by up to 77%, the maximum loss in the reward is increased by up to 53%, and the total loss in the reward is increased by up to 59% compared to benchmark attacks depending on the maximum number of jammed resource blocks with the reinforcement learning based attack. Fig. 3 shows the change of the reward over time after the attack ends when the maximum number of jammed resource blocks is 5.

The advantage of the reinforcement learning based attack comes from the smallest reward when the attack ends. Thus, we also check the reward under different attacks. We evaluated the performance of the victim's reinforcement learning under different attacks. The results are shown in Fig. 4. The average reward is shown over the past 1000 slots, so we observe that the performance is high at the beginning and rapidly decreases over time. Under both random jamming or reinforcement learning based jamming attacks, the performance still remains low. On the other hand, the performance under myopic jamming attack keeps increasing. The deterministic nature of the myopic algorithm causes this. The behavior of the myopic jamming attack can easily be learned by the victim reinforcement learning algorithm used for RAN slicing and its effect can be mitigated.

V. Conclusion

We introduced a novel over-the-air attack on network slicing for NextG RAN by jamming the resource blocks and manipulating the reinforcement learning algorithm used for resource allocation. We exploited the property that jamming a resource block that is assigned to a request not only reduces the associated reward to zero but also manipulates the update of the reinforcement learning algorithm that takes this reward as the input. Therefore, the effect of the jamming attack continues and it takes a while for the reinforcement learning algorithm for network slicing to update itself and recover from the performance loss even after the jamming attack ends. Subject to an energy budget, the goal of the jammer is set as selecting which resource blocks to jam in order to maximize the number of failed network slicing requests over time. By monitoring the spectrum, the jammer builds a surrogate reinforcement learning model to learn how the reinforcement learning algorithm for network slicing operates and jams the resource blocks to reduce its performance. Our results demonstrated that this jamming attack is much more effective in reducing the network slicing performance (namely, increasing the recovery time and reducing the achieved reward) compared to myopic and random jamming attacks, and poses a major threat to network slicing in the NextG RAN.

References

[1] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94-100, May 2017.

[2] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira, "Network slicing for 5G with SDN/NFV: Concepts, architectures, and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 80–87, May 2017.

[3] P. Rost, C. Mannweiler, D. S. Michalopoulos, C.Sartori, V. Sciancalepore, N. Sastry, O. Holland, S. Tayade, B. Han, D. Bega, D. Aziz, and H. Bakker, "Network slicing to enable scalability and flexibility in 5G mobile networks," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 72–79, May 2017.

[4] A. Kaloxylos, "A survey and an analysis of network slicing in 5G networks," *IEEE Communications Standards Magazine*, vol. 2, no. 1, Apr. 2018.

[5] S. D'Oro, F. Restuccia, A. Talamonti, and T. Melodia, "The slice is served: Enforcing radio access network slicing in virtualized 5G systems," *IEEE INFOCOM*, 2019.

[6] T. Erpek, T. O'Shea, Y. E. Sagduyu, Y. Shi, and T. C. Clancy, "Deep learning for wireless communications," *Development and Analysis of Deep Learning Architectures, Springer*, 2019.

[7] A. Nakao, and P. Du, "Toward in-network deep machine learning for identifying mobile applications and enabling application specific network slicing," *IEICE Transactions on Communications*, 2018.

[8] A. Thantharate, R. Paropkari, V. Walunj, C. Beard, "DeepSlice: A deep learning approach towards an efficient and reliable network slicing in 5G networks," *IEEE 10th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, 2019.

[9] R. Li, Z. Zhao, Q. Sun, C.-L. I, C. Yang, X. Chen, M. Zhao, and H. Zhang, "Deep reinforcement learning for resource management in network slicing," *arXiv preprint* arXiv:1805.06591, 2020.

[10] J. Koo, M. R. Rahman, V. B. Mendiratta, and A. Walid, "Deep reinforcement learning for network slicing with heterogeneous resource requirements and time varying traffic dynamics," *arXiv preprint* arXiv:1908.03242, 2019.

[11] H. Wang, Y. Wu, G. Mina, J. Xu, and P. Tang, "Data-driven dynamic resource scheduling for network slicing: A deep reinforcement learning approach," *Information Sciences*, 2019.

[12] Q. Liu and T. Han, "When network slicing meets deep reinforcement learning," *International Conference on Emerging Networking Experiments and Technologies*, 2019.

[13] Z. Xu, Y. Wang, J. Tang, J. Wang, and M. C. Gursoy, "A deep reinforcement learning based framework for power-efficient resource allocation in cloud RANs," *IEEE International Conference on Communications (ICC)*, 2017.

[14] Y. Shi, Y. E. Sagduyu, and T. Erpek, "Reinforcement learning for dynamic resource optimization in 5G radio access network slicing," *IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2020.

[15] Y. Shi, P. Rahimzadeh, M. Costa, T. Erpek, and Y. E. Sagduyu, "Deep reinforcement learning for 5G radio access network slicing with spectrum coexistence," *TechRxiv. Preprint.* https://doi.org/10.36227/techrxiv.16632526.v1

[16] A. Nassar and Y. Yilmaz, "Deep reinforcement learning for adaptive network slicing in 5G for intelligent vehicular systems and smart cities," *arXiv preprint* arXiv:2010.09916, 2020.

[17] K. Suh, S. Kim, Y. Ahn, S. Kim, H. Ju and B. Shim, "Deep reinforcement learning-based network slicing for beyond 5G," *IEEE Access*, vol. 10, pp. 7384–7395, 2022.

[18] Y. E. Sagduyu, Y. Shi, T. Erpek, W. Headley, B. Flowers, G. Stantchev, and Z. Lu, "When wireless security meets machine learning: Motivation, challenges, and research directions," *arXiv preprint* arXiv:2001.08883, 2020.

[19] D. Adesina D, C. C. Hsieh, Y. E. Sagduyu, and L. Qian, "Adversarial machine Learning in wireless communications using RF data: A review," *arXiv preprint* arXiv:2012.14392, 2020.

[20] C. Zhong, F. Wang, M. C. Gursoy, and S. Velipasalar, "Adversarial jamming attacks on deep reinforcement learning based dynamic multichannel access," *IEEE Wireless Communications and Networking Conference (WCNC)*, 2020.

[21] F. Wang, C. Zhong, M. C. Gursoy, and S. Velipasalar, "Defense strategies against adversarial jamming attacks via deep reinforcement learning," *IEEE Conference on Information Sciences and Systems (CISS)*, 2020.

[22] Y. Shi, Y. E Sagduyu, T. Erpek, K. Davaslioglu, Z. Lu, and J. Li, "Adversarial deep learning for cognitive radio security: Jamming attack and defense strategies," *IEEE International Conference on Communications (ICC) Workshop on Promises and Challenges of Machine Learning in Communication Networks*, 2018.

[23] T. Erpek, Y. E. Sagduyu, and Y. Shi, "Deep learning for launching and mitigating wireless jamming attacks, *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 1, pp. 2–14, Mar. 2019.

[24] B. Kim, Y. E. Sagduyu, K. Davaslioglu, T. Erpek, and S. Ulukus, "Over-the-air adversarial attacks on deep learning based modulation classifier over wireless channels," *Conference on Information Sciences and Systems (CISS)*, 2020.

[25] B. Kim, Y. E. Sagduyu, K. Davaslioglu, T. Erpek and S. Ulukus, "Channel-aware adversarial attacks against seep learning-based wireless signal classifiers," *IEEE Transactions on Wireless Communications*, vol. 21, no. 6, pp. 3868–3880, June 2022.

[26] B. Kim, Y. E. Sagduyu, T. Erpek, K. Davaslioglu, and S. Ulukus, "Channel effects on surrogate models of adversarial attacks against wireless signal classifiers," *IEEE International Conference on Communications (ICC)*, 2021.

[27] Y. Shi and Y. E. Sagduyu, "Adversarial machine learning for flooding attacks on 5G radio access network slicing," *IEEE International Conference on Communications (ICC) Workshops*, 2021.

[28] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The feasibility of launching and detecting jamming attacks in wireless networks," *ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2005.

[29] Y. E. Sagduyu, R. Berry, and A. Ephremides, "Jamming games in wireless networks with incomplete information," *IEEE Communications Magazine*, 2011.

[30] J. Koo, V. B. Mendiratta, M. R. Rahman, and A. Walid, "Deep reinforcement learning for network slicing with heterogeneous resource requirements and time varying traffic dynamics," *IEEE International Conference on Network and Service Management (CNSM)*, 2019.