A Deep Reinforcement Learning-Based Dynamic Traffic Offloading in Space-Air-Ground Integrated Networks (SAGIN)

Fengxiao Tang, *Member, IEEE*, Hans Hofner, *Student Member, IEEE*, Nei Kato, *Fellow, IEEE*, Kazuma Kaneko, Yasutaka Yamashita, and Masatake Hangai, *Senior Member, IEEE*

***Abstract*— Space-Air-Ground Integrated Networks (SAGIN) is considered as the key structure of the next generation network. The space satellites and air nodes are the potential candidates to assist and offload the terrain transmissions. However, due to the high mobility of space and air nodes as well as the high dynamic of network traffic, the conventional traffic offloading strategy is not applicable for the high dynamic SAGIN. In this paper, we propose a reinforcement learning based traffic offloading for SAGIN by considering the high mobility of nodes as well as frequent changing network traffic and link state. In the proposal, a double Q-learning algorithm with improved delay-sensitive replay memory algorithm (DSRPM) is proposed to train the node to decide offloading strategy based on the local and neighboring historical information. Furthermore, a joint information collection with hello package and offline training mechanism is proposed to assist the proposed offloading algorithm. The simulation shows that the proposal outperforms conventional offloading algorithms in terms of signaling overhead, dynamic adaptivity, packet drop rate and transmission delay.**

***Index Terms*— Space-air-ground integrated networks (SAGIN), satellite communication, UAV, traffic offloading, reinforcement learning (RL), double Q-learning.**

I. Introduction

DESPITE the recent advances in terrestrial communication technology, the lack of resources and the inability to serve increasing traffic demands still remains a prominent challenge researchers face to this day. This is because the rapid increase in user-equipment outpaces the development done in wireless communication networks. The new concept of a Space-Air-Ground Integrated Networks (SAGIN) [1] has thus been presented as a possible solution and has since then taken the forefront in future advanced network research. A SAGIN is a network in which all major networks are fully integrated to complement the terrestrial connection. Such a network provides multiple advantages thanks to the large resource availability pool and large area coverage as well as the added flexibility regarding resource management, such as moving devices which allow network managers to instantly increase resources to meet a forecasted high traffic demand. Despite the enhancements that SAGIN would provide, there are still prominent challenges that the utilization of SAGIN would yield. In particular, it is not obvious how the management of traffic flow via offloading is to be conducted in such a complex network with heterogeneous structure. It is especially crucial that a robust and smart traffic offloading scheme is utilized to adjust the complex environment. Considering that there is an abundant amount of research regarding traffic routing protocols that work efficiently for certain devices, it is of even more importance that the proposal of an offloading solution that allows the use of these diverse routing protocols is made. Unfortunately applying conventional traffic offloading methods to solve this challenge is inadequate because they were designed for static, homogeneous networks that represent the exact opposite of the network topology of a SAGIN because they neither account for future hop bandwidth and link states, or the live topological information of the network.

While SAGIN related research is relatively young, there are still a few proposals that attempt to tackle related issues to traffic offloading in SAGIN-like environments [2]–[4]. F. Mendoza *et al.* attempts to utilize the satellite sector to distribute traffic via an offloading-like algorithm that depends on a calculated bit rate allocation for every link in the network [5]. While effective in a space-ground type of environment, their method would fail at more dynamic and complex networks, such as SAGIN. On a similar basis, N. Chang *et al.* attempts to apply a smarter computation offloading technique with machine learning in SAGIN [6]. This method while effective in offloading computational tasks would struggle in larger state and action-space networks such as traffic distribution in SAGIN. For example, in [7], B. Fan et al propose an offloading system for Vehicle-to-everything (V2X) that utilizes a Software Defined Network (SDN) centralized approach and distributed approach to reduce control complexity. This method works well for devices with 2-D movement spaces but begin to falter as devices increase and their movements are upgraded to the 3-D space.

The utilization of smart, machine learning algorithms in wireless communication networks has proven itself an efficient solution tool to improve and speed up complex networks [8], making upcoming systems truly "next-generation". However,

Manuscript received June 13, 2021; revised September 12, 2021; accepted October 18, 2021. Date of publication November 8, 2021; date of current version December 17, 2021. *(Corresponding author: Fengxiao Tang.)*

Fengxiao Tang, Hans Hofner, and Nei Kato are with the Graduate School of Information Sciences (GSIS), Tohoku University, Sendai 980-8579, Japan (e-mail: fengxiao.tang@it.is.tohoku.ac.jp; hans.hofner@it.is.tohoku.ac.jp; kato@it.is.tohoku.ac.jp).

Kazuma Kaneko, Yasutaka Yamashita, and Masatake Hangai are with the Information Technology Research and Development Center, Mitsubishi Electric Corporation, Kamakura 247-8520, Japan (e-mail: kaneko.kazuma@dfmitsubishielectric.co.jp; yamashita.yasutaka@abmitsubishielectric.co.jp; hangai.masatake@cw.mitsubishielectric.co.jp).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2021.3126073.

Digital Object Identifier 10.1109/JSAC.2021.3126073

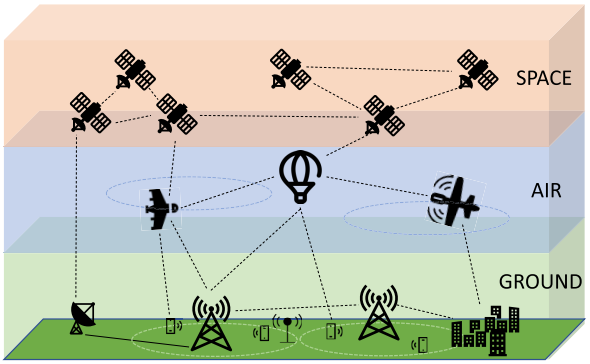


Fig. 1. The space-air-ground integrated network.

it's important that the correct tools are utilized and aggregated in a way that makes sense. In this research, we contribute a novel traffic offloading system that utilizes a novel delay-sensitive algorithm together with a well-established neighbor information discovery protocol that efficiently allows nodes in a network to captures salient neighborhood information in real time, which is fed into a smart and fast machine learning model that can determine the correct action within a large action space. To further adjust to the high dynamic network, a delay sensitive based replay memory algorithm called as DSRPM is proposed to improve the training efficiency of the learning system. Unlike other traffic offloading proposals that only consider very niche environments or simplistic network states, our research employs a scalable and flexible solution that works for complex and dynamic networks.

The remainder of this paper will continue as follows: In this Section, the research background as well as the objective of the research is presented. Section II highlights important related literature and challenges that researchers are faced when attempting to apply conventional or other applied methods to traffic offloading in SAGIN. In Section III we then propose our distributed reinforcement learning based traffic offloading system to flexibly make traffic offloading decisions. We present the usage of a complex reinforcement learning algorithm named Double Deep Q-Learning and the reason for our chosen model structure. Section IV presents performance evaluation of our proposal via simulations and their results.

II. Related Works

In this section, we give an introduction to the background and environment of the research by presenting important and related research literature pertaining to dynamic offloading in SAGIN, as well as challenges that are faced when attempting to find an optimal solution. Additionally, we discuss the motivation of this research and then continue by highlighting the physical environment in which our solution is targeted towards to and why this is important.

A. Offloading Solutions to Satellite and UAV

The work in [9] proposes Satellite can be used to offload multimedia traffic from 5G ground communication. In this work, a multicast subgrouping-maximum satisfaction index metric is proposed to measure the priority of candidate paths. Meanwhile, the authors in [5] mainly focus on the measurement of link state of both satellite and terrestrial links and maximize the network utility function by offloading the traffic to satellite when the terrestrial links is week. Du et. al proposes a auction game based traffic offloading for satellite terrestrial network [10].

Besides satellite, the work of [11] proposes a traffic offloading algorithm of using UAV in the air to offload the traffic from cellular network. The work try to ensure the offloading fairness of users by optimizing the bandwidth allocation and location of UAVs. Besides, the work in [12] considered using UAV to offload traffic from device-to-device (D2D) network with game theory based channel allocation algorithm. To jointly optimize the QoS and energy consumption in UAV assistant network, the work of [13] transfers the joint traffic offloading problem to a mix-integer nonconvex optimization problem and proposes a successive convex approximation based algorithm to solve it. The simulation result shows that the proposal efficiently improves the overall energy consumption and network performance. However, the above researches all considered the only element of satellite or UAVs in the SAGIN. Besides, the existing works only consider direct connection or connection in single cell where the chain reaction of long-hops transmission is not fully considered. The existing fixed rule based offloading algorithms are hard to adjust to the complex environment of SAGIN.

B. Reinforcement Learning in Wireless Network

Benefiting from the development of the powerful intelligent tool, the machine learning especially the reinforcement learning algorithms are widely proposed to be the potential solution to handle the dynamic scheduling problem in complex environment [14].

For example, the deep reinforcement is widely used in channel allocation in wireless network. The researchers in [15] proposes a deep Q-learning based channel allocation algorithm to extract the network feature and predict the future network state without knowing the system statistics. The authors in [16] proposes a deep reinforcement learning based multi-access control and dynamic energy harvesting algorithm. And the work in [17] uses deep Q-learning for dynamic channel allocation for unlicensed spectrum in Long Term Evolution (LTE). The Long short-term memory (LSTM) is used in the above two works for improve the prediction ability of time series data. The experimental results show that the above proposals achieves much better performance in complex situations with dynamic traffic and channel changing compared with fix-rule based algorithms.

Besides channel allocation, the deep reinforcement learning is also exploited for network traffic control. The work in in [18] employs deep reinforcement learning to explore the potential next-hop selection in dynamic wireless network. Instead of central control, the authors in [19] propose a distributed multi-agent reinforcement learning based routing strategy in decentralized networks. However, all those works never consider the high dynamic scenario with changing network topology. Besides, due to the high delay consed by the long communication distance and changing topology, the training data collection scheme needs to be carefully considered in SAGIN.

C. Reinforcement Learning for Offloading Solutions in SAGIN

While SAGIN is a relatively new research field with a few papers, researchers have attempted to tackle similar challenges following the objectives of this research. However, due to the complexity of SAGIN, the state of network changes frequently and affected by factors such as UAV data transmission rate, link state, network topology, other users' decision, and traffic load which are dynamic and correlated, the transition probability of state is totally unknown and hard to model. Therefore, the model-free approaches with reinforcement learning are widely explored to address the offloading problem in SAGIN.

Cheng et. al in [20] attempt to improve computation offloading for Internet of Thing (IoT) devices in a SAGIN environment by employing a Markov decision process model that is fed information from an online reinforcement learning algorithm which in turn utilizes only a single neural network. While effective for computational tasks, it falls short for traffic distribution and offloading as resource requests for traffic are more heterogeneous and can create an even larger action space, which paired with a single neural network would introduce large biases. In the same vein, Zhou et. al [6] also attempt to solve computational offloading in a SAGIN environment by utilizing a heuristic multi-stochastic conditional constrained Markov decision model. The proposed solution is to be applied in a niche environment were relay node count is low, and does not promise good scalability in larger node count environments. Moreover, the large dynamic property of a SAGIN environment is not addressed, and thus this is something that can not be directly applied to traffic offloading. In a more similar research, Wang et. al propose a traffic offloading technique that utilizes a deep reinforcement learning algorithm as well as a state-aware two-hop protocol to find the optimal traffic offloading route in a Low Earth orbit (LEO) satellite network [21]. While the system provides a flexible and scalable state-aware system, it falls short in heterogeneous environments such as SAGIN because the research is catered to a high-bandwidth network such as LEO Satellites. While a SAGIN might contain LEO-Satellite networks and clusters, a large part of the SAGIN network will also account for other networks containing devices with much smaller bandwidth capabilities [1].

D. Challenges of Link State for Offloading in SAGIN

In this part, we discuss a challenge researchers are faced when attempting to improve the performance of networks similar to SAGIN through offloading. In particular, the heterogeneity of SAGIN-like networks is that one can not always assume the same bandwidth link 2 or 3 hops out when traffic offloading. This is apparent when attempting to apply current conventional traffic offloading in a SAGIN. Traffic offloading was a solution developed for previous generation networks as a solution to reduce the load of a network onto a higher bandwidth network and as a result, the conventional traffic offloading method is simple in nature due to the fact that there has been not much drive for innovation. Since conventional traffic offloading assumes the device it is offloading to has

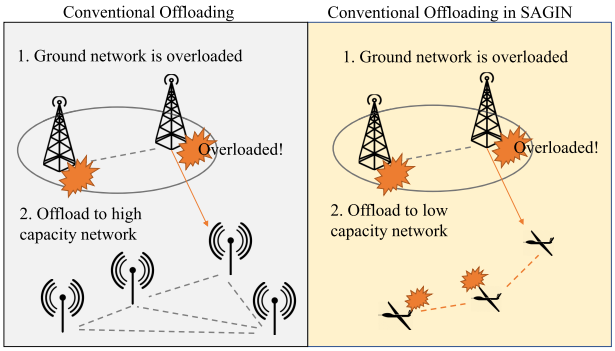


Fig. 2. Offloading scenarios.

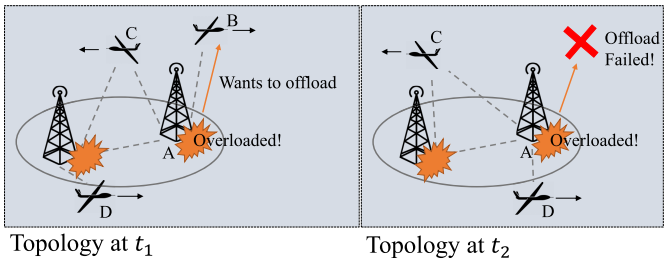


Fig. 3. Consequence of dynamic topology.

connections of similar bandwidth, large data may be offloaded to a device which has only limited bandwidth in their second, or third hop connections.

This is apparent in Fig. 2, as conventional offloading always assumes future hops to have the same bandwidth as the first node it is offloading to. This is because in past systems, there was no consideration of a SAGIN-like networks were devices with different bandwidth and processing capabilities are fully intertwined and connected. Moreover, conventional methods take no care of understanding the state of the surrounding network because of the previous mentioned fact. Therefore it is apparent that we need a smarter, more state-aware system for future generation networks if we want to utilize the full extent of the future networks such as SAGIN.

E. Challenges of Dynamic Environments for Offloading in SAGIN

Networks such as SAGIN encompass a large array of resources and hardware, containing the largest dynamics of all proposed researched networks to this day. It is not immediately obvious how that can impact performance with the solutions presented previously. A challenge that continues to exist is how to deal with the complex nature of dynamic devices, such as UAVs in a wireless communication network. A particular problem is how to minimize the error in offloading procedures when neighboring devices are dynamic. Related to understanding the network state, it is important that devices in a network are aware of the surrounding neighbors with dynamic properties. A consequence of attempting to utilize a basic communication protocol or conventional traffic offloading method in a highly dynamic network can be seen in Fig. 3 in which a device attempts to process an offloading procedure to a device that will no longer be in range to collect the data. There have been a few attempts to manage

this dynamic problem such as [22] and [23]. Utilizing large computational or any other method provides a large overhead just to manage the positional data of every device among a network. Moreover, this system scales poorly as more devices with faster and more dynamic movement are introduced, the overhead begins to grow exponentially. However, research into heterogeneous networks has brought us scalable network discovery protocols to understand the network efficiently such as the OLSR protocol [24] and can be considered to be used as a basis for solving these challenges by exploring local and neighboring information. In the next section, we describe the network system and communication constraints caused by the high dynamic environment of SAGIN. The problem is formulated with objective function and constraints.

III. System Model and Problem Statement

A. Network Model

As a SAGIN is considered to connect all major communication networks into one, it is important to capture the generality of the network. In this section we present the network formulation and the problem statement. We choose to model the SAGIN as a weighted, bi-directional graph $G = (N, E)$ with vertices $N = \{\boldsymbol{n_0}, \boldsymbol{n_1}, \ldots, \boldsymbol{n_m}\}$ representing the devices in the network. Within all nodes, there are some source nodes $NS = \{ns_0, ns_1, \ldots, ns_s\}$ generate network traffic and send to other nodes. The weighted edges $E = \{e_{01}, e_{10}, e_{02}, \ldots, e_{mm}\}$ representing a link between a device in the network with non-present links being simply represented as a nullified weight. Each device $\boldsymbol{n_i} = \{a_{i0}, a_{i1}, \ldots a_{iu}\}$ contains a set of both static and dynamic parameters that represent the device type such as the generalized mobility model, traffic generation rate, and queue capacity extent. For every device at time t, a data flow of a set of packets with a normally distributed size $\mu \in \mathcal{N}(f, \sigma^2)$ and defined routing path $rp = \{n_s \rightarrow n_f\}$ is being sent to node $n_f \in N$, which can be further represented as $P_{n_s \rightarrow n_f} = \{\mu_{rp}, rp, d\}$ with μ_{rp}, d as the data size and the average end transmission-delay of the packets respectively.

B. Mobility Model

There are four types of nodes in SAGIN, except for the ground base station (BS.) is fixed, the ground devices, UAVs, satellites (Sat.) all have their mobility model. In space, we only considered the LEOs that move around the earth in time period t_s with speed v_s. When the communication distance between satellite and other nodes is bigger than a threshold, the link breaks. In the air, the UAV is feasible to connect to ground users, UAVs, and satellite. The UAVs with high mobility change the distance to ground users and satellites frequently, causing frequent link breaks and relink. In the considered network, we consider a mobility model [25], in which UAVs move with a fixed speed v_u in one direction in a predefined period. After a while, the moving direction of the UAV changes randomly within certain angle sin_u with the same moving speed. In the ground, the devices use a random walk model to move with speed v_d [26], and the ground base stations are with fixed locations.

C. Communication Model

The distance between nodes in SAGIN is highly dynamic and significantly affect the packets transmission rate. In this part, we model the communication model of link transmission rate with dynamic connection distance. Here we define the transmission rate between UAV and ground users in time slot t based on the communication model in [27] as:

$$R_{uav} = Wlog_2(1 + \frac{P_t \cdot 10^{\frac{P_{loss}}{10}}}{\sigma^2}). \tag{1}$$

where W is the up/down-link channel bandwidth between transmitter and receiver. p_t denotes the transmit power and the σ^2 denotes the noise power, and the p_{loss} denotes the average path loss between UAV and BS.

Based on the formulation in [28], the path loss is calculated as:

$$P_{loss} = 10\alpha log(d) + C + X_\sigma. \tag{2}$$

where d is the distance between transmitter and receiver, α is the path loss exponent which is considered as a constant value and the C also a constant value depending on operating frequency and antennas gain. The X_σ is the Gaussian random variable.

As the distance between satellite and UAVs is much bigger than the UAV moving distance. In this paper, we consider the moving speed of the UAVs and ground users not affect their distance to satellites. Thus the channel gain is formulated based on Weibull-based channel model [29] as:

$$G_c = \frac{G_t G_r \lambda^2}{4\pi d^2} 10^{-\frac{F_{rain}}{10}}. \tag{3}$$

where the G_t and G_r represent the antenna gain in transmitter and receiver respectively. λ is the wavelength of the propagation signal and the F_{rain} is the distribution of rain attenuation follows Weibull-based channel model [29].

Thus, the transmission rate between satellite and UAV as well as the ground nodes can be calculated following [6] as:

$$R_{sat} = Wlog_2(1 + \frac{P_t \cdot |G_c|^2}{\sigma^2}), \tag{4}$$

In the high dynamic SAGIN, due to the frequent link break and relink, we further define the transmission rate of both satellite and UAVs of link e_{ij} is suddenly change with link break as follow:

$$R_{i,j} = \begin{cases} R_\epsilon & \text{if } (R_{ij} >= R_\epsilon), \\ 0 & \text{if } (R_{ij} < R_\epsilon). \end{cases} \tag{5}$$

where, R_ϵ denotes a threshold of the transmission rate of link e_{ij}. When the calculated transmission rate of the corresponding link is bigger than R_ϵ means the link is correctly connected otherwise it is broken due to the long distance or big noise. When the link is broken, the transmission rate is constrained as empty.

D. Transmission Model

The wireless communication system, the end-to-end transmission delay is constructed with three parts, the propagation delay, transmission delay and queue delay. When the routing path rp is decided, the final end-to-end delay can be formulated as the sum of multiple-hop delays:

$$\mathcal{D}(rp) = \sum_{rp_{i,j} \in rp} d_{rp_{i,j}} \tag{6}$$

where, $rp_{i,j}$ means the one-hop transmission from node i to j of the routing path rp, and $d_{rp_{i,j}}$ represents the one-hop transmission delay, which can be calculated with:

$$d_{rp_{i,j}} = \frac{d_{i,j}}{v_p} + \frac{l_p}{R_{i,j}} + n_q \frac{l_p}{R_{i,j}} \tag{7}$$

where, $d_{i,j}$ is the distance from node i to j, v_p is the signal propagation speed, l_p is the length of transmission packet and n_q is the number of queuing packets in the transmission buffer.

E. Problem Formulation

We express the problem with an objective function that attempts to minimize the packet delay over an entire network for all nodes over an expressed time T by application of a traffic offloading method $F_{\rho,h}(n_s \rightarrow n_f, RP)$ to the defined packet-path $P_{n_s \rightarrow n_f}$ at time τ dependent on explicit parameters of ρ and h. Here, the RP is a set of candidate offloading paths $RP = \{rp_1, \ldots, rp_{|H|}\}$, $\rho \in \{0, 1\}$ represents offloading decision, and h means the h-th path. $rp = n_r \rightarrow n_f$ is the offloading path which is decided by the relay node n_r, in the considered SAGIN, the relay node can be the satellite and one of the ground node or UAVs.

Thus, the final optimization objective of this work can be formulated as:

$$\begin{aligned} &\arg\min_{\rho,h} \sum_{T} \sum_{N} \mathcal{D}(F_{\rho,h}(n_s \rightarrow n_f, RP)), \\ &\text{s.t.} \quad \forall_{e_{ij}} \sum_{N} \mu_{rp} \leq R_{ij}, \mu_{rp} \in p_{n_s \rightarrow n_f}, \\ &\qquad n_s \in NS, \quad n_f \in N, \\ &\qquad rp_h \in RP, \quad n_r \neq n_s. \end{aligned} \tag{8}$$

where $\mathcal{D}(\cdot)$ calculates the overall packet delay of the input array with equation.(6). We attempt to minimize the parameters of the traffic offloading method that allows us to minimize global delay of the network.

IV. Distributed Reinforcement Learning Based Traffic Offloading

To achieve the formulated objective function, in this section we present a novel deep reinforcement learning based traffic offloading scheme that utilizes both delay state gathering and delay sensitive memory replay methods.

As we discussed earlier, utilizing smart and state-of-the-art methods is becoming essential in future generation networks. Since the traffic offloading problem is a high dimensional and high state/action problem, the application of optimized generalization algorithms such as deep reinforcement algorithms is not only essential, but also crucial for employment of successful applications for future communication networks. Moreover, because of the capability of these efficient algorithms, utilizing the information of the surrounding devices as a state is possible, which is something that most researchers tend to ignore. Therefore we propose a smart deep reinforcement learning algorithm paired together with an efficient state gathering method that allows us to streamline salient surrounding device information to fully make use of the learning ability of reinforcement learning models. Besides, in order to adjust to the delay sensitive scenario in SAGIN, we propose a sumtree based prioritized replay algorithm which efficiently sellect the training samples not fully random but more relies on the delay performance of historical actions.

A. Learning Model

As shown in previous related research [30], [31], utilization of smart learning algorithms show high performance improvements over old and obsolete protocols. In particular, the Q-Learning method [32] has shown notable success as the reinforcement learning algorithm of choice for novel wireless network challenges [33] because of how the method efficiently learns the optimal policy for any finite Markov decision process via a simple reinforcement algorithm. The Q-Learning utilizes a representation of the problem or challenge using an agent that makes decisions, called actions, given a particular environment, also referred to as the state.

Q-Learning works by attempting to learn some model that tells us what the best action is in any state. It does that by modeling it as the method $Q(s_t, a_t)$ which returns an individual value that is termed the reward, signifying how effective that action a_t is at state s_t. In the formulated problem, action a_t is a index combination of $\{\rho, rp_j\}$ Thus, the best offloading action for objective function of equation. (8) can be rewritten with Q-formulation as:

$$\hat{F}(n_s \rightarrow n_f) = \arg\max_{\rho,j} Q(s_t, \{\rho, rp_j\}). \tag{9}$$

However, since in many reinforcement learning applications what effect a particular action has on the future is of high importance, some method needs to update $Q(s_t, a_t)$ such that we can accurately generalize or approximate action effects on future states. This is done via the Bellman equation,

$$\begin{aligned} Q(s_t, a_t) = &\, Q(s_t, a_t) \\ &+ \alpha(R_{t+1} + \gamma \max Q(s_{t+1}, a) - Q(s_t, a_t)). \end{aligned} \tag{10}$$

where R_{t+1} is the feedback reward in time slot $t+1$, α is the learning rate and γ is the learning discount. The $Q(s_t, a_t)$ function, which can also be referred to as the Q-Matrix, as in the original paper [32] a deep neural network was used, is iteratively updated via equation. (11).

To make up for larger state and action domain problems, an extension to the Q-Learning method is the Deep Q-Learning method that has recently emerged as the preferred advanced method. This method utilizes a neural network instead of a simple, matrix-form table, hence the addition of the word "Deep". In [34], Huang *et al.* apply a deep q-learning algorithm to predict future traffic loads as well as to estimate

an offloading decision in a static macro and micro cell environment. The applied method proves the superiority of such methods because instead of using a simplified data structure such as a matrix to represent $Q(s_t, a_t)$, a neural network is utilized, which can allow us to approximate the method for larger state spaces.

However, this particular solution would have trouble in even much larger state and action environments such as an ultra-heterogeneous network such as SAGIN that heavily relies on devices with high dynamic properties. The reason for this is that while effective, the Q-Learning and in turn the Deep Q-Learning algorithm updates a certain value from the past state of the same value and may create a large bias.

This is apparent when looking at the above Bellman equation's $\max Q(s_{t+1}, a)$ expression, which in particular inflates any overestimated values which in turn create an estimation bias. This is especially troublesome to large state and action problems because any overestimation of actions will not allow the algorithm to generalize the environment to the most optimized state. Even when using a single neural network to approximate the Q-value rather than a matrix, the algorithm tends to skew toward overestimated actions. To combat this, we decided to utilize a Double Deep Q-Learning algorithm [35] that reduces optimization biases by using two neural networks, or Q-value estimators, to evaluate each others' action selection and evaluations. The two newly introduced networks are called the Policy and Target networks.

Instead of using the same Q-Matrix, or Q-Network in this case, to update itself, we split the parameters into two different Q-Networks that contain their own set of parameters, weights and biases so that any inflation due to noise is spread differently within the two networks.

With the utilization of the Policy Q-Network and Target Q-Network, $Q(s, t)$ and $Q^*(s, t)$ respectively, we can create an unbiased estimating solution. The updating procedure then changes to:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(R_{t+1} + \gamma Q^*(s_{t+1}, a) - Q(s_t, a_t)). \tag{11}$$

where at every iteration we interchange weights and bias to allow the target network $Q^*(s, t)$ to minimize parameter efficiency decline since the target network does not get updated. In Fig. 4, a high-level overview of the Double Deep Q-Learning system can be seen, which shows the general flow of data. The details of the loss function and replay memory will be defined in the following sub-sections.

It's important to note however that when applying a neural network to approximate the Q-Matrix as a function of both the state and action such as in our previous work [36], it is particularly inefficient to probe for every action. So instead, the Q-Network will instead take the state as a singular input and return the expected Q-values for every action.

B. Distributed or Centralized Approach

The hyper-heterogeneity nature of SAGIN asks the question of whether a centralized or distributed approach should be taken. We decided to take advantage of a distributed approach,

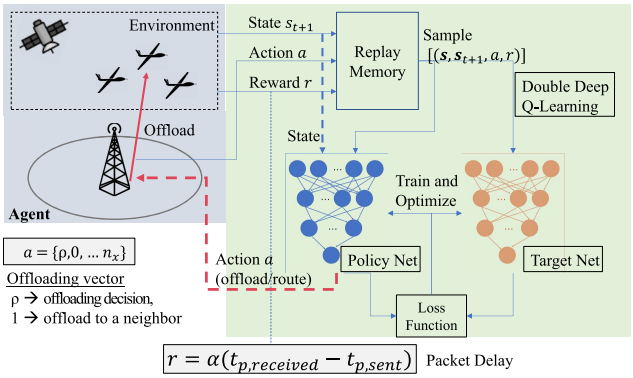


Fig. 4. Double deep Q-learning system overview.

due to several factors such as the increase in cheap modern hardware processing, which makes choosing to utilize a reinforcement learning algorithm in a distributed traffic offloading approach a sensible approach. The benefits of distributed communication has been studied extensively in networks such as ones that utilize device-to-device relaying [37] or in ad-hoc networks [38].

Furthermore, when allowing single devices to process their own traffic offloading decision, you allow a more flexible system that can utilize both centralized and distributed approaches for different use-cases. Lastly, efforts fall short when applying centralized traffic offloading approaches to heterogenous environments with inherently "distributed" devices such as UAV's due to their mobility and the problems you're faced when attempting to coordinate centralized servers and moving devices. Lastly, since the environment of dynamic devices differ greatly from device to device, utilizing a centralized system requires more overhead as management of the different models for different devices becomes needed, and it is also for this reason that we believe a distributed approach is most efficient.

C. State and Action Representations

In reinforcement learning, the overarching representation of all the algorithms is that of an agent and environment pair which interact with each other. The agent attempts to learn about the environment through a trial and error process, or in other words, it attempts to build a model of the environment through an algorithmic approach. It can also be said that the agent attempts to learn the best action depending on the state of the environment. In our method, we chose to model the state as the most relevant surrounding environment of devices and their parameters. This is important to note because in many proposed traffic offloading solutions, researchers fail to provide the most relevant piece of information for their traffic offloading models, which is the surrounding information of the topology.

As mentioned above, the algorithm will be deployed in a distributed manner and as such, each device is responsible for collecting and managing their own state at regular intervals. Given the then processed state, individual devices process and learn the correct traffic offloading action at any time t. The traffic offloading action can be described as either initializing an offloading request with another neighbor device, or continue to route the data via the originally defined traffic

distribution method. With this semi-binary action approach, this traffic offloading method does not disrupt any other defined traffic routing methods and can be used together. This is important especially in a SAGIN because of the diverse set of traffic routing protocols that might be defined [1].

We choose to represent the state as a multi-dimensional array which contains relevant information of all surrounding one and two-hop neighbors as normalized scalars of different device parameters or attributes that allow us to explain the total state of all surrounding devices. These attributes can be specifically defined by the network operators and can be diversified to fit the needs for each device, which is a design-choice we decided to take that aligns with a distributed solution.

$$S_{n_i,t} = \{\mu_1, \ldots, \mu_i \ldots\}, \quad i \in \{N_{adj}(n_i), N_{adj}(N_{adj}(n_i))\}. \tag{12}$$

where $N_{adj}(\cdot)$ represents the set of adjacent vertices, or nodes. The size of the state space is equal to the max number of potential neighbors of two-hops and is also equal to the square of the number of possible sub-connections. Case studies show that utilizing a two-hop approach of collecting neighboring information is effective [39], [40].

The action space $a_{n_i,t} = \{\rho, pr_j\}$ represents the offloading decision and the offloading path of node n_i in time slot t. The ρ represents offloading decision, indicating whether the device will offload the packet. The offloading action is limited with single path. In the SAGIN, the choice of offloading path pr_j is equal to the choice of next relay node $ar_{n_i,t}$ (i.e., the satellite or one of the neighboring UAVs), as the relay node well automatically decide the path with default routing protocol or another offloading decision. We represent the offloading decision path as a pseudo-binary vector, where the element of "1" in the vector denotes the selected offloading node including all neighboring UAVs and satellite. The length of the vector is equal to the max number of neighbors, when the practical candidate is smaller than max value (e.g., no enough neighbor or link is break), the elements in the vector are filled with 0.

$$\begin{aligned} a_{n_i,t} &= \{\rho, ar_{n_i,t}\}, \\ ar_{n_i,t} &\in \{0, \ldots, n_x\}, \quad x \leq |N(n_i)|. \end{aligned} \tag{13}$$

Given the state, the Double Deep Q-Learning module will learn to map the defined state of the network to a simple array list that represents the nodes to offload to or not. Since the state contains information of two-hop neighbors and is updated at high intervals, we are able to keep a robust knowledge of any devices in a relevant time period.

D. Training and Running Phase

Given the environment, the agent must learn what the best action to make is depending on the current state of the environment, and this is done through what is called the training phase. In previous Q-Learning methods, the agent would normally look at the current state, decide an action, and depending on a returned reward value (a value that defines how good the action was or not) it will update the Q-Matrix, which represents the model of the environment that describes how good certain actions are at certain states. In a Double Deep Q-Learning method however, the model is not represented as a matrix, but as a set of weights and biases in a neural network, which can be referred to as a Q-Network (analogous to the Q-Matrix). Due to this, when updating the Q-Network, the addition of something called replay memory, shown in Fig. 5, is incorporated to optimize learning. Replay memory is a key technique to training Q-Learning models that allows the agent to learn from earlier "memories", which in this case are just recorded instances of past states, actions, rewards and the following states given the chosen action. The Replay Memory in simple terms in just a data structure that will contain past state, action and reward samples.

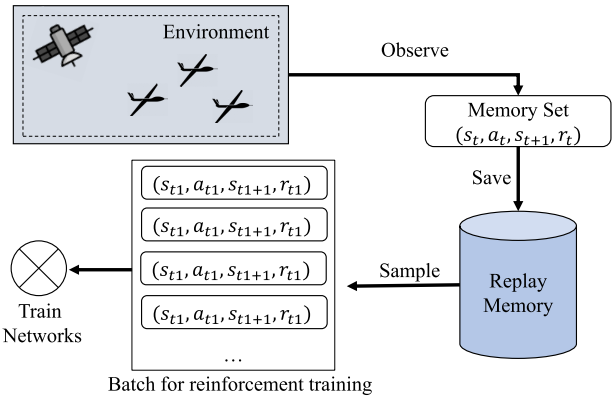


Fig. 5. Replay memory.

In normal double deep Q-learning, the sampling from replay memory is totally random. However, the random sampling method ignores the timeliness and performance bias of different actions which is not eifficent in the time-sensitive scenario. To improve the sampling and training performance, we propose a delay sensitive replay memory algorithm (DSRPM).

In DSRPM, a sumtree [41] is employed to record training samples based on their weight. The weight w_i of sample i is calculated based on the delay performance as:

$$w_i = \tanh k \times \frac{1}{d_i} \tag{14}$$

where, d_i is the delay of sample i, k is the delay influence coefficient.

After the weight of the sample is calculated, we add the sample i and corresponding wight w_i to the sumtree as a new leef and update the sumtree. Then, when training begin, the learning system will choose the samples with highest wight (leef node) to training.

After sampling, to train the models from start, we have the two previously mentioned Policy and Target networks initialize with randomized weights and biases, and make decisions based on the ϵ-greedy model, which forces the model to occasionally take a random action rather than follow its policy. This is to allow the agent to "explore" the action space and not miss any actions that are actually high-rewarding but seem initially low-rewarding.

$$\epsilon \leftarrow \epsilon_{end} + (\epsilon_{start} - \epsilon_{end}) * e^{\frac{\tau}{\epsilon_{decay}}}. \tag{15}$$

where, the ϵ is the adaptive exploring ratio changing with learning process. At some defined interval we sample the replay memory structure and feed the data into the policy and reward networks. It is important to note that both the training phases and running phases are expected to run in parallel and that the network never stops learning. This allows the network to continuously modify its' weights and biases according to any changes in the environment.

At every timed iteration, the state is fed into both the Policy and Target Q-Networks, $Q(s)$ and $Q^*(s)$, and their outputs are processed as actions, seen in equation. (13). The Policy network output action is used as the actual action of the device, whereas the Target network output is utilized to update the networks via a loss function that compares the output of both the Policy and Target network. This is similar to the original Deep Q-Learning algorithm where it uses a loss function with itself. We chose to utilize the Huber loss function because of its ability to ignore anomalities in data.

$$L_\delta(y-\hat{y}) = \begin{cases} 1/2(y-\hat{y})^2 \\ \delta|y-\hat{y}| - 1/2\delta^2 \end{cases} \tag{16}$$

where, the y is the out put of learning system and $\hat{y}$ is the targeted value, and $\delta = |y|$ is the scalar of output vector. is. We utilize the loss function seen in equation. (16) to calculate the error, or the difference of the expected output and the predicted output. Once the loss has been calculated, we feed it into what we call an optimizer so we can understand how to update the weights and biases of the Policy and Target networks. In this system, we apply the popular RMSProp (root mean square) optimizing algorithm. The RMSprop is an adaptive learning algorithm that evaltuates gradient updates to improve the parameters of the network.

$$E[g^2]_t = \gamma E[g^2]_{t-1} + \eta g_t^2, \tag{17}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t. \tag{18}$$

where, the g_t is the gradient, η is constant value called forgetting factor. θ represents the wight and bias in the neural network. We can then apply these incremental changes continuously as the device processes offloading procedures at the same time. We incrementally update the Policy network via equation. (11). Since the objective is to decrease the total packet delay, we consider the reward R_{t+1} from equation. (11) to be the reciprocal of the packet delay:

$$R = \frac{1}{\nu(d_{P_{n_s \to n_f}})} \tag{19}$$

where, ν is a constant discount value. Utilizing the reward, we can update the networks during training and running. A pseudo code representation of the training and running phase is present in Algorithm 1.

E. Proposed State Gathering Method

In this part, we present our proposal of a novel state-gathering method and algorithm that utilizes the well known Hello-packets protocol to capture relevant surrounding information that will represent the state of the network.

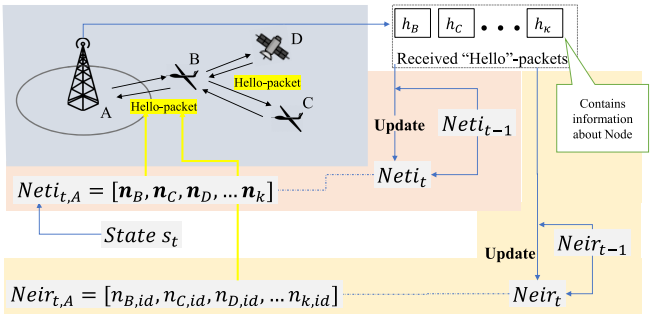


Fig. 6. NETI and NEIR tables.

As alluded in the previous sections, it is important that any traffic offloading system utilizing a machine learning model collects the correct information to make decisions, and it is especially important in a dynamic and heterogenous environment that the information collection method is robust and can capture topological information. The question here is then, how do we capture that information effectively. However, research has shown that the usage of the well known "Hello"-packet protocol can be successful in capturing the most relevant topology information just by collecting information of the first and second-hop neighbors [24]. By sending out special packets periodically, we can establish and collect all possible network connections available to a device in its vicinity. As a device collects Hello packets periodically, our proposed system will build two important tables that will give enough salient information of the surrounding environment, or surrounding topology, which is to be fed into the machine learning module as the state.

As Hello packets are collected from neighboring devices, we split the associated data from the packets into two tables, the network-info table (NEIR) and the network-reference table (NETI). The Hello packets are expected to contain the calculated NEIR and NETI tables as well as other data points about the device who is sending the Hello-packet, these tables are then calculated directly from these received Hello-packets.

The NEIR table will contain information to make the correct traffic offloading decision such as power consumption levels, current queue capacity, and energy constraints of the node's neighbors, and second-hop neighbors. Utilizing this two-hop state approach, we are able to catch enough relevant topological information of the network. The efficiency of gathering surrounding topology information via two-hop neighbor references has been proven to be effective via the application of several protocols such as the Optimized Link State Routing protocol (OLSR) which also utilizes Hello-packets and a Topology-control algorithm. Unlike OLSR however, our proposal shifts the next-hop decision making to another domain rather than encapsulate it together in one algorithmic flow, as well as separate irrelevant data that would skew the machine learning model. The NEIR table will then be directly fed into the Double Deep Q-Learning system and can be said to represent the state of the network.

The NETI table however will contain information not relevant to the machine learning model but are necessary for actual traffic offloading such as any parameters pertaining to the transmission protocol. This allows the traffic offloading

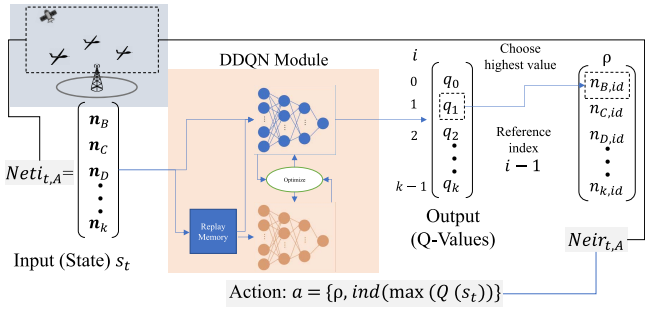


Fig. 7. Input and output data flow.

system to again work in tandem with a variety of protocols that network-administrators might require. The reason for this exclusion is to allow the model to learn the important features of the topology without any possibility of becoming skewed due to irrelevant data inclusions of physical parameters. The arriving Hello-packets at time slot t are then represented as $H = \{h_{n_j}\}$, where n_j is the j-th node. Hello-packets is at the very minimum size which maintains information of queue size, offloading method, and a unique packet ID. Then, the NETI table of node n_i is constructed with information in hello-packets as follow:

$$NETI_{n_i} = [h_{n_0}[p_\rho], \ldots, h_{n_j}[p_\rho], \ldots, h_{n_{N_c}}[p_\rho]],$$
$$n_i \in N, \quad n_\mu \in \{N_{adj}(n_i), N_{adj}(N_{adj}(n_i))\}, \ h \in H. \tag{20}$$

where, $N_c = |\{N_{adj}(n_i), N_{adj}(N_{adj}(n_i))\}|$ is the total number of neighboring nodes within two hopes. The parameters of node n_j collected from Hello-packet are considered to be $h_{n_j}[p_\rho] \in h_{n_j}$. These parameters can be self defined by the network operator, but in our case we have chosen to utilize the queue size at time t as the main parameter.

Given these definitions, we propose the utilization of an updating algorithm that can capture salient state information. At initialization, we first populate the NEIR and NETI tables with the first Hello-packets that arrive and append N_c counters to the ID in the NETI table. Then we reference the parameters defined by the network operator that they deem important in the Hello packets and append the contents to the NEIR table to be used for the state later on. The NETI and NEIR tables are updated every M_c seconds also defined by network and device operators. Since the method is distributed, we are allowed to make these single definitions for every device or cluster. When it comes time to re-populate or update the tables, we sort by the N_c counter to allow nodes that have been seen often to always stay referenced in the tables. This allows static nodes such as base stations to always maintain their presence in the tables. When no Hello-packets are available for a table entry, the N_c counter decreases to allow devices that are moving to be removed from the tables. To mitigate error from natural link-state exceptions, we utilize a reverse exponential decrease in the N_c counter so we can ensure nodes with high count stay referenced.

F. Input and Output Flow

Having now understood how the state is collected and represented as well as what the action space is, in this sub-section we will discuss the traffic offloading system flow and how all the system pieces work together.

Algorithm 1 NEIR and NETI Table Update Process

Input : Array of Hello-Packets
Output: Populated NEIR and NETI Tables
forall the HelloPackets H **do** Update
  Collect ID and offloading method of device to NETI table;
  NETI $\leftarrow \{h_{id}, h_{of}, N_c\}$
  if *isempty(*NEIR*)* **then**
    for *all* ρ **do**
      Append parameter to NEIR table;
      NEIR of current id $\leftarrow h[p_\rho]$
    end
  else
    forall the NETI *entries sort by* N_c **do**
      if *NETI entry in* H **then**
        NEIR of current id $\leftarrow h[p_\rho]$
        $N_c = N_c + \frac{1}{1+e^{N_c}}$
      else
        $N_c = N_c - \frac{1}{1+e^{N_c}}$
        if $N_c \leq 0$ **then**
          Remove from NETI and NEIR
        end
      end
    end
  end
end

With the arrival of Hello-packets and after a certain time $\tau\Delta_t$, where τ is the relay steps and Δ_t is the time period of time slot t. the NETI and NEIR tables are uploaded via the state update gathering method explained in the previous sub-section. These data structures are then utilized whenever a packet has been received and processed as ready for transmission. The agent or relaying device with capability to offload will then feed the current state from the NETI to the Double Deep Q-Learning module and process whether the current processed packet should offload or not, and if so, to which neighboring device it should offload to. Given the output of the Policy network, which are Q-values for every action, we use the index of the largest Q-value in the output and reference that index in the NETI tables, which contains the information relevant to processing an offloading request, as well as the device to offload ID. This flow can be seen in Fig. 7 which shows how we expect to reference the data from the NETI table. The NETI table can also be considered to be a tool to seperate the offloading logic from the state or environment learning.

In this section, we presented our traffic offloading system that utilizes an effective unification of a novel state-gathering algorithm and an improved reinforcement learning model. We discussed the reasoning toward our design choices, as well as presented the problem statement. We chose to utilize the Deep Q-Learning framework to overcome the large state and action spaces that are difficult to solve without approximation functions such as a neural network. In particular, the choice of using two neural networks to balance out any biases strikes

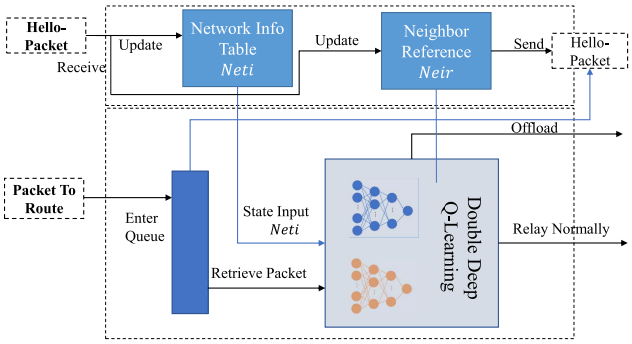


Fig. 8. Simplified overview of proposed method.

a balance between efficiency and computational correctness. Given the learning model of choice, we explained our novel state-gathering method that is able to capture just the right amount of salient data points from the network topology, which can be directly applied to the Deep Q-learning model as the input, or as the state of the environment, which is then processed to create an accuracte offloading decision.

An overview of the proposed solution can be seen in Fig. 8.

V. Performance Evaluation

In this section, we present results from our simulation and experiments. In particular, by applying our method versus an approximated version of conventional traffic offloading as well as normal routing, we show that our method achieves more optimal results and outperforms conventional offloading. Additionally, we present the simulation environment and the chosen parameters for both the simulation and the model.

A. Signaling Overhead Analysis

In this part, we at first simulated the signaling overhead of our proposed distributed learning with NETI and NEIR state gathering compared with centralized control method and distributed control method with gathering of global information. We consider the total nodes count from 50 to 300 and simulate the signaling process in one iteration. As the result shown in the Fig. 9. Compared with distributed control method with gathering of global information, our proposal always shows much smaller signaling overhead. Besides, compared with centralized control, the signaling overhead of our proposal is higher than the centralized control method when the network scale is small (less than 100 nodes) but smaller than it when the network scale is big. And the signaling superiority becomes significant when the network is large-scaled. In the next part, we detailed evaluate the network performance of our proposal ignoring the signaling overhead.

B. Simulation Environment

The simulation environment is modeled after a typical SAGIN environment [1] in which the devices are split into three different abstract kinds of device types, namely a source-node device type, a relay-node device type, and a destination-node device type. The source nodes are responsible for generating data and the destination types are responsible

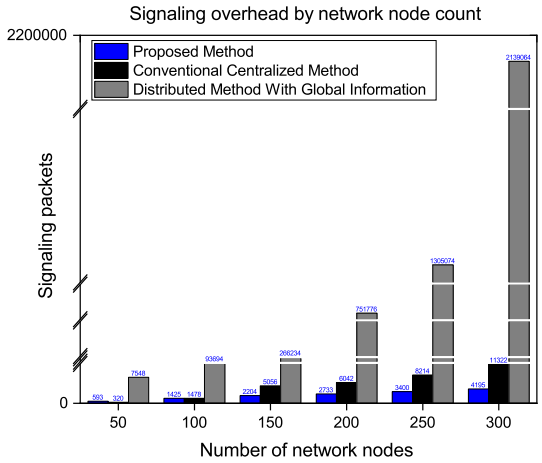


Fig. 9. Signaling overhead per increasing node count.

for receiving data. The relay nodes are expected to relay the data based on the Open Shortest Path First (OSPF) traffic routing protocol, as well as whatever defined offloading system is applied. The relay device types are modeled as UAV-Base Stations with mobility and normal, static Base Stations and satellite with slightly larger processing capabilities. As mentioned in section.III, the parameters used are based on the setting used in [6], [20], [28]. Three LEOs are considered in this simulation, the satellite orbiting time t_s is set as 105 minutes with with altitude of speed v_u of 7.35 km/s [42]. There are 6 moving UAVs. The altitude of UAV is set to 5km, and the moving speed v_u of the UAV is set as 5m/s in the first part and changes from 5m/s to 30m/s in the second part, the angle sin_v is $180°$. The moving speed v_d of ground devices is 1.4m/s. The source and node types are modelled after general User Equipment (UE) devices which utilize a normally distributed data generation rate. There are 100 to 400 source nodes and 80 destination nodes on the ground with 8 fixed base stations as relays. The link-states error percentage for the entire network is considered to be negligible.

C. Simulation Parameters

When choosing to utilize a deep neural network, in addition to finding the most efficient hyperparameters, it's also important to understand and choose the most optimal neural network architecture. This important because SAGIN provides a wide-range of devices with different processing capabilities, and when choosing a large network architecture with many layers that may be more accurate, the processing speed might be too slow for any use. Based on previous research by [31] that utilizes artifical neural networks (ANNs) in wireless networks, we've chosen to apply a neural network architecture with two hidden layers that reduce the size by 2/3 of the previous layer.

An overview of all the the key simulation parameters regarding both the environmental and machine learning model can be found listed in Table. I.

D. Simulation Results

To further add clarity to how our method compares, we also ran simulations in the same environment that utilize a simple

TABLE I
SIMULATION PARAMETERS

name	value	name	value
SN Count s	100 to 400	Replay Mem. Size	600 packets
SN Mobility Model	Random-Waypoint	Learning discount γ	0.798
SN Generation Rate	$\mathcal{N}(1e6, \sigma^2)$ Mbps	Training batch size	30
Device→ BS. &. UAV R_ϵ	1.5 Mbps	ϵ_{start}	0.8
Bs.→ Device &. Bs &. UAV R_ϵ	80 Mbps	ϵ_{end}	0.8
Bs.→ Sat. R_ϵ	15 Mbps	ϵ_{decay}	200
Sat.→ Sat. &. Bs. &. UAV R_ϵ	15 Mbps	Training period ω	0.05s
UAV→ Sat. &. UAV &. Bs. R_ϵ	20 Mbps	Simulation Area	100km^2
Bs. Count	8	Sat. count	3
UAV Count	6	Destination Node count	80

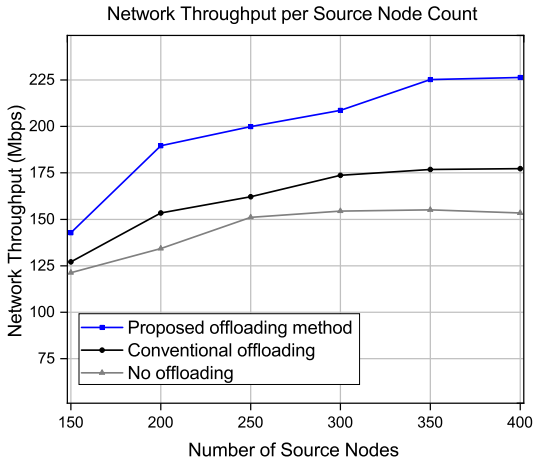


Fig. 10. Throughput per increasing source node count.

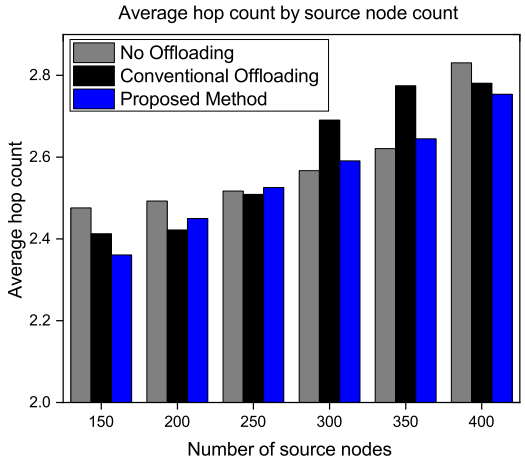


Fig. 11. Average hop count per increasing source node count.

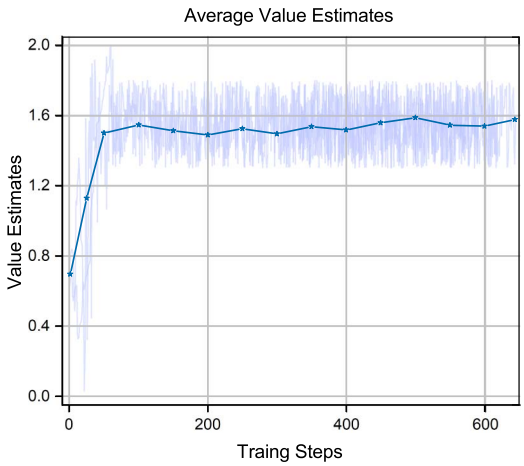


Fig. 12. Average calculated Q-values.

traffic offloading schema that resembles conventional traffic offloading used in today's networks [43]–[46]. Those conventional algorithms utilizes greedy algorithm which judge the offloading objectives based on the current network state. The conventional greedy based offloading solution in those cases is also distributed, and whenever a device begins to drop packages, the greedy based offloading algorithm attempts to begin an offloading procedure with the nearest neighbor. This solution takes no precautions in the state of the surrounding network. We dub this solution as the conventional method and compare it together with our method, as well as with a simple applied OSPF traffic offloading algorithm simulation that contains absolutely no traffic offloading, in which any transfer of data can only happen between devices of the same kind. In all simulations, we increased the source node count to observe how each method impacts how each device work to maintain the overall efficiency of the network. We have chosen to use the overall network throughput, packet drop rate and packet delay as the appropriate evaluation metrics.

Fig. 10 shows the superiority of our method against the pseudo-conventional method as well as regular traffic distribution method. It is expected for both offloading methods to out-perform simulations with no offloading method at all and this can be confirmed in the results. The three systems perform at near identical rates from 100 to 150 source nodes but after that the compounded distribution performance boosts begin to show from nodes 200 to 250 where our conventional traffic offloading method shines. Our proposed method pushes the upper-bound limit of the network to it's theoretical throughput capability. Besides, the average hop count of packets are demonstrated in Fig. 11. The hop count in our proposal is not always better than benchmarks as the hop-count is relatively not that critical factor in our work. Due to the unfair link-state of hops and long queue delay caused by congestion, the transmission routing path of less hop count can also with longer delay than the one of bigger hop count.

To evaluate the performance of training, we can see in Fig. 12 that the average value estimate of the Q-Values from several simulations begin to converge after the 150 iterations. The average training loss of abundant simulations shown in Fig. 13 allow us to see that the method quickly converges to near zero loss rapidly at around 600 training iterations.

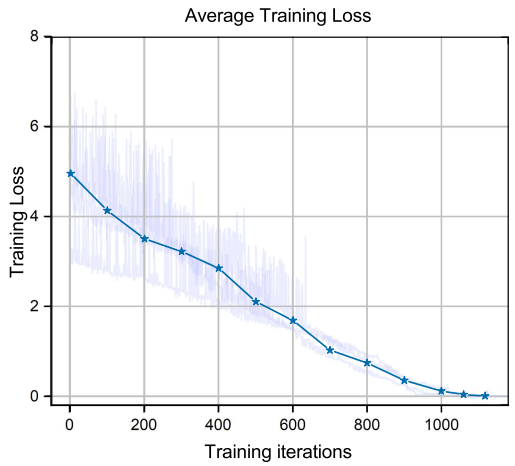


Fig. 13. Average training loss.

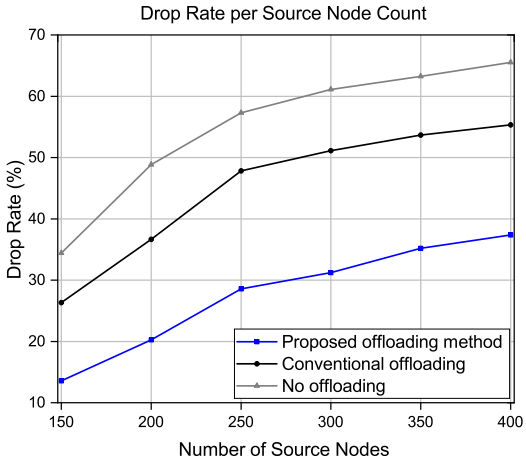


Fig. 14. Drop rate per increasing source node count.

This is due to the fact that training operations happen at only certaining iterations and thus the delay of action calculation propagates much later.

The performance increase by utilizing our proposed method can also be further examined and confirmed from Fig. 14 and Fig. 15 in which the drop rate performance as well as the average packet delay show significant improvements. Utilizing our offloading method, we are able to keep a more constant drop rate increase rate than the other methods. Increasing the source node count seems to bombard both the conventional offloading and no offloading networks such that they can not meet demands despite that the resource capability is present.

Lastly we can see the performance difference in packet delay shown in Fig. 15, over the three different simulations. Our method is able to consistently keep the average packet delay increase to near linear increase whereas the other methods expand exponentionally. Similar to the other figures, the obvious performance disparity is seen near the 200 source node count position.

To further show the performance of our proposal in the scenario with different mobility, we further simulate the network performance of packet drop rate and throughput. We change the moving speed of UAVs from 5m/s to 30m/s to see both the network performance. As shown in Fig. 16 and Fig. 17, the packets drop rate decreases and throughput decreases with

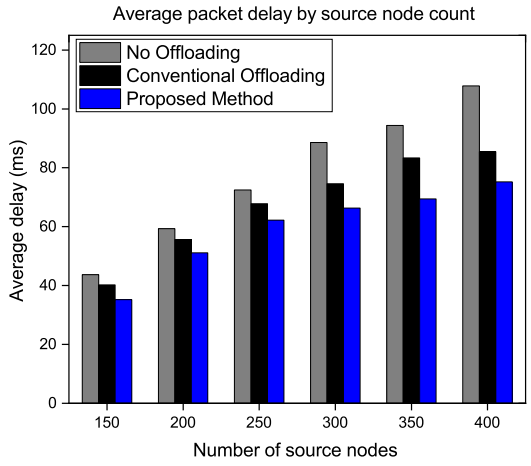


Fig. 15. Packet delay per increasing source node count.

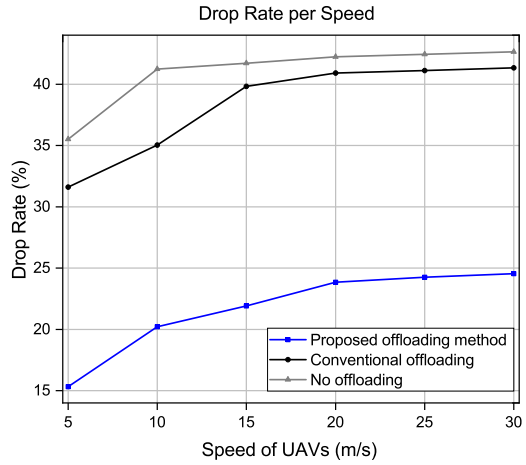


Fig. 16. Packet drop rate with increasing moving speed of UAVs.

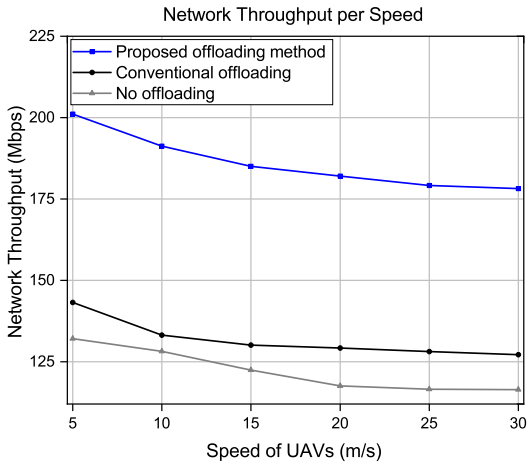


Fig. 17. Throughput with increasing moving speed of UAVs.

moving speed increase. However, both of the packets drop rate and throughput of our proposal are better than the two benchmarks. With this, we understand the overall efficiency and superiority when applying our system to solve traffic offloading in SAGIN.

VI. Conclusion

The idea of a Space-Air-Ground Integrated Network is an incoming research topic that is becoming increasingly relevant as we enter the age of advanced communications and as such,

it is important that we improve and innovate away existing protocols that are inadequate for future advanced networks such as SAGIN. In this paper, we present a novel distributed reinforcement learning based traffic offloading system for usage in SAGIN. Initially we present relevant challenges that can not be solved with conventional traffic offloading and that are not addressed by other relevant research. We then propose our reinforcement learning based system that utilizes a novel network state information gathering protocol. This protocol allows us to capture the information of the surrounding network in a flexible manner that can be fed to a reinforcement learning algorithm that can make the best offloading decision based on the network state. We then presented simulation results that compares our solution with a conventional offloading method and shows that our method is superior in reducing package loss and increasing network throughput as well as improving packet delay.

REFERENCES

[1] J. Liu, Y. Shi, Z. M. Fadlullah, and N. Kato, "Space-air-ground integrated network: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2714–2741, 4th Quart., 2018.

[2] S. Gu, Q. Zhang, and W. Xiang, "Coded storage-and-computation: A new paradigm to enhancing intelligent services in space-air-ground integrated networks," *IEEE Wireless Commun.*, vol. 27, no. 6, pp. 44–51, Dec. 2020.

[3] Z. M. Fadlullah *et al.*, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2432–2455, 1st Quart., 2017.

[4] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar, "Towards the deployment of machine learning solutions in network traffic classification: A systematic survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1988–2014, 2nd Quart., 2019.

[5] F. Mendoza, R. Ferrus, and O. Sallent, "A traffic distribution scheme for 5G resilient backhauling using integrated satellite networks," in *Proc. 13th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2017, pp. 1671–1676.

[6] C. Zhou *et al.*, "Delay-aware IoT task scheduling in space-air-ground integrated network," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[7] B. Fan, Z. He, Y. Wu, J. He, Y. Chen, and L. Jiang, "Deep learning empowered traffic offloading in intelligent software defined cellular V2X networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 13328–13340, Nov. 2020.

[8] H. Ye, L. Liang, G. Y. Li, and B.-H. F. Juang, "Deep learning-based end-to-end wireless communication systems with conditional GANs as unknown channels," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3133–3143, May 2020.

[9] G. Araniti, I. Bisio, M. De Sanctis, A. Orsino, and J. Cosmas, "Multimedia content delivery for emerging 5G-satellite networks," *IEEE Trans. Broadcast.*, vol. 62, no. 1, pp. 10–23, Mar. 2016.

[10] J. Du, C. Jiang, H. Zhang, Y. Ren, and M. Guizani, "Auction design and analysis for SDN-based traffic offloading in hybrid satellite-terrestrial networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2202–2217, Oct. 2018.

[11] J. Lyu, Y. Zeng, and R. Zhang, "Spectrum sharing and cyclical multiple access in UAV-aided cellular offloading," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2017, pp. 1–6.

[12] F. Tang, Z. M. Fadlullah, N. Kato, F. Ono, and R. Miura, "AC-POCA: Anticoordination game based partially overlapping channels assignment in combined UAV and D2D-based networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 2, pp. 1672–1683, Feb. 2018.

[13] H. Guo and J. Liu, "UAV-enhanced intelligent offloading for Internet of Things at the edge," *IEEE Trans. Ind. Informat.*, vol. 16, no. 4, pp. 2737–2746, Apr. 2020.

[14] F. Tang, B. Mao, Y. Kawamoto, and N. Kato, "Survey on machine learning for intelligent end-to-end communication toward 6G: From network access, routing to traffic control and streaming adaption," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 1578–1598, 3rd Quart., 2021.

[15] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari, "Deep reinforcement learning for dynamic multichannel access in wireless networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 4, no. 2, pp. 257–265, Jun. 2018.

[16] M. Chu, H. Li, X. Liao, and S. Cui, "Reinforcement learning-based multiaccess control and battery prediction with energy harvesting in IoT systems," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2009–2020, Apr. 2018.

[17] U. Challita, L. Dong, and W. Saad, "Proactive resource management for LTE in unlicensed spectrum: A deep learning perspective," *IEEE Trans. Wireless Commun.*, vol. 17, no. 7, pp. 4674–4689, Jul. 2018.

[18] R. Ding, Y. Xu, F. Gao, X. Shen, and W. Wu, "Deep reinforcement learning for router selection in network with heavy traffic," *IEEE Access*, vol. 7, pp. 37109–37120, 2019.

[19] R. Ding, Y. Yang, J. Liu, H. Li, and F. Gao, "Packet routing against network congestion: A deep multi-agent reinforcement learning approach," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2020, pp. 932–937.

[20] N. Cheng *et al.*, "Space/aerial-assisted computing offloading for IoT applications: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, May 2019.

[21] C. Wang, H. Wang, and W. Wang, "A two-hops state-aware routing strategy based on deep reinforcement learning for LEO satellite networks," *Electronics*, vol. 8, no. 9, p. 920, 2019.

[22] X. Liu, Y. Liu, and Y. Chen, "Reinforcement learning in multiple-UAV networks: Deployment and movement design," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 8036–8049, Aug. 2019.

[23] L. Liu, S. Zhang, and R. Zhang, "CoMP in the sky: UAV placement and movement optimization for multi-user communications," *IEEE Trans. Commun.*, vol. 67, no. 8, pp. 5645–5658, Aug. 2019.

[24] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, "Optimized link state routing protocol for ad hoc networks," in *Proc. IEEE Int. Multi Topic Conf., Technol. 21st Century (IEEE INMIC)*, Dec. 2001, pp. 62–68.

[25] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Commun. Mobile Comput.*, vol. 2, no. 5, pp. 483–502, Sep. 2002.

[26] B. Jabbari, Y. Zhou, and F. Hillier, "Simple random walk models for wireless terminal movements," in *Proc. IEEE 49th Veh. Technol. Conf.*, vol. 3. May 1999, pp. 1784–1788.

[27] X. Lyu *et al.*, "Optimal schedule of mobile edge computing for Internet of Things using partial information," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2606–2615, Nov. 2017.

[28] A. Al-Hourani and K. Gomez, "Modeling cellular-to-UAV path-loss for suburban environments," *IEEE Wireless Commun. Lett.*, vol. 7, no. 1, pp. 82–85, Feb. 2018.

[29] S. A. Kanellopoulos, C. I. Kourogiorgas, A. D. Panagopoulos, S. N. Livieratos, and G. E. Chatzarakis, "Channel model for satellite communication links above 10 GHz based on Weibull distribution," *IEEE Commun. Lett.*, vol. 18, no. 4, pp. 568–571, Apr. 2014.

[30] B. Mao *et al.*, "Routing or computing? The paradigm shift towards intelligent computer network packet transmission based on deep learning," *IEEE Trans. Comput.*, vol. 66, no. 11, pp. 1946–1960, Nov. 2017.

[31] F. Tang, B. Mao, Z. M. Fadlullah, J. Liu, and N. Kato, "ST-DeLTA: A novel spatial-temporal value network aided deep learning based intelligent network traffic control system," *IEEE Trans. Sustain. Comput.*, vol. 5, no. 4, pp. 568–580, Oct. 2020.

[32] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[33] H. Tsuchida *et al.*, "Efficient power control for satellite-borne batteries using Q-learning in low-earth-orbit satellite constellations," *IEEE Wireless Commun. Lett.*, vol. 9, no. 6, pp. 809–812, Jun. 2020.

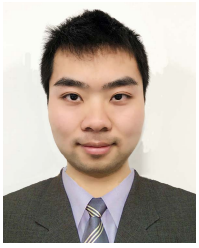
[34] C. Huang and P. Chen, "Mobile traffic offloading with forecasting using deep reinforcement learning," *CoRR*, vol. abs/1911.07452, pp. 1–9, Nov. 2019.

[35] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 30, Mar. 2016, pp. 2094–2100.

[36] F. Tang, Y. Zhou, and N. Kato, "Deep reinforcement learning for dynamic uplink/downlink resource allocation in high mobility 5G HetNet," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 12, pp. 2773–2782, Dec. 2020.

[37] A. Asadi, Q. Wang, and V. Mancuso, "A survey on device-to-device communication in cellular networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 1801–1819, 4th Quart., 2014.

[38] T. K. Forde, L. E. Doyle, and D. O'Mahony, "Ad hoc innovation: Distributed decision making in ad hoc networks," *IEEE Commun. Mag.*, vol. 44, no. 4, pp. 131–137, Apr. 2006.
[39] S. Yang, W. Song, and Z. Zhong, "Packet-level performance analysis for video traffic over two-hop mobile hotspots," *IEEE Wireless Commun. Lett.*, vol. 1, no. 2, pp. 137–140, Apr. 2012.
[40] Z. Jin, W. Su, J. Cho, and E. K. Hong, "An analytic model for the optimal number of relay stations in two-hop relay networks," *IEEE Commun. Lett.*, vol. 17, no. 2, pp. 285–288, Feb. 2013.
[41] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. ICLR*, 2016, pp. 1–21.
[42] B. S. Yeo, "An analysis of the impact of earth rotation on LEO satellite mobility models," in *Proc. 57th IEEE Semiannu. Veh. Technol. Conf. (VTC-Spring)*, Apr. 2003, pp. 1376–1380.
[43] X. Kang, Y.-K. Chia, S. Sun, and H. F. Chong, "Mobile data offloading through a third-party WiFi access point: An operator's perspective," *IEEE Trans. Wireless Commun.*, vol. 13, no. 10, pp. 5340–5351, Oct. 2014.
[44] H. Yu, M. H. Cheung, G. Iosifidis, L. Gao, L. Tassiulas, and J. Huang, "Mobile data offloading for green wireless networks," *IEEE Wireless Commun.*, vol. 24, no. 4, pp. 31–37, Aug. 2017.
[45] A. Y. Ding, Y. Liu, S. Tarkoma, H. Flinck, H. Schulzrinne, and J. Crowcroft, "Vision: Augmenting WiFi offloading with an open-source collaborative platform," in *Proc. 6th Int. Workshop Mobile Cloud Comput. Services*. New York, NY, USA: Association for Computing Machinery, 2015, pp. 44–48.
[46] J. Korhonen, T. Savolainen, A. Y. Ding, and M. Kojo, "Toward network controlled IP traffic offloading," *IEEE Commun. Mag.*, vol. 51, no. 3, pp. 96–102, Mar. 2013.



Fengxiao Tang (Member, IEEE) received the B.E. degree in measurement and control technology and instrument from the Wuhan University of Technology, Wuhan, China, in 2012, the M.S. degree in software engineering from Central South University, Changsha, China, in 2015, and the Ph.D. degree from the Graduate School of Information Sciences (GSIS), Tohoku University, Japan. He is currently an Associate Professor at GSIS, Tohoku University. His research interests include unmanned aerial vehicles systems, IoT security, game theory optimization, network traffic control, and machine learning algorithm. He was a recipient of the Prestigious Dean's and President's Awards from Tohoku University in 2019. He was also a recipient of the several best paper awards from conferences, including IC-NIDC 2018 and GLOBECOM 2017/2018, and the prestigious Funai Research Award in 2020.

Hans Hofner (Student Member, IEEE) received the B.E. degree from The University of New Mexico, Albuquerque, NM, USA, in 2018, and the M.S. degree from the Graduate School of Information Sciences (GSIS), Tohoku University, Japan, in 2021. His research interests include machine learning, SAGIN, and network traffic control.

Nei Kato (Fellow, IEEE) is currently a Full Professor (the Deputy Dean) with the Graduate School of Information Sciences (GSIS) and the Director of Research Organization of Electrical Communication (ROEC), Tohoku University, Japan. He has been engaged in research on computer networking, wireless mobile communications, satellite communications, ad hoc and sensor and mesh networks, smart grid, and pattern recognition. He has published more than 400 papers in peer-reviewed journals and conference proceedings. He is a fellow of The Engineering Academy of Japan and IEICE. He was the Vice-President (Member and Global Activities) of the IEEE Communications Society from 2018 to 2019, the Chair of the IEEE Communications Society Sendai Chapter, the Editor-in-Chief of *IEEE Network Magazine* from 2015 to 2017, a Member-at-Large on the Board of Governors of the IEEE Communications Society from 2014 to 2016, the Vice Chair of the Fellow Committee of IEEE Computer Society in 2016, and a member of the IEEE Communications Society Award Committee from 2015 to 2017. He has been the Editor-in-Chief of IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY since 2017. He has also served as the Chair for the Satellite and Space Communications Technical Committee from 2010 to 2012 and the Ad Hoc and Sensor Networks Technical Committee of IEEE Communications Society from 2014 to 2015. He is a Distinguished Lecturer of the IEEE Communications Society and the IEEE Vehicular Technology Society.

Kazuma Kaneko received the B.E. degree in information engineering and the M.S. and Ph.D. degrees in information science from the Graduate School of Information Science (GSIS), Tohoku University, Sendai, Japan, in 2013, 2015, and 2018, respectively. He joined Mitsubishi Electric Corporation, Kanagawa, Japan, in 2018. He received the 2014 IEEE ComSoc Sendai Chapter Student Excellent Research Award, the Prestigious Deans Awards from Tohoku University in 2015, the Japan Society for the Promotion of Science Fellowship in 2016, and the Best Paper Award from the IEEE International Conference on Communications in 2017.

Yasutaka Yamashita received the B.E. and M.E. degrees in electrical engineering and information systems from The University of Tokyo, Japan, in 2013 and 2015, respectively. He joined the Information Technology Research and Development Center, Mitsubishi Electric Corporation, Kamakura, Japan, in 2015. His research interests include satellite communication systems and signal processing.

Masatake Hangai (Senior Member, IEEE) received the B.E., M.E., and Ph.D. degrees in electrical engineering from Tohoku University, Sendai, Japan, in 2000, 2002, and 2012, respectively. In 2002, he joined the Information Technology Research and Development Center, Mitsubishi Electric Corporation, Kamakura, Japan, where he has been engaged in research and development of microwave control circuits, microwave amplifiers, MMICs, modems, and active phased array antenna systems for satellite applications. He is a Senior Member of the Institute of Electronics, Information and Communication Engineers (IEICE), Japan. He was a recipient of the 2011 IEEE MTT-S Japan Young Engineer Award and the 2011 Michiyuki Uenohara Memorial Award.