RoLocMe: A Robust Multi-agent Source Localization System with Learning-based Map Estimation

Thanh Dat Le, **Lyuzhou Ye**, **Yan Huang**
University of North Texas
thanhle5@my.unt.edu, lyuzhou.ye@unt.edu, yan.huang@unt.edu

Abstract

This paper addresses the source localization problem by introducing **RoLocMe**, a multi-agent reinforcement learning system that integrates *SkipNet* — a skip-connection-based RSS estimation model — with parallel Q-learning. SkipNet predicts RSS propagation of the entire search region, enabling agents to explore efficiently. The agents leverage dueling DQN, value decomposition, and λ-returns to learn cooperative policies. RoLocMe converges faster and achieves at least 20% higher success rates than existing methods in both dense and sparse reward settings. A drop-one ablation study confirms each component's importance and RoLocMe's effectiveness for larger teams.

1 Introduction

Source localization is a well-established research problem with numerous applications, including forest fire tracking [Sharma *et al.*, 2020], dangerous gas leakage detection [Lu *et al.*, 2014], radioactive material localization [Morelande *et al.*, 2007; Zhao *et al.*, 2022], search-and-rescue operations [Atif *et al.*, 2021], and security surveillance [Rybski *et al.*, 2000].

Recent deep learning-based source localization methods [Zhang *et al.*, 2016; Zubow *et al.*, 2020; Zhan *et al.*, 2021; Wang *et al.*, 2022; Lin *et al.*, 2022; Raj and B. S., 2023] learn directly from data and often outperform traditional techniques, especially under imperfect conditions such as sparse sensor coverage or signal degradation, where classical approaches tend to falter. However, extreme signal attenuation challenges both. Although traditional methods frequently fail outright, deep learning models, though affected, tend to be more robust, leveraging weak signal patterns to maintain some level of performance. Mobile sensors address this shared vulnerability by actively seeking higher-quality data to reduce uncertainty and improve localization accuracy.

Early research on mobile sensing for source localization primarily relied on path-planning strategies [Koutsonikolas *et al.*, 2006; Hu *et al.*, 2008; Jiang *et al.*, 2011; Rezazadeh *et al.*, 2014], which were straightforward to implement but struggled to generalize in complex environments. To overcome these limitations, recent efforts have shifted toward data-driven reinforcement learning approaches, broadly categorized into single-agent and multi-agent methods. Single-agent approaches [Wu, 2019; Ebrahimi *et al.*, 2021; Proctor *et al.*, 2021; Zhao *et al.*, 2022] have demonstrated adaptability in various environments but often result in prolonged searches and lack generalizability for multi-agent scenarios. Recent multi-agent reinforcement learning studies [Alagha *et al.*, 2022; Alagha *et al.*, 2023; Wickramaarachchi *et al.*, 2024] have shown promise but face challenges in addressing complex environments and severe signal attenuation.

1.1 Our Contributions

This paper presents a multi-agent approach to source localization, addressing two critical challenges: (1) Adaptability in complex environments, where agents must handle multi-path fading in unknown environments with dense obstacles, and (2) Extreme signal attenuation, where latent information from weak signal samples is difficult to discern, posing challenges for effective decision-making. To overcome these challenges, we make the following contributions:

- We propose RoLocMe, a novel system that integrates a radio map estimation module, with the Value Decomposition Network (VDN) [Sunehag *et al.*, 2017] for agent training. Unlike recent methods relying solely on sampled signals and agent positions, an explicit signal propagation map estimation module is seamlessly integrated with a reinforcement learning process, augmenting previous observations to improve navigation and planning.
- We evaluate RoLocMe across various scenarios involving complex environments and extreme signal attenuation and test the system by placing agents at significant distances from the transmitter. Our system achieves at least a 20% improvement in success rate and a 20% reduction in both localization time and the number of actions taken. We further examine the system's scalability by training on various team sizes, demonstrating strong performance across all configurations.
- We conduct a drop-one ablation study by removing individual modules from RoLocMe. The study showed that λ-returns are important in dense reward. And the radio map estimation component proves crucial to the system, boosting agents' performance by at least 15% across different scenarios.

2 Related Work

Our work is most related to mobile sensing and agent coordination for source localization, including path planning-based and reinforcement learning-based approaches.

2.1 Path Planning-based Approaches

Early studies on source localization focused on path-planning, where an agent follows a fixed trajectory—such as Hilbert curves, spiral curves, LMAT trajectories, or Z-curves [Koutsonikolas *et al.*, 2006; Hu *et al.*, 2008; Jiang *et al.*, 2011; Rezazadeh *et al.*, 2014]—until reaching the target or within a threshold. Although these deterministic paths are straightforward to implement, they often lack flexibility and can be challenging to optimize in complex environments.

2.2 Reinforcement Learning-based Approaches

Recent research has shifted toward data-driven reinforcement learning (RL) approaches, which are commonly categorized into single-agent and multi-agent methods.

Single-Agent Approaches
A study introduced multimodel Q Learning, which employs a tabular Q-learning algorithm with pattern recognition on sampled measurements to locate the source [Wu, 2019]. Another approach, PC-DQN [Zhao *et al.*, 2022], applies DBSCAN to sampled measurements before using Deep Q-learning (DQN) [Mnih, 2013], thus enabling efficient searches and demonstrating strong transferability to related tasks. Other work involves training an agent on a predefined trajectory and then fine-tuning its policy with DQN [Ebrahimi *et al.*, 2021]. Meanwhile, source measurements can be transformed into two-dimensional matrices and processed by Double Deep Q-learning [Liu and Abbaszadeh, 2019]. Finally, RAD-A2C combines Proximal Policy Optimization (PPO) [Schulman *et al.*, 2017] with a recurrent neural network to refine search policies [Proctor *et al.*, 2021].

Multi-Agent Approaches
The OMDTL method introduces a novel observation design and a team-based reward function, relying on PPO to foster cooperative agent behaviors [Alagha *et al.*, 2022]. Building on OMDTL, two enhanced methods, MDRL SR and MDRL DC [Alagha *et al.*, 2023], both employ an autoencoder for environment compression to augment observations during PPO training, thereby improving agents' performance. While MDRL SR uses dense rewards, MDRL DC leverages expert demonstrations under sparse rewards. However, neither MDRL SR nor MDRL DC addressed complex environments, and the agents' observation can be further simplified. Another approach proposes multiple reward functions and employs a similar PPO-based framework for multi-agent source localization [Wickramaarachchi *et al.*, 2024].

Our work focuses on multi-agent source localization. We primarily compare our approach with [Alagha *et al.*, 2023] due to the similarity of the problem setting. In contrast, while [Wickramaarachchi *et al.*, 2024] tackles a comparable multi-agent task, their sensing strategy allows agents to sample multiple values in a small area around them rather than relying on individual readings at specific positions, making a direct comparison less applicable to our setup.

3 Proposed RoLocMe System

This section introduces RoLocMe's system design, integrating a multi-agent reinforcement learning strategy with SkipNet, a deep learning radio signal prediction model.

3.1 Problem Definition and Modeling

Consider a geographic Region of Interest (RoI) discretized into a two-dimensional grid of $\mathcal{H} \times \mathcal{W}$, where $\mathcal{H}$ and $\mathcal{W}$ are the height and width of the grid, respectively, with $(\mathcal{H}, \mathcal{W} \in \mathbb{Z}^+)$. A set of agents is deployed on random cells throughout this RoI. Each agent can choose from nine possible actions at each time step: idle or move between cells in one of the eight directions (up, down, left, right, or any of the four diagonals).

Additionally, each agent can observe Received Signal Strength (RSS) at its current cell, know the whole environment, and communicate with other agents. The goal is to efficiently locate and approach the unknown transmitter within a distance of x cells.

Based on the defined problem above, we model it as Partially Observable Markov Decision Process (POMDP) [Oliehoek *et al.*, 2016] because the agents cannot observe RSSs of the entire environment and can only rely on the limited samples based on itself and other agents. POMDP can be described as a tuple $< \mathcal{N}, \mathcal{S}, \mathcal{A}, \oplus, \mathcal{P}, r, \gamma >$, where:

- $\mathcal{N} = \{1 \ , ..., \ n\}$ denotes the set of n agents (robots) where $\mathcal{N} \geq 1$.
- $\mathcal{S} = \{s_1, \ ..., \ s_k\}_{k \in \mathbf{Z}^+}$ denotes the set of finite states.
- $\mathcal{A} = \underbrace{A \times A \times \cdots \times A}_{n}$ is the set of joint actions, where A is the set of actions for each agent to choose, while $\mathbf{a}_t = (a_t^1, ..., a_t^n) \in \mathcal{A}$ denotes a joint actions at time step t.
- $\oplus = \underbrace{\Phi \times \Phi \times ... \times \Phi}_{n}$ is the set of joint observations, where Φ is the finite set of observations of each agent, and $\phi_t = (\phi_t^1, ..., \phi_t^n) \in \oplus$ is a joint observations at time step t.
- $\mathcal{P}(s_{t+1}, \phi_{t+1} | s_t, \mathbf{a}_t) \in [0, 1]$ denotes the probability that at step t, taking joint actions $\mathbf{a}_t$ in state s_t results in a transition to state s_{t+1} and joint observations ϕ_{t+1}.
- $\mathcal{R}(s, a) : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$ is a reward function that outputs the team reward when taking joint action $\mathbf{a}$ at state s. $\mathcal{R}(s, a)$ is from hence forward referred to as r_t for the rest of the paper.
- $\gamma \in [0, 1]$ is the discount factor.

At each time step, every agent i observes $\phi_t^i \in \Phi$ and selects an action $a_t^i \in A$ according its policy, denoted by π^i. Once all agents have acted simultaneously, they collectively receive a team reward r_t. Each agent's objective is to maximize the team's expected cumulative reward over the entire episode.

3.2 RoLocMe System Overview

RoLocMe addresses source localization by integrating two deep-learning models: SkipNet [Locke *et al.*, 2023] and the VDN [Sunehag *et al.*, 2017]. SkipNet is a pre-trained signal map estimator that enhances each agent's observations by providing a best effort understanding of signal propagation across the entire RoI. On the other hand, VDN, leveraging DQN architecture and auxiliary reinforcement learning components, facilitates the coordination of multiple agents for collaborative source localization. In training and inference, all agents utilize the same trained model, but each acts independently based on its observation.

As shown in Figure 1, RoLocMe begins by constructing shared observations that include a building map (updated with all agents' visited positions) and sampled signal points to date from the entire region of interest. SkipNet processes these shared observations to produce a refined signal distribution map. The resulting map is then combined with the shared observations and downsized to form individual observations ϕ_t^i for each agent based on its current position. Guided by ϕ_t^i, agents use an ϵ-greedy strategy to select joint actions a_t^i and explore new cells. As agents gather new signal samples, the shared observations are updated, allowing SkipNet to refine its predictions and help agents navigate more effectively iteratively. This process repeats until the agents exhaust the search time or when an agent reaches the source.

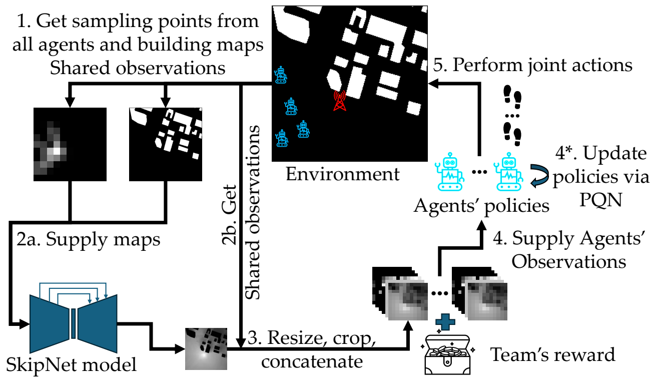
1. Get sampling points from all agents and building maps
Shared observations
5. Perform joint actions
Environment
4*. Update policies via PQN
Agents' policies
2a. Supply maps
2b. Get Shared observations
4. Supply Agents' Observations
3. Resize, crop, concatenate
Team's reward
SkipNet model

Figure 1: RoLocMe workflow: (1) Obtaining shared observations, (2) SkipNet predicts the RSS map, (3) Create Agents' observations from shared observations and SkipNet prediction, (4) Supply observations to the Agents and Agents select actions, and (5) Agents perform action and transition occurs until termination.

3.3 SkipNet

SkipNet [Locke *et al.*, 2023] plays a critical role in the success of RoLocMe by leveraging shared observations collected from all visited positions and sampled points across agents. It estimates the signal distribution for the entire RoI. Inspired by the autoencoder architecture proposed in [Teganya and Romero, 2021], SkipNet introduces skip-connections between the encoder and decoder, which resembles a U-Net architecture as shown in Figure 2. These connections minimize information loss during encoding, especially when sampling points are sparse, and enhance gradient flow between the decoder and encoder. This design facilitates efficient and unrestricted information transfer during backpropagation, improving overall estimation.

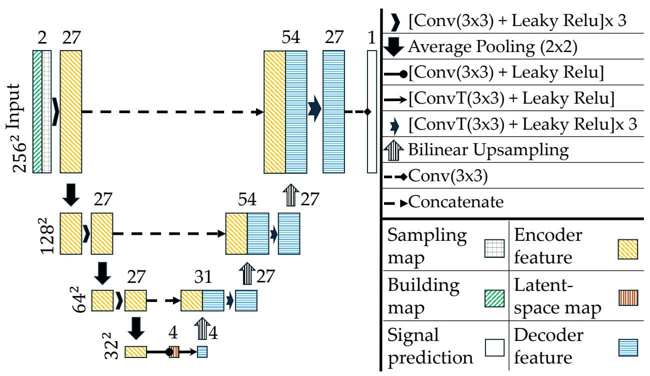

Figure 2: SkipNet architecture.

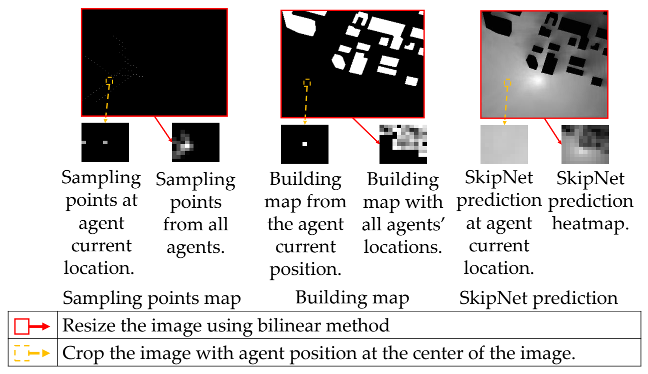
Sampling points at agent current location.
Sampling points from all agents.
Building map from the agent current position.
Building map with all agents' locations.
SkipNet prediction at agent current location.
SkipNet prediction heatmap.
Sampling points map
Building map
SkipNet prediction
Resize the image using bilinear method
Crop the image with agent position at the center of the image.

Figure 3: Top: Shared observations across the RoI. Bottom: Downsized local and global views for each agent.

3.4 Agent Observation Preprocessing

As illustrated in Figure 3, each agent utilizes two key components to construct its observation:

- **Shared Observations** includes two maps: the **Sampling Points Map**, which aggregates RSS readings collected over time and scaled between $[0, 1]$, the **Building Map**, which identifies building pixels (-1), visited cells (1), and unvisited cells (0).
- **SkipNet Estimation**: A predicted signal propagation map generated by SkipNet, scaled between $[0, 1]$.

Shared observations enhance agents' awareness of each other's positions and provide source estimations based on sampled points. SkipNet generates a best effort signal propagation map for the entire RoI, improving agents' understanding of the signal distribution. An agent i^{th} observation, ϕ_t^i, combines shared observations with SkipNet predictions, downsized through bilinear interpolation, and cropped around the agent's position to form a stack of six $(d \times d)$ maps. This downsizing and cropping, shown to improve convergence without compromising performance [Alagha *et al.*, 2022], effectively balances global context with local focus.

3.5 Agent Architecture

Figure 4 illustrates the high-level design for our multi-agent system and a detailed view of each agent's neural network architecture. We employ VDN [Sunehag *et al.*, 2017] as a general design for tackling multi-agent cooperatives, Dueling DQN [Wang *et al.*, 2016] for the agent's network design, λ-returns [Sutton and Barto, 2018], and [Gallici *et al.*, 2024] suggestions for stabilize DQN in training.

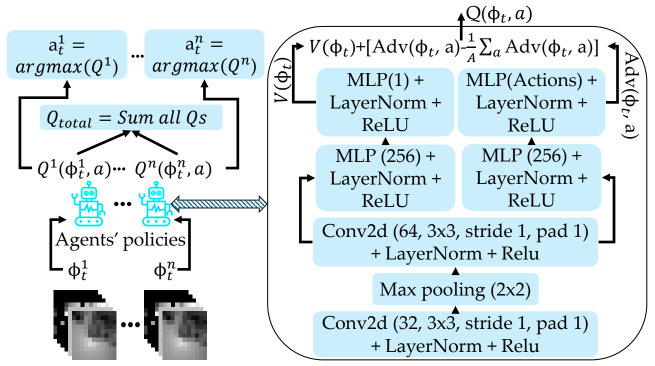
$Q(\phi_t, a)$
$a_t^1 = argmax(Q^1)$
...
$a_t^n = argmax(Q^n)$
$Q_{total} = Sum\ all\ Qs$
$Q^1(\phi_t^1, a)$ ... $Q^n(\phi_t^n, a)$
Agents' policies
ϕ_t^1
ϕ_t^n
$V(\phi_t)+[Adv(\phi_t, a)-\frac{1}{A}\sum_a Adv(\phi_t, a)]$
$V(\phi_t)$
$Adv(\phi_t, a)$
MLP(1) + LayerNorm + ReLU
MLP(Actions) + LayerNorm + ReLU
MLP (256) + LayerNorm + ReLU
MLP (256) + LayerNorm + ReLU
Conv2d (64, 3x3, stride 1, pad 1) + LayerNorm + Relu
Max pooling (2x2)
Conv2d (32, 3x3, stride 1, pad 1) + LayerNorm + Relu

Figure 4: Agent network architecture.

Value Decomposition Network

Value Decomposition Networks (VDN) [Sunehag *et al.*, 2017] provides a straightforward yet powerful approach for coordinating multiple agents. Let θ denotes the network parameters in VDN. VDN assumes the team's optimal Q-value, $Q^{\text{team},\theta}$, is the sum of each individual agent's Q-value, $\widetilde{Q}^{i,\theta}$. Formally:

$$Q^{\text{team},\theta}(\phi_t, a_t) = \sum_{i=0}^{n-1} \widetilde{Q}^{i,\theta}(\phi_t^i, a_t^i). \quad (1)$$

Agents can then greedily select actions from their local Q-values to maximize $Q^{\text{team},\theta}$. During training, the agents share a common reward and update their local Q-values by back-propagating the team's Q-value via Q-learning.

Deep Q-learning

DQN [Mnih, 2013] utilize deep neural network to estimate the action's Q-value directly from raw inputs and update θ according to:

$$\theta \leftarrow \theta + \alpha\big[r_t + \gamma \max_{a_t} Q(s_{t+1}, a_{t+1}; \theta) - Q(s_t, a_t; \theta)\big] * \nabla_\theta Q(s_t, a_t; \theta). \quad (2)$$

To stabilize learning, DQN employs a large experience replay buffer (for randomized updates) and a target network (updated at a slower pace) to reduce oscillations.

Auxiliary Components

Dueling DQN. Dueling DQN [Wang *et al.*, 2016] separates Q-value estimation in DQN into two distinct streams: the *Value* stream (V), which estimates the overall value of a state, and the *Advantage* stream (Adv), which captures the relative benefit of each action in that state. The Q-value is computed as:

$$Q(s_t, a_t) = V(s_t) + \Big(Adv(s_t, a_t) - \frac{1}{Acts}\sum_{a_{t+1}} Adv(s_t, a_{t+1})\Big) \quad (3)$$

, where s_t represents the state, a_t is the current action, a_{t+1} is the action in next state, and $Acts$ is the number of actions. This design mitigates the overestimation of action values and clarifies the state value independently of specific actions.

Parallel Q-learning - PQN. DQN [Mnih, 2013] relies on large replay buffers and a target network for stability but can be slow or prone to divergence without them. Parallel Q-learning [Gallici *et al.*, 2024] tackles this by adding layer normalization to the network and training over multiple vectorized environments in parallel. These changes allow DQN to achieve competitive performance—comparable to policy-gradient methods like PPO [Schulman *et al.*, 2017]—without large replay buffers and a target network.

$\lambda -$ returns. $\lambda -$ returns [Sutton and Barto, 2018] is a target estimation method in reinforcement learning that utilizes bootstrapping, state samples, and rewards. This method weighs short-term and long-term returns by combining n-step returns with the λ parameter, suitable for improving training stability in a partially observable task, as shown in Equation 4. A λ value near 0 favors immediate rewards. In contrast, a value near 1 prioritizes long-term returns.

$$R_t^\lambda = R_t^1 + \gamma\lambda\Big[R_{t+1}^\lambda - \max_{a_{t+1}} Q_\theta(\phi_{t+1}, a_{t+1})\Big],$$
$$\text{where } R_t^1 = r_t + \gamma \max_{a'} Q(\phi_{t+1}, a'). \quad (4)$$

Reward Design

We investigate the system performance on two reward definitions—*dense* and *sparse*— [Alagha *et al.*, 2023]. Breadth-First Search (BFS) distances between agents and the transmitter are computed at each step.

- **Dense rewards:** Let $\mathbf{M}_t$ be the set of agent-to-transmitter distances at time t and $\min(\mathbf{M}_t)$ their minimum. Let m be the number of agents that moved at time t. Then,

$$r_t = \begin{cases} -m+1, & \text{if } \min(\mathbf{M}_{t+1}) < \min(\mathbf{M}_t), \\ -m-1, & \text{otherwise.} \end{cases} \quad (5)$$

- **Sparse rewards:**

$$r_t = \begin{cases} -m+100, & \text{if transmitter is localized,} \\ -m, & \text{otherwise.} \end{cases} \quad (6)$$

Termination Conditions

An episode terminates when an agent comes within L meters of the transmitter, marking success or when the maximum step limit is reached, indicating failure.

Training Process

Algorithm 1 details the entire procedure. After each update cycle, multiple validation episodes across diverse settings are conducted, and the best model is saved based on success rates. Before integrating into RoLocMe for agent training, SkipNet undergoes pre-training using mean squared error (MSE) loss.

4 Experiments and Results

This section presents experiments conducted to evaluate the RoLocMe system in comparison with the state-of-arts methods and provides an in-depth analysis of the significance of each component in its design.

Algorithm 1 RoLocMe Training

```
1: $\theta \leftarrow$ Initialize network parameters
2: $TSC \leftarrow 0$;  $t \leftarrow 0$                         ▷ TSC = Training step count
3: SkipNet $\leftarrow$ Load the pretrained SkipNet model
4: Shared_Obs$_t$ $\leftarrow$ Env_Reset                        ▷ this t is 0
5: SkipNet_Pred$_t$ $\leftarrow$ SkipNet(Shared_Obs$_t$)
6: Fuse_Obs $\leftarrow$ Concat(SkipNet_Pred$_t$, Shared_Obs$_t$)
7: $\phi_t \leftarrow$ Preprocessing_Combine_Obs(Fuse_Obs$_t$)   ▷ $\phi_t = [\phi_t^0, \dots, \phi_t^{n-1}]$
8: while $TSC \leq$ Max Training Steps do
9:     for each env $e \in \{0 : N_{\text{env}} - 1\}$ in parallel do
10:        for each agent $i \in \{0 : n-1\}$ do
11:            $a_t^i \leftarrow \pi_{\epsilon\text{-greedy}}(\phi_t^i)$
12:        end for
13:        $\mathbf{a}_t = [a_t^0, \dots, a_t^{n-1}]$
14:        Shared_Obs$_{t+1}$, $r_t$, $terminal_t$ $\leftarrow$ Env_Step$_e$($\mathbf{a}_t$)
15:        SkipNet_Pred$_{t+1}$ $\leftarrow$ SkipNet(Shared_Obs$_{t+1}$)
16:        Fuse_Obs $\leftarrow$ Concat(SkipNet_Pred$_{t+1}$, Shared_Obs$_{t+1}$)
17:        $\phi_{t+1} \leftarrow$ Preprocessing_Combine_Obs(Fuse_Obs$_{t+1}$)   ▷ $\phi_{t+1} = [\phi_{t+1}^0, \dots, \phi_{t+1}^{n-1}]$
18:        Store ($\phi_t, \mathbf{a}_t, \phi_{t+1}, r_t, terminal_t$) in buffer
19:        $t \leftarrow t + 1$
20:    end for
21:    if $t \mod T = 0$ then                                   ▷ Start an update cycle
22:        for epoch $\in \{1 : \text{num epochs}\}$ do
23:            Shuffle buffer by environment trajectories
24:            Create mini-batches grouped by environment
25:            for each mini-batch - mb do
26:                Compute $R_{[t-T:t]}^{\lambda,mb}$ using Equation (4)
27:                Compute $Q^{\text{team},\theta}$ using Equation (1)
28:                Perform gradient descent on
29:                $||\text{StopGrad}(R_t^{\lambda,mb}) - Q^{\text{team},\theta}(\phi_t, a_t)||^2$
30:                with respect to the network parameters $\theta$
31:            end for
32:        end for                                              ▷ End an update cycle
33:        Clear buffer
34:    end if
35:    $TSC \leftarrow TSC + N_{\text{env}}$
36: end while
```

4.1 Dataset

We use the RadioMapSeer dataset [Yapar *et al.*, 2022], which consists of 701 environment maps sourced from OpenStreetMap [OpenStreetMap, 2023] and represents various areas across major European cities. Each map features 80 transmitter locations, resulting in 80 distinct signal propagation maps generated using the Dominant Path Model method in the WinProp program [Hoppe *et al.*, 2017]. We divide the dataset by environment, allocating 501 maps for training, 100 for validation, and 100 for testing.

- **SkipNet dataset:** We created a low-sampling dataset by generating five sampled signals for each propagation map in the training and validation sets using a uniform sampling method with sampling rates between 0.01% and 0.1% of the total pixels.
- **RoLocMe dataset:** Randomly selected 100 environments each from the training and validation sets, with 5 transmitter placements per environment, totaling 1,000 unique transmissions for end-to-end GPU training. Final results use all test maps.

4.2 Methods

Q-Learning hyperparameters	Values
Learning rate	5e−4
Gradient clipping by global normalization	100
Discount factor (γ)	0.99
Decay rate (λ)	0.65
Number of epochs (num epochs)	2
Epsilon (ϵ)	0.99
Epsilon decay rate	4e−4
Number of parallel environments (N_{env})	128
Mini-trajectory size (T)	100
Mini batch size	16
Number of mini-batches (mb)	8
Number of validation episodes	100
Agent's observations input size (d)	7
Optimizer ([Liu *et al.*, 2019])	RAdam
Environment training parameters	**Values**
Max Training Steps	5e+7
Maximum step per episode	100
Meters per step	5
Terminate distance (L)	5
Distance deployments of each agent	100 and above
SkipNet hyperparameters	**Values**
Learning rate	5e-4
Optimizer ([Kingma, 2014])	Adam
LeakyRelu's α	0.3

Table 1: Hyperparameters used for training RoLocMe and SkipNet

We introduce the state-of-art models [Alagha *et al.*, 2023] used for performance comparison:

- MDRL SR-Dense rewards: Multi-agents model trained with Proximal Policy Gradient with dense rewards
- MDRL SR-Sparse rewards: Multi-agents model trained with Proximal Policy Gradient with sparse rewards
- MDRL DC: MDRL SR-Sparse rewards with a guided expert model

To ensure consistency, all comparisons are conducted using a team of four agents and evaluated under the same testing conditions. Each method is trained with 10 different seeds, and performance is recorded across 100 unique starting positions for each proximity zone within every signal propagation map in the testing set. This results in a total of 800,000 starting positions per promimity zone. To achieve a more robust estimate and reduce the impact of outliers, the interquartile mean is used for evaluation [Agarwal *et al.*, 2021]. Table 1 presents the parameters used during the training of RoLocMe.

4.3 Metrics

We introduce three key metrics to evaluate the performance:

- **Success rate**: Success rate is the percentage of successful localizations, as defined in Equation 7.

$$\text{Success Rate}(\%) = \frac{\text{\#locate transmitter within 5 meters}}{\text{\#trials}} \tag{7}$$

- **Localization time**: how long a team of agents can successfully localize the transmitter.
- **Actions**: total number of moves of the agents in the team. The lower the actions, the better the energy efficiency, as each movement costs some energy.

4.4 Performance Comparison with State of Arts Methods

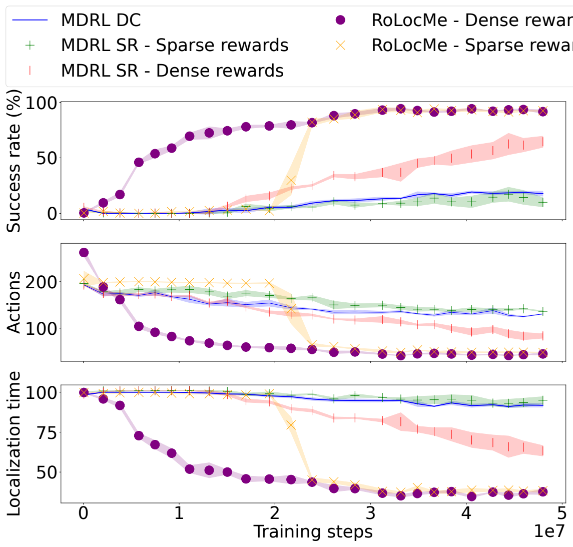

Figure 5: Convergence time of each metric in training 4 agents.

Figure 5 illustrates the convergence rate of each model across various metrics during training. RoLocMe, under both rewarding systems, consistently converges faster than the other models. Among the compared models, MDRL with dense rewards demonstrates slower convergence than RoLocMe.

In Figure 6, as the agents' starting positions move further from the transmitter, the performance of all models degrades. However, RoLocMe consistently outperforms the other models, achieving at least a 30% improvement in the first two proximity groups and approximately 20% in the last two groups. Furthermore, although the number of actions and localization times increases across all methods as proximity increases—due to the longer traversal paths—RoLocMe maintains superior performance compared to the other models.

Additionally, RoLocMe in both rewards functions perform quite similarly. However, there is a slight increase in localizing time and actions taken roughly 3%, which is insignificant. Based on the performance, training RoLocMe with dense reward often yields a slightly better policy than sparse reward. However, using dense or sparse rewards for RoLocMe depends on whether performing repeated BFS calculations for every step during training is feasible.

4.5 Drop-One Component Ablation Study

In Figure 7, RoLocMe's full configuration converges faster and more consistently than its ablation variants (i.e., without λ-returns, layer normalization, dueling, or SkipNet) in both

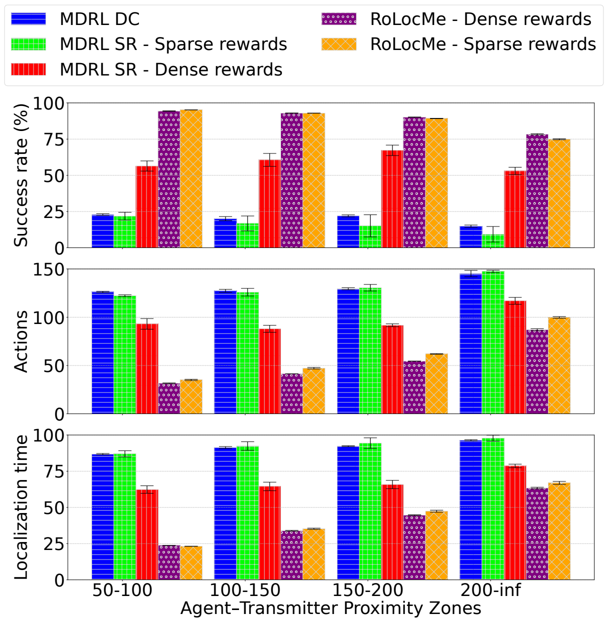

Figure 6: Performance benchmark of 4 agents of RoLocMe with other RL models in the testing set.

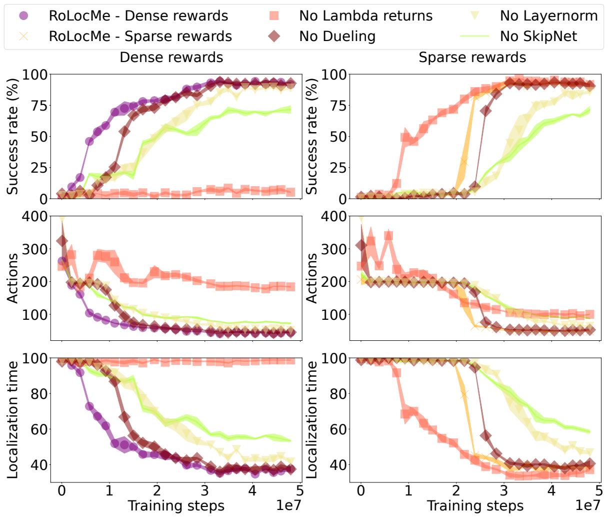

Figure 7: Convergence time of each drop-one RoLocMe's component in two different reward functions in training 4 agents.

dense and sparse reward settings. These ablated models converge more slowly, especially under sparse rewards, where their early success rates remain near zero for a considerable period. By contrast, RoLocMe trained with dense rewards rapidly achieves near-100% success, requiring fewer actions and shorter localization times.

Figure 8 displays each ablation's performance under both reward functions across various proximity zones. In all cases, RoLocMe maintains higher success rates, shorter localization times, and fewer actions—especially when the agent starts far from the transmitter. While $\lambda-$ return is also important to the system, removing SkipNet erodes the agents' ability to learn

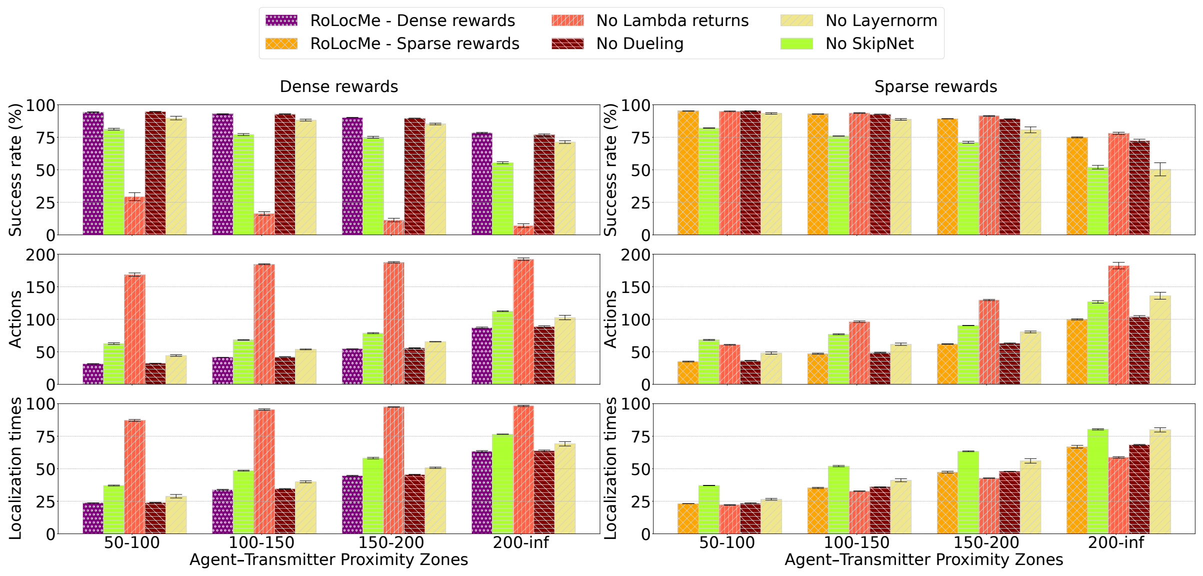

Figure 8: Performance of each drop-one RoLocMe component in the testing set for dense rewards and sparse rewards in training 4 agents.

the dynamic of signal propagation, leading to significantly degraded performance. However, the complete RoLocMe design generalizes effectively across all proximities.

Under dense rewards, omitting dueling or layer normalization produces success rates similar to RoLocMe but requires about 2% more time and actions. In this dense reward setting, successes are not distinctly marked and can be obscured by smaller rewards or failures, making λ-returns crucial for properly reinforcing the final objective; without λ-returns, the success rate drops by at least 75%, with localization times tripling and actions increasing by at least 1.5 times compared to the full RoLocMe. Another key component is SkipNet: eliminating it deprives the agent of the ability to learn the dynamic of signal propagation in the RoI, causing success rates to fall by about 15% and time and action costs to rise by roughly 20%.

In the sparse reward setting, models lacking dueling, λ-returns, or layer normalization initially achieve comparable success rates within the first three proximity zones; however, the model without λ-returns substantially increases localization time and actions. In the absence of SkipNet, the success rate diminishes, and both localization time and actions increase significantly, reflecting patterns seen in the dense-reward case. Generally, models missing SkipNet have lower performance and require more time and action to search.

4.6 RoLocMe Scalability on Various Team Sizes

Figure 9 illustrates the performance of the RoLocMe system across various team sizes, ranging from 2 to 16 agents and different proximity groups. For closer proximities (50–100 and 100–150), success rates remain near 100% regardless of team size. However, as proximities increase (150–200 and beyond), larger teams show a decline in performance. While larger teams can cover greater distances more effectively, the increased coordination overhead negatively impacts success rates, leading to higher localization times and more executed actions. We suspect this limitation arises from using VDN, which struggles to optimize policies for large groups of agents.

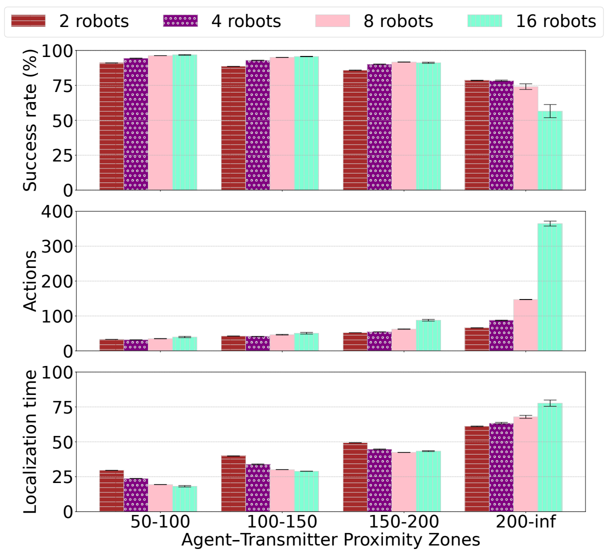

Figure 9: Performances of RoLocMe in various numbers of agents.

5 Conclusion

In this paper, we present RoLocMe, a system comprising two deep learning models that complement each other for source localization tasks. By fusing the signal estimation from the SkipNet model into the agents' observations and designing the agents via VDN and DQN, we observed a notable performance boost compared to existing methods. Additionally, we performed an ablation study to illustrate how each component contributes to the agents' performance across various metrics.

Acknowledgments

Research was partially sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-23-2-0014. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

[Agarwal *et al.*, 2021] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

[Alagha *et al.*, 2022] Ahmed Alagha, Shakti Singh, Rabeb Mizouni, Jamal Bentahar, and Hadi Otrok. Target localization using multi-agent deep reinforcement learning with proximal policy optimization. *Future Generation Computer Systems*, 136:342–357, 2022.

[Alagha *et al.*, 2023] Ahmed Alagha, Rabeb Mizouni, Jamal Bentahar, Hadi Otrok, and Shakti Singh. Multiagent deep reinforcement learning with demonstration cloning for target localization. *IEEE Internet of Things Journal*, 10(15):13556–13570, 2023.

[Atif *et al.*, 2021] Muhammad Atif, Rizwan Ahmad, Waqas Ahmad, Liang Zhao, and Joel J. P. C. Rodrigues. Uav-assisted wireless localization for search and rescue. *IEEE Systems Journal*, 15(3):3261–3272, 2021.

[Ebrahimi *et al.*, 2021] Dariush Ebrahimi, Sanaa Sharafeddine, Pin-Han Ho, and Chadi Assi. Autonomous uav trajectory for localizing ground objects: A reinforcement learning approach. *IEEE Transactions on Mobile Computing*, 20(4):1312–1324, 2021.

[Gallici *et al.*, 2024] Matteo Gallici, Mattie Fellows, Benjamin Ellis, Bartomeu Pou, Ivan Masmitja, Jakob Nicolaus Foerster, and Mario Martin. Simplifying deep temporal difference learning. *arXiv preprint arXiv:2407.04811*, 2024.

[Hoppe *et al.*, 2017] Reiner Hoppe, Gerd Wölfle, and Ulrich Jakobus. Wave propagation and radio network planning software winprop added to the electromagnetic solver package feko. In *International Applied Computational Electromagnetics Society Symposium - Italy (ACES)*, pages 1–2, 2017.

[Hu *et al.*, 2008] Zhen Hu, Dongbing Gu, Zhengxun Song, and Hongzuo Li. Localization in wireless sensor networks using a mobile anchor node. In *2008 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 602–607, 2008.

[Jiang *et al.*, 2011] Jinfang Jiang, Guangjie Han, Huihui Xu, Lei Shu, and Mohsen Guizani. Lmat: Localization with a mobile anchor node based on trilateration in wireless sensor networks. In *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, pages 1–6, 2011.

[Kingma, 2014] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Koutsonikolas *et al.*, 2006] D. Koutsonikolas, S.M. Das, and Y.C. Hu. Path planning of mobile landmarks for localization inwireless sensor networks. In *26th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'06)*, pages 86–86, 2006.

[Lin *et al.*, 2022] Meiyan Lin, Yonghui Huang, Baozhu Li, Zhen Huang, Zihan Zhang, and Wenjie Zhao. Deep learning-based multiple co-channel sources localization using bernoulli heatmap. *Electronics*, 11(10), 2022.

[Liu and Abbaszadeh, 2019] Zheng Liu and Shiva Abbaszadeh. Double q-learning for radiation source detection. *Sensors*, 19(4), 2019.

[Liu *et al.*, 2019] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

[Locke *et al.*, 2023] William Locke, Nikita Lokhmachev, Yan Huang, and Xinrong Li. Radio map estimation with deep dual path autoencoders and skip connection learning. In *2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1–6, 2023.

[Lu *et al.*, 2014] Qiang Lu, Qing-Long Han, and Shirong Liu. A finite-time particle swarm optimization algorithm for odor source localization. *Information Sciences*, 277:111–140, 2014.

[Mnih, 2013] Volodymyr Mnih. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[Morelande *et al.*, 2007] Mark Morelande, Branko Ristic, and Ajith Gunatilaka. Detection and parameter estimation of multiple radioactive sources. In *2007 10th International Conference on Information Fusion*, pages 1–7, 2007.

[Oliehoek *et al.*, 2016] Frans A Oliehoek, Christopher Amato, et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.

[OpenStreetMap, 2023] OpenStreetMap, 2023.

[Proctor *et al.*, 2021] Philippe Proctor, Christof Teuscher, Adam Hecht, and Marek Osiński. Proximal policy optimization for radiation source search. *Journal of Nuclear Engineering*, 2(4):368–397, 2021.

[Raj and B. S., 2023] Nibin Raj and Vineeth B. S. Indoorrssinet - deep learning based 2d rssi map prediction for indoor environments with application to wireless localization. In *2023 15th International Conference on COMmunication Systems NETworkS (COMSNETS)*, pages 609–616, 2023.

[Rezazadeh *et al.*, 2014] Javad Rezazadeh, Marjan Moradi, Abdul Samad Ismail, and Eryk Dutkiewicz. Superior path planning mechanism for mobile beacon-assisted localization in wireless sensor networks. *IEEE Sensors Journal*, 14(9):3052–3064, 2014.

[Rybski *et al.*, 2000] Paul E. Rybski, Sascha A. Stoeter, Michael D. Erickson, Maria Gini, Dean F. Hougen, and Nikolaos Papanikolopoulos. A team of robotic agents for surveillance. In *Proceedings of the Fourth International Conference on Autonomous Agents*, page 9–16, New York, NY, USA, 2000. Association for Computing Machinery.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[Sharma *et al.*, 2020] Amit Sharma, Pradeep Kumar Singh, and Yugal Kumar. An integrated fire detection system using IoT and image processing technique for smart cities. *Sustainable Cities and Society*, 61:102332, 2020.

[Sunehag *et al.*, 2017] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

[Sutton and Barto, 2018] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[Teganya and Romero, 2021] Yves Teganya and Daniel Romero. Deep completion autoencoders for radio map estimation. *IEEE Transactions on Wireless Communications*, 21(3):1710–1724, 2021.

[Wang *et al.*, 2016] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003, 2016.

[Wang *et al.*, 2022] Wenyu Wang, Lei Zhu, Zhen Huang, Baozhu Li, Lu Yu, and Kaixin Cheng. Mt-gcnn: Multi-task learning with gated convolution for multiple transmitters localization in urban scenarios. *Sensors*, 22(22), 2022.

[Wickramaarachchi *et al.*, 2024] Helani Wickramaarachchi, Michael Kirley, and Nicholas Geard. Cooperative multi-agent reinforcement learning with dynamic target localization: A reward sharing approach. In *AI 2023: Advances in Artificial Intelligence*, pages 310–324, Singapore, 2024. Springer Nature Singapore.

[Wu, 2019] Guangyu Wu. Uav-based interference source localization: A multimodal q-learning approach. *IEEE Access*, 7:137982–137991, 2019.

[Yapar *et al.*, 2022] Çağkan Yapar, Ron Levie, Gitta Kutyniok, and Giuseppe Caire. Dataset of pathloss and ToA radio maps with localization application. *arXiv preprint:2212.11777*, 2022.

[Zhan *et al.*, 2021] Caitao Zhan, Mohammad Ghaderibaneh, Pranjal Sahu, and Himanshu Gupta. Deepmtl: Deep learning based multiple transmitter localization. In *2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 41–50, 2021.

[Zhang *et al.*, 2016] Wei Zhang, Kan Liu, Weidong Zhang, Youmei Zhang, and Jason Gu. Deep neural networks for wireless localization in indoor and outdoor environments. *Neurocomputing*, 194:279–287, 2016.

[Zhao *et al.*, 2022] Yong Zhao, Bin Chen, XiangHan Wang, Zhengqiu Zhu, Yiduo Wang, Guangquan Cheng, Rui Wang, Rongxiao Wang, Ming He, and Yu Liu. A deep reinforcement learning based searching method for source localization. *Information Sciences*, 588:67–81, 2022.

[Zubow *et al.*, 2020] Anatolij Zubow, Suzan Bayhan, Piotr Gawłowicz, and Falko Dressler. Deeptxfinder: Multiple transmitter localization by deep learning in crowdsourced spectrum sensing. In *2020 29th International Conference on Computer Communications and Networks (ICCCN)*, pages 1–8, 2020.