Fully-Decentralised Multi-agent Reinforcement Learning in Fog-based Internet-of-Things

XXX YYYY, and XXX ZZZZ

***Abstract*—Reinforcement learning (RL) algorithms**

***Index Terms*—Reinforcement learning (RL), Internet-of-Things (IoT), Fog-based IoT, Markov decision process (MDP), machine learning (ML).**

I. Background

THE rapid increase in the use of wireless devices and technologies for Internet-of-Things (IoT) applications, such as Industrial IoT, military surveillance, health monitoring, and smart cities, has brought about an increasing need for frameworks on connectivity prediction, information control, and failure analysis to improve network reliability [1]. Interestingly, with the proliferation of more IoT devices, which widely make use of the unlicensed industrial, scientific, and medical (ISM) frequency band, it is anticipated that the shared frequency spectrum will become more crowded as more and more devices will contend for the wireless medium, resulting poor network performance and increased application failures.

Fog computing platform extends the cloud by bring processing, computation, decision making and control closer to the edge of the network [2]. First, the IoT paradigm can effectively leverage on fog devices to deliver reliable network performance by minimizing communication outages caused by poor communication link between communicating parties, and also handle cross-technology interference (CTI) caused by heterogonously deployed IoT devices. Second, via local control, the fog devices can help in extending the longevity of the IoT network.

II. Problem Definition

In this section, we provide full description of the system model, as well the RL approach used to address the problem. The Mobile Fog Relay Agent (MFRA) and its environment are discussed below.

A. MFRA environment

States: The states are defined as a tuple, $\langle$Outage communication cost ($\mathcal{P}_{out}$) /Energy status of the fog relay (J) /Energy status of the IoT sensor (J)$\rangle$.

- Outage communication cost: Outage observations from the environment is estimated using (1) from [3], which gives an estimate of the communication outage when the agent takes an action, such as changing power levels or location, or both.
- Energy expended by fog relay: This observation gives the agent insight on how much energy by the fog agent when following policy $\omega_i \in \omega_{fog}$. If the fog agent continues to take sub-optimal actions, it depletes its energy and dies out.
- Energy expended by IoT sensor: This observation gives the agent insight on how much energy the IoT sensor has used up by following a policy $\omega_i \in \omega_{IoT}$. If the IoT sensor continues to take sub-optimal actions, it depletes its energy and dies out.

$$\mathcal{P}_{out} = 1 - (1 + 2\Psi^2 \ln \Psi) \exp\Big(- \frac{N_0 \tilde{\kappa}}{P_I (D_I + \delta)^{-\sigma}} \Big), \quad (1)$$

where $\Psi = \sqrt{(N_0\tilde{\kappa})/(P_R(D_S+\delta)^{-\sigma})}$, and $\mathcal{P}_{out}$ is an expression for the outage probability with values between 0 and 1. We assume a predefined threshold $\tilde{\kappa}$ which determines the outage in communication, P_I is transmit power of the IoT sensor, P_R is transmit power of the fog relay agent, D_I is the distance between IoT sensor and fog relay agent, and D_S is the distance between fog relay agent and destination node. We assume a small change in the position of the fog relay agent, $\delta = \pm 0.25m$, N_0 to be the channel noise, and σ to be the path-loss exponent.

B. MRFA agent

We apply a the Q-Learning algorithm, an RL approach which requires no prior knowledge of the environment by the agent. In Q-learning, the agent interacts with the environment over periods of time according to a policy ω. At every time-step $k \in N$, the environment produces an observation $s_k \in \mathbb{R}^{D_s}$. By sampling, the agent then picks an action a_k over $\omega(s_k)$, $a_k \in \mathbb{R}^{D_a}$, which is applied to the environment. The environment consequently produces a reward $r(s_k, a_k)$ and may end the episode at state s_N or transits to a new state s_{k+1}. The agent's goal is to minimize the expected cumulative cost, $\min_\omega \mathbb{E}_{s_0,a_0,s_1,a_1,\ldots,s_N}\Big[\sum_{i=0}^{N} \gamma^i \mathcal{C}(s_i)\Big]$, where $0 \leq \gamma \leq 1$ is the discount factor, and $\mathcal{C}$ is the overall cost function of our model.

First, the agent takes an initial random action a_k and gets observations from the environment which corresponds to that action, as well as a reward. It then discretizes the continuous observations emanating from the environment into a $50 \times 10 \times 10$ state space corresponding to the tuple, $\langle$Outage communication cost ($\mathcal{P}_{out}$) /Energy status of the fog relay (J) /Energy status of the IoT sensor (J)$\rangle$. The agent then updates it's Q-values at each time-step k following (3).

Manuscript received March 14, 2019; revised July X, 2019.

$$Q(s_k, a_k) := Q(s_k, a_k) + \alpha\Big[r_{k+1} + \gamma \max_a Q(s_{k+1}, a) - Q(s_k, a_k)\Big], \quad (2)$$

where α is the learning rate, which determines the impact of new experience on the Q-value, r_{k+1} is the reward the agent receives by being in s_{k+1} from s_k. Based on the policy followed by the agent, it gets observations and rewards from the environment.

Action space: The actions are move and transmit by fog relay, and select a power-level and transmit by IoT end-device, which make up eight possible actions.

- Mobility: Move by $\pm\delta$ and transmit, where $\delta = \pm 0.25m$ and mobility range (m) = [-30, 30]
- Power-level: Choose power-level and transmit, P, where transmit power ranges (W) = [0.001, 0.01, 0.15, 0.2, 0.25, 0.3]

Goal: The goal is for the agent to learn to minimize the overall cost $\mathcal{C}$ in the tuple, $\langle$Outage communication cost ($\mathcal{P}_{out}$) /Energy status of the fog relay (J) /Energy status of the IoT sensor (J)$\rangle$, by keeping all nodes within the link alive while ensuring that the packets received in each transmission does not fall below the pre-defined threshold, which was set at 95%.

Rewards: The reward function used is given in (3) as

$$R = \begin{cases} 100, & \text{if } goal == Reached \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

Metrics: Outage probability, i.e. the ratio of the number of packet lost to those transmitted, which we measure in percentages, and energy status of the fog relay agent and the IoT sensor, i.e. the ratio of depleted energy to the initial capacity in Joules, which we measure as a percentage.

The MFRA's learning process is summarized in Algorithm 1. A new learning episode is terminated when the agent attains the pre-defined goal of minimizing the communication outage in the link, or when either the fog relay or the IoT sensor dies out due to taking sub-optimal actions without getting to the goal. When an fog relay moves closer to the communicating parties, the IoT sensor uses a lower power level as compared to when it is far away, hereby saving IoT sensor energy. However, mobility have some cost and if the fog relay continues to move in order to minimize the communication outage, it may die out soon, hereby causing a point-of-failure to the network. As each episode is completed, a reward of 100 points is given to the agent if it reaches its goal and a 0 points otherwise. The reward is updated in the Q-learning table, with environmental information updated as well.

Algorithm 1 MFRA Learning Process

Initialize: Power levels (W) = [0.001, 0.01, 0.15, 0.2, 0.25, 0.3], $\delta = \pm 0.25m$ and mobility range (m) = [-30, 30]
top:
ResetEnvironment()
state ← MapLocalObservationToState(*env*)
action ← QLearning.SelectAction(*state*)
if *action* == "move close and Tx" **then**
    Env.EstimateOutage (1)
    Env.EstimateFogEnergyStatus
else if *action* == "move away and Tx" **then**
    Env.EstimateOutage (1)
    Env.EstimateSensorEnergyStatus
else if *action* == "choose power level and Tx" **then**
    Env.EstimateOutage (1)
    Env.EstimateSensorEnergyStatus
endif
InvokePolicy(ExponentialDecay)
UpdateQLearningProcedure() (3)
CurrentState ← NewState
if *goal* == "Reached" **then return** Reward = 100,
else if *goal* != "Reached" or Agent == Death **then return** Reward = 0
endif
EndEpisode **goto** *top*.

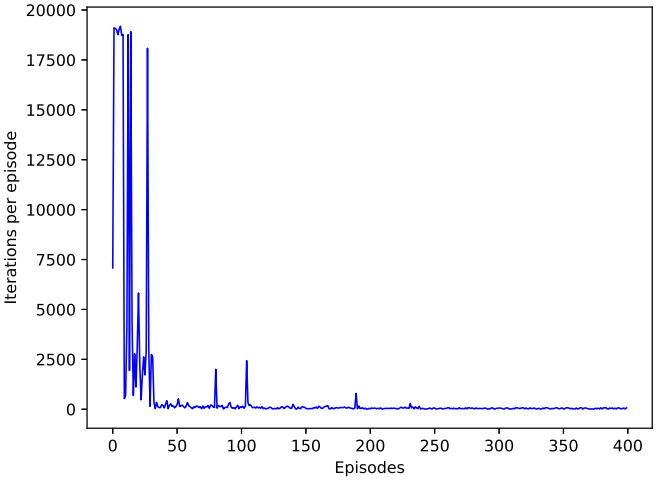

Fig. 1. Number of iteration over episodes prior to the point of failure.

III. Experimental Setup

A. General settings

We carried out experimentation with 150 fog relays and 1000 IoT sensors randomly deployed.

Table I shows a summary of the parameters used in simulating

B. Baselines

C. MFRA scenarios

D. Indicators

IV. Related Works

References

[1] U. Wetzker, I. Splitt, M. Zimmerling, C. A. Boano and K. Römer, "Troubleshooting Wireless Coexistence Problems in the Industrial Internet of Things," 2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous

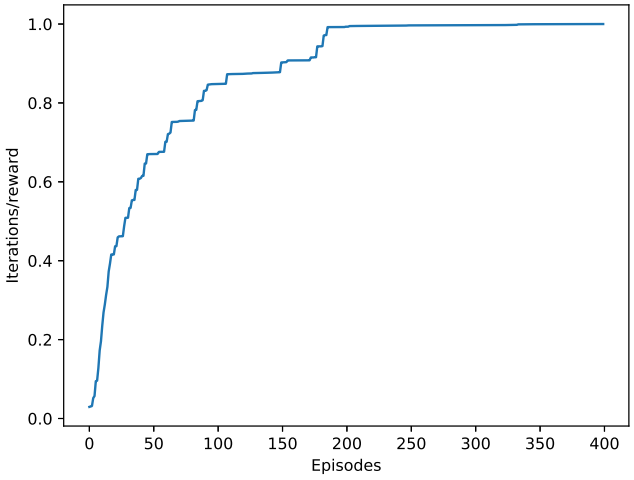

Fig. 2. Normalized Iterations per reward over 1000 episodes.

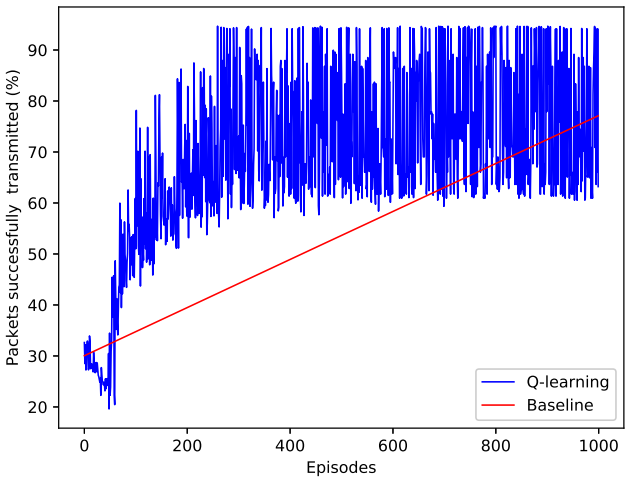

Fig. 3. Percentage of packets successfully transmitted via the fog agent.

Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES), Paris, 2016, pp. 98-98.

[2] B. Omoniwa, R. Hussain, M. A. Javed, S. H. Bouk and S. A. Malik, "Fog/Edge Computing-based IoT (FECIoT): Architecture, Applications, and Research Issues," in IEEE Internet of Things Journal.

[3] B. Omoniwa et al., "An Optimal Relay Scheme for Outage Minimization in Fog-based Internet-of-Things (IoT) Networks," in IEEE Internet of Things Journal.

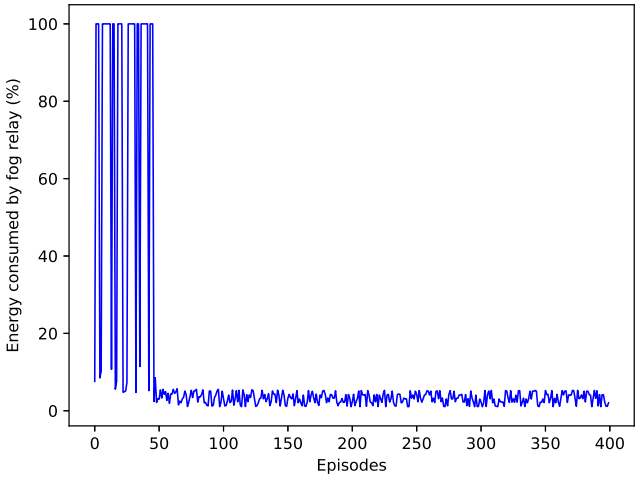

Fig. 4. Average energy consumed by fog agent over episodes.

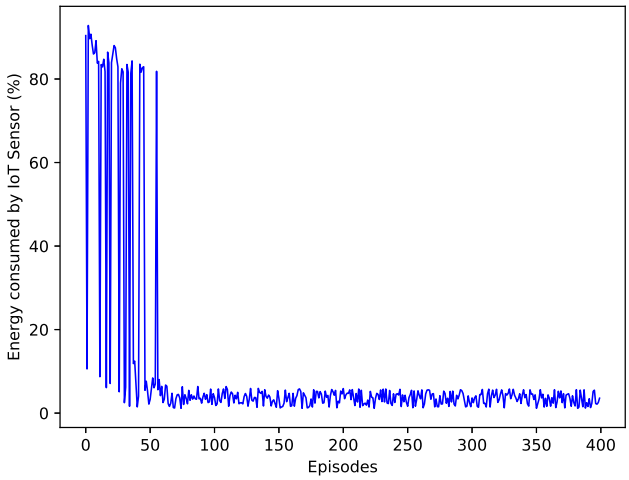

Fig. 5. Average energy consumed by IoT sensor over episodes.

TABLE I
SIMULATION PARAMETERS

Parameter	*Values*
D_I	40 metres
P_I	[0.001, 0.3] Watts
D_S	35 metres
P_R	0.3 Watts
δ	± 0.25 metres
Mobility bound	[-35, 35] metres
Noise power N_0	2×10^{-7} Watts
Path-loss exponent σ	3
Pre-defined threshold κ	1
Discount factor γ	0.9
Learning rate α	0.1
Episodes N	1000
Iteration runs	100000
Policy ϵ	$e^{-0.0015N}$