Ad Hoc Networks 132 (2022) 102869

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Reinforcement learning based reliability-aware routing in IoT networks

Kazim Ergun [a,*], Raid Ayoub [b], Pietro Mercati [b], Tajana Rosing [a]

[a] Department of Electrical and Computer Engineering, University of California San Diego, USA
[b] Intel Corporation, USA

ARTICLE INFO

Keywords:
IoT networks
Routing
Reliability

ABSTRACT

The unprecedented scale and ubiquity of the Internet of Things (IoT) introduce a maintainability challenge. IoT networks operate in diverse and harsh environments that impose thermal stress on IoT devices. The lifetime of these networks can be limited by hardware failures resulting from exacerbated reliability degradation mechanisms at high temperatures. In this paper, we propose a novel adaptive and distributed reliability-aware routing protocol based on reinforcement learning to mitigate the reliability degradation of IoT devices and improve the network Mean Time to Failure (MTTF). Through routing, we curb the utilization of quickly degrading devices, which helps to lower the device power dissipation and temperature, thus reducing the effect of temperature-driven failure mechanisms. To quantify and optimize networking performance besides reliability, we incorporate Expected Transmission Count (ETX) in our formulations as a measure of communication link quality. Our proposed algorithm adapts routing decisions based on the current reliability status of the devices, the amount of degradation they are likely to experience due to communication activity, and network performance goals. We extend the ns-3 network simulator to support our reliability models and evaluate the routing performance by comparing with state-of-the-art approaches. Our results show up to a 73.2% improvement in reliability for various communication data rates and the number of nodes in the network while delivering comparable performance.

1. Introduction

The Internet of Things (IoT) continues to rapidly develop as it is adopted progressively across many domains such as logistics, farming, industrial and environmental monitoring, healthcare, and smart infrastructures. The number of interconnected IoT devices will reach 40 billion by 2025 and worldwide spending on the IoT is already more than $750 billion [1]. Unfortunately, coupled with such dramatic growth, the inherent large-scale of the IoT brings a maintainability challenge with high costs. Currently, the operational expenses reach up to 80% of the overall cost [2], of which a significant fraction is due to hardware failures. While meeting the needs of a growing range of applications, it is also a crucial requirement for IoT devices and networks to operate reliably for long periods, otherwise, maintenance investments can become a critical bottleneck for the growth of IoT.

Recent advances in energy harvesting techniques combined with energy-efficient approaches at different layers of the networking stack made it possible for IoT devices to have substantially prolonged battery lifetimes. With batteries continuously being recharged by energy harvesting sources, energy-neutral operation [3] for the network can be ensured. In such networks, since the risk of batteries running out of energy is diminished, the limiting factor for network lifetime are hardware failures due to reliability issues. As a result of aging and degradation, components in IoT devices lose reliability and eventually fail, leading to a permanent loss of functionality.

Previous research has shown that reliability degradation of electronics worsens exponentially with increasing temperature due to intensified effects of various mechanisms such as Time-Dependent Dielectric Breakdown (TDDB), Electromigration (EM), Bias Temperature Instability (BTI), and Hot Carrier Injection (HCI) [4–6]. IoT devices are often deployed in harsh environments, resulting in stress on the hardware to reduce their reliability and mean time to failure (MTTF). The majority of them typically do not have active cooling to mitigate the thermal stress. In such cases, curbing power dissipation of devices helps to lower the device temperatures and scaling down the effect of temperature-driven failure mechanisms to achieve a better MTTF. Network routing can be useful in this regard; it is possible to place the IoT devices into low-power states by avoiding them in communication paths. In this way, low reliability devices in the network are utilized less to reduce thermal stress and slow down degradation.

Many energy-based routing algorithms have been proposed [7–9] with the goal of extending the battery lifetime of IoT networks, but no work considers the reliability of IoT devices. Here we refer

* Corresponding author.
E-mail address: kergun@ucsd.edu (K. Ergun).

https://doi.org/10.1016/j.adhoc.2022.102869
Received 25 October 2021; Received in revised form 11 April 2022; Accepted 14 April 2022
Available online 25 April 2022

(a) Reliability problems

(b) Our solution vs traditional solutions

Fig. 1. Reliability-driven routing in IoT networks.

specifically to the aging and reliability degradation of the hardware of an IoT device, not the communication reliability or soft errors that are broadly studied. The literature on network routing does not scrutinize the problem of hardware failures and reliability issues as a bottleneck for the lifetime of networks. To improve the MTTF of IoT networks, a reliability-aware routing should be designed following these principles:

(1) Avoid the weakest nodes: As shown in Fig. 1(a), there are many situations that may result in an unbalanced reliability degradation in IoT networks. We call this phenomenon *reliability gap*, the situation in which there is a significant reliability difference between different nodes of the network. Reliability gap can arise as a result of convergecast traffic patterns, congestion, environmental stress, and disproportionate communication distances (Fig. 1(a)). For networks with convergecast traffic patterns and local congestions, some regions in the network may have more traffic to forward, and hence, the nodes here are active more often than the others. Similarly, some nodes may be exposed to higher thermal stress due to their physical location, especially in applications such as industrial and environmental monitoring. These nodes may be the bottleneck for network lifetime since their reliability degrades rapidly and will be the first to fail because of shortened MTTF. The networking load should be distributed as a function of the reliability state of the nodes – besides network performance – by using reliability-aware routing, thus avoiding the weakest nodes.

(2) Use efficient communication links: If the communication link is of low quality and inefficient, then the transmitter node must send many copies of the same packet to be correctly captured by the receiver. Multiple retransmissions mean that the transmitter and receiver will stay active and experience reliability degradation until a successful reception takes place. Using better links improves both communication performance and device reliability. Therefore, the routing protocol should be aware of the link quality and its influence on device reliability.

The typical approach for routing is to model the network as a weighted directed graph and then find paths with the minimum cumulative weight. The weights of the graph edges and vertices traditionally include a variety of node and link metrics: latency, hop count, stability, bandwidth, throughput, and energy, or may be a composite of multiple metrics [10,11]. The impact of degradation mechanisms on device reliability has not been taken into account by routing techniques to date. A simple example of how reliability might affect routing is shown in Fig. 1(b). Node 2 is in a location exposed to higher temperature and thus has much higher thermal stress than the other nodes. A traditional routing solution, shown in yellow, in the bottom right figure, selects the least hop path between the source (S) and the destination (D) nodes, routing through node 2, thus causing its early failure. Our solution, shown in the top right figure, in green, takes a slightly longer path through nodes 1 and 3 but avoids the early failure of node 2. Performance and reliability trade-off over a continuous sample space of possible routing strategies is shown on the left of Fig. 1(b). Traditional routing solutions purely aim at maximizing the aforementioned performance-related metrics. In contrast, the goal of a reliability-aware routing solution is to find a favorable middle ground between performance and reliability.

In recent years, the complexity, dynamism, and heterogeneity of modern IoT networks have driven a recent development of routing techniques based on reinforcement learning (RL) [12,13]. Traditional routing techniques, which are based on statistical assumptions regarding traffic flows and network conditions, are more and more perceived as inefficient to suit the diverse, complex, and highly changing conditions of IoT networks. RL-based routing techniques have been shown to successfully address these challenges; they automatically learn the dynamics of networks, such as new flow arrivals, congestion points, topology changes, quality of links, and adapt to it.

In this paper, we propose R3-IoT, a distributed reinforcement learning based reliability-aware routing protocol for IoT networks. We maximize the reliability and hence the MTTF of the most degraded nodes by (i) avoiding them in the communication path to minimize their traffic, (ii) using high quality, reliable links to reduce retransmissions. Reliability and Expected Transmission Count (ETX) [14] metrics are incorporated in the reinforcement learning formulation. The routing policy learns from experience at runtime, using the past routing decisions and their outcomes, to achieve high performance and to prolong the network lifetime. To the best of our knowledge, we are the first to present a routing solution that explicitly addresses the reliability degradation problem in IoT networks. We conduct extensive simulations using a real-world ambient temperature dataset from the National Solar Radiation Database (NSRDB) [15]. Our evaluation uses large-scale sensor network data, along with real device measurements and models from the High-Performance Wireless Research and Education Network (HPWREN) [16]. Since reliability is difficult to evaluate in practice, we use simulations based on an extended version of ns-3 [17] to demonstrate that our routing approach can achieve similar performance compared to the state-of-the-art while showing up to 73.2% improvement in reliability for various communication data rates and number of nodes in the network.

2. Related work

2.1. Reliability in routing

The term *reliability*, especially in networks, is associated with many different types of failures. Almost all of the literature on network reliability focuses on communication link reliability, that is, the situations where the connection between two nodes in the network fails. In some papers, node failures are also included, but they can be categorized into three groups, none of which handle hard errors: soft errors (causing random bit flips) [18], software reliability issues [19], or batteries running out of energy [20,21]. For example, in [19], software failures, message congestion, VM failures on IoT devices are considered, and the failures are modeled as a Poisson process with an average failure rate. There are also some hardware failures discussed in various works (such as [22]), but they consist of superficial models of sensor faults;

short faults, constant faults, and noise faults. These types of failures are transient and can be more easily fixed, whereas hard failures are not recoverable. We propose a routing solution that explicitly addresses the reliability degradation due to hardware failure mechanisms, which is different from previous works.

Hard failures, caused by well-known thermally-driven mechanisms in silicon, such as TDDB, EM, BTI, result in a need to replace that electronic component in the field, leading to high maintenance and replacement costs. Hard failure models have been studied extensively at the circuit and chip level [4–6], and adopted for dynamic voltage & frequency scaling, task scheduling, and power gating strategies in multi-core system-on-a-chips [23]. Prior to our work in [24], nobody has considered how hard failures affect problems at the network level and how networking might affect the electronics reliability. This paper extends and improves [24] by introducing a new reinforcement learning based protocol, whereas only a routing metric was proposed in the former.

2.2. Maximum lifetime routing

The problem of maximum lifetime routing has been extensively researched over the last two decades for Wireless Sensor Networks (WSNs) and more recently for IoT networks. Since the majority of WSN and IoT devices are battery-operated, the works in this domain are directed towards improving battery lifetimes. In contrast to the approaches that aim at minimizing the total or average energy consumption (e.g., LEACH [25], GEAR [26], ER-RPL [9]), maximum lifetime routing can ensure a balanced depletion of energy among the network nodes. By incorporating the residual energy of node batteries into routing decisions, quickly draining nodes are avoided in the communication paths, and hence network lifetime is extended. A detailed survey on this topic can be found in [20,27,28]. We next discuss a few representative publications.

Chang and Tassiulas [21] define the communication link cost as a function of remaining node energy and the required transmission energy for using that link. By using the Bellman–Ford shortest path algorithm for the computed link costs, the least cost path – whose residual energy is the largest – is found. Following similar methodologies, metrics such as link quality [29], throughput [30], queue utilization [31], and so on were combined with residual energy to achieve different objectives along with the network lifetime. The authors of [32,33] proposed evolutionary algorithms for balancing the load and energy consumption of the nodes. In [34], the traffic load is estimated and the optimal data path is computed to avoid energy holes by efficiently utilizing the nodes that are susceptible to congestion. Although a myriad of studies analyzed the network lifetime problem from an energy optimization perspective, to date, there is no work that addresses the reliability issues in the way we do in this paper.

Only a small subset of the proposed routing algorithms have found application in practice. There are prevalent routing protocols that have gone through the standardization process, which demands a lot of time, effort, and is expensive. Thus, if impact is needed, building a protocol completely from scratch is undesirable. Following this philosophy, modifications and improvements to various standardized protocols such as AODV [35], OLSR [36], RPL [37] were proposed with a focus on extending network lifetime. Many studies engineered new routing metrics that take into account residual battery energies [38–40]. Further modifications are proposed in [41,42]. Similarly, in our paper we adopt and build upon the AODV (Ad hoc On-Demand Distance Vector) protocol.

2.3. Reinforcement learning based routing

Reinforcement learning based routing is gaining importance with the ever-increasing complexity and dynamism of IoT networks and the recent advancements in machine learning [12]. The first application of RL in routing by Boyan and Littman [43] demonstrated that RL is indeed a promising solution for complex communication networks. Since then, many studies have been conducted using variants of RL algorithms with different networking objectives and requirements. Most of the works in RL-based routing have utilized the well-established Q-routing algorithm [43] as their underlying idea, albeit with some improvement [44–46]. Q-routing is based on the traditional Q-learning model in which each node makes its routing decision based on the local routing information. Among many different objectives employed using Q-routing based algorithms, the work in [46] reduces end-to-end delay. Work in [44] couples Q-routing with on-policy Monte Carlo to reduce energy consumption and enhance the network lifetime. The authors of [45] introduce a dynamic discount factor to Q-learning for reducing the amount of route discovery processes after a link failure occurs.

A few recent works consider a combination of reinforcement learning with AODV routing. For example, Q-learning AODV (QLAODV) [47] is a routing protocol that considers link stability and bandwidth efficiency. In [48] the authors use a Bayesian Network to estimate congestion levels and tune the learning weights where they consider signal to noise ratio, delay, and throughput for making routing decisions. Residual battery levels and energy efficiency were also explored as routing objectives in [49], where they are utilized to adjust the willingness of nodes to participate in AODV routing with the SARSA learning algorithm. In contrast to the previous works on RL based routing, we consider the reliability of network devices which was not studied before and we try to maximize hardware lifetime. Reinforcement learning based AODV was particularly studied in MANETs and VANETs due to their erratic mobility, energy consumption, and traffic profiles. It has shown promising results because of its adaptability in such highly dynamic network conditions. Following this rationale, we propose a novel distributed Q-learning based adaptive AODV routing approach. In our work, we model the dynamic factors such as ambient temperature and computation workloads of IoT devices, as well as their effect on reliability. We assume that IoT devices run various workloads which contribute to their heating, combined with the thermal stress imposed by the environment. Our proposed approach is able to adapt these variations in the network and discover better routes without having to know the network topology and traffic patterns in advance.

To summarize, our main contributions are as follows:

- We explicitly consider hardware reliability in IoT network routing to reduce failures and improve network lifetime.
- We incorporate node reliability and ETX metrics into Q-learning updates in our reinforcement learning based approach. Through ETX, we assess the expected communication link performance as well as the expected reliability degradation of a node. Thus, routing decisions are driven by the current reliability of the nodes on the path, amount of degradation they will experience due to retransmissions, and networking performance.
- We implement our routing mechanism in a novel routing protocol called R3-IoT.
- We model reliability in the ns-3 network simulator. To accomplish this, we include the ambient temperature and computation workloads of IoT devices in our simulations.
- We compare R3-IoT to state-of-the-art routing protocols and show that the network reliability is significantly improved while achieving similar performance.

3. Reliability modeling and simulation

3.1. Device modeling

Reliability is defined as the probability of not having failures up to a given time t. The reliability function $R(t)$, in general, can be expressed as a function of *failure rate* $\lambda_f(t)$ [5]:

$$R(t) = e^{-\int_0^t \lambda_f(t')dt'} \quad (1)$$

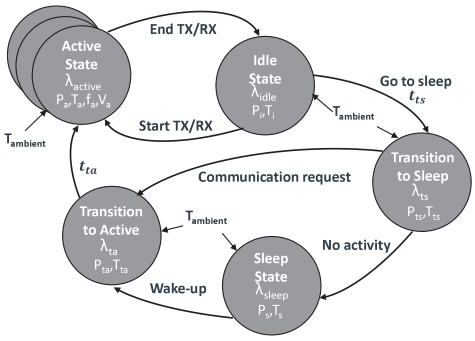
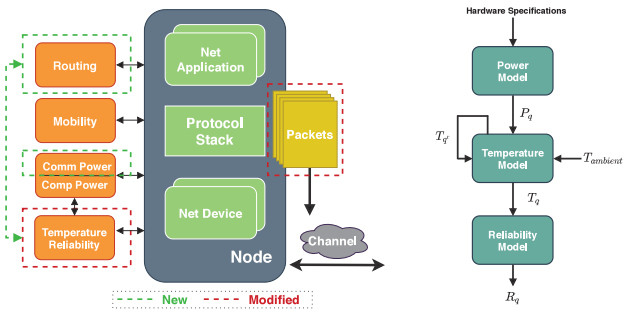


Fig. 2. Device state model.



(a) ns-3 node structure (b) Reliability computation

Fig. 3. Reliability simulation in ns-3.

From Eq. (1), the rate of how quickly reliability is degrading is determined by the failure rate, which depends on temperature, aging, device power state, and switching frequency between power states.

In our reliability analysis, we focus on hard failure mechanisms that cause irrecoverable device failures. Mechanisms such as Time-Dependent Dielectric Breakdown (TDDB), Electromigration (EM), Bias Temperature Instability (BTI), and Hot Carrier Injection (HCI) induce reliability degradation, and thus eventually cause failures. Failure rate models have been developed for each mechanism, which show an exponential dependence on temperature that can be described as follows:

$$\lambda_q = A_0 \eta_q e^{-\frac{E_a}{kT_q}} \tag{2}$$

$$\forall q \in \{Active, Idle, Sleep, \ldots TransitionToSleep, TransitionToActive\}$$

where A_0 is an empirically determined constant, E_a is the activation energy, k is the Boltzmann's constant, and η_q is a constant depending on the respective mechanism and device. Here, we consider temperature T_q of a device as a function of its power state, ambient temperature, and time. When a device switches to a different power state, temperature increases/decreases until it converges to a new steady-state value after a certain time. We assume that the IoT devices can be in various operational states (e.g., *Active*, *Idle* etc.) denoted q, characterized by power dissipation, voltage, and frequency.

Fig. 2 depicts a sample power state diagram of the device with the state transition mechanisms and the parameters characterizing the states. Such state-based model is able to represent the dynamics of IoT devices for many applications, yet convenient and adaptable for simulation purposes. For IoT devices, switching between power states using duty-cycling and wake-up radio techniques – usually implemented at Medium Access Control (MAC) layer – is common for energy saving purposes [50]. The objective of state transitions (represented with arcs) is to put the IoT device in low power modes when not communicating. In the idle state, the system-on-a-chip (SoC) of the device is powered on but not communicating or processing any packets. In the sleep state, most of the SoC subsystems are power-gated. In the active state, the device is busy transmitting/receiving and processing packets. Power scaling methods such as DVS policies can be used for transition between active states for some IoT devices, either down-scaling to reduce power consumption or up-scaling to meet an application performance criteria [3]. Transition to sleep and transition to active states model the time and power consumption required to enter and exit the sleep state. Transition times to/from low-power states follow average transition times t_{ts}, t_{ta}, respectively. Failure rates and the amount of induced reliability degradation change with each power state since different levels of power consumption result in different temperature profiles. Ambient temperature heavily influences the device's internal temperature and has an effect in every operational state.

3.2. Reliability simulation for IoT networks

Analytical models for power, temperature, and reliability should be used to enable reliability evaluation and analysis in network simulations. In this work, we leverage the recently proposed *RelIoT* [51] framework for the ns-3 simulator and enhance it according to our reliability modeling discussion.

The original framework offers an application-based power model that characterizes power consumption of different applications running on IoT devices, particularly targeting edge computing scenarios. Fig. 3(a) depicts the node structure in ns-3 augmented with the *RelIoT* framework, our modifications, and new additions. We implement a new power state machine model as in Fig. 2, for the communication component of a node's power consumption module. The state transitions take place according to the node's communication activity. To compute reliability, power/temperature/reliability model flow is initiated by state transitions as shown in Fig. 3(b). We adopt *RelIoT*'s first order differential temperature model, which incorporates the dependence of node's internal temperature on power and ambient temperature. However, we change the temperature update mechanism. The model dynamically updates device temperature when ambient temperature changes or a state transition occurs. During the transient period, temperature increases/decreases until converging to the steady-state temperature of the new operational state. Our modified reliability model dynamically updates the node's reliability through Eq. (3) by recursively subtracting the degradation induced between consecutive state transitions. The current implementation uses the reliability model presented in [6].

$$R_q = R_{q'} - \Big(\underbrace{R(t_{q'}, T_q) - R(t_q, T_q)}_{\text{degradation}} \Big) \tag{3}$$

The subscripts q and q' indicate the current and previous states respectively. T_q is the temperature experienced by the device between two state transitions from time instants $t_{q'}$ to t_q. $R(\cdot)$ is the static reliability function described in Eq. (1). Finally, the reliability model connects with our routing protocol so that reliability values can be monitored by the routing algorithm. Packet structures are modified to accommodate for the requirements of our learning algorithm, which are explained into more detail in Section 5.

In real systems, reliability tracking is possible with degradation, stress, or aging monitors [52–54]. These monitors, based on ring oscillators that convert temperature and voltage stress into oscillation frequency, give information on the accumulated stress of the SoC, which is used to estimate failure rates and reliability degradation caused by different mechanisms (e.g., TDDB, EM, NBTI) [52,53]. As an example use for IoT, the authors in [54] implement an on-chip stress monitor for IoT devices and pave the way for its future usage in IoT maintenance and management. The system reliability degradation status is usually an input into dynamic reliability management (DRM) techniques [55] to improve device usage. In our case, we propose to use reliability status as an input to our network routing protocol to drive routing decisions.

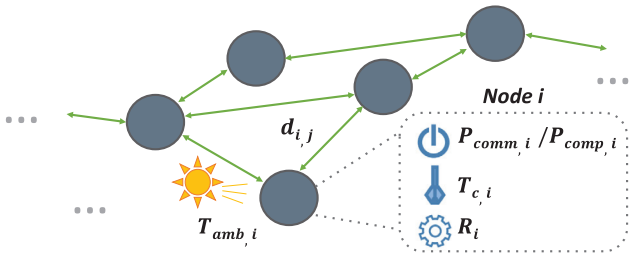


Fig. 4. System model.

4. Reliability-aware routing with reinforcement learning

In the following, we first describe the system model and the problem. We next present our proposed reinforcement learning based routing algorithm.

4.1. System model

Consider an ad hoc wireless IoT network deployed in a mesh topology. The network consists of heterogeneous nodes that we classify into three categories: sensors, gateways, and servers. Each category can have devices of different types. For example, the sensor node can have ARM Cortex-M4 or ARM Cortex-A7 processors. The sensor nodes are 'source' nodes that generate data, which needs to be communicated to a sink (i.e. server). Since the network topology is mesh, nodes can cooperate to distribute and relay data in a multi-hop fashion. We assume that gateways can do data processing.

We investigate a network with many sensor and gateway nodes but a single sink node, similar to related works [47–49]. There are a total of $N+1$ nodes, where nodes $i = 1, \ldots, N$ denote sensor and gateway nodes and $i = N+1$ denotes the sink node. Source (sensor) nodes generate data at rates r_i, so the sink node receives their sum $\sum r_i$. The distance between nodes i and j is $d_{i,j}$. Let N_i denote the set of neighboring nodes to which node i can send packets to. Then, $N_i = \{j : d_{i,j} < d_{max}\}$, where d_{max} is the distance of transmission with maximum power. The notation $j \in N_i$ is used to show that node j is a neighbor of node i and they can communicate. The nodes are static with fixed distances, so neighbors do not change. However, the communication between neighbors is not perfect at all times because we assume lossy communication links, which cause random packet drops.

As shown in Fig. 4, nodes are characterized by their power consumption, temperature, reliability, and the ambient temperature around them. We assume an energy harvesting network, so the residual energy of batteries is not included in our model. In a heterogeneous network, the power, thermal, and reliability characteristics vary between different nodes. Each node consumes the power amount $P_{comp,i}$ for computation and $P_{comm,i}$ for communication. Even though our routing approach only has an impact on communication power consumption, the routing decisions are made by taking into account the overall device temperature and reliability. IoT devices usually run various workloads throughout their operation that contribute to their power consumption. Therefore, we also consider the computation power consumption for the nodes. The core temperature $T_{c,i}$ of the nodes is influenced by their overall power dissipation $P_{comm,i} + P_{comp,i}$ and by the ambient temperature $T_{amb,i}$. Finally, R_i denotes the reliability, which is heavily affected by temperature $T_{c,i}$. For the routing algorithm, the computation power consumption $P_{comp,i}$ and ambient temperature $T_{amb,i}$ are 'external' factors that influence node reliabilities. A routing algorithm cannot control these variables, but the routing decisions should be made in consideration of their effect on the overall reliability.

In such a setting, we seek to improve the network's mean time to failure (MTTF) by means of routing, while keeping performance at adequate levels. MTTF of a single node can be expressed through Eq. (4) as a function of reliability:

$$MTTF = \int_0^\infty R(t)dt \tag{4}$$

where $R(t)$ is the reliability if a device at time t. Then, network MTTF is the minimum of any node in the network (i.e. $\min_{i \in N} MTTF_i$). Here we assume that the network lifetime is defined as the time of first node's failure, which is a common assumption. This definition is one of the most prevalent in literature [20] and was used in many recent works [42,56]. Improving the network's MTTF, or its lifetime in general, can be accomplished by maximizing the reliability of the weakest nodes in the network and minimizing the overall reliability degradation on the nodes. If the weakest node in the network has high reliability, then the time it takes for the first node to fail on average will be extended. As stated previously, how quickly a node's reliability degrades relates to its communication activity, which is influenced by routing decisions. Hence, routing can help prolong network lifetime.

Our goal is to have a protocol that improves the reliability, and hence the MTTF of the most degraded nodes, as well as takes into account the performance in its routing decisions. The routing protocol should (i) help avoiding paths with the minimum reliability nodes, (ii) utilize efficient communication links, and (iii) lead to decisions that will induce minimal reliability degradation.

4.2. Reinforcement learning: Background

Reinforcement learning (RL) [57] is a framework in which an *agent* learns control policies based on experience, for making decisions in an *environment*, to optimize a given notion of rewards. The agent interacts over time with its environment, collects information, and selects the *action* to be applied according to its goal and current state. What is good or what is bad for the agent is defined by the *reward signal*. The environment returns a reward to the agent on each time step to provide feedback about the effect of the recent taken action. The total amount of reward an agent can expect to accumulate, starting from the current state, is estimated by the *value function*. Reward signals indicate what is good (or bad) in an immediate sense, whereas value functions indicate what is good (or bad) in the long run. Therefore, usually value functions are the primary criteria when making and evaluating action decisions.

The problem of reinforcement learning is mathematically framed using a 4-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ the set of actions, $\mathcal{P}$ the state transition probabilities, and $\mathcal{R}$ the rewards. The agent and environment interact at discrete time steps (also called epochs), $t = \{0, 1, 2, 3, \ldots\}$. At each time step t, the agent observes state $s_t \in \mathcal{S}$, then selects an action $a_t \in \mathcal{A}$. As a consequence of its action, one time step later, the agent receives a reward, $r_{t+1} \in \mathcal{R}$, and finds itself in a new state, s_{t+1}. The probability of moving from state s to state s' by taking action a is:

$$p(s'|s,a) = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$$
$$\sum_{s' \in \mathcal{S}} p(s'|s,a) = 1 \tag{5}$$

The objective of the agent while selecting a certain action is to maximize the total cumulative reward it receives over its lifetime. Thus, the agent should learn which of its actions are desirable in the long run. Based on this, actions are selected such that the *return* G_t, expressed as the sum of discounted rewards, is maximized:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{6}$$

where $\gamma \in [0, 1]$ is the discount factor. If $\gamma = 0$, the agent is called 'myopic', only being concerned with maximizing the immediate reward. As γ approaches 1, the future rewards are taken into account more.

The rewards the agent can expect to receive in the future depend on which actions it will take. A *policy* π defines how the agent selects

its actions. Formally, the policy $\pi(s,a) : \mathcal{S} \times \mathcal{A} \mapsto [0,1]$ is a mapping from states to probabilities of selecting each possible action. The value function $V^{\pi}(s)$ is the expected return of following a policy π when at a state s:

$$V^{\pi}(s) = \mathbb{E}_{\pi}[G_t|s_t = s] = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s\right] \tag{7}$$

Here, $\mathbb{E}_{\pi}[\cdot]$ denotes the expected value if the agent follows the policy π. The function V^{π} is particularly called as the *state-value* function. A similar function, denoted $Q^{\pi}(s,a)$, is defined for the expected return of starting from state s and taking action a under policy π. It is called as the *action-value* function and expressed as: $Q^{\pi}(s,a) = \mathbb{E}_{\pi}[G_t|s_t = s, a_t = a]$.

The goal of reinforcement learning is to find a policy that results in maximum reward in the long run. A policy π is better than a policy π' if its expected return is greater than that of π' for all states, i.e., $\pi > \pi' \Leftrightarrow V^{\pi}(s) > V^{\pi'}(s), \forall s \in \mathcal{S}$. The *optimal policy* π^* has the optimal state-value function V^* and action-value function Q^*, given as follows:

$$V^*(s) = \max_{\pi} V^{\pi}(s), \quad \forall s \in \mathcal{S}$$
$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a), \quad \forall s \in \mathcal{S}, \forall a \in \mathcal{A}(s) \tag{8}$$

The optimal value function $V^*(s)$ can be expressed through Eq. (9) without reference to any policy, in a special form called the *Bellman equation* [57].

$$V^*(s) = \max_{a \in \mathcal{A}} \left[r + \gamma \sum_{s' \in S} p(s'|s,a)V^*(s')\right] \tag{9}$$

Once V^* is obtained, the optimal policy can be determined by solving a system of equations. However, explicitly solving the Bellman equation is rarely practical due to the large solution space present in the majority of RL problems. There are many different methods that approximately solve the Bellman equation at reasonable computational cost. In our problem, we use such a method – Q-learning – since we deal with large networks and the learning has to take place on resource-constrained IoT devices.

4.3. Q-learning based routing algorithm design

In this paper, we use Q-learning for routing, which is a model-free off-policy temporal difference reinforcement learning approach [57]. Q-learning is based on the value of state–action pairs $Q(s,a)$, called *action-value* function. We define the value of taking action a in state s under a policy π, $Q^{\pi}(s,a)$ as the expected return from taking such an action and thereafter following policy π:

$$\begin{aligned} Q^{\pi}(s,a) &= \mathbb{E}_{\pi}[G_t|s_t = s, a_t = a] \\ &= \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s, a_t = a\right] \\ &= r + \gamma \sum_{s' \in S} p(s'|s,a)V^{\pi}(s') \end{aligned} \tag{10}$$

Thus, from Eqs. (9) and (10), we have:

$$V^*(s) = \max_{a \in \mathcal{A}} Q^*(s,a) \tag{11}$$

In Q-learning, the agent learning consists in a sequence of stages, called epochs. In epoch t, the agent is in state s_t, it performs action a_n, it receives a reward r_t, and it moves to state s_{t+1}. The action value is updated as follows:

$$Q(s_t,a_t) \leftarrow (1-\alpha)Q(s_t,a_t) + \alpha[r_{t+1} + \gamma \max_{a \in \mathcal{A}} Q(s_{t+1},a_{t+1})] \tag{12}$$

where α is the learning rate. The agent learns optimal policy with the help of a greedy policy where an action can be chosen just by taking the one with the maximum Q-value for the current state. Q-learning is a suitable method for online 'runtime problems' such as routing in our case, because it only uses the most recent decisions to update its policy. Also, it converges to the optimum action-values with probability 1 as long as all actions can be randomly sampled in all states [58]. For this reason, it is an effective technique for learning from delayed reinforcement, i.e. learning based on the reward that can be received far in the future. To map a routing problem to a Q-learning problem, one needs to design the state space, action space, reward function, and learning parameters:

- *State (s):* The current state of the agent is the index of the node holding the packet.
- *Action (a):* Selection of the node for the next hop.
- *Reward (r):* Higher rewards earned if the action brings the packets closer to the destination node.
- *Learning Parameters (γ, α):* Described in terms of reliability and performance related metrics of the nodes.

4.3.1. States and actions

For learning a routing strategy, each node is associated with a state s, and for each of its neighbor s', there is a corresponding action. Executing action a at s means forwarding the packet to the neighbor s' corresponding to that action. Let $s = i$ denote the node holding a packet P to forward and $Q_i(des,j)$ denote the Q-value of node i forwarding the packet to destination d through next-hop node $s' = j$. Then, the action-value updates are expressed through Eq. (13).

$$Q_i(des,j) = (1-\alpha)Q_i(des,j) + \alpha[r + \gamma \max_{k \in N_j} Q_j(des,k)] \tag{13}$$

Node i maintains a table of Q-values $Q_i(des,j)$ for each neighbor j and destination des, which can be regarded as its routing table, telling which neighbor to forward the data. With the help of this routing table, the optimal routing path can be constructed by a sequence of table look-up operations. We treat the network as the environment and the nodes as the entities where the agents reside. For a completely distributed Q-learning based routing, we assume that there exists an agent at each node. The agents try to improve the current solution while switching between exploration and exploitation of the solution space. The exploration and exploitation processes of Q-learning have their own interpretations for routing. In our approach we adopt the $\epsilon - greedy$ strategy. By default, a node selects the next hop node which has maximal Q-value to forward data, which is called exploitation. However, with some probability ϵ it chooses a random node which does not have maximal Q-value, this is called exploration.

4.3.2. Reward function

The reward function is critical to Q-learning, as it determines the behavior and performance of the agent. The goal of the routing algorithm employing Q-learning is to get the packet delivered to the destination (the sink node) with maximum expected return, i.e., minimum cost. To prevent loops and ensure forwarding the packets towards the destination node, we need increasing Q-values in the direction of the destination. Hence, we employ the following reward function:

$$r = \begin{cases} 1 & \text{if } i \in N_{des} \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

where N_{des} is the set of neighbors of the destination des. This means that if a node receives a packet from the destination, the reward will be 1 and otherwise 0. Therefore, only the immediate neighbors of the sink node receives a reward of 1. By using this reward function, closer nodes to the destination attain high Q-values. This reward propagates to the nodes away from the destination, but it gets discounted more and more as it travels further.

4.3.3. Reliability-aware learning parameters

To include both reliability and performance aspects in our Q-value updates, we propose a composite discount factor (γ) function of a node's *reliability* and its *expected transmission count* (ETX) over communication links. Using this approach, Q-value of a node decreases at each update if its reliability and link quality is low. The optimal policy is then the one which picks both reliable nodes and reliable communication links for routing.

ETX is one of the most frequently used metrics in routing protocols. It estimates the number of data transmissions required to send a packet over a link and get acknowledged, including retransmissions. It is computed as:

$$ETX = 1/(p_{i\mapsto j} \cdot p_{j\mapsto i}) \tag{15}$$

Here, the notation $p_{i\mapsto j}$ denotes the probability of successful packet delivery from packet source node i to destination node j. We obtain both probabilities $p_{i\mapsto j}$ and $p_{j\mapsto i}$ with a message exchange protocol that is explained in Section 5. Through ETX, we assess the link performance as well as the expected reliability degradation for using that link. To calculate the reliability degradation induced, we first estimate the traffic that the node has to forward. Let j denote the node of interest, then the total allocated traffic for j is the sum of traffic it generates TR_j^{gen} and the traffic TR_i^{total} incoming from its neighbors i:

$$TR_j^{total} = TR_j^{gen} + \sum_{i|i\mapsto j} TR_i^{total} \tag{16}$$

Multiplying this estimate of total traffic TR_j^{total} with the expected transmission count $ETX_{j\mapsto i}$ and then dividing by the data rate r_j of the node, we compute the expected communication time through Eq. (10).

$$t_j^{comm} = (TR_j^{total} \cdot ETX_{j\mapsto i})/r_j \tag{17}$$

Finally, in Eq. (11), we estimate the reliability degradation D_j induced by using Eqs. (1) and (3).

$$D_j = R(t_{s,j}, T_{s,j}) - R(t_{s,j} + t_j^{comm}, T_{s,j}) \tag{18}$$

Since D is proportional to ETX, its value will be higher for low-quality links. Selecting paths with the minimum D utilizes efficient links (better performance with fewer packet loss) and induces minimal reliability degradation.

As the second part of our composite function, we need a component that focuses on the bottleneck in network MTTF: the node with minimum reliability. Thus, the discount factor should be directly proportional to the current reliability R_j of the nodes. Together with both the degradation and current reliability metrics, a node computes its discount factor γ as

$$\gamma_j = R_j \frac{2}{\pi} \arctan\left(\frac{\gamma_0}{D_j}\right) \tag{19}$$

where γ_0 is a predefined constant. We use inverse tangent function to contain the value of γ_j in the interval [0,1]. Using this discount factor, Q-value is updated based on the current reliability of the nodes, the amount of degradation they will experience due to retransmissions, and networking performance (by the implicit use of ETX in the DEG function). The information is discounted for each node it passes through and is also discounted according to the reliability and the expected degradation of the nodes. In this way, we ensure that the route selected has less hops and is more reliable.

Intuitively, by designing such a discount factor we do not only consider individual node reliabilities at each single hop, on the contrary, we consider the overall system reliability from the source to the destination. From Eqs. (10) and (12), we observe that the discount factor is multiplied for each node Q-value is computed. A cumulative equivalent discount factor for the overall system, from a source node src to a destination node des, can be expressed as $\gamma_{equ} = \prod_{j=src}^{des} \gamma_j = \prod_{j=src}^{des} R_j \frac{2}{\pi} \arctan\left(\frac{\gamma_0}{D_j}\right)$. This multiplicative property of the discount factor exhibits a meaningful resemblance to a common system reliability definition used in industry. The system of n components fails if any of its components fails – as we have assumed in our network lifetime definition – for systems organized in a 'series' structure. The series system reliability is then given as follows [5]:

$$R_{system}(t) = \prod_{i=0}^{n} R_i(t) \tag{20}$$

at any time t throughout the operation of the system. It can be seen that the cumulative discounted rewards over a packet route has a similar form to Eq. (20).

In our work, we model the dynamic factors such as ambient temperature and computation workloads of IoT devices, as well as their effect on reliability. We assume that IoT devices run various workloads which contribute to their heating, combined with the thermal stress imposed by the environment. Through this reliability-aware learning parameter, our Q-learning approach is able to adapt these variations in the network and discover better routes without having to know the network topology and traffic patterns in advance.

5. R3-IoT protocol design

In this section, we describe several design considerations for our routing protocol, which is a modified version of the Ad-Hoc On-demand Distance Vector (AODV) [35] protocol. We explain implementation details including the route discovery mechanism, the packet structures, the routing table, and the metadata exchanged between devices. We also discuss the overhead involved. Finally, we present results on a small 'toy' example to demonstrate the route discovery process of our approach.

5.1. Route discovery mechanism

Route discovery is carried out by Route Request (RREQ) and Route Reply (RREP) packets as in the AODV protocol. The source node floods the network with RREQ packets, which are forwarded through multiple hops until they reach to the destination. The RREQ packets are only forwarded to the neighbor with the highest Q-value. When the RREQ packet reaches the destination, the destination node returns an RREP packet through the same path that the RREQ packet followed. In the original AODV protocol, each node broadcasts RREQ packets to all of its neighbors. We choose where the packet should be forwarded independently at each node based on greedy exploitation, that is, the packet is only sent to a single node of highest Q-value. This significantly reduces the number of packets need to be communicated as there is no need for broadcasting.

5.2. Packet structure and metadata exchange

To determine the ETX metric of a link between two nodes, ETX based protocols (e.g., AODV-ETX [59]) use Low Power Payload (LPP) packets. The successful packet delivery probabilities $p_{i\mapsto j}$ and $p_{j\mapsto i}$ in Eq. (15) are estimated with the LPP packets that are broadcasted over the network. Each node broadcasts LPP at an average period τ and remembers the LPPs received from its neighbors over the last w seconds, allowing to compute the probability $p_{j\mapsto i}$ at any time t through Eq. (21). Using relatively small size LPPs, this process incurs only a marginal overhead and saves bandwidth.

$$p_{j\mapsto i} = \frac{count(t, t-w)}{w/\tau} \tag{21}$$

where $count(t, t-w)$ is the number of LPPs received by node i and w/τ is the number of LPPs that was sent by node j during the window w. For the link $i \mapsto j$, this allows node i to simply estimate $p_{j\mapsto i}$ by counting successfully received LPPs from j. On the other hand, to compute $p_{i\mapsto j}$,

Table 1
Structure of the LPP packet.

Field name	Size	Description
Type	8b	Indicates that the packet is of type LPP
LPP ID	8b	Identification number of LPP
Originator IP Addr.	32b	IPv4 address of the node that generates the LPP packet
Originator Seq. No.	32b	Sequence number of the node that generates the LPP packet
No. of Neighbors (n)	8b	The number of neighbors whose LPPs are received by this node
Neighbor IP Addr.	n*40b	IPv4 address of the neighbor from which an LPP packet is received in the last w seconds
Forward LPP Count		The number of LPP packets received in the last w seconds from the neighbor
Max Q-Value	32b	The max Q-value among the neighbors of the node that generates the LPP packet
Reliability	32b	The current reliability of the node that generates the LPP packet

node j includes the number of LPPs it received from i sent during the last w seconds in its each LPP. This way, node i can also estimate $p_{i\mapsto j}$ by using this information.

We modify the LPP packets to also include the fields that are needed for Q-learning. The structure of our modified LPP packet is shown in Table 1 with the description of the fields. The fields *Neighbor IP Address* and *Forward LPP Count* are repeated for each neighbor. The number of neighbors is indicated by n. *Max Q-Value* and *Reliability* are floating point numbers, so are not suitable for serialization and deserialization of packets. We represent these values as integers by 10^k for k decimal digit resolution before the serialization and deserialization processes.

5.3. Routing table

In AODV, the routing table entries are classified by the destination addresses. If there are more than one route to the destination, the best route is selected as the one with the least number of hops. We extend the original routing table of AODV with an additional Q-Table. Since we assumed that there exists an agent at each node for completely distributed Q-learning, every node has to store and maintain the Q-values of its next-hop neighbors. We use a dynamic Q-Table, such that the size of the Q-Table of a node is determined by the number of destination nodes and neighbor nodes.

In addition to Q-values, Eqs. (7)–(12) show that R, ETX, and the maximum Q-Values of each neighbor are needed for action-value updates. Every node has a table for their neighbors with records of this metadata, which is extracted from the LPP packets they receive. Also, the table needs new fields regarding the forward and reverse LPP counts for each neighbor of a node to calculate ETX. Overall, the table has the following fields per each neighbor:

- *NeighborIpAddress:* contains the IP address of the neighbor,
- *ReverseLppCount:* tracks the number of LPP packets received from the neighbor with respective IP address,
- *ForwardLppCount:* tracks the number of LPP packets that the neighbor with respective IP address received from this node,
- *QValue:* contains the Q-Value for the neighbor with respective IP address,
- *MaxQValue:* contains the maximum Q-value in the Q-table of the neighbor with respective IP address,
- *Reliability:* contains the reliability value of the neighbor with respective IP address.

The maintenance of the entries of Q-Tables is ensured through LPP packets. Each node exchanges information with neighboring nodes and updates its Q-Table periodically. *ReverseLppCount* is obtained by counting received LPPs from each neighbor. *ForwardLppCount* is obtained from the received LPP packets. ETX metric is calculated for each neighbor from these two values, as shown in Eq. (21). The entry, *MaxQValue*, contains the maximum Q-value in the Q-Table of the neighbor. As shown in Table 1., the neighbor node finds the maximum Q-value of its own neighbors (i.e., $\max_{k\in N_j} Q_j(des,k)$) and puts it in the LPP packet. This value is received and stored, then is used for learning updates.

5.4. Protocol overhead analysis

The overhead of communication comes from metadata exchanging with LPP packets. As Table 1 shows, the extra header of each LPP packet includes Q-value, maximum Q-value, and reliability. All the other metadata is already present in the default LPP packet structure. All added fields are represented with a 32-bit word. Therefore, in total, there are additional 24 *bytes*. The frequency of broadcasting LPP packets is usually very low compared to data communication, and hence, this part of overhead is negligible. In our experiments, we set the LPP period $\tau = 1$ s so the LPP overhead is only 24 bytes/s compared to any ETX based protocol.

For the Q-learning algorithm, each node has to carry out computations to update Q-values when an LPP packet exchange occurs. As shown in Eq. (6), this computation is a simple multiply-add operation that introduces very modest delay and power consumption, much smaller than that of communications. The computation overhead is also negligible. Additional computations required by our technique involve only two floating-point additions and multiplications per iteration of Q-learning, which is very minor relative to what is required for the rest of the computation and communication. This overhead scales linearly with the number of node's neighbors. However, even though the number of nodes reach thousands, the number of neighbors per node stay in the order of tens [60]. Most other routing protocols have similar communication and computation overhead. Regarding the training overhead of our Q-learning method, we leverage the distributed structure of Q-learning to reduce this cost where the training overhead per individual node is small and scales with the number of node's neighbors. In large networks, the traffic distribution is not expected to be uniform across all nodes in most cases where a fraction of nodes may stay dormant/less active; hence, the frequently accessed nodes become more critical and can be trained faster by levering the distributed nature of our Q-learning. The training of the Q-learning model is done completely online (no offline phase) using temporal difference learning method where the model continues adapting at runtime, using runtime observations.

Besides exchanging metadata and computing Q-values, nodes need to store these metadata – some in the form of routing tables – for all of their neighbors. At runtime, the amount of required metadata storage may vary because the number of neighbors and the communication paths in the network can dynamically change. Again, this metadata is much smaller compared to the storage capacity IoT devices have and the data payload they hold. In the case of very large-scale networks with a high number of nodes, approximate Q-learning can be implemented to dramatically reduce the size of Q-tables. By using function approximation, Q-learning can scale to handle very large state-spaces [61]. In particular, deep Q-learning is a promising solution that proposes neural network function approximation for Q-tables [62,63].

5.5. Small-scale example

We use the simple network shown in Fig. 5 to demonstrate the route discovery mechanism of our approach. The source node S wants to send packets to the destination node D. Then, the possible non-cyclic routes that the packet can take are: S-2-D, S-1-2-D, S-1-3-D, S-2-3-D, and S-1-2-3-D. The particular route to be selected for packet transmission depends on the routing protocol. We assume that at the beginning

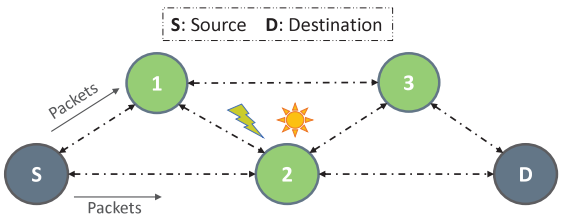
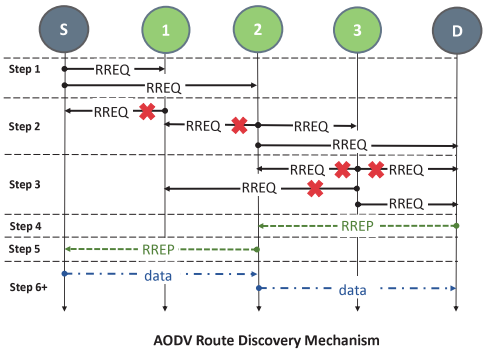
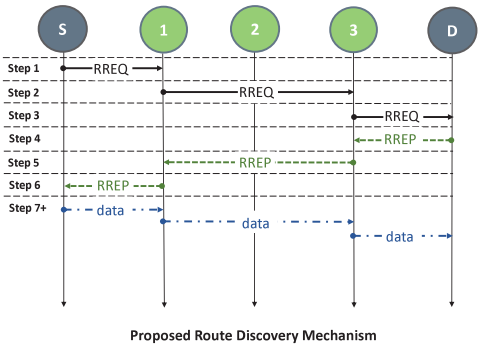


Fig. 5. Simple network example.



(a)



(b)

Fig. 6. Route discovery in the simple network example.

of route discovery node 2 is highly degraded with a reliability value $R_2 = 0.70$ under the influence of environmental stress, whereas other intermediate nodes have reliability values $R_1 = R_3 = 0.90$. Here, the reliability values depict the probability of not having failures before the given time instant, defined in the interval [0,1]. The route discovery procedure for the default AODV and our proposed approach are shown in Fig. 6, as a flow diagram of RREQs and RREPs.

In AODV, the source node S first broadcasts an RREQ, which is received by both node 1 and node 2. In our approach instead, RREQs are not broadcasted, they are forwarded to the neighbor with the highest Q-value. As node 2 has the lower reliability in this example, its Q-value is small as well. Therefore, an RREQ is sent to node 1. As a second step, AODV again broadcasts RREQs from each node that received RREQ in the previous step. If the received RREQ packet has the same ID which was already seen, it is discarded, else, the 'hop count' value of the RREQ is increased by 1, and the packet is broadcasted again until it reaches the destination. Since in our approach RREQs have only a single receiver at each step, they are not discarded at any point. Finally, for both approaches, when the destination node D receives RREQ, it generates a unicast RREP packet and sends it back to the source node. When the source node receives this RREP, it then has the route to the destination and can start sending data packets.

For the example network in Fig. 5, the default AODV protocol chooses the least hop path S-2-D, whereas our proposed approach chooses path S-1-3-D, avoiding the most degraded node 2. Results show that node 2 degrades much faster than the other nodes using default AODV and becomes the bottleneck for the network lifetime. When our approach is used, degradation of node 2 slows down and the reliabilities of all nodes meet at the value $R = 0.42$. After that point, all nodes degrade at the same rate. In this way, the MTTF of the simple network is increased by 2.55x. For this particular small network of 5 nodes, this improvement comes with a cost of one extra hop in the path (from one hop to two hop route) and an increase of 16.7% end-to-end delay. In general networks have many more hops from the source to destination than this example. The performance penalty of our approach is not as large with only a few extra hops, for which we provide detailed results in the following section.

6. Evaluation

We conduct simulations based on a scenario of environmental monitoring. We refer to an example of real-world deployment, the High-Performance Wireless Research and Education Network (HPWREN) [16]. HPWREN is a heterogeneous wireless sensor network, deployed in the Southern California area. In HPWREN, there are many types of computing systems ranging from the small wireless sensor nodes, single-board computers, to the high-performance server systems at the UCSD Supercomputer Center. It comprises several subnetworks, but we only focus on a 2 km^2 region of the Santa Margarita Ecological Reserve (SMER) network covered with a mesh topology [64]. We use data collected from HPWREN to model IoT devices in ns-3 and configure the parameters of the simulation. In our simulations, IoT devices (nodes) are randomly distributed over a field of 1000 m x 1000 m. We conduct experiments for networks of 50, 100, 150, 200, and 250 nodes. A subset of 20 nodes generate Constant Bit Rate (CBR) traffic – typical of sensors that sample at regular intervals – and transmit UDP data with 512 bytes packets to a sink node in an ad hoc fashion. We chose data rates of 20, 40, 60, 80, 100, and 120 kbps for evaluation. Wireless links between nodes are assumed lossy and have a bandwidth of 2 Mbps, so successful packet transmission is not guaranteed. The lossy communication environment is simulated using the *HybridBuildingsLossModel* in ns-3. Experiments last 730 s, are repeated 100 times with different random seeds and averaged to reduce the randomness in results for achieving high confidence. In our experiments we use the IEEE 802.11b standard for the MAC layer because it the most matured communication standard implementation in ns-3. There are efforts on modeling low-rate and low-power standards for IoT, but they are not fully developed yet. Hence, we modify the 802.11 PHY and MAC layer parameters and scale data rate and power values to imitate communication in an IoT environment. All the communication-related parameters used in our simulations are summarized in Table 2. For Q-learning, we set the learning rate $\alpha = 0.8$, the discount rate constant $\gamma_0 = 10^{-5}$, and the exploration parameter $\epsilon = 0.1$. The respective values were determined by carrying out a grid search and finding the best performing values. Therefore, the values used in the paper are optimized and represent the best achievable results using our method.

Environment: Reliability heavily depends on the temperature of the environment, so we consider realistic, varying ambient temperature conditions. We use a temperature dataset which contains half-hourly ambient temperature measurements of 210 locations over a year [15]. Moreover, we consider the effects of the device being placed in different locations by selecting the temperature as $T_{amb} \pm U(-10, +10)$, where U is a uniform distribution. For example, a device placed in a closed container under the sun, with airflow around the device is restricted, will have a much higher ambient temperature than a device placed under a shade in open air. Reliability is evaluated considering the TDDB failure mechanism [6], which is a commonly used model in industry today. Other reliability modes can easily be added as needed, since they all

Table 2
Simulation parameters.

Parameter	Value
Simulation area	1000 m × 1000m
Number of nodes	50, 100, 150, 200, 250
Routing protocol	AODV, AODV-ETX, R3-IoT
MAC layer	IEEE 802.11b
Traffic type	CBR UDP
Data rate	20, 40, 60, 80, 100, 120 kbps
Packet size	512 bytes
Bandwidth	2 Mbps
Loss model	ns3::HybridBuildingsLossModel

exponentially depend on temperature. We scale the impact of reliability degradation in the simulations to reflect 1 year of degradation for 365 s of simulation time.

Target Platforms: To capture the heterogeneity of IoT networks, we use models of 3 different embedded devices in our simulations. The target IoT devices are Raspberry Pi 0, Raspberry Pi 2, and ESP8266 microcontroller. We estimate the CPU and WiFi power consumption and temperature of the edge devices by collecting measurements of various applications under different ambient temperatures on the actual devices. We configure the parameters of the heterogeneous node models in ns-3 so that they follow the power, temperature, and reliability characteristics of the modeled platforms. We randomly choose the number of each device platform in the network for every simulation instance.

Workloads: We assume IoT devices run various computation workloads as well as communication data. Therefore, device reliability is also affected by these workloads, which can be considered as an external factor. In our experiments, we consider the ML classification and regression tasks characterized for edge computing settings in [65] with their corresponding power consumption values. The simulated workloads are generated randomly from the measured set of tasks with the execution times sampled from an exponential distribution with a mean of 10 s.

We compare our proposed approach (referred to as R3-IoT) with the following techniques:

- *AODV* [35] is the original Ad-Hoc On-demand Distance Vector routing protocol. It routes the packets through the least hop path.
- *AODV-ETX* [59] finds the path with the least total expected transmission count. The goal is to optimize the packet delivery ratio.
- *Q-AODV* [66] is a Q-learning based energy-aware maximum lifetime routing approach. Network nodes learn to adjust its route-request packets according to their energy profile.

All the models and solutions are implemented in the ns-3 network simulator, leveraging *RelIoT* framework [51] for power, temperature, and reliability simulation. We modified the original models in this framework to support our routing implementation. We simulate the following three different scenarios:

(1) *Constant Uniform Ambient Temperature and No Computation Workload.* When the ambient temperature is the same for all nodes and they do not run any computation, it is expected that only communication would make a difference in their reliability and hence lifetime. In this experiment, we evaluate how good our approach is without the influence of external factors (workloads, ambient temperature) affecting reliability.

(2) *Varying Ambient Temperatures and No Computation Workload.* The most important external factor that affects reliability is the ambient temperature. Since it is an uncontrollable element, the routing decisions would not change it. However, the routing protocol should still be aware of the impact of ambient temperature on reliability over time to be able to moderately utilize the nodes which are under environmental stress. In this experiment, we evaluate if our approach can learn the pattern of ambient temperature changes and the magnitude of its impact on reliability over time.

(3) *Varying Ambient Temperatures and Stochastic Computation Workloads.* Computation is an orthogonal component to communication, which also degrades the reliability of IoT devices. In this case, we model the possible workloads that can be running on IoT devices. These workloads cause extra heating of the device. The routing protocol should learn to avoid the devices which heat up because of running high workloads and operating under high ambient temperatures. We randomly pick workloads from our task dataset and set their service times by sampling from an exponential distribution. Not all nodes do computation, we also randomly select the ones that run workloads.

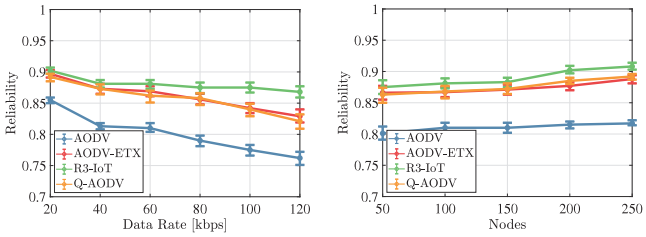


Fig. 7. Scenario 1: Reliability for different data rates and number of nodes.

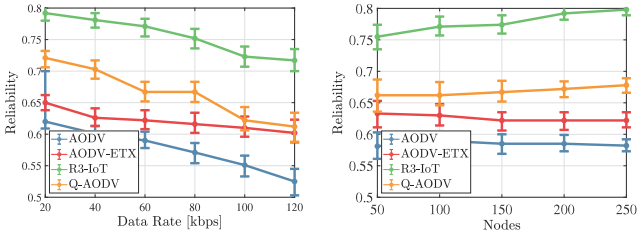


Fig. 8. Scenario 2: Reliability for different data rates and number of nodes.

In the following, we present results for reliability (probability of not having failures up to the given time) and performance (packet delivery ratio, end-to-end delay, and convergence rate). The reliability results are represented as probabilities that take values in $[0, 1]$.

Reliability Results: Figs. 7–9 show reliability results for different data rates and number of nodes in all scenarios, averaged over randomized simulation runs. The data rate results were collected for networks with 100 nodes, then different numbers of nodes were tested at a data rate of 100 kbps. The reliability values denote the *minimum* reliability in the network since we have defined the network lifetime as the time which the first node fails. In all scenarios and for all approaches, reliability degrades with increasing data rate as expected due to the increased communication activity. AODV performs the worst because it always chooses the same nodes which result in the least hop in the communication path. AODV-ETX changes the nodes used because of randomness in expected transmission counts due to link qualities. This is the reason why it performs similar to R3-IoT in Scenario 1 (Fig. 7), because the only source of degradation is due to communication and retransmissions due to link losses. Q-AODV also shows a similar characteristic to AODV-ETX because energy consumption and expected transmission count is highly correlated, so they inherently have parallel goals. Again in Scenario 1, for the increasing number of nodes, there is not much difference in the minimum reliability in the network because the most degraded nodes in this scenario are always the ones closer to the sink.

As shown in Fig. 8, when an external factor, temperature, that affects reliability is involved, R3-IoT performs much better than both

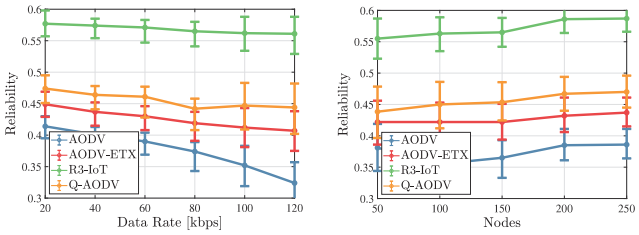


Fig. 9. Scenario 3: Reliability for different data rates and number of nodes.

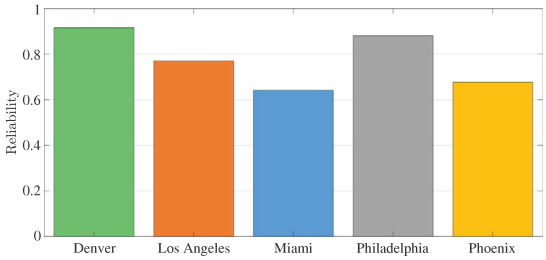
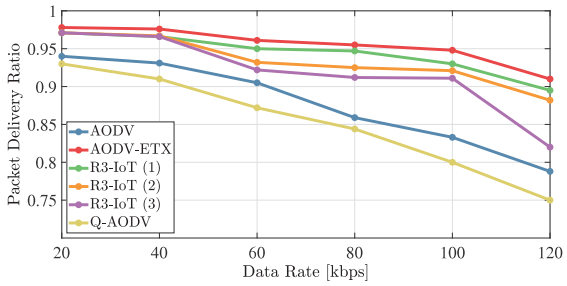
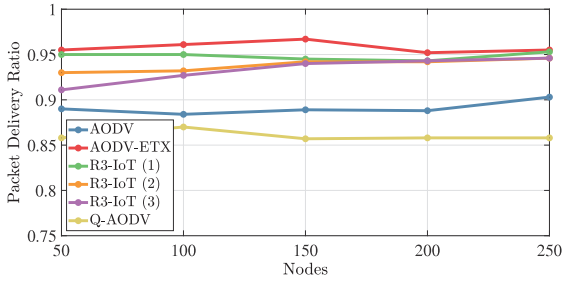


Fig. 10. Scenario 2: Reliability for cities with different average ambient temperatures.



(a) Packet delivery ratio vs data rate



(b) Packet delivery ratio vs number of nodes

Fig. 11. Performance results.

AODV, AODV-ETX, and Q-AODV. R3-IoT learns the routes that bypass the nodes under high thermal stress. Q-AODV also results in fairly high reliability because the nodes that are highly degraded usually also has lower energy. Hence, Q-AODV avoid the low-reliability nodes too. When there are a high number of nodes in the network, then R3-IoT can find more routes that include only high-reliability nodes. The discrepancy between R3-IoT and other approaches is further exacerbated when computation workloads are added to nodes (Fig. 9). In this scenario, at the highest data rate 120 kbps, the most degraded node has 73.2%, 37.8%, and 26.4% more reliability with R3-IoT compared to AODV, AODV-ETX, Q-AODV approaches respectively. Similarly, for the networks with 250 nodes, improvements of 51.8%, 38.6%, and 24.9% are observed.

Finally, for scenario 2, we run simulations under various average ambient temperature values to show its impact on resulting reliability. We use hourly values from 2 years long temperature data collected for cities: Denver, Los Angeles, Miami, Philadelphia, and Phoenix. The respective 2 year average temperatures between 2015–2017 of these cities are: 9.8 °C, 17.8 °C, 25.1 °C, 12.4 °C, and 22.4 °C. Fig. 10 shows that a network deployed in a city with hot weather (Miami) can have as much as 42% less reliability after 2 years compared to a network deployed in a city with cold weather (Denver).

Performance Results: We evaluate the performance of all methods under all scenarios. Fig. 11(a) shows the packet delivery ratios (PDR) for different data transmission rates. Error bars are omitted for performance results since the variation between simulation runs was negligible unlike the reliability results. For all cases, PDR drops with the increasing transmission rate, however, the drop is more severe for AODV and Q-AODV. Both AODV-ETX and R3-IoT chooses better communication links because they utilize the ETX metric, which results in a fewer number of packet losses over links. For all three scenarios, AODV, AODV-ETX, Q-AODV performances do not change because they are agnostic of the changes in node temperature and workloads. Therefore, we use only a single line on the plots to depict the performance of these approaches on the three scenarios. On the other hand, as temperature increases and nodes start running workloads, R3-IoT favors the nodes with higher reliability in the route instead of the nodes with higher quality links. This is the reason performance drops slightly from scenario 1 (R3-IoT (1) on the plot) to scenario 3 (R3-IoT (3) on the plot).

In Fig. 11(b) we compare the PDRs for varying numbers of nodes in the network. Similar to the previous case, Q-AODV performs the worst while R3-IoT performs very close to AODV-ETX. There is a slight increase in PDR as the number of nodes increases for all approaches. This is due to the fact that there are more possible routes that packets can take if the number of nodes is high. For more nodes, the performance of R3-IoT in both Scenario 1 and Scenario 2 approaches to its performance in Scenario 1. Even though R3-IoT tends to choose the nodes with high reliability instead of the ones that can result in better performance, there are more nodes that satisfy both properties in a network with large number of nodes.

We further present comparison for average end-to-end delay of all methods under all scenarios in Fig. 12. The results are plotted only for varying number of nodes because no significant changes were observed for different data rates. Similar to PDR results, AODV, AODV-ETX, Q-AODV delays do not change under varying node temperature and workloads. Q-AODV performs the worst because it does not include any mechanism to determine the shortest paths or the most reliable links. AODV and AODV-ETX can find paths with minimal number of hops and hence acquire very small delays. Our approach is similar to AODV in Scenario 1, but it gravitates towards balancing reliability when ambient temperature and workloads are introduces to the simulations.

Fig. 13 shows the convergence performance of our Q-learning approach for networks with varying number of nodes. The corresponding experiments were conducted at a data rate of 100kbps. We omit the results for varying data rates as we have not observed any significant changes at different data rates, the convergence rates were similar across the board. The values plotted are the normalized sum of the Q-values of all the nodes in the network. Epochs denote the number of Q-value updates, that is, the operation described in using Eq. (12). The time each epoch takes can be configurable in our approach and is equal to the LPP packet period since Q-value updates take place when

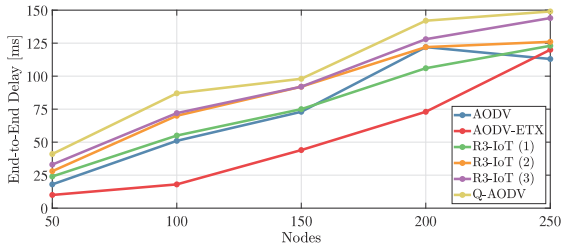


Fig. 12. Average end-to-end delay.

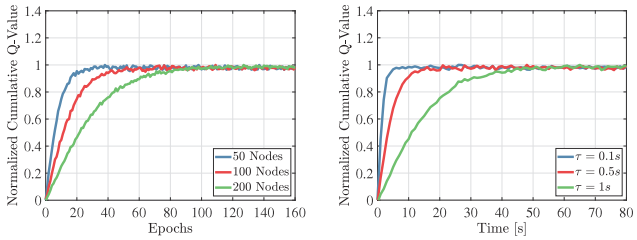


Fig. 13. Convergence of Q-values.

LPP packets are exchanged. In the second plot we present convergence in actual clock time metric for different LPP periods, simulated on a 100 node network. Results show that convergence is reached in fairly low iterations, especially for networks with a small number of nodes. This implies an efficient use of samples, that is, the sampling complexity of the proposed distributed reinforcement learning approach is low. It should be noted that these results show a learning curve for a "cold-start", when the algorithm is run right after the network is set up. Similar initialization periods are common for many routing protocols for building their routing tables. After the initial phase, it only needs a few epochs for the Q-learning algorithm to react to the changes in the system (e.g. variations in the external factors such as ambient temperature and workloads).

7. Discussion

Q-learning can be computationally costly to implement in large-scale networks due to the growing size of Q-tables. In an ad hoc network, each node may have to store every next-hop and destination pair for all of the nodes in the network. However, the nodes have only one destination, i.e., the sink node, in the scenario we considered. It is only crucial to keep the next-hop information for this specific destination node. On the other hand, the Q-values are kept for only the neighbors of the nodes, which is not a large number when compared to the total number of nodes in the network. If the nodes have very large number of neighbors, then approximate Q-learning can be implemented to dramatically reduce the size of Q-tables. By using function approximation, Q-learning can scale to handle very large state-spaces [61]. In particular, deep Q-learning is a promising solution that proposes neural network function approximation for Q-tables [62,63]. In future work, we want to utilize deep reinforcement learning solutions to overcome such scaling issues.

Our general Q-learning methodology can be adopted and implemented into different protocols albeit with modifications, but the specific implementation in this paper is based on AODV. For example, a more common routing protocol is RPL [37] for many low-power IoT applications. One of the reasons that we used AODV is that it is the most mature communication standard implementation in ns-3. Moreover, it is specifically designed for 'dynamic' networks such a MANETs where the topology change over time. There are efforts on modeling low-power and lossy network protocols such as RPL for IoT, but they are not fully developed yet. Hence, we propose to extend our approach to such protocols in future work.

8. Conclusion

In this paper, we explored the problem of maintaining IoT device reliability from the perspective of network routing. The literature on network routing up to date did not study hardware related reliability issues. We proposed a distributed reinforcement learning based routing approach to improve network MTTF, which learns to make its decisions based on the current reliability of the nodes, the amount of degradation they will experience, and networking performance. We extended the ns-3 AODV protocol with a Q-learning algorithm and demonstrated improved network MTTF compared to AODV, AODV-ETX, and Q-AODV methods. Our results show up to a 73.2% improvement in reliability for various communication data rates and the number of nodes in the network while delivering comparable performance.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was partially supported by SRC task #2805.001, NSF grants #1911095, #1826967, #1730158 and #1527034, and by KACST.

References

[1] IDC, The growth in connected IoT devices, 2019, [Online].
[2] Cisco Jasper, The hidden costs of delivering IIoT services: Industrial monitoring & heavy equipment, 2016.
[3] A. Kansal, J. Hsu, S. Zahedi, M.B. Srivastava, Power management in energy harvesting sensor networks, ACM Trans. Embed. Comput. Syst. (TECS) 6 (4) (2007).
[4] J.W. McPherson, Reliability challenges for 45nm and beyond, in: 43rd ACM/IEEE Design Automation Conference, 2006.
[5] T.S. Rosing, K. Mihic, G. De Micheli, Power and reliability management of SoCs, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 15 (4) (2007) 391–403.
[6] C. Zhuo, D. Sylvester, D. Blaauw, Process variation and temperature-aware reliability management, in: Proceedings of the Conference on Design, Automation and Test in Europe, European Design and Automation Association, 2010, pp. 580–585.
[7] T.M. Behera, S.K. Mohapatra, U.C. Samal, M.S. Khan, M. Daneshmand, A.H. Gandomi, Residual energy-based cluster-head selection in WSNs for IoT application, IEEE Internet Things J. 6 (3) (2019) 5132–5139.
[8] A. Dhumane, R. Prasad, J. Prasad, Routing issues in internet of things: a survey, in: Proceedings of the International Multiconference of Engineers and Computer Scientists, 2016.
[9] M. Zhao, I.W.-H. Ho, P.H.J. Chong, An energy-efficient region-based RPL routing protocol for low-power and lossy networks, IEEE Internet Things J. 3 (6) (2016).
[10] R. Baumann, S. Heimlicher, M. Strasser, A. Weibel, A survey on routing metrics, TIK Rep. 262 (2007) 1–53.
[11] M.E.M. Campista, P.M. Esposito, I.M. Moraes, L.H.M. Costa, O.C.M. Duarte, D.G. Passos, C.V.N. De Albuquerque, D.C.M. Saade, M.G. Rubinstein, Routing metrics and protocols for wireless mesh networks, IEEE Netw. 22 (1) (2008) 6–12.
[12] Z. Mammeri, Reinforcement learning based routing in networks: Review and classification of approaches, IEEE Access 7 (2019) 55916–55950.
[13] A. Valadarsky, M. Schapira, D. Shahaf, A. Tamar, Learning to route, in: Proceedings of the 16th ACM Workshop on Hot Topics in Networks, 2017.
[14] D.S. De Couto, D. Aguayo, J. Bicket, R. Morris, A high-throughput path metric for multi-hop wireless routing, in: Proceedings of the 9th Annual International Conference on Mobile Computing and Networking, 2003, pp. 134–146.
[15] NREL, National Solar Radiation Database (NSRDB), https://maps.nrel.gov/nsrdb-viewer, [Online].

[16] High Performance Wireless Research and Education Network (HPWREN), http://hpwren.ucsd.edu/.
[17] The ns-3 Network Simulator, https://www.nsnam.org/, [Online].
[18] S. Baskar, V. Dhulipala, Comparative analysis on fault tolerant techniques for memory cells in wireless sensor devices, Asian J. Res. Soc. Sci. Humanit. 6 (cs1) (2016) 519–528.
[19] J. Yao, N. Ansari, Fog resource provisioning in reliability-aware IoT networks, IEEE Internet Things J. 6 (5) (2019) 8262–8269.
[20] I. Dietrich, F. Dressler, On the lifetime of wireless sensor networks, ACM Trans. Sensor Netw. (2009).
[21] J.-H. Chang, L. Tassiulas, Maximum lifetime routing in wireless sensor networks, IEEE/ACM Trans. Netw. (2004).
[22] Y. Peng, W. Qiao, L. Qu, J. Wang, Sensor fault detection and isolation for a wireless sensor network-based remote wind turbine condition monitoring system, IEEE Trans. Ind. Appl. 54 (2) (2017) 1072–1079.
[23] P. Mercati, F. Paterna, A. Bartolini, L. Benini, T.Š. Rosing, Warm: Workload-aware reliability management in linux/android, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 36 (9) (2016) 1557–1570.
[24] K. Ergun, R. Ayoub, P. Mercati, T. Rosing, Improving mean time to failure of IoT networks with reliability-aware routing, in: 2021 10th Mediterranean Conference on Embedded Computing (MECO), IEEE, 2021, pp. 1–4.
[25] M. Handy, M. Haase, D. Timmermann, Low energy adaptive clustering hierarchy with deterministic cluster-head selection, in: 4th International Workshop on Mobile and Wireless Communications Network, IEEE, 2002, pp. 368–372.
[26] Y. Yu, R. Govindan, D. Estrin, Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks, 2001.
[27] A. Sarkar, T.S. Murugan, Routing protocols for wireless sensor networks: What the literature says? Alex. Eng. J. 55 (4) (2016) 3173–3183.
[28] J.V. Sobral, J.J. Rodrigues, R.A. Rabêlo, J. Al-Muhtadi, V. Korotaev, Routing protocols for low power and lossy networks in internet of things applications, Sensors (2019).
[29] D. Zhang, G. Li, K. Zheng, X. Ming, Z.-H. Pan, An energy-balanced routing method based on forward-aware factor for wireless sensor networks, IEEE Trans. Ind. Inf. 10 (1) (2013) 766–773.
[30] S.B. Shah, Z. Chen, F. Yin, I.U. Khan, N. Ahmad, Energy and interoperable aware routing for throughput optimization in clustered IoT-wireless sensor networks, Future Gener. Comput. Syst. 81 (2018) 372–381.
[31] R. Ullah, Y. Faheem, B.-S. Kim, Energy and congestion-aware routing metric for smart grid AMI networks in smart city, IEEE Access 5 (2017) 13799–13810.
[32] A.A. Rahat, R.M. Everson, J.E. Fieldsend, Hybrid evolutionary approaches to maximum lifetime routing and energy efficiency in sensor mesh networks, Evol. Comput. 23 (3) (2015) 481–507.
[33] X. Li, B. Keegan, F. Mtenzi, T. Weise, M. Tan, Energy-efficient load balancing ant based routing algorithm for wireless sensor networks, IEEE Access (2019).
[34] Z. Wadud, N. Javaid, M.A. Khan, N. Alrajeh, M.S. Alabed, N. Guizani, Lifetime maximization via hole alleviation in iot enabling heterogeneous wireless sensor networks, Sensors 17 (7) (2017) 1677.
[35] C.E. Perkins, E.M. Royer, Ad-hoc on-demand distance vector routing, in: Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications, IEEE, 1999, pp. 90–100.
[36] T. Clausen, P. Jacquet, C. Adjih, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, L. Viennot, Optimized link state routing protocol (OLSR), 2003.
[37] T. Winter, P. Thubert, A. Brandt, J.W. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J.-P. Vasseur, R.K. Alexander, et al., RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks, Rfc 6550, 2012, pp. 1–157.
[38] E. Yitayal, J.-M. Pierson, D. Ejigu, A balanced battery usage routing protocol to maximize network lifetime of MANET based on AODV, in: International Conference on Next Generation Wired/Wireless Networking, Springer, 2014, pp. 266–279.
[39] P.O. Kamgueu, E. Nataf, T.D. Ndié, O. Festor, Energy-based routing metric for RPL, 2013.
[40] J.-M. Kim, J.-W. Jang, AODV based energy efficient routing protocol for maximum lifetime in MANET, in: Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services (AICT-ICIW'06), IEEE, 2006, p. 77.
[41] O. Iova, F. Theoleyre, T. Noel, Improving the network lifetime with energy-balancing routing: Application to RPL, in: 2014 7th IFIP Wireless and Mobile Networking Conference (WMNC), IEEE, 2014, pp. 1–8.
[42] O. Iova, F. Theoleyre, T. Noel, Using multiparent routing in RPL to increase the stability and the lifetime of the network, Ad Hoc Netw. 29 (2015) 45–62.
[43] J.A. Boyan, M.L. Littman, Packet routing in dynamically changing networks: A reinforcement learning approach, in: Advances in Neural Information Processing Systems, 1994, pp. 671–678.
[44] W. Naruephiphat, W. Usaha, Balanced energy-efficient routing in MANETs using reinforcement learning, in: 2008 International Conference on Information Networking, IEEE, 2008, pp. 1–5.
[45] D. Macone, G. Oddi, A. Pietrabissa, MQ-Routing: Mobility-, GPS-and energy-aware routing protocol in MANETs for disaster relief scenarios, Ad Hoc Netw. 11 (3) (2013) 861–878.
[46] Y.-H. Chang, T. Ho, L.P. Kaelbling, Mobilized ad-hoc networks: A reinforcement learning approach, in: International Conference on Autonomic Computing, 2004. Proceedings., IEEE, 2004, pp. 240–247.
[47] C. Wu, K. Kumekawa, T. Kato, A MANET protocol considering link stability and bandwidth efficiency, in: 2009 International Conference on Ultra Modern Telecommunications & Workshops, IEEE, 2009, pp. 1–8.
[48] K. Wang, W.-C. Wong, T.Y. Chai, A MANET routing protocol using Q-learning method integrated with Bayesian network, in: 2012 IEEE International Conference on Communication Systems (ICCS), IEEE, 2012, pp. 270–274.
[49] S. Chettibi, S. Chikhi, Dynamic fuzzy logic and reinforcement learning for adaptive energy efficient routing in mobile ad-hoc networks, Appl. Soft Comput. 38 (2016) 321–328.
[50] A. Kozłowski, J. Sosnowski, Energy efficiency trade-off between duty-cycling and wake-up radio techniques in IoT networks, Wirel. Pers. Commun. (2019).
[51] K. Ergun, X. Yu, N. Nagesh, L. Cherkasova, P. Mercati, R. Ayoub, T. Rosing, RelIoT: Reliability simulator for IoT networks, in: Internet of Things - ICIOT, 2020.
[52] A. Grenat, S. Sundaram, S. Kosonocky, R. Rachala, S. Sambamurthy, S. Liepe, M. Rodriguez, T. Burd, A. Clark, M. Austin, et al., 4.2 Increasing the performance of a 28nm x86-64 microprocessor through system power management, in: ISSCC, IEEE, 2016.
[53] R. Baranowski, A. Cook, M.E. Imhof, C. Liu, H.-J. Wunderlich, Synthesis of workload monitors for on-line stress prediction, in: 2013 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS), IEEE, 2013, pp. 137–142.
[54] K. Takeuchi, M. Shimada, S. Konishi, D. Oshida, N. Ota, T. Yasumasu, K. Shibutani, T. Iwashita, T. Kokubun, F. Tsuchiya, Experimental implementation of 8.9 kgate stress monitor in 28nm MCU along with safety software library for IoT device maintenance, in: 2019 IEEE International Reliability Physics Symposium (IRPS), IEEE, 2019.
[55] P. Mercati, A. Bartolini, F. Paterna, T.S. Rosing, L. Benini, Workload and user experience-aware dynamic reliability management in multicore processors, in: 2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC), IEEE, 2013.
[56] H. Sharma, A. Haque, Z.A. Jaffery, Maximization of wireless sensor network lifetime using solar energy harvesting for smart agriculture monitoring, Ad Hoc Netw. 94 (2019) 101966.
[57] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.
[58] C.J. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (3–4) (1992) 279–292.
[59] N.J. Jevtic, M.Z. Malnar, Novel ETX-based metrics for overhead reduction in dynamic ad hoc networks, IEEE Access (2019).
[60] H. Li, D. Yu, A statistical study of neighbor node properties in ad hoc network, in: Proceedings. International Conference on Parallel Processing Workshop, IEEE, 2002, pp. 103–108.
[61] D. Pandey, P. Pandey, Approximate Q-learning: An introduction, in: 2010 Second International Conference on Machine Learning and Computing, IEEE, 2010, pp. 317–320.
[62] J. Fan, Z. Wang, Y. Xie, Z. Yang, A theoretical analysis of deep Q-learning, in: Learning for Dynamics and Control, PMLR, 2020, pp. 486–489.
[63] P. Xu, Q. Gu, A finite-time analysis of Q-learning with neural network function approximation, in: International Conference on Machine Learning, PMLR, 2020, pp. 10555–10565.
[64] Santa Margarita Ecological Reserve (SMER), https://fsp.sdsu.edu//.
[65] W. Cui, Y. Kim, T.S. Rosing, Cross-platform machine learning characterization for task allocation in IoT ecosystems, in: 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), IEEE, 2017, pp. 1–7.
[66] S. Chettibi, S. Chikhi, Adaptive maximum-lifetime routing in mobile ad-hoc networks using temporal difference reinforcement learning, Evol. Syst. 5 (2) (2014) 89–108.

Kazim Ergun is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA, USA. He is a Member of the System Energy Efficiency Lab, where he works on optimization and control for the Internet of Things, with the focus on reliability and energy efficiency. His research interests also include edge computing, wireless communications, and machine learning.

Raid Ayoub received the Ph.D. degree in computer engineering from the Department of Computer Science and Engineering, University of California, San Diego, La Jolla, in 2011. He is currently a research scientist with Intel labs. His current research interests include online optimizations and control, high-level system modeling with a focus on communications fabric, Internet-of-things, machine learning, design automation, and system energy efficiency.

Pietro Mercati is a Research Scientist in Intel Labs. He obtained his Ph.D. in Computer Science in 2017 from the University of California San Diego. His research is in hardware and software design and optimization for energy-efficient and reliable computing. He is interested in mathematical optimization, machine learning, control theory, algorithms, Bayesian methods.

Tajana Šimunić Rosing received the M.S. degree in engineering management concurrently with the Ph.D. degree from Stanford University, Stanford, CA, USA, in 2001 with the Ph.D. topic "Dynamic Management of Power Consumption". She is a Professor, a Holder of the Fratamico Endowed Chair, and the Director of System Energy Efficiency Lab, University of California San Diego, La Jolla, CA, USA. From 1998 to 2005, she was a full-time Research Scientist with HP Labs, Palo Alto, CA, USA, while also leading research efforts with Stanford University. She was a Senior Design Engineer with Altera Corporation, San Jose, CA, USA. She is leading a number of projects, including efforts funded by DARPA/SRC JUMP CRISP program with focus on design of accelerators for analysis of big data, DARPA and NSF funded projects on hyperdimensional computing and SRC funded project on IoT system reliability and maintainability. Her current research interests include energy efficient computing, cyber–physical, and distributed systems.