QELAR: A Machine-Learning-Based Adaptive Routing Protocol for Energy-Efficient and Lifetime-Extended Underwater Sensor Networks

Tiansi Hu, *Student Member*, *IEEE*, and Yunsi Fei, *Member*, *IEEE*

Abstract—Underwater sensor network (UWSN) has emerged in recent years as a promising networking technique for various aquatic applications. Due to specific characteristics of UWSNs, such as high latency, low bandwidth, and high energy consumption, it is challenging to build networking protocols for UWSNs. In this paper, we focus on addressing the routing issue in UWSNs. We propose an adaptive, energy-efficient, and lifetime-aware routing protocol based on reinforcement learning, QELAR. Our protocol assumes generic MAC protocols and aims at prolonging the lifetime of networks by making residual energy of sensor nodes more evenly distributed. The residual energy of each node as well as the energy distribution among a group of nodes is factored in throughout the routing process to calculate the reward function, which aids in selecting the adequate forwarders for packets. We have performed extensive simulations of the proposed protocol on the Aqua-sim platform and compared with one existing routing protocol (VBF) in terms of packet delivery rate, energy efficiency, latency, and lifetime. The results show that QELAR yields 20 percent longer lifetime on average than VBF.

Index Terms—Routing protocols, distributed networks, wireless communication, mobile communication systems.

1 Introduction

Underwater sensor network (UWSN) is an emerging and promising networking technique which has attracted more and more attention in recent years. It facilitates humans to sense and monitor the vast unexplored underwater domain of the earth. UWSNs can be employed in a wide spectrum of aquatic applications [1], such as environmental observation for scientific exploration [2], coastline surveillance and protection, commercial exploitation, disaster prevention, assisted navigation [3], and mine detection [4].

Deploying UWSNs, however, is much more difficult than the better investigated terrestrial sensor networks due to the harsh environment and high system costs. Although there exist many mature protocols for terrestrial sensor networks, they cannot be applied directly to UWSNs because of the fundamental differences in both the communication medium and network features.

Communications in UWSNs have to be done through acoustic channels, because electromagnetic radio signals attenuate quickly in water. The speed of sound in water (approximately 1.5×10^3 m/s) is five-order slower than the speed of light (3×10^8 m/s), which brings long propagation latency and end-to-end delay to networking services. The bandwidth of an acoustic channel is low (up to 20 kbps), and the error rate is high [5]. These physical limitations render many traditional networking protocols for terrestrial sensor networks, such as carrier sensing in media access control (MAC), table-based routing, and transporting, infeasible for UWSNs. In addition, time synchronization and localization services, on which geographical routing protocols are based, are generally not available in underwater acoustic networks. Moreover, many aquatic applications, like long-term estuary monitoring, rely on mobile underwater sensor networks to achieve in-situ monitoring and observations. The mobility of underwater sensor nodes in such three-dimensional context results in dynamically changing network topologies, which compounds the design of networking protocols and algorithms for UWSNs.

Energy efficiency has also been a major design concern for sensor networks. In underwater environment, because of the much higher transmission and receiving power consumptions of acoustic modems (e.g., 10 W and 3 W of transmission and receiving power, respectively, for the popular WHOI Micro-Modem [6] versus 105 mW and 54 mW for the Crossbow Mica2 [7]), and the harsh environment for sensor battery replacement and energy harvesting, energy efficiency becomes even more critical and challenging. For a sensor network to fulfill its duty for as long as possible, e.g., for long-term environment monitoring application, lifetime-awareness should be considered as well in the early design stage of UWSNs. Network lifetime is the time span from the deployment of a network to the instant when the network is considered nonfunctional [8]. There are several types of lifetime definitions, such as network lifetime based on the number of alive nodes, based on sensor coverage, and based on connectivity. We evaluate our routing protocol by two commonly used demarcation in

- *The authors are with the Department of Electrical and Computer Engineering, University of Connecticut, 371 Fairfield Way, U-2157, Storrs, CT 06269-2157. E-mail: {tiansi, yfei}@engr.uconn.edu.*

Manuscript received 5 Dec. 2008; revised 8 June 2009; accepted 10 Sept. 2009; published online 3 Feb. 2010.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2008-12-0489.
Digital Object Identifier no. 10.1109/TMC.2010.28.

1536-1233/10/$26.00 © 2010 IEEE Published by the IEEE CS, CASS, ComSoc, IES, & SPS

the paper, the time when the first sensor node dies and the time when the network gets partitioned.

In view of the dynamic nature of UWSNs, their long propagation delay and stringent energy resource, an UWSN should be self-configuring and adaptive to the environment in order to minimize the energy consumption and extend the network lifetime, through revolutionary redesign at every level of the networking protocol stack.

Typical routing protocols in ad hoc networks employ shortest path algorithms based on either hop count or energy consumption to achieve low latency, high throughput, and high energy efficiency. However, low energy consumption for networking services does not necessarily lead to long lifetime, in that not only the total amount of energy consumption, but the distribution of energy consumption across network nodes determines the network lifetime as well. Hence, to prolong the network lifetime, traffic in the network should be balanced well among sensor nodes to avoid hot spots in the network, i.e., sensor nodes that are more frequently used. Otherwise, the hot spot nodes will be drained faster than other parts of the network, and the network will be partitioned with "void" area and stop functioning.

1.1 Related Work

We next briefly review related work on routing protocols along four lines, traditional algorithms, energy-efficient and lifetime-aware algorithms, specific designs for underwater environment, and machine learning-based protocols.

Conventional routing protocols developed for ad hoc and sensor networks can be classified into three different categories, which are routing table-based proactive, on-demand reactive, and geographical protocols [1]. Route discovery and maintenance in both proactive and reactive routing protocols are costly, especially in the context of underwater networks with large delay and limited bandwidth. In geographical routing protocols, nodes use location information for packet delivery in multihop networks [9]. However, in UWSNs, it is difficult to obtain the position information efficiently and dynamically.

As battery energy capacity is a major constraint for sensor networks, many research efforts have focused on energy efficiency and lifetime of sensor networks. BAR [10] is a battery-aware routing protocol based on the observation that the battery can provide more energy capacity if it is not used continuously. Hence, the lifetime of a network is prolonged by allowing interleaving recovery time during battery usage. However, it assumes geographical information known to each node, which is difficult to get in UWSNs. There also exist routing protocols for maximizing the lifetime of sensor networks, including linear optimization [11], [12], [13] as well as nonlinear optimization [14]. The algorithm presented in [11] is centralized, which is too costly to be applied in realistic UWSNs. Other algorithms in [12], [13] suffer from potential large performance degradation in routing path discovery. In [14], the lifetime is taken as a sort of resource of each node, which is optimally allocated using a utility-based distributed algorithm to make each node evenly used. However, the mathematical model does not extend to a practical protocol that can be implemented in a real wireless sensor network. Overall, most of the routing approaches mentioned above are not designed specifically for UWSNs, and may not overcome the unique fundamental difficulties in UWSNs.

Recently, there have emerged some works on UWSN-specific routing protocols. Vector-based-forwarding (VBF) is a geographic routing approach [15], where the position information of source, sink, and intermediate forwarders is attached to each packet. On receiving a packet, each node calculates its own position based on the location of the predecessor node embedded in the packet and the angel of arrival (AOA) of the signal. Only sensor nodes that fall in a "routing pipe" centered around the source-sink vector are eligible for forwarding. The VBF protocol requires a special device on every sensor node that measures the angle of incoming signals. The method of position computation assumes a static network. Moreover, the nodes in the pipe will become hot spots and drain much faster than those outside the pipe, resulting in uneven usage of sensor nodes and unbalanced workload, and thus, will shorten the network lifetime. In [16], a traffic-aware routing protocol for underwater 3D geographic routing is presented. The relative location of each undersea sensor node can be computed via the exchanges of acoustic beacons between neighbors. A traffic-aware routing tree algorithm is proposed based on the gravity between two neighboring node clusters, which is a function of the distance and the virtual bandwidth between them. Although the algorithm achieves higher throughput and lower latency compared to other geographical protocols, the assumption of geographic information is not always valid in underwater environment. Depth-based-routing (DBR) [17] is another geographic routing protocol for UWSNs. In DBR, with the depth of each node known from pressure sensors, a greedy algorithm is employed to forward packets along the upwards direction from the bottom to any surface node, and relay them to the land destination via radios on surface nodes. The packet delivery rate depends upon the node distribution and the number of sinks. A packet of a node will be dropped if there is no sink around to receive it. In addition, the assumption that the packets only need to be delivered from bottom to surface does not always hold, e.g., in a sensor network where sinks need to send command packets or interested data query to each sensor node (i.e., down-link data). In [18], an energy-efficient routing protocol for UWSNs is presented. In its network setting, each sensor is anchored to the bottom of the ocean and equipped with a floating buoy that can control the depth of the sensor. The network is layered, and the sink is on the surface of the ocean. Nodes can only communicate to the ones in the neighbor layers, and all the packets are only sent to the sink, which limits its application. In [19], a resilient routing algorithm for long-term underwater monitoring applications is presented, which consists of two phases. The first phase discovers optimal node-disjoint primary and backup multihop data paths to minimize the energy consumption. In the second phase, an online distributed algorithm monitors the network and switches to the backup paths in case of severe failures. This protocol, however, is limited to long-term applications, and assumes that all the nodes are fixed and anchored to the bottom of sea. Focused Beam Routing (FBR) is presented in [20], which is a cross-later approach and designed for networks containing both static and mobile nodes. The source node must be aware of its own location and the location of its final destination.

There are also some routing protocols based on *Machine Learning* techniques. Q-Routing, which is designed for packet routing in telephone networks, is one of such early works based on Q-learning technique [21]. It finds the best routing path by considering not only the number of hops to the destination, but also the contribution of traffic jams, and hence, queue length of each node to the latency. The simulation shows that under high network load, the algorithm outperforms the shortest-path algorithm, which considers only the number of hops to the destination. It also performs well under changing network topologies. Since it uses only local information at each node and is fully distributed, it can be adapted to wireless networks as well. An algorithm named *Dual Reinforcement Q-Routing* is later presented to make the algorithm converge faster [22].

AdaR [23] is a self-configured routing protocol for wireless sensor networks based on reinforcement learning. However, in this protocol, packets are delivered from sensors to a centralized base station to learn the environment, which is costly to apply in the context of UWSNs. When one node forwards a packet to another according to the current routing policy, both the action and associated reward are appended to the packet. When the base station receives the packet which contains samples along the routing path, the base station calculates new weights in the linear quality function (Q) and broadcasts them throughout the whole network to improve the forwarding policy.

Ant colony optimization (ACO) is another *Machine Learning* technique used in routing for mobile ad-hoc networks. ACO is inspired by the nature of ants that exchange information with each other by laying down pheromone on their way back to the nest when they have found food. In ACO routing protocols, packets act like artificial ants that put pheromone on the nodes they visited from source (nest) to sink (food) so as to achieve optimal paths for the following packets. AntNet [24] is a proactive ACO-based algorithm, in which every node in the network generates a forward ant packet at regular intervals. After the forward ant reaches the destination, a backward ant packet is dispatched and sent all the way back. ARA [25] is a reactive ACO-based algorithm which divides the routing process into route discovery phase and route maintenance phase. Both of the protocols achieve a high delivery rate and almost shortest path. However, they also inherit disadvantages from proactive and reactive routing protocols. As a proactive routing protocol, AntNet introduces too much overhead in that nodes regularly send ant packets; on the other hand, the route discovery phase is long for ARA. In [26], a geographical ACO-based routing protocol POSANT is presented. However, the assumption of GPS being available does not hold in UWSNs.

1.2 Paper Overview and Contributions

In this paper, we propose an adaptive, energy-efficient, and lifetime-aware routing protocol, QELAR, to address the issues mentioned above based on Q-learning technique, which is a reinforcement learning technique that solves decision problems. By learning the environment and evaluating an action-value function (Q-value), which gives the expected reward of taking an action in a given state, the distributed learning agent is able to make a decision automatically.

We find that Q-learning can perform well in UWSNs in the following aspects to utilize or overcome the important properties of UWSNs:

- **Low Overhead**. In the underwater environment with low bandwidth, high delay, and high acoustic communication power consumption, it is costly to broadcast data throughout the network. Our proposed routing protocol is a hybrid of proactive and reactive protocol, where nodes only keep the routing information of their direct neighbor nodes which is a small subset of the network, so that the drawbacks of both proactive and reactive routing protocols are avoided. The routing information is updated by one-hop broadcasts rather than flooding.
- **Dynamic Network Topology**. In the harsh underwater environment, link quality is not guaranteed. A node in the transmission range may not be available due to link corruption, which results in topology change. Meanwhile, the movement of nodes also causes topology changes. Without GPS available underwater, a routing protocol in such an environment should be able to adapt to the topology changes. As we can see later, Q-learning algorithm learns from the network environment, and allows a fast adaptation to the current network topology.
- **Load Balance**. Shortest path routing algorithms reduce both the energy consumption and end-to-end delay by enforcing packets to go along the shortest path. However, nodes on the shortest path will drain more quickly than others, so that shortest-path algorithms may reduce the lifetime of the network. In contrast, after we take nodes energy into consideration in Q-learning, alternative paths can be chosen to use network nodes in a fair manner.
- **General Framework**. Q-learning is a framework that can be easily extended. Its behavior is largely determined by the reward function, which is used to give a positive or negative reinforcement to the agent after it makes a decision. By judiciously designing the reward function, we make our routing protocol energy-efficient and lifetime-aware. The optimal policy (routing forwarders selection) is derived through iterative Q-learning of the residual energy of each node and the energy distribution among a group of nodes. Meanwhile, we can easily integrate other factors such as end-to-end delay and node density for extension and can balance all the factors according to our need by tuning the parameters in the reward function.

More specifically, in our Q-learning-based routing algorithm, the energy consumption of sensor nodes and residual energy distribution among groups of nodes are novel considerations that are put into the reward function. With such a reward function, the forwarding policy can be improved at runtime, so as to achieve high energy efficiency and uniform energy distribution, and thus, prolong the network lifetime. The effects of different weights in the reward function on the network performance are thoroughly evaluated in simulations. Without any assumption of the underlying MAC, the routing protocol monitors the environment closely and learns to extract needed information, such as the network topology and residual energy of surrounding nodes, since nodes are able to directly exchange the information with its one-hop neighbor nodes to aid each node in choosing adequate forwarders for the next hop. The overhead of information exchange is small according to our analysis and experiments. The Q-learning process can keep up with dynamic changes of the network. Thus, the proposed routing protocol coping with the mobility of sensor nodes specifically is suitable for mobile underwater sensor networks. In our simulations, for typical marine science applications where the maximum velocity of nodes driven by water is 2-3 knots (1-1.5 m/s) [27], QELAR is able to achieve high energy efficiency and long lifetime.

The major contribution of our approach is that for the first time, we apply the Q-learning technique in a distributed routing protocol for UWSNs to balance workload among sensor nodes for longer network lifetime, to reduce networking overhead for higher energy efficiency, and to learn the environment effectively and efficiently for better adaptability to dynamic networks.

The remainder of this paper is organized as follows: Section 2 gives a brief introduction on reinforcement learning technique and formulates our routing problem in UWSNs. In Section 3, we describe the proposed routing algorithm in detail. We demonstrate the evaluation results in Section 4, followed by conclusions drawn in Section 5.

2 Reinforcement Learning Technique and System Modeling

In this section, we first give a brief introduction of general *Reinforcement Learning* technique (RL), which is the theoretical basis for our protocol. We then describe the Q-learning algorithm, and adopt it in our routing protocol. We map the routing problem to the state space formulated in the general framework of RL. A critical component of RL, the reward function, is proposed in our routing protocol, and we analyze how each reward parameter may affect the behavior of the protocol.

2.1 Reinforcement Learning Technique

The technique of *Reinforcement Learning* (RL) provides us a framework, by which a system can learn to achieve a goal in control problems based on its experience. Fig. 1 depicts the framework of RL. An agent in RL chooses actions according to the current state of a system and the reinforcement it receives from the environment. Most RL algorithms are based on estimating *value functions*, functions of states (or of state-action pairs), which evaluate *how good* it is for the agent to be in a given state (or how good it is to perform an action in a given state) [28].

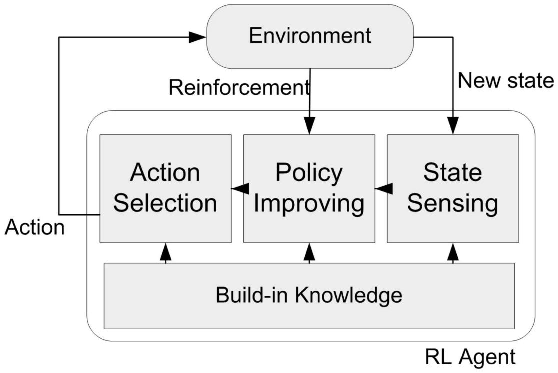


Fig. 1. The framework of reinforcement learning.

A reinforcement learning task satisfying the Markov property is called a Markov Decision Process (MDP) [28], which is of great importance to understand the concept of RL. A tuple of (S, A, P, R) is used to describe the characteristics of a particular MDP, where S, A, P, R are the set of states, actions, state transition probabilities, and rewards, respectively. P is composed by $P^a_{ss'}$, which is the probability of going to the next state s' from state s with a given action a, as shown in (1), and $R^a_{ss'}$ is the associated direct reward expected from the environment for taking the action a at s and entering s'.

$$P^a_{ss'} = Pr\{s_{t+1} = s'|_{s_t=s,a_t=a}\}, \tag{1}$$

where

$$\sum_{s'\in S} P^a_{ss'} = 1.$$

A direct reward is received after taking an action from state s_t at time t:

$$\begin{aligned} r_t &= r(s,a)|_{s=s_t,a=a_t} \\ &= \sum_{s_{t+1}\in S} P^{a_t}_{s_t s_{t+1}} R^{a_t}_{s_t s_{t+1}}. \end{aligned} \tag{2}$$

In the framework of RL, a policy π is mapping from each state, $s \in S$, and action, $a \in A(s)$, to the probability $\pi(s,a)$ of taking action a when in state s. The *value* of a state s under a policy π, denoted as $V^\pi(s)$, is defined as the expected total reward that can be received by the agent under the policy:

$$\begin{aligned} V^\pi(s) &= E_\pi\{R_t|_{s_t=s}\} \\ &= E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k}\bigg|_{s_t=s}\right\}, \end{aligned} \tag{3}$$

where $E_\pi\{\}$ denotes the expected value under the policy π, which means the total reward that can be received after taking an action according to policy π at time t and following the policy π thereafter. In particular:

$$\begin{aligned} E_\pi\{r_t\} &= E_\pi\{r(s_t,a_t)\} \\ &= \sum_{a_t\in A} \pi(s_t,a_t) r_t \\ E_\pi\{r_{t+1}\} &= E_\pi\{r(s_{t+1},a_{t+1})\} \\ &= \sum_{a_t\in A} \pi(s_t,a_t) \left\{ \sum_{s_{t+1}\in S} P^{a_t}_{s_t s_{t+1}} \right. \\ &\qquad \left. \left[\sum_{a_{t+1}\in A} \pi(s_{t+1},a_{t+1}) r_{t+1} \right]\bigg|_{s_t=s} \right\} \\ &\cdots \quad \cdots \quad \cdots \end{aligned}$$

γ in (3) is the discount factor in the range of $[0,1)$, which is used to discount the rewards in the future, with the consideration that recent actions affect the current value more than future ones. γ determines how important the future rewards are. When γ is set to 0, the system only considers the current reward and it acts similarly to a greedy algorithm, but the local optimum does not necessarily lead to global optimum. When γ is set to 1, the system will strive for a long-term high reward. However, the future reward cannot be estimated accurately, and hence, good performance is not guaranteed. Therefore, to balance the two factors, the typical value of γ is within $[0.5, 0.99]$.

With the definitions above, we obtain the *optimal value* of a state, which can be derived if an optimal policy is used thereafter. The value function is unique and is the solution to the *Bellman Equations* [28]:

$$
\begin{aligned}
V^*(s) &= \max_\pi V^\pi(s) \\
&= \max_\pi E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k} \Bigg|_{s_t=s} \right\} \\
&= \max_\pi E_\pi \left\{ r_t + \sum_{k=1}^{\infty} \gamma^k r_{t+k} \Bigg|_{s_t=s} \right\} \\
&= \max_a \left[r_t + \gamma \sum_{s_{t+1}\in S} P^a_{s_t s_{t+1}} V^*(s_{t+1}) \Bigg|_{s_t=s} \right],
\end{aligned} \tag{4}
$$

where $P^a_{ss'}$ is defined in (1). It is the system model that captures the future system behavior.

2.2 Q-Learning Technique

Theoretically, plugging everything into (4) and solving it directly will obtain an optimal policy. However, it requires a lot of computing power, which typically can not be provided by the resource-constrained sensor nodes deployed underwater. There are some estimation methods that can yield near-optimal policies without much computations and the underlying model. Q-learning is such an off-policy temporal difference approach [23], in which the agent approximates the Q-values iteratively.

Q-learning is based on the value of state-action pairs $Q(s,a)$. 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 [28]:

$$
\begin{aligned}
&Q^\pi(s,a) \\
&\quad = E_\pi\{R_t|_{s_t=s,a_t=a}\} \\
&\quad = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k} \Bigg|_{s_t=s,a_t=a} \right\} \\
&\quad = E_\pi \left\{ r_t + \sum_{k=1}^{\infty} \gamma^k r_{t+k} \Bigg|_{s_t=s,a_t=a} \right\} \\
&\quad = \left(r_t + \gamma \sum_{s_{t+1}\in S} P^{a_t}_{s_t s_{t+1}} V^\pi(s_{t+1}) \right) \Bigg|_{s_t=s,a_t=a}.
\end{aligned} \tag{5}
$$

Thus, from (4) and (5), we can get:

$$
V^*(s) = \max_a Q^*(s,a), \tag{6}
$$

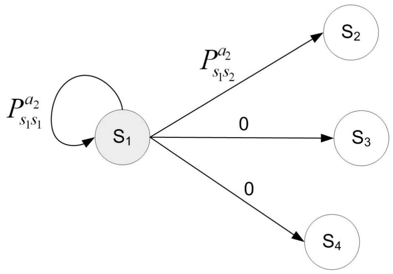


Fig. 2. Example of state transitions.

and from (5) and (6), we get

$$
Q^*(s_t,a_t) = r_t + \gamma \sum_{s_{t+1}\in S} P^{a_t}_{s_t s_{t+1}} \max_a Q^*(s_{t+1},a). \tag{7}
$$

$Q^*(s_t,a_t)$ denotes the expected reward that can be received by taking an action a_t at the state s_t, and following the optimal policy thereafter. It can be approximated by iterations:

$$
Q(s_t,a_t) \leftarrow (1-\alpha)Q(s_t,a_t) + \alpha\Big[r_t + \gamma \max_a Q(s_{t+1},a)\Big], \tag{8}
$$

where $\alpha \in (0,1]$ is the learning rate, which models the rate of updating Q-values.

Simply employing the model-free Q-learning in (8) will lead to slow convergence of $V^*(s) = \max_a Q^*(s,a)$, because in model-free Q-learning, we need to sufficiently frequently visit every state in the system. This is infeasible in network applications, because a single update of V needs enumerating all of the possible actions, e.g., forwarding packets to all the neighbors in routing. Typically, in a moderate network, V needs to be updated several times to get converged. Since the behavior of the network before convergence is undefined, the slow convergence of *V value* leads to undesired behaviors which result in large latency, especially for the first several packets. We observed that in a network with around 100 nodes, the first several packets took more than 100 hops before they were delivered.

To make V converge faster, we need to exploit the underlying system model, and use (7) directly. The advantage of this model-based Q-learning is that it allows each node to do virtual experiments by simply calculating Q values of all the actions based on the system model to update V, instead of really taking the forward actions to each of the neighbors, which is costly. We will give an example in Section 2.4.

2.3 State Space for the Routing Problem

In UWSN routing, we consider the whole network as a system. The definition of system states is related to individual packets. For a certain source-destination pair, let s be the node which holds a packet. For example, in an n-node network, when node s_1 has a packet, the system state related to the packet is s_1. Let $a_{s'}$ be the action that a node forwards the packet to node s'. All the states and the actions related to each node in the network makes up the set of states S and the set of actions A. The state of the system related to a packet transits only when the packet is forwarded from one node to another. Each forwarding action may succeed or fail. As shown in Fig. 2, s_1 may choose s_2 as the next forwarder, i.e., take action a_2 according to its policy. The success rate is $P^{a_2}_{s_1 s_2}$ and the failure rate is $P^{a_2}_{s_1 s_1} = 1 - P^{a_2}_{s_1 s_2}$.

In general, each action a_m issued by s_n has its associated $P^{a_m}_{s_n s_m}$ and $P^{a_m}_{s_n s_n} = 1 - P^{a_m}_{s_n s_m}$. Specially, if $m = n$, then $P^{a_m}_{s_n s_m} = 1$. Thus, the transition probabilities when the system is in state s_1 can be characterized by the matrix:

$$P(s_1) = \begin{array}{c} \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{array} \!\!\! \begin{array}{c} \begin{array}{cccc} s_1 & s_2 & \cdots & s_n \end{array} \\ \begin{pmatrix} P^{a_1}_{s_1 s_1} & 0 & \cdots & 0 \\ P^{a_2}_{s_1 s_1} & P^{a_2}_{s_1 s_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ P^{a_n}_{s_1 s_1} & 0 & \cdots & P^{a_n}_{s_1 s_n} \end{pmatrix} \end{array}. \qquad (9)$$

The transition probabilities as shown in (9), i.e., the underlying system model, are generally unknown. However, we can estimate it at runtime based on the history, which is explained in Section 3.

2.4 Lifetime-Aware Reward Function and Analysis

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 with maximum reward, i.e., minimum cost.

Consider the scenario that node s_n forwards a packet to s_m. If the forwarding attempt is successful, the reward function $R^{a_m}_{s_n s_m}$ for the state-action pair (s_n, a_m) is defined as:

$$\begin{aligned} R^{a_m}_{s_n s_m} &= -g - \alpha_1[c(s_n) + c(s_m)] \\ &\quad + \alpha_2[d(s_n) + d(s_m)], \end{aligned} \qquad (10)$$

where $c(s)$ and $d(s)$ are residual energy-related rewards which will be explained later, and α_1 and α_2 are their weights. g is the constant *punishment* or *cost* when a node attempts to forward a packet. It is based on the fact that each packet forwarding attempt consumes energy, occupies channel bandwidth, and contributes to the number of hops to the destination (i.e., delay). With this constant punishment, the agent is compelled to choose the relatively shorter paths to the destination, and hence, the routing delay is minimized. In a network, the farther an intermediate node is from the destination node in terms of number of hops, the more negative reward it would receive, and therefore, its *V value*: $V(s_n) = \max_a Q(s_n, a)$ is lower. As the greedy Q-learning algorithm always chooses the action with the highest Q, this *V value* gradient leads packets to be relayed from the source to the destination with minimum number of hops. Because of the importance of the term $-g$, its weight is set to be 1, and the other weights (α_1 and α_2) are all lower than 1.

However, shortest path does not guarantee longest lifetime, because overuse of some nodes on the shortest path will introduce unbalanced workload, and hence, shorten the network lifetime. As the energy distribution affects the lifetime of the whole network, we need to further differentiate sensor nodes by their energy characteristics. By using the nodes in a fair way, the residual energy of nodes will be more evenly distributed, which implies longer network lifetime.

Therefore, other two functions of nodes, $c(s_n)$ and $d(s_n)$, are introduced to the reward function to make the protocol lifetime-aware. $c(s_n)$ is a cost function of residual energy of node n, as defined below.

$$c(s_n) = 1 - \frac{E_{\text{res}}(s_n)}{E_{\text{init}}(s_n)}, \qquad (11)$$

where $E_{\text{res}}(s_n)$ and $E_{\text{init}}(s_n)$ are the residual energy and initial energy of node n, respectively. With the initial energy $E_{\text{init}}(s_n)$ the same for all sensor nodes, the less residual energy node n has, the higher cost $c(s_n)$ is. Higher cost means that the two nodes involved in the transmission are more reluctant to send and receive packets.

$d(s_n)$ is the reward of energy distribution in a group of sensor nodes, which includes the node n that holds a packet and all its direct neighbors in the transmission range. Let $E_{res}(s_n)$ be the residual energy of node n, and $\overline{E}(s_n)$ be the average residual energy in the group of node n. The reward of energy distribution in the group is defined as:

$$d(s_n) = \frac{2}{\pi}\arctan(E_{\text{res}}(s_n) - \overline{E}(s_n)). \qquad (12)$$

From (12), we can see that the larger the difference between the residual energy of a node and its group average, the more advantage for the node to be chosen as the next forwarder over other candidates.

By definition, both $c(s_n)$ and $d(s_n)$ are in the range of $[-1, 1]$, which enables us to balance all the parameters in (10) simply by tuning the weights only.

The weights of the terms in the reward function should be tuned such that it weighs most on the constant cost, because it can greatly reduce the number of hops to the destination as mentioned above. However, detouring routes with more hops should also be allowed, which lead to evenly distributed residual energy. According to the guideline and the experiments given in Section 4, we set $\alpha_1 = 0.5$ and $\alpha_2 = 0.05$.

If the packet forwarding attempt from s_n to s_m fails, the reward function is defined as:

$$R^{a_m}_{s_n s_n} = -g - \beta_1 c(s_n) + \beta_2 d(s_n), \qquad (13)$$

where g, $c(s_n)$ $d(s_n)$ are defined above, and β_1, β_2 are weights that we can tune. The guideline for choosing α_1 and α_2 can also be applied here. One possible setting can be $\beta_1 = 0.5$ and $\beta_2 = 0.05$.

With $R^{a_m}_{s_n s_m}$ and $R^{a_m}_{s_n s_n}$, we can define our reward function as:

$$r_t = P^{a_m}_{s_n s_m} R^{a_m}_{s_n s_m} + P^{a_m}_{s_n s_n} R^{a_m}_{s_n s_n}. \qquad (14)$$

From the definitions of $R^{a_m}_{s_n s_m}$ and $R^{a_m}_{s_n s_n}$, we can see that both these two functions are nonpositive, i.e., the agent will always get a negative reward after they forward packets. Therefore, the *V value*s of all the nondestination nodes are negative. The *V value* of the destination should be the largest among all the nodes, and thus, is set at 0.

The following example illustrates how Q-learning works. In the network shown in Fig. 3, the circles represent sensor nodes, and the edges represent direct links. Initially, let all the Q values and V values be 0, and we let $\gamma = 0.5$, $g = 1$. We do not consider the energy-related terms ($c(s)$ and $d(s)$) of the reward function, and let $P^{a_m}_{s_n s_m} = 1$ for the sake of simplicity.

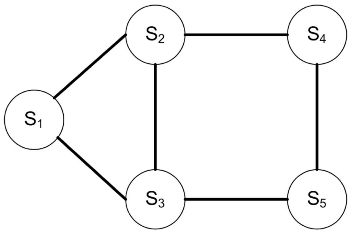


Fig. 3. A network topology.

Node 1 does virtual experiments by calculating:

$$\begin{aligned}Q(s_1,a_2) &= r_t + \gamma\big(P^{a_2}_{s_1s_2}V(s_2) + P^{a_2}_{s_1s_1}V(s_1)\big)\\ &= -1 + 0.5 \times V(s_2)\\ &= -1\end{aligned}$$

and similarly, $Q(s_1,a_3) = -1$. Since $Q(s_1,a_2) = Q(s_1,a_3)$, Node 1 randomly chooses a node, say Node 2, and updates its own $V(s_1) = \max_a Q(s_1,a)$ to -1, as Step I in Fig. 4. Node 2 then finds out that Node 4 is the destination, so it forwards the packet to Node 4 directly and updates its $V(s_2)$:

$$\begin{aligned}V(s_2) = Q(s_2,a_4) &= r_t + \gamma\big(P^{a_4}_{s_2s_4}V(s_4) + P^{a_4}_{s_2s_2}V(s_2)\big)\\ &= -1.\end{aligned}$$

as Step II in Fig. 4.

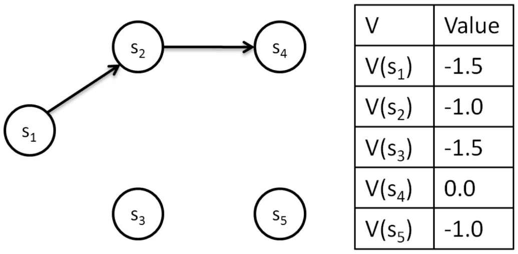
Before sending the second packet, Node 1 does virtual experiments again by calculating $Q(s_1,a_2)$ and $Q(s_1,a_3)$. Since $V(s_3)$ is not changed, $Q(s_1,a_3)$ still equals to -1 as seen by the previous packet, but

$$\begin{aligned}Q(s_1,a_2) &= r_t + \gamma\big(P^{a_2}_{s_1s_2}V(s_2) + P^{a_2}_{s_1s_1}V(s_1)\big)\\ &= -1 + 0.5 \times V(s_2)\\ &= -1.5.\end{aligned}$$

Because $Q(s_1,a_3)$ is larger than $Q(s_1,a_2)$, the second packet is forwarded to Node 3 as Step III in Fig. 4.

Similarly, Node 3 calculates $Q(s_3,a_1) = -1.5$, $Q(s_3,a_2) = -1.5$, and $Q(s_3,a_5) = -1.0$, so Node 3 forwards the packet to Node 5 as Step IV in Fig. 4. Node 5 finds Node 4 is the destination node and forwards the packet to it as Step V in Fig. 4.

2.5 Convergence of V Values



V	Value
V(s1)	-1.5
V(s2)	-1.0
V(s3)	-1.5
V(s4)	0.0
V(s5)	-1.0

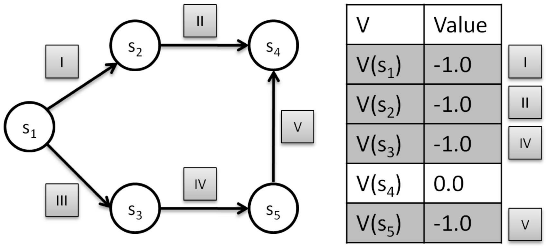
Fig. 5. Converged *V value*s with only the g term included in the reward function.

Because the *V value* of Node 4 is fixed to 0 by definition, the *V value*s of the nodes that are one hop away from Node 4, i.e., Node 2 and Node 5, converge to -1.0 according to (7), as shown in the given example. Before the *V values* of Node 2 and Node 5 converge, the *V values* of other nodes further away from Node 4 depend upon random actions taken by those nodes with nonconverged *V values*. However, after the convergence of the one-hop neighbors of the sink node, the nodes that are two hops away converge. In the given example, after Node 2 and Node 5 converge to -1.0, Node 3 has:

$$\begin{aligned}Q(s_3,a_2) &= r_t + \gamma\big(P^{a_2}_{s_3s_2}V(s_2) + P^{a_2}_{s_3s_3}V(s_3)\big)\\ &= -1 + 0.5 \times V(s_2)\\ &= -1.5.\end{aligned}$$

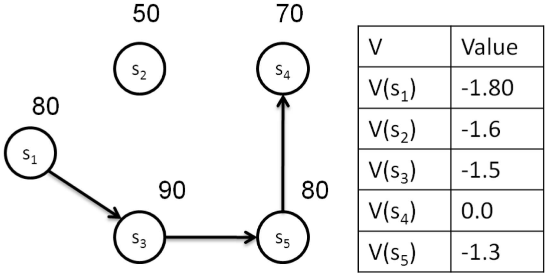
The similar thing happens if Node 3 attempts to forward packets to Node 5, so that the *V value* of Node 3 gets constant at -1.5. As the process goes on, the convergence of values will reversely propagate from the sink to the source. After several packets are delivered, as shown in Fig. 5, the *V value*s of Node 2 and 5 converge to -1.0, and the *V value*s of Node 1 and 3 converge to -1.5. As we can see it, converged *V value* of each node is proportional to its distance to the destination in terms of hop count. With the converged values, nodes are able to find an optimal path that can receive maximum reward. In the example, the path is Node 1-> Node 2-> Node 4, as shown in Fig. 5, for the fewest number of hops (when only the g term is considered in the reward function.)

With energy considered, things become a little more complicated, for the network does not always choose the shortest routing path. Fig. 6 demonstrates a snap-shot of the network during the routing process. The number besides each node denotes its residual energy level. As we can see that since Node 2 falls on the shortest path, it is used most often, and hence, has the lowest residual energy. Since a lower reward will be received if a packet is forwarded to a node with lower residual energy, the value of the state-action pair decreases. In Fig. 6, $Q(s_1,a_2)$ decreases as the



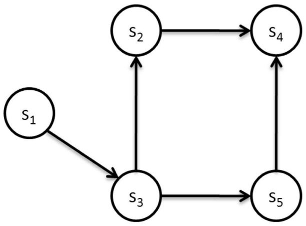
V	Value
V(s1)	-1.0
V(s2)	-1.0
V(s3)	-1.0
V(s4)	0.0
V(s5)	-1.0

Fig. 4. An example of model-based Q-learning.



V	Value
V(s1)	-1.80
V(s2)	-1.6
V(s3)	-1.5
V(s4)	0.0
V(s5)	-1.3

Fig. 6. Decision with energy considered in the reward function.



V	Value
V(s$_1$)	-1.75
V(s$_2$)	-1.0
V(s$_3$)	-1.5
V(s$_4$)	0.0
V(s$_5$)	-1.0

Fig. 7. Reconverged *V value*s after topology change without energy considered in the reward function.

residual energy of Node 2 decreases. After $Q(s_1, a_2)$ drops below $Q(s_1, a_3)$, Node 1 will forward the packet to Node 3 instead of Node 2. With the same reason, Node 3 forwards the packet to Node 5. Therefore, higher residual energy overrides the shorter path to achieve evenly distributed energy, and thus, longer network lifetime.

In UWSNs, nodes are typically moving. The mobility leads to network topology change. Even in static UWSNs, temporary losses of connectivity can be experienced due to high bit error rates in underwater environment. With various causes, the common net effect is that one neighbor which is previously in the transmission range can not be reached any more. The following example shows that if the topology changes, new converged *V value*s can be reestablished so as to adapt to the new topology.

For example, if the link between Node 1 and Node 2 is disrupted, Node 1 detects the failure (the failure detection algorithm will be introduced in the next section), and forwards the packet to Node 3, updating its *V value* to

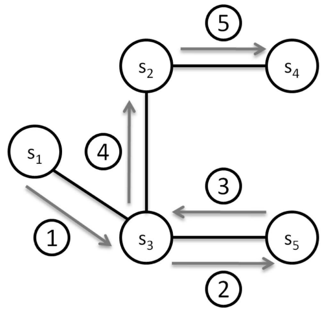
$$\begin{aligned} V(1) = Q(s_1, a_3) &= r_t + \gamma\big(P^{a_3}_{s_1 s_3} V(s_3) + P^{a_3}_{s_1 s_1} V(s_1)\big) \\ &= -1.75. \end{aligned}$$

Fig. 7 depicts the converged values after the topology change. The topology is changed to be: Node 2 and Node 5 are still the sink node's one-hop neighbors, Node 3 is the only two-hop neighbor, and Node 1, which used to be a two-hop neighbor, now becomes the three-hop neighbor with the s_1-s$_2$ link failure. After the reconvergence of the *V value*s, two possible shortest paths are Node 1 -> Node 3 -> Node 2 -> Node 4 and Node 1-> Node 3 -> Node 5 -> Node 4. In this scenario, Node 2 and Node 5 are in equal positions and can both be chosen by Node 3. They can be further differentiated by their energy characteristics, which we omitted in this example.

In certain scenario, a node cannot find the next forwarder, e.g., in sparse network, a backtracking procedure needs to be activated. We next consider the case where Node 5 could not reach Node 4, and a packet generated by Node 1 has been forwarded to Node 5 via Node 3. Node 5 notices that Node 3 is the only neighbor, so it has to forward the packet back to Node 3 and updates its value to:

$$\begin{aligned} V(5) = Q(s_5, a_3) &= r_t + \gamma\big(P^{a_3}_{s_5 s_3} V(s_3) + P^{a_3}_{s_5 s_5} V(s_5)\big) \\ &= -1.75 \end{aligned}$$

which is smaller than the *V Value* of Node 3's other neighbor—Node 2. Therefore, Node 3 forwards the packet to Node 2 instead of Node 5, and Node 2 sends the packet to Node 4 as shown in Fig. 8. The V value of Node 5 will prevent Node 3 from sending following packets to Node 5 again.



V	Value
V(s$_1$)	-1.75
V(s$_2$)	-1.0
V(s$_3$)	-1.5
V(s$_4$)	0.0
V(s$_5$)	-1.75

Fig. 8. An example of backtracking.

3 The Protocol Design

In this section, we describe several salient considerations for our routing protocol, including the packet structure, exchanging metadata among sensor nodes for selecting adequate forwarders, coping with transmission failures, and overhead analysis.

3.1 Packet Structure

The structure of packets in the network is depicted in Fig. 9. In the figure, the shaded part is packet header, which includes three parts: two packet-related fields, a tuple related to the previous forwarder and the routing decision made by the previous forwarder.

The two packet-related fields are

- *Packet ID*, the unique ID of the packet, and
- *Destination Address*, the address where the packet should be ultimately send to.

They are determined by the source node of the packet. Typically, they do not change during the entire lifetime of the packet.

Before forwarding the packet, a node should put its own information in the tuple shown in Fig. 9, which includes:

- *V Value*, the current *V value*,
- *Residual Energy*, the current residual energy,
- *Average Energy*, the average residual energy in its local group, and
- *Previous-hop Node*, the ID or address of the previous hop.

Upon receiving or overhearing a packet, every node retrieves the information of the previous forwarder from

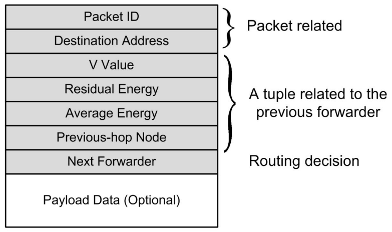


Fig. 9. Packet structure.

```
OnHearing(Packet p)
begin
  UpdateSenderInfo(p);
  if (isDataPacket(p))
    if (!isForwardedToMe(p))
      DropPacket(p);
    else
      Calculate $\overrightarrow{Q}$
      a = FindNeighborWithMax($\overrightarrow{Q}$)
      AttachLocalInformation(p);
      ForwardPacket(p, a, $Q_{max}$);
    end if
  end if
```

Fig. 10. The basic routing protocol.

the tuple and updates its own local copy, which helps them making correct routing decisions.

After making a routing decision, the node puts the address of the next forwarder to the fields *Next Forwarder*, so that the next forwarder indicated will route the packet again. Other neighbors overhearing the packet simply drop the packet after they get the information of the previous node as mentioned above.

Other than the packet header, the *Payload Data* is optional. It may contain metadata from upper-level protocols and user message. When the payload data is absent, the packet is used only for routing metadata exchanging, which will be discussed next.

3.2 Metadata Exchange

Equations (6)-(7) and (10)-(14) show that all we need for calculating the *Q value* for each state-action pair (s_n, a_m) are the *residual energy*, $E_{res}(s_n)$ and $E_{res}(s_m)$, *average energy of group*, $\overline{E}(s_n)$ and $\overline{E}(s_m)$, and the *V value* $V(s)$ of each neighbor node of s_n. The tuple of $(E_{res}(s_n), \overline{E}(s_n), V(s_n))$ of each node will be referred to as *metadata* below. To make the algorithm distributed, all the nodes are responsible for sending the local metadata to their neighbors.

Each node attaches its metadata to the data packets that it sends. Because every node in the UWSN is overhearing the traffic, it can obtain the metadata of its neighbors from the incoming packets even if the packets are not intended for it.

In addition, every node in the network should periodically broadcast control packets with no payload, which is a complementary method for metadata exchange. This kind of periodic one-hop broadcasts ensure that all nodes, including those that do not have data traffic, broadcast their metadata, so that neighbors can make correct decisions. The period should be set according to the network applications. Shorter period is desired in a more dynamic network and also in those short-term and real-time applications, however, more energy consumption can be anticipated due to the frequent broadcast.

Every node maintains a table for its known neighbors with records of their metadata, which is extracted from the packets heard. In UWSNs, the number of neighbors for a node may change due to unexpected node failures under certain circumstances or topology changes caused by node mobility. To keep the status information updated, entries in the table expire if the corresponding nodes are not heard for a certain time.

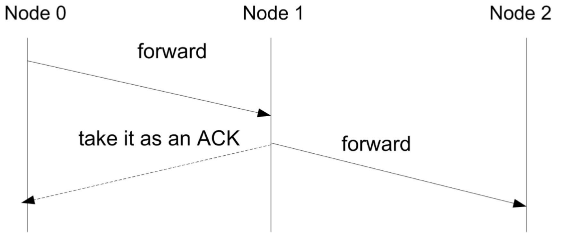


Fig. 11. Case of transmission success.

3.3 QELAR Protocol

3.3.1 Basic Protocol

Fig. 10 is the pseudocode for the routing algorithm. As described above, when a node hears a packet (either control packet or data packet), no matter whether or not it is designated as the next forwarder, it extracts the sender neighbor's information, such as *residual energy*, *group average energy*, and *V value*, from the header of the packet and updates the corresponding entry in the local neighbor list.

If the received packet is a data packet and the field of *Forwarder ID* in the packet indicates that the node is not eligible to forward the packet, the packet is simply dropped. Otherwise, the eligible forwarder will calculate the Q values associated with each of its neighbors by (7), and choose the next forwarder with the highest Q value. Before sending the packet, the node replaces the old metadata in the packet header with its own information.

3.3.2 Transmission Failure Detection

In the environment of UWSNs, packets may not always be able to get delivered successfully due to noise in the channel, collisions or topology change. One of the most direct-forward confirmation solutions is sending ACK packets back to the previous-hop node on receiving a packet. However, this naïve solution will lead to large communication overheads and take much network bandwidth in UWSNs.

Since all the nodes are listening to the traffic, one node can get to know whether a packet is successfully delivered to other nodes or not by analyzing the outgoing traffic of the receiver node. After a node sends a packet, it stores the packet in the memory rather than removes it from the buffer immediately. If the next forwarders (not the destination node) successfully receive the packet, they will forward the packet further along the next hop, and the returning packet heard by the previous forwarder will be taken as an acknowledgment. The data flow when a packet is successfully delivered from Node 0 to Node 1 is shown in Fig. 11. Node 0 releases the memory for the packet on hearing the *ACK*.

If the transmission fails, the packet in the sender's memory will not be heard by itself in some time, and therefore, retransmission will be triggered, shown in Fig. 12, in the hope of the connection may recover later. If the number of retransmissions for one packet exceeds a predefined constant *max_trans*, the forwarding attempt is regarded as a failure and no more retransmission. If

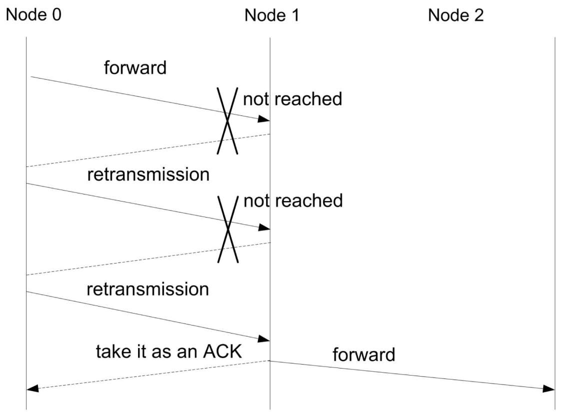


Fig. 12. Case of transmission failure & retransmission.

max_trans is set to 0, the network is very strict and takes the no-response as a failure right away.

Our QELAR routing protocol adapts to transmission failures gradually by sensing the environment and updating the *V values*. For calculating the *Q value* of the pair of node and action, every sensor node except the sink keeps a history of forwarding actions to its neighbors so that it can estimate $P^{a_m}_{s_n s_m}$ and $P^{a_m}_{s_n s_n}$ in (7). In the example given in Section 2, when the link between Node 1 and Node 2 is disrupted, Node 1 senses the failure based on the above mechanism. Upon failure, Node 1 gradually increases $P^{a_2}_{s_1 s_1}$ which is the failure rate of forwarding packets to Node 2, and decreases $P^{a_2}_{s_1 s_2}$ which is the success rate. Since Node 2 is chosen in the first place rather than Node 3, $Q(s_1, a_2)$ is larger than $Q(s_1, a_3)$ initially. With several unsuccessful attempts (usually not many), $Q(s_1, a_2)$ gradually decreases because more and more negative reward is received. After $Q(s_1, a_2)$ drops below $Q(s_1, a_3)$, Node 1 will forward the packet to Node 3.

3.4 Protocol Overhead Analysis

The overhead of acoustic communications for metadata exchanging comes from two places: the header of data packets and the broadcast control packets with no payload. As Fig. 9 shows, the extra header of each packet includes *V value*, *residual energy*, *average energy*, *sender's ID*, and *forwarder's ID*. In a realistic network implementation, both *sender's ID* and *forwarder's ID* can each be represented by a byte so that a maximum of 256 sensor notes are supported. The *V value*, *residual energy*, and *average energy* can each be represented by a 16-bit word. Therefore, these fields will be no more than 10 bytes. In a network, where every node has chances to forward data packets which help exchange information between nodes, we do not need broadcast special control packets. The frequency of broadcast packets can be set very low, and hence, this part of overhead is ignored.

Besides metadata exchanging, nodes need to store metadata for all the neighbors. The total amount of metadata may vary at runtime because of different number of neighbors and different number of paths (pairs of source and destination) existing in the network. With the cheap storage today, the storage overhead can be ignored.

Finally, although in the Q-learning-based routing protocol, each node has to carry out some computations of *Q value*s frequently, the computations are simple and their delay and power consumption are much smaller than that of acoustic communications (two-order lower) [6]. Hence, the computation overhead is ignored.

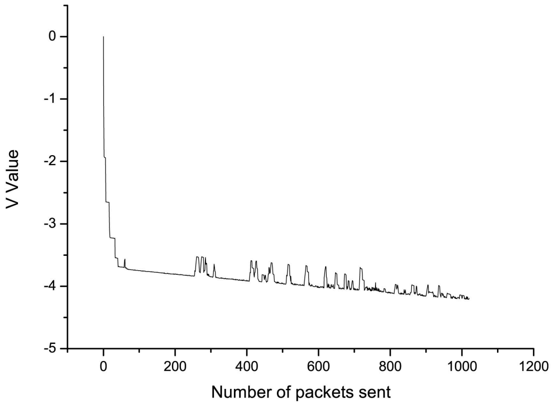


Fig. 13. *V Value* of the source node.

4 Evaluations

In this section, we present simulation results of our routing protocol, QELAR, by a network simulator Aqua-sim [15], which is extended for aquatic environment based on the widely used Network Simulator—NS2 [29]. We first analyze the algorithm, then evaluate how various design parameters of our protocol are affecting the network performance, such as delivery rate, latency, energy consumption, and residual energy distribution, and finally perform comparisons between QELAR and VBF [15].

4.1 Analysis on Convergence of the Algorithm

In this section, we deploy 125 static sensor nodes uniformly in a 3D space of $200 \times 200 \times 200\ \mathrm{m}^3$ to simulate some monitoring applications. The 125 nodes are in 5 equally spaced layers, each of which contains 25 (5×5) nodes. Packets are generated by a node on the bottom layer at a constant rate of 1 packet per second and sent to one of the nodes on the top layer. The one-hop transmission range is set to 75 m. The acoustic transmission power, receiving power and idle power of sensor nodes are set to 10 W, 3 W, and 30 mW, respectively.

Fig. 13 shows the *V value* of the source node after each packet is sent. Since the source node is the farthest from the sink, it takes the longest time for their *V value* to converge. The curve shows that the *V value* drops dramatically at the beginning, and gets steady after about 50 packets are sent, which means that the routing agent in the source node learns the environment and has reached an accurate approximation of the environment, including the source node's geographical distance from the sink node and energy distribution among the source node's one-hop neighbors. The curve decreases slowly after 50 packets are sent. This is an evidence that the distance part of the *V value* has converged, and the *V value* tracks the residual energy very well. Since the residual energy is decreasing as more packets are sent, the *V value* is decreasing according to the cost function defined. In addition, small fluctuations are observed after 200 packets are sent. Because the system tries

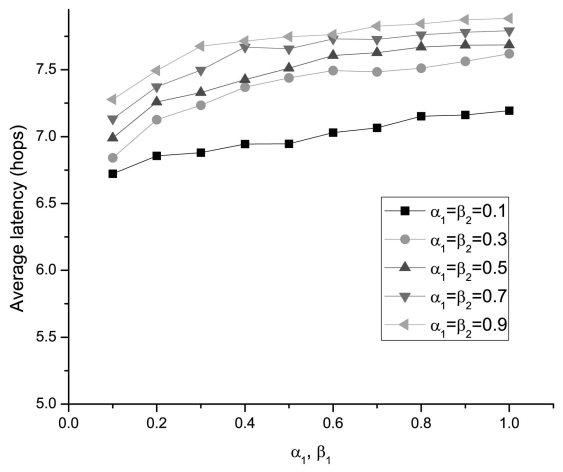


Fig. 14. Average latency of the delivered packets.

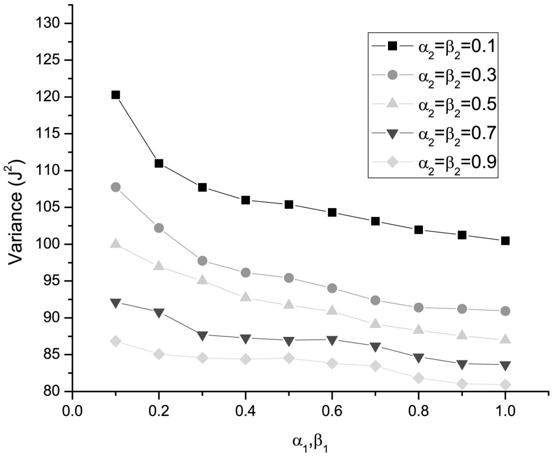


Fig. 16. Variance of the residual energy after delivering 1,000 packets.

to make energy evenly distributed, and hence, selects alternative paths that may be longer than the shortest path, the forwarders on the detouring routes increase the *V value* temporarily by changing the logic distance of the source node from the sink.

4.2 Performance Evaluation of the Protocol

We next evaluate the effect of various design parameters, including α_1, α_2, β_1, and β_2 in the reward functions in (10) and (13) with $g = 1$, on network performance. Because α_1 and β_1 are in equal positions, we set them the same values in the following simulations, so are α_2 and β_2. With α_2 and β_2 varying between 0.1 and 0.9 with a step size of 0.2, Fig. 14 depicts the average latency of each packet in terms of hop count. The figure shows that the routing agents try to find alternative paths to strive for more evenly distributed residual energy with larger weights α_1, α_2, β_1, and β_2. However, the cost is that packets need to detour so as to avoid those nodes closer to the sink but with relatively lower energy, which increases the latency. For the same reason, as shown in Fig. 15, the energy consumption for delivering 1,000 packets in the network also increases with larger α_2 and β_2. Larger α_1 and β_1 also lead to higher energy consumption. However, overall the energy consumption does not change much (fluctuation within 4 percent). This can be explained by the fact that there exists multiple paths that are shortest in terms of hop count, and therefore, larger weights for the energy terms in the reward function force the agents to alternatively choose a path among them for uniformly distributed residual energy, without increasing hop counts. Since the hop count does not vary much, the overall energy efficiency will not change much.

Fig. 16 shows that the normalized variance of residual energy in the network changes as those weights increase. The larger α_1 and β_1 will make nodes with higher residual energy more favorable than those with lower residual energy but closer to the sink. The effect is that the residual energy is more uniformly distributed in the network, i.e., lower variance. The impact of α_2 and β_2 on the variance is also significant in that they push for more evenly distributed energy in the local groups, hence, a lower variance is also achieved with higher α_2 and β_2.

We next introduce errors to packet transmissions in the network, and examine how transmission errors in the underlying protocol layer such as MAC affect the performance of QELAR. We vary the transmission error rate between 0 and 0.5, and Fig. 17 shows the results in which all the metrics, including the packet delivery rate, total energy

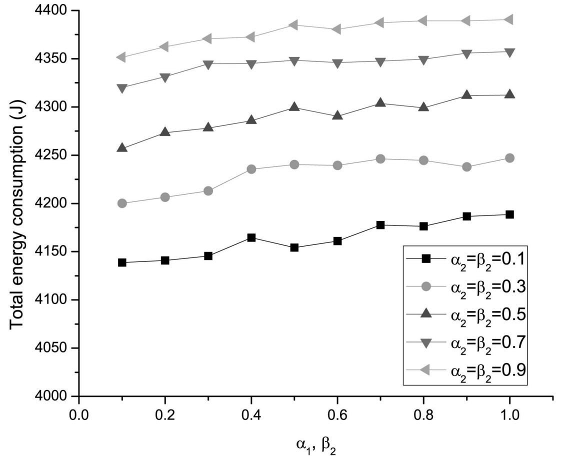


Fig. 15. The normalized energy consumption for delivering 1,000 packets.

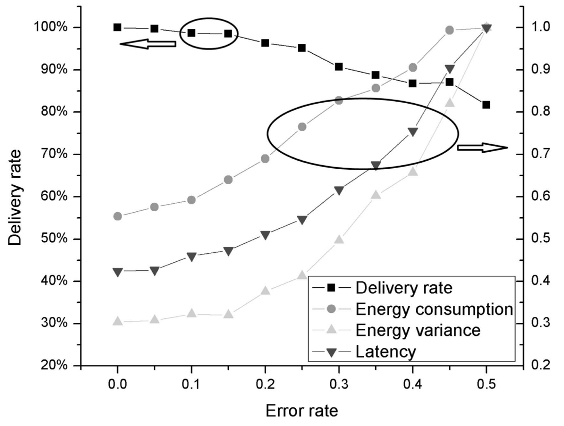


Fig. 17. The protocol performance with transmission errors.

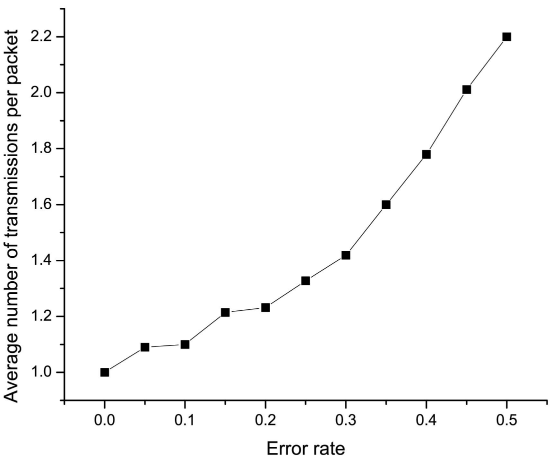


Fig. 18. Number of transmissions versus error rate.

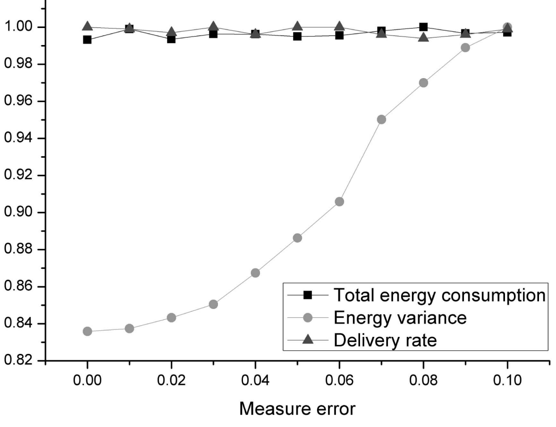


Fig. 19. Number of transmissions versus error rate.

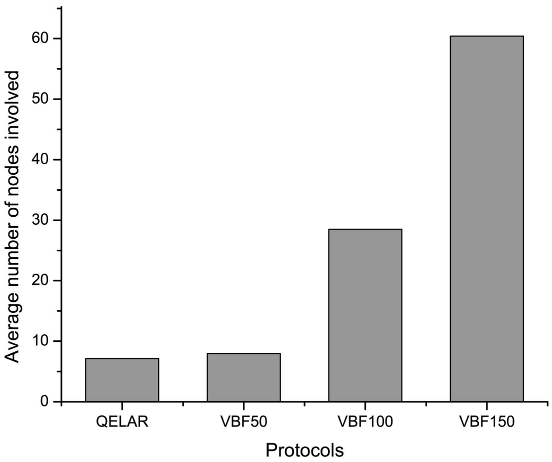


Fig. 20. Average number of nodes involved for a packet in different routing protocols.

consumption, residual energy variance, and average latency, are normalized to [0, 1]. We observe that even with high error rates, a high delivery rate (>85 percent) is achieved because of retransmission. Some packets may detour because of errors and may be dropped due to exceeding the hop count limit. That is why the delivery rate decreases a little with high error rate. The other three metrics, however, will significantly degrade as the error rate increases. In Fig. 18, we can see that the average number of transmissions for a single packet between two nodes increases as the error rate goes from 0 to 0.5. Retransmissions consume extra energy and delay the packet delivery. Meanwhile, because of the detouring caused by errors, packets may not always be able to go along the best path that helps balance the workload among sensor nodes, and thus, the variance also increases.

Since the residual energy of each node is an important factor in our routing protocol, and it may not be accurate with the current estimation system in sensor nodes, we then evaluate the effect of inaccurate battery measurement on the protocol performance. Each time the hardware reports a residual energy value, we introduce a random noise with the maximum value from 0 percent to 10 percent of the residual energy. Fig. 19 shows the effects of the inaccuracy on total energy consumption, energy variance as well as packet delivery rate. We can see that because of the inaccurate residual energy, the energy variance becomes larger as the measurement error increases, which means residual energy is distributed less and less uniformly. However, for the same routing tasks, the energy consumption almost keeps the same, because the number of hops to destination does not change much with the inaccurate energy measurement. At the same time, residual energy error almost has no influence on the packet delivery rate.

4.3 Comparisons between QELAR and VBF

In this section, we compare the performance of QELAR and VBF with different pipe radiuses of 50 m, 100 m, and 150 m in the same 3D dynamic network. We deploy 250 nodes in a $500\ \mathrm{m} \times 500\ \mathrm{m} \times 500\ \mathrm{m}$ space with transmission range 100 m, and set the random speed of all the intermediate nodes with a maximum speed of 1 m/s. The source and the sink are fixed on the bottom and surface of the network, respectively, while all other nodes are mobile. We evaluate the delivery rate, energy efficiency, routing efficiency, and network lifetime, which are all normalized to $[0, 1]$. Energy efficiency is defined as

$$\frac{\text{total packet delivered}}{\text{energy consumption}}.$$

Routing efficiency is defined as

$$\frac{1}{\text{average latency in terms of hop count}}.$$

Although there are many other possible definitions of network lifetime, we define network lifetime as the total routing time until the first node dies, which normally gives a lower bound of network lifetime. With a more uniformly distributed energy offered by our protocol, the lifetime should be longer under more realistic definitions.

In QELAR, packets are forwarded based on unicast and retransmission rather than multicast in VBF. Therefore, it is reasonable to see from Fig. 20 that the number of nodes involved for delivering a single packet in QELAR is much smaller than that of VBF. Meanwhile, as the radius of pipe in VBF increases, more nodes will be involved.

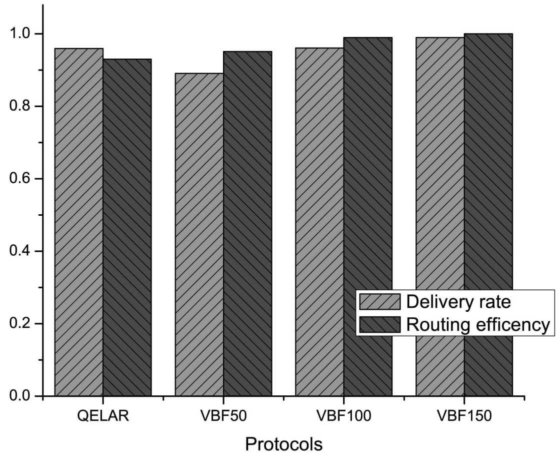


Fig. 21. Comparisons in terms of delivery rate and routing efficiency.

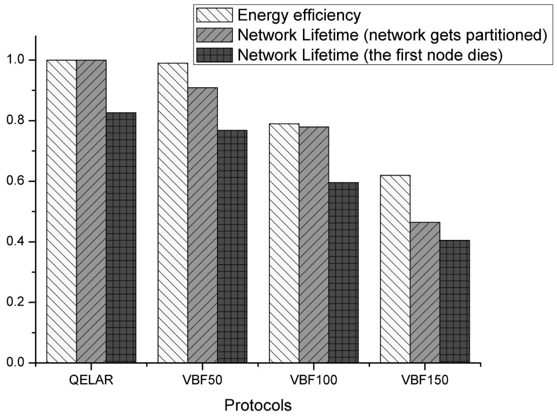


Fig. 22. Comparisons in terms of energy efficiency and lifetime.

The consequences are twofold. On the one hand, more involved nodes lead to high delivery rate as well as routing efficiency (i.e., low end-to-end delay). From Fig. 21, we can see that as the radius of the pipe increases (from 50 m to 100 m to 150 m), higher packet delivery rates are achieved by VBF, and the delivery rate of QELAR is comparable to VBF100. VBF also has advantages in routing efficiency as multicasting leads to lower average latency. However, since the sensor nodes are equipped with a learning agent, QELAR is able to find a path of the length near the shortest one, and thus, the gap between QELAR and VBF in terms of routing efficiency is not large.

On the other hand, since much less nodes are used when delivering a single packet compared to VBF, QELAR consumes much less energy as shown in Fig. 22. Meanwhile, QELAR is also able to balance the workload by distributing routing traffic to nearly all the nodes in the network, i.e., different packets travel through different groups of sensor nodes in the network. In contrast, VBF only uses the nodes within the pipe as forwarders and drains them fast. Because less energy is consumed in QELAR and the residual energy is distributed more uniformly, the most frequently used node lasts longer in QELAR than in VBF, which leads to a longer lifetime defined by the death of the first node. We also tested another network lifetime definition—the time that the network gets partitioned, which is later than the time that the first node dies. We observe the same trend as shown in Fig. 22. In QELAR, with more uniformly distributed energy, the network is unlikely to be partitioned at an early time, while VBF drains the nodes in the pipe fast, and thus, makes the network partitioned. In general, QELAR achieves average 20 percent longer lifetime than VBF with the three radiuses, for both the two different network lifetime definitions.

5 Conclusion

In this paper, we have proposed a novel distributed routing protocol, QELAR, for underwater sensor networks based on Q-learning. Unlike other routing protocols designed for UWSNs, few assumptions are made. Thus, it is more realistic to apply our routing protocol in a mobile UWSN without special devices for AOA and detailed geographical information, and with need of downlink communications. Nodes in the network are responsible for learning the environment to take the optimal action and improve the performance of the whole network gradually.

Our protocol has been examined by simulations with various different network configurations. Based on these results, QELAR can be easily tuned to trade latency or energy efficiency for lifetime, and thus, it can be applied to various different applications. Meanwhile, the simulation results have shown that even in a moderately sparse network, our protocol can achieve high delivery rate and energy efficiency. Moreover, because of its early awareness of residual energy distribution, the lifetime of networks can be prolonged significantly compared to another UWSN routing protocol.

Acknowledgments

The work was supported in part by the US National Science Foundation under grant 0709005 and in part by the University of Connecticut Research Foundation under large grant FRS 448483.

References

[1] I.F. Akyildiz, D. Pompili, and T. Melodia, "State of the Art in Protocol Research for Underwater Acoustic Sensor Networks," *Proc. SIGMOBILE Mobile Computing Comm. Rev.*, vol. 11, no. 4, pp. 11-22, 2007.

[2] G. Acar and A.E. Adams, "ACMENet: An Underwater Acoustic Sensor Network for Real-Time Environmental Monitoring in Coastal Areas," *IEE Proc. Radar, Sonar, and Navigation*, vol. 153, pp. 365-380, 2006.

[3] J. Rice, "Acoustic Communication and Navigation Networks," *Proc. Int'l Conf. Underwater Acoustic Measurements: Technologies & Results*, July 2005.

[4] L. Freitag, M. Grund, C. von Alt, R. Stokey, and T. Austin, "A Shallow Water Acoustic Network for Mine Countermeasures Operations with Autonomous Underwater Vehicles," *Underwater Defense Technology*, 2005.

[5] J. Heidemann, Y. Li, A. Syed, J. Wills, and W. Ye, "Research Challenges and Applications for Underwater Sensor Networking," *Proc. IEEE Wireless Comm. & Networking Conf.*, Apr. 2006.

[6] L. Freitag, M. Grund, S. Singh, J. Partan, P. Koski, and K. Ball, "The WHOI Micro-Modem: An Acoustic Communications and Navigation System for Multiple Platforms," *Proc. IEEE Oceans*, 2005.

[7] Crossbow Inc., Mica2 Data Sheet, http://www.xbow.com/products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf, 2008.

[8] Y. Chen and Q. Zhao, "On the Lifetime of Wireless Sensor Networks," *IEEE Comm. Letters,* vol. 9, no. 11, pp. 976-978, Nov. 2005.

[9] S. Lee, B. Bhattacharjee, and S. Banerjee, "Efficient Geographic Routing in Multihop Wireless Networks," *Proc. Int'l Symp. Mobile Ad Hoc Networking & Computing,* pp. 230-241, 2005.

[10] C. Ma and Y. Yang, "Battery-Aware Routing for Streaming Data Transmissions in Wireless Sensor Networks," *Mobile Networks and Applications,* vol. 11, no. 5, pp. 757-767, 2006.

[11] J.-H. Chang and L. Tassiulas, "Fast Approximate Algorithms for Maximum Lifetime Routing in Wireless Ad-Hoc Networks," *Proc. Int'l Conf. Broadband Comm., High Performance Networking, and Performance of Comm. Networks,* pp. 702-713, 2000.

[12] J.-H. Chang and R. Tassiulas, "Energy Conserving Routing in Wireless Ad-Hoc Networks," *Proc. IEEE INFOCOM,* pp. 22-31, 2000.

[13] A. Sankar and Z. Liu, "Maximum Lifetime Routing in Wireless Ad-Hoc Networks," *Proc. IEEE INFOCOM,* pp. 1089-1097, 2004.

[14] Y. Cui, Y. Xue, and K. Nahrstedt, "A Utility-Based Distributed Maximum Lifetime Routing Algorithm for Wireless Networks," *Trans. Vehicular Technology,* special issue on cross-layer design in mobile ad hoc networks & wireless sensor networks, vol. 55, pp. 797-805, May 2006.

[15] P. Xie, J.-H. Cui, and L. Lao, "VBF: Vector-Based Forwarding Protocol for Underwater Sensor Networks," *Proc. Networking,* pp. 1216-1221, 2006.

[16] L. Zhang, T.-H. Kim, and C. Liu, M.-T. Sun, and A. Lim, "Traffic-Aware Routing Tree for Underwater 3D Geographic Routing," *Proc. Int'l Conf. Intelligent Sensors, Sensor Networks, and Information Processing,* pp. 487-492, Dec. 2008.

[17] H. Yan, J. Shi, and J.-H. Cui, "DBR: Depth-Based Routing for Underwater Sensor Networks," *Proc. Networking,* pp. 72-86, 2008.

[18] C.-H. Yang and K.-F. Ssu, "An Energy-Efficient Routing Protocol in Underwater Sensor Networks," *Proc. Int'l Conf. Sensing Technology,* pp. 114-118, Dec. 2008.

[19] D. Pompili, T. Melodia, and I.F. Akyildiz, "A Resilient Routing Algorithm for Long-Term Applications in Underwater Sensor Networks," *Proc. Mediterranean Ad Hoc Networking Workshop,* 2006.

[20] J.M. Jornet, M. Stojanovic, and M. Zorzi, "Focused Beam Routing Protocol for Underwater Acoustic Networks," *Proc. ACM Int'l Workshop Underwater Networks,* pp. 75-82, Sept. 2008.

[21] J.A. Boyan and M.L. Littman, "Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach," *Proc. Advances in Neural Information Processing Systems,* vol. 6, pp. 671-678, 1994.

[22] S. Kumar and R. Miikkulainen, "Dual Reinforcement Q-Routing: An On-Line Adaptive Routing Algorithm," *Proc. Artificial Neural Networks in Eng. Conf.,* 1997.

[23] P. Wang and T. Wang, "Adaptive Routing for Sensor Networks Using Reinforcement Learning," *Proc. IEEE Int'l Conf. Computer and Information Technology,* p. 219, 2006.

[24] G.D. Caro and M. Dorigo, "Ant Colonies for Adaptive Routing in Packet-Switched Communications Networks," *Proc. Fifth Int'l Conf. Parallel Problem Solving from Nature,* pp. 673-682, 1998.

[25] M. Güneş, U. Sorges, and I. Bouazizi, "ARA—The Ant-Colony Based Routing Algorithm for MANETs," *Proc. Int'l Conf. Parallel Processing Workshop,* Aug. 2002.

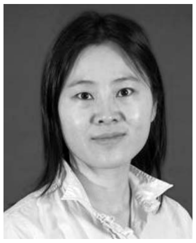
[26] S. Kamali and J. Opatrny, "A Position Based Ant Colony Routing Algorithm for Mobile Ad-Hoc Networks." *J. Networks,* vol. 3, no. 4, pp. 31-41, 2008.

[27] J.-H. Cui, J. Kong, M. Gerla, and S. Zhou, "Challenges: Building Scalable Mobile Underwater Wireless Sensor Networks for Aquatic Applications," *Proc. IEEE Network,* special issue on wireless sensor networking, pp. 12-18, 2006.

[28] R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction.* The MIT Press, Mar. 1998.

[29] S. McCanne and S. Floyd, *ns-Network Simulator,* http://www-mash.cs.berkeley.edu/ns/, 2008.

Tiansi Hu received the BS degree in electrical engineering from Peking University, Beijing, China, in 2007, and he is currently working toward the PhD degree at the University of Connecticut, Storrs. His research interests include optimization, algorithm design in mobile computing, and underwater sensor networks. He is a student member of the IEEE.



Yunsi Fei received the BS and MS degrees in electronic engineering from Tsinghua University, Beijing, China, in 1997 and 1999, respectively, and the PhD degree in electrical engineering from Princeton University, New Jersey, in 2004. She is currently an assistant professor in the Electrical and Computer Engineering Department at the University of Connecticut, Storrs. Her research interests include configurable and extensible processors, secure computer architecture, mobile computing and underwater sensor networks, low-power hardware and software design, and computer-aided design of integrated circuits and systems. She is the recipient of the US National Science Foundation CAREER award. She has served on the technical program committees for a number of conferences, including ISLPED, IPDPS, SASP, ISCAS, and VLSI-SOC. She is a member of the IEEE.