Utilizing Kinematics and Selective Sweeping in Reinforcement Learning-Based Routing Algorithms for Underwater Networks

Randall Plate and Cherry Wakayama
Maritime Systems Division
SPAWAR Systems Center Pacific
San Diego, CA, USA
Email: {randall.plate,cherry.wakayama}@navy.mil

***Abstract*—Effective utilization of mobile ad-hoc underwater distributed networks is challenging due to high system costs and the harsh environment characterized by low bandwidth, large latency, high energy consumption, and node mobility. This work addresses the routing issue, which is critical in successfully establishing and utilizing an underwater network. In particular, it focuses on reinforcement learning (RL)-based routing algorithms, which possess the ability to explore the network environment and adapt routing decisions to the constantly changing topology of the network due to node mobility and energy usage. It begins with a baseline implementation of a routing algorithm based on Q-learning known as QELAR. Although QELAR adapts to the network topology, its rate of convergence may be unacceptable for certain scenarios such as networks comprised of a large number of nodes and/or nodes with high mobility, resulting in excess resource usage. This paper presents two additional features that can improve the convergence rate and ability to track changes in the network: the use of kinematic information of the neighbor nodes and selective backward exploration. Simulation results have been obtained using NS-2 to compare the energy efficiency, convergence, and delivery performance of an RL algorithm with these two additional features to the baseline algorithm on networks with both fixed and mobile nodes.**

I. Introduction

Conventional routing protocols developed for ad-hoc networks can be classified into three different categories: table-based proactive, on-demand reactive, and geographical [1]. Each of these has inherent weaknesses when applied in underwater environments. Proactive routing has high maintenance costs for storing information about neighbor nodes, while reactive routing has high route discovery costs to gather neighbor information as it is needed. Geographic routing relies on position and/or velocity information, which can be difficult to obtain or inaccurate depending on node capabilities. Additionally, many existing works on routing protocols for underwater networks depend on various assumptions and limiting scenarios. For example, battery-aware routing protocol assumes geographical information is known at each node [2], vector-based-forwarding protocol assumes a static network [3], and depth-based-routing protocol assumes that the packets only need to be delivered from the bottom to the surface [4]. Combining the strengths of the three basic approaches can produce a more robust routing approach. Reinforcement Learning (RL) algorithms provide a convenient method of doing this when a robust yet generic solution is required. One RL technique, Q-learning, is implemented in an algorithm called QELAR [5], and can be considered as a hybrid of proactive and reactive protocols. It is a fully distributed architecture, where nodes compute their own routing decisions by storing routing information (Q values) of their direct neighbor nodes. In QELAR, the Q-value estimates consider the energy consumption of sensor nodes and residual energy distribution among neighboring nodes to optimize total energy consumption and network lifetime. The environment is learned as the estimated Q-values converge to reflect the network topology. However, convergence could occur slowly for certain network configurations such as those comprised of a large number of nodes and/or nodes with high mobility, resulting in excess resource usage.

We investigate two additional features that can improve the convergence rate of the Q-learning algorithm and its ability to track changes in the network while balancing the routing overhead. The first is the use of kinematic information to add a geographic component to the RL approach. Nodes transmit their own position and velocity estimates and store those of their neighbors to enable more accurate estimation of successful transmission capability, resulting in fewer failed transmissions. We assume that a node can estimate its own position and velocity but is unaware of the destination location. The second is the addition of selective backward exploration to the forward exploration of QELAR, such that nodes actively propagate significant changes in their Q values back toward the source. Although exploration overhead is increased with the addition of each feature, the improved convergence rate and tracking of network changes results in an overall improvement in energy consumption and/or latency as compared to the baseline Q-routing approach. Our objective is to understand the trade-offs achievable when employing these algorithms in different network scenarios so that the best features/parameter values can be chosen for maximal efficiency.

This paper proceeds as follows. In Section II, we introduce Q-learning and a baseline algorithm, QELAR. In Section III we discuss two additional features – utilizing kinematic

information and selective sweeping – which we have developed to improve the convergence rate. Section IV presents NS-2 simulation results that demonstrate the performance of the two additional features; the energy usage, convergence, and delivery performance results are compared to that of the baseline QELAR algorithm. Conclusions and future work are provided in Section V.

II. Q-Learning-Based Routing Algorithms

The sequential routing decision system can be modeled as a Markov Decision Process (MDP). The MDP is characterized by its state set, action set, dynamics (set of state transition probabilities), and expected immediate cost. Let s_t and a_t denote a state and action (decision), respectively, at time t; for the underwater network system, the state represents the location of the packet in the network and the action represents which of the one-hop neighbor nodes of s to forward to. Let $P^{a_t}_{s_t s_{t+1}}$ denote the probability of going from the current state, s_t, to the next state, s_{t+1}, when taking action a_t; $R^{a_t}_{s_t s_{t+1}}$ is the expected immediate cost for taking action a_t at state s_t and arriving at state s_{t+1}. Cost may include energy usage, network lifetime, delay, and congestion depending on the mission operational objectives. The objective is to minimize the expected sum of costs of actions leading from the source node to the destination node. According to Bellman's principle of optimality in Dynamic Programming, the optimal solution can be obtained once the optimal value functions are found. Value functions are functions of the states that provide a measure of how good it is to be in a given state (V), or of state-action pairs that estimate how good it is to perform a given action in a given state (Q). Optimal value functions, V^* and Q^*, satisfy the Bellman equations [6]:

$$V^*(s_t) = \min_{a_t} \sum_{s_{t+1}} P^{a_t}_{s_t s_{t+1}} \left[R^{a_t}_{s_t s_{t+1}} + \gamma V^*(s_{t+1}) \right], \quad (1)$$

$$Q^*(s_t, a_t) = \sum_{s_{t+1}} P^{a_t}_{s_t s_{t+1}} \left[R^{a_t}_{s_t s_{t+1}} + \gamma \min_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) \right], \quad (2)$$

where $0 \leq \gamma \leq 1$ is a parameter that determines how important future costs are. The relation between V^* and Q^* is given by:

$$V^*(s_t) = \min_{a_t} Q^*(s_t, a_t). \quad (3)$$

Value functions evaluated at given states (V) are referred to as V values, while functions evaluated as state-action pairs (Q) are referred to as Q values. For mobile ad hoc underwater networks, where the system model of future behavior is not known at each node, the optimal value functions are not known a priori, and also change as the network changes. In this situation, methods of estimating the optimal value functions must be employed to find good routing decisions. Q-learning is a reinforcement learning method which iteratively approximates Q^*.

A. QELAR Algorithm

QELAR is a recently-developed routing algorithm for underwater sensor networks that implements a model-based Q-learning approach [5]. The Q values represent the estimated cost (based on energy usage) for a packet to reach the destination node from the neighboring nodes. Nodes use Eq. (2) to evaluate neighbor node Q values and choose the neighbor with the minimum Q value (Eq. (3)). Transition probabilities to neighbors are estimated based on the success/failure history of a node's forwarding actions. The cost function considers the initial node energy, residual energy, and the energy distribution of the neighboring nodes to balance choosing short paths with maximizing network lifetime. When nodes overhear neighboring forwarding transmissions, they use the information obtained to update estimates of their neighbors' V values. In order to keep up with changing network topology, nodes also periodically transmit header-only packets with their V value and energy information to neighbors. Thus, a combination of virtual experiments (iterating equation (2)) and actual observation (overhearing neighbor transmissions) are employed to make routing decisions.

III. Improvements for System Model and Convergence Rate

For networks comprised of nodes with high mobility, QELAR requires frequent periodic pinging to allow nodes to update their P and V estimates; inaccurate estimates result in many transmission failures and therefore energy waste and potential network flooding. As an approach to compensate for this issue and improve the ability to predict state transition probabilities and enhance V estimate convergence, we discuss the utilization of neighbor kinematic data (position and velocity) and selective sweeping.

A. Utilizing Kinematics

Although QELAR's scheme of learning transition probabilities from the success/failure rate of its history of forwarding actions can be effective for static networks, the mismatch between estimates and the actual environment can become significant for networks with high mobility, resulting in increased transmission failure rate. In order to achieve a better system model for networks with mobile nodes, we utilize the kinematic state of the neighbors. Along with their V values, nodes include their kinematic states (position and velocity) in transmitted packet headers so that their locations can be tracked by their neighbors. We assume that each node has some capability of estimating its own position and/or velocity. In the best case, this could be a Global Positioning System (GPS)-based system for surface nodes (buoys); submerged nodes (e.g. autonomous/unmanned underwater vehicles (AUV/UUVs)) may only have a basic inertial navigation system; fixed nodes (sensors) could be programmed with their locations at installation. For nodes that are drifting with currents and do not have such navigation capabilities, a localization algorithm could be utilized that is based on communications with neighbor nodes. Note that this is a

more relaxed requirement than typical of geographic routing approaches in that we do not require any node to know the location of the destination.

Nodes employ a propagation model of the medium (either known a priori or learned) to estimate the probability of successful transmission to its neighbors based on transmission distance. For the NS-2 simulation employed, we have generated a transmission success probability model as a function of transmission power and distance based on Monte-Carlo simulations

$$P = -\frac{s_{size}}{\pi} \arctan \frac{(r - r_{0.5}) * s_{slope}}{r_{0.5}} + v_{shift} \qquad (4)$$

where s_{size} is an overall scaling parameter set to 1.3, s_{slope} is a scaling of the slope of the curve computed from a polynomial as a function of transmission power, and v_{shift} is a vertical shift parameter computed from another polynomial as a function of transmission power. The parameter $r_{0.5}$ is the range at which the probability of successful transmission is 0.5 and is computed from two different polynomials as a function of transmission power as well. The resulting curves for three different transmission powers are shown in Figure 1 where we see good agreement between the model and the Monte-Carlo results.

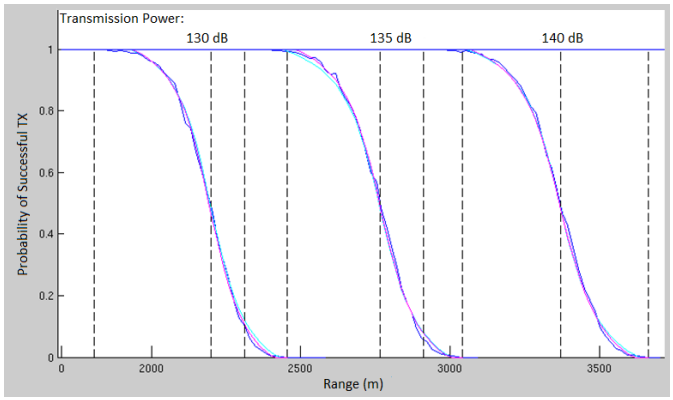

Fig. 1. Transmission probability of success as a function of transmission power and distance. Each curve actually consists of three lines, nearly overlapping: Monte-Carlo simulation result (blue), custom fit curve for each transmission power (cyan), and generic model that accepts any transmission power as an input (magenta).

B. Selective Backward Sweeping

Selective backward sweeping centers around the concept of actively propagating changes in nodes' V values (due to learning or movement) back toward the source rather than waiting for them to percolate backward as a natural product of forwarding packets. Just as in QELAR, each time a node receives or overhears a neighbor's transmissions, it updates its own estimate of that node's V value. However, instead of waiting until its next transmission time to compute its own V value, it does so immediately and compares its change in V value, δV, to a predefined threshold, Θ. If $\delta V > \Theta$, it initiates forwarding a header-only packet with its updated information so that its neighbors have access to this information.

To control the spreading of backward sweeping packets, an additional set of V values are maintained at each node corresponding to the estimated minimum cost of going backward to reach the source node: thus, Q^s (V^s) and Q^d (V^d) denote the Q values (V values) associated with directing packets toward the source and destination, respectively. This is similar to the approach used in [7], although they use this information differently to avoid bottlenecks. The 'selective' aspect of the approach is implemented by including a prioritized list of nodes in each packet header that indicates which ones are "allowed" to initiate backward packets. The nodes are chosen according to minimum V^s value, which results in the backward packets being steered toward the source. Thus, the header of each packet now includes the node's estimate of V_t^s in addition to V_t^d, and also a list of backward forwarder ID's. A depth parameter is included which defines how many nodes to select as backward forwarders at each transmission.

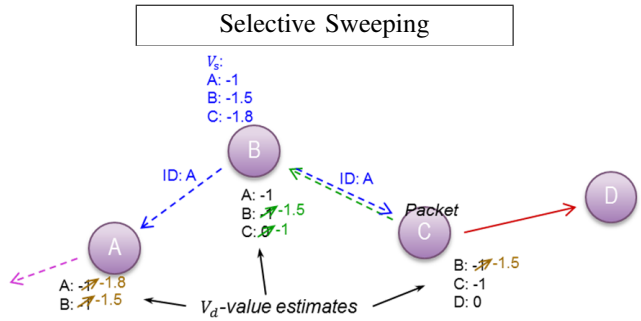

Fig. 2. Five-step selective sweeping operation example. **1 (red):** Node C forwards a packet to node D. **2 (green):** Node B overhears the packet and updates its V_d estimates of C and itself. **3 (blue):** B initiates a backward packet to A since A has max V_s value. **4 (brown):** A and C both hear the packet and update their V_d estimates. **5 (magenta):** Node A continues sending the backward packet since it was addressed to A.

IV. Simulation Results

The popular NS-2 network simulator was used to implement both the baseline and modified algorithms [8]. Several extensions have been leveraged to make use of existing underwater modeling (DESERT [9] and WOSS [10]) as well as NS-MIRACLE [11] to support modularization of the code.

While multiple sources/destinations could exist in the simulator, the results for this paper include a single source and single destination. The source node generates packets using the constant bit-rate (CBR) module provided in the DESERT framework which produces packets at random times at a specified average period, which has been set to 60 seconds. Two different scenarios are run, one with a stationary destination (node 25), and one with a mobile destination (node 26), which could be considered a UUV. The remaining nodes, aside from the mobile node, are organized in a grid pattern with 1 km spacing as shown in Figure 3; the path of the mobile node is shown in blue.

An energy module has been developed to record energy usage at each node. It models the transmission, receiving, and

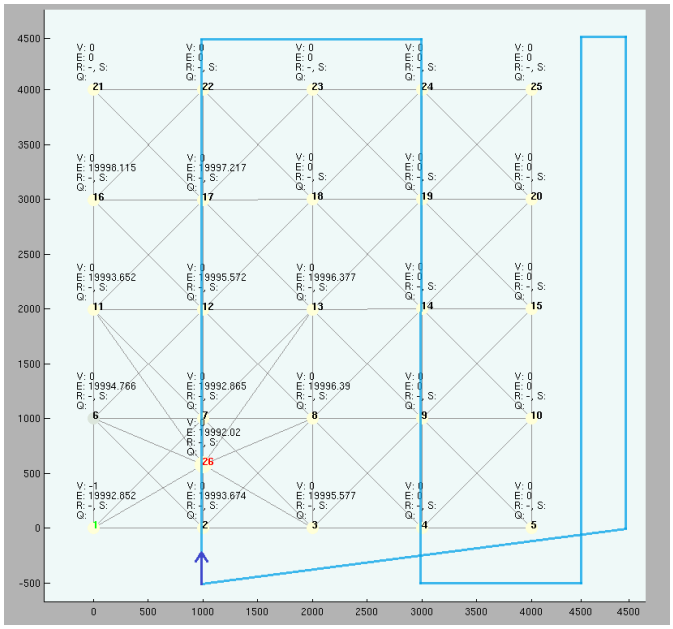

Fig. 3. Field layout; UUV starts at (1000,-500) and follows path shown in blue when present; shown here at position (1000,550) part way into run. The source is indicated by the node numbered in green (node 1) and the destination is numbered in red (node 26 - UUV). Gray lines indicate node connections with a probability of success greater than 0.5.

idle power levels to be 10 W, 3 W, and 0.03 W, respectively, and deducts energy from an initial available amount (e.g. 10 kJ) as nodes spend periods of time in each of these states. The simulation continues for a specified period of time, 10,000 seconds (2.7 hours) in this case.

Packets consist of a header and a payload. Header fields depend on the algorithm implemented (see Table I) and serve to direct the packet along its path and relay routing information among nodes. The size of the header is estimated by allowing one byte for each field that is a node ID (enabling 256 nodes to be present in the network - this could be expanded if necessary) and two bytes for each numerical value. The payload section contains the actual data being transmitted by the source and is fixed at 200 bytes for all packets.

A. Numerical Metrics

A number of metrics have been computed to compare algorithm performance under both static and mobile scenarios (Table II). Results are shown for four configurations of the modified algorithm, which we will refer to as QKS (Q-learning utilizing Kinematics and Sweeping): tracking only, selective sweeping with depths of 1 and 3, and uniform sweeping. 'Total Energy' is the sum of all energy consumed by all nodes during the simulation. 'Total TX' and 'Failed TX' are the total transmissions of all nodes, and the number of transmissions that failed to reach their intended next hop, respectively. 'Sweep TX' denotes the number of transmissions initiated as part of the selective sweeping functionality (QKS) and 'Periodic TX' is the number of periodic transmissions (QELAR). 'Avg Path Len' is the average path length considering all paths from

TABLE I

PACKET HEADER FIELDS FOR QELAR (Q), MODIFIED ALGORITHM WITH KINEMATIC TRACKING ONLY (T), AND MODIFIED ALGORITHM WITH BACKWARD SWEEPING ONLY (S). NOTE THAT 'NEXT BACKWARD ID' IS A VECTOR OF ID'S, WITH LENGTH DEPENDING ON THE SWEEP DEPTH USED; 3 BYTES ARE LISTED AS A REPRESENTATIVE SIZE.

Field	Bytes	Q	T	S
Packet ID	1	x	x	x
Dest Node ID	1	x	x	x
Sender V_d	2	x	x	x
Energy (res)	2	x	x	x
Energy (avg)	2	x	x	x
Prev Hop Node ID	1	x	x	x
Next Forwarder ID	1	x	x	x
Sender Pos	6		x	
Sender Vel	6		x	
Sender V_s	2			x
Next Backward ID	3			x

source to destination used by packets during the simulation.

A Monte-Carlo simulation mode has been employed, where identical node configurations are used, but the random aspects of the simulation will vary (e.g. packet errors due to noise, packet transmission times, etc.). Each algorithm is run many times and the results averaged to provide a more representative result.

We first observe that both algorithms show similar performance in the static case. In this scenario, QELAR is able to estimate and maintain sufficiently accurate estimates of successful transmission probabilities (P values) along the path to the destination as they remain constant over time, and the V values change only gradually due to energy usage. Thus, the extra overhead of QKS does not result in much improvement, although it does show slight benefit due to the geographic information providing more accurate/immediate P value information.

The mobility case is more challenging and we see decreases in performance in both algorithms, however the affect on QELAR is much more significant, where we see a total energy increase of approximately 2.5 times, and failed transmissions increasing by a factor of over 7. Comparatively, QKS increased by only about a factor of 1.5 with respect to total energy, and failed transmissions less than tripled. This demonstrates the advantage of using geographic information in the mobile node case: the tracking component of the QKS algorithm allows nodes to maintain accurate estimates of the UUV, resulting in better estimates of P values. This, in turn, produces better routing decisions and therefore fewer failed transmissions and significantly less energy used.

Three columns are shown under the QELAR heading: the first corresponds to the static scenario, and the second and third both correspond to the mobile scenario. Column two describes a configuration where periodic transmissions were not used, whereas in column three periodic transmissions were configured to be initiated after 2000 seconds of a node not having transmitted. This offers some improvement, as the mobile node will now transmit its information more frequently, but it still suffers considerably compared to even the modified

TABLE II

METRICS FOR BASELINE (QELAR), WITH AND WITHOUT PERIODIC TRANSMISSIONS, AND NEW ALGORITHM (QKS) USING TRACKING (T) AND SELECTIVE SWEEPING (S) WITH VARIOUS DEPTHS: 1 NEIGHBOR (S(1)), 3 NEIGHBORS (S(3)), OR UNIFORM (S(U)). SCENARIOS FOR STATIONARY AND MOBILE DESTINATION (2 M/S) ARE SHOWN.

Algorithm	QELAR			QKS - T Only		QKS - T & S(1)	QKS - T & S(3)	QKS - T & S(U)
Dest Vel (m/s)	0	2	2	0	2	2	2	2
Total Energy (J)	20614	51716	46333	20100	30148	29454	27675	26381
Total TX	1414	4146	3601	1361	2141	2068	1923	1847
Failed TX	357	2658	2128	326	892	854	769	678
Sweep TX	0	0	0	0	0	13.9	24.2	31.4
Periodic TX	0	0	42.1	0	0	0	0	0
Avg Path Len	6.7	23	19.3	6.5	10.4	10	9.7	8.6

algorithm with tracking only. More frequent periodic transmissions resulted in more energy usage overall. Adding various levels of selective sweeping to the modified algorithm reveals further reductions in both failed transmissions and total energy used. We can see the selective sweeping is more efficient in that fewer sweeping transmissions are made as compared to the periodic transmissions made by QELAR due to the fact that they are triggered based on changes in V value, not simply time.

Note that we have also simulated a case we refer to as "uniform" sweeping, where instead of selectively choosing nodes with the minimum V values in the direction of the source node, all neighbors are allowed to transmit sweeping packets. This is therefore the upper limit on the amount of sweeping that can be performed.

B. V Value Estimation

Figures 4 and 5 show the error of V value estimates at all nodes as time progresses through the simulation. The actual V values for each node are computed at each time instant, and each node's estimated V value is compared to the correct value; the deviations between these two are summed for all nodes. This provides an overall metric of how well the network as a whole has converged at any given time; note that zero-error convergence can never happen because the network is attempting to converge on a "moving target" since the V values are constantly changing. It should also be noted that some level of overall error in V value estimate in the network is not necessarily a bad thing, because if nodes are not located along a path used to get to the destination, it may be acceptable to let their V estimates be inaccurate and not waste resources updating them. However, this metric still gives valuable insight into routing performance.

For the static case (Figure 4), we observe almost identical performance between the two algorithms, as was evidenced from the metrics; we can, however, observe a slightly better steady state convergence for the QKS algorithm due to the better P accuracy.

In the mobile case (Figure 5), we observe more significant differences. While the QELAR algorithm converges faster, this is due to initial packets doing more "exploration" of the network due to inaccurate knowledge of the network, and thus more nodes change their V values sooner. As can be seen from Figure 3, the UUV (destination) begins close to the source node in the mobile case, and thus packets do not need to venture far into the network to reach it. Thus, the QKS algorithm is actually performing better by not sending packets throughout the network, which has the side effect that these nodes do not converge in V value until later in the scenario. Also note that increasing the backward sweeping increases the V convergence by essentially carrying out more exploration, even to the point where it exceeds the QELAR algorithm, yet is able to maintain better energy performance at the same time due to the efficient use of the sweeping packets.

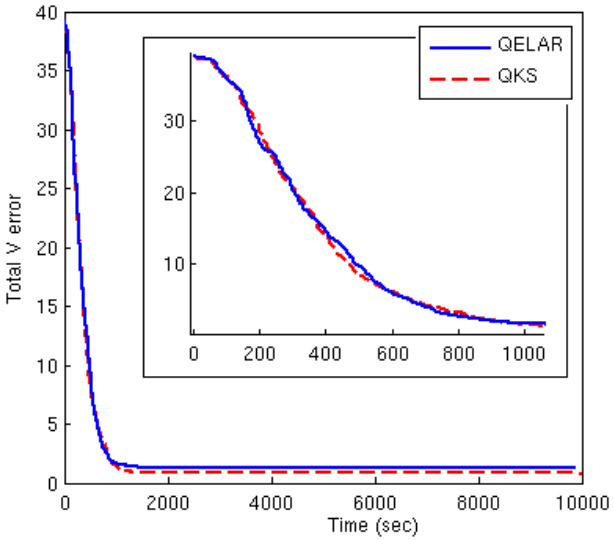

Fig. 4. Total V value error for all nodes - static scenario. Inset shows a zoomed view of the first 1000 seconds to show similar convergence rate/characteristic.

The more significant effect we observe is the two humps around 4500 seconds and 7500 seconds, which correspond to times when the UUV leaves the node field. We note that the tracking alone helps with maintaining good estimates during this time over the QELAR algorithm, with selective sweeping helping even further. In fact, with uniform sweeping, we observe the fastest initial convergence and lowest error during the remainder of the simulation of any of the algorithms.

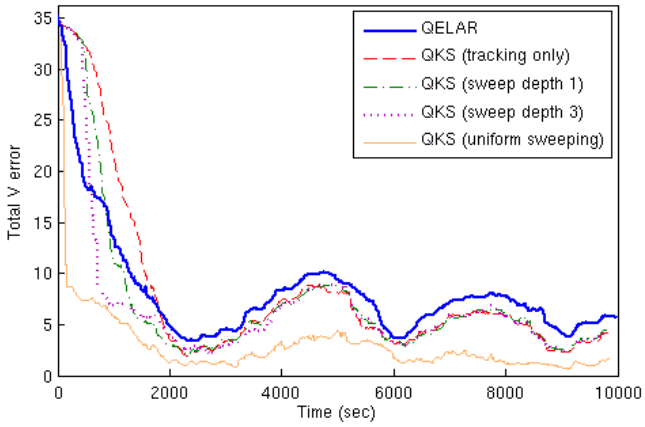

Fig. 5. Total V value error for all nodes - mobile scenario.

C. P Value Estimation

We can also examine the accuracy of the P value estimates (probabilities of successful transmission) of the two algorithms, as shown in Figure 6; we provide only the mobile result here as the static result showed similar trends. Here, the error between the estimated P values at each node and the actual probability of successful transmission based on node distance is computed at each time instant. This shows most clearly, perhaps, the benefit of the modified algorithm: all four implementations show distinct improvement over QELAR in terms of ability to track the continually changing P values due to estimating neighboring node kinematic states and tracking their position between communications with them. This improved ability to estimate transmission probabilities is primarily due to the kinematic tracking component of the algorithm and therefore we see little difference between the selective sweeping levels used. The lower value of P error is one factor influencing the reduced V value error we examined above.

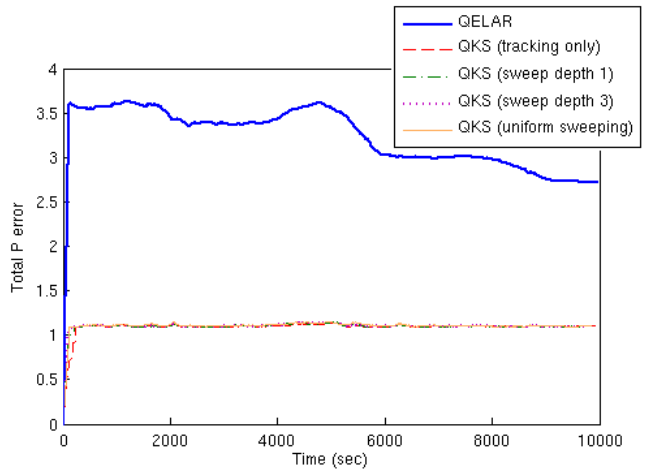

Fig. 6. P estimate error for all nodes - mobile scenario.

V. Conclusion and Future Work

Our RL-based routing algorithm employing the features of tracking using neighbor node kinematic information and prioritized sweeping based on changes in V value estimates demonstrates faster convergence and better estimates of dynamic networks than the baseline algorithm, QELAR. This results in decreased overall energy usage, which is a key limitation in underwater networks. The simulation results presented are still early in the development work being done, and we plan to perform more exhaustive simulations as we continue. Specifically, we are in the process of incorporating a random node motion model, and we also plan to perform a more rigorous parameter study of the algorithm in the near term. Following this, we plan to further expand the algorithm to include a multi-modal capability, where it would allow nodes to not only determine routing paths, but also dynamically adapt transmission mode (acoustic, optic, etc.) to best fit the situation at a given time.

Acknowledgment

This work is being funded under the Office of Naval Research (ONR) In-House Laboratory Research (ILIR) program at SSC Pacific.

References

[1] D. Pompili and I. F. Akyildiz, "Overview of networking protocols for underwater wireless communication," *IEEE. Commun. Mag., Feature Topic on Underwater Wireless Communications*, Jan. 2009.
[2] C. Ma and Y. Yang, "Battery-aware routing for streaming data transmissions in wireless sensor networks," in *Proc. Intl. Symp. Mobile Ad hoc Networking & Computing*, 2005, pp. 230–241.
[3] P. Xie, J.-H. Cui, and L. Lao, "VBF: Vector-based forwarding protocol for underwater sensor networks," in *Proc. Networking*, 2006, pp. 1216–1221.
[4] H. Yan, J. Shi, and J.-H. Cui, "DBR: Depth-based routing for underwater sensor networks," in *Proc. Networking*, 2008, pp. 72–86.
[5] T. Hu and Y. Fei, "QELAR: A machine-learning-based adaptive routing protocol for energy-efficient and lifetime-extended underwater sensor networks," *IEEE Trans. Mobile Computing*, vol. 9, no. 6, pp. 796–809, Jun. 2010.
[6] R. S. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, Massachusetts: The MIT Press, 1998.
[7] S. Kumar and R. Miikkulainen, "Dual reinforcement q-routing: An online adaptive routing algorithm," in *Proc. Artificial Neural Networks in Eng. Conf.*, 1997.
[8] NS-2, "http://www.isi.edu/nsnam/ns/."
[9] DESERT, "http://nautilus.dei.unipd.it/desert-underwater."
[10] WOSS, "http://telecom.dei.unipd.it/ns/woss/."
[11] NS-MIRACLE, "http://telecom.dei.unipd.it/pages/read/58/."