I J C T A, 10(8), 2017, pp. 257-268 ISSN: 0974-5572

Q-learning based fault tolerant routing with congestion control for Wireless Sensor Networks

S. Sridevi* and M. Usha**

ABSTRACT

Fault tolerance and congestion control have been identified as two important methods for improving the lifetime of Wireless Sensor Networks. However, the routing algorithms existing in the literature rarely address these two issues simultaneously. This paper proposes a Q-learning based routing algorithm to select next hop nodes by considering both fault tolerant characteristics and congestion degree. The information about congestion and node fault are updated according to the current network condition. Negative rewards are assigned in case of congestion and node fault. Learning rate of the Q-learning algorithm is set according to the congestion status. Extensive simulation results shows that the proposed algorithm reduces packet drop ratio, energy consumption and improves the network lifetime and throughput.

***Index Terms*:** Wireless Sensor Networks, Congestion control, Q-learning, Fault tolerance, multi hop

1. INTRODUCTION

Wireless Sensor Networks (WSNs) consists of a collection of sensor nodes distributed in a region of interest to monitor for the occurrence of specific events. These sensor nodes are limited by power, memory and processing capacity constraints. WSNs are used in wide variety of applications like environment monitoring, habitat monitoring and military applications [1]. In the event driven applications of WSNs, sensor nodes generate huge amount of data upon the occurrence of specific event. When these data are sent to the sink through multiple hops, due to the many-to-one nature of data transfer, the load on the network may become greater than the capacity it can handle, thereby creating congestion [2]. Congestion increases packet loss, packet delay and reduces the useful lifetime of the sensor nodes. Many routing algorithms are developed in recent years to reduce network congestion [3].

Due to the limited power supply battery and the deployment in an unattended environment, these WSNs are frequently affected by node and link failure. The sensor nodes may fail either permanently or temporarily due to battery depletion and factors like electromagnetic radiation respectively. These faults create malfunctioning of the sensor nodes and hence nodes with these faults should be properly diagnosed [4].

In this paper, Q-learning algorithm is used to route the data in multiple hops to the sink node. The next hop node is selected dynamically to forward the packet based on congestion, transient fault and permanent fault. In each node, the best link for forwarding the packet is selected based on the past experience and the present reward. Learning rate of the Q-learning algorithm is adjusted according to the current congestion level.

The rest of the details discussed in the paper are organized as follows: In section II, related work on various congestion control, fault tolerant and reinforcement algorithms are presented. Section III discusses

* United Institute of Technology, Coimbatore, Tamilnadu, India, *Email: srive71@gmail.com*

** Sona College of Technology, Salem, Tamilnadu, India, *Email: usha@sonatech.ac.in*

the proposed Q-learning based routing protocol in detail. In section IV, simulation setup and the results are presented. Section V gives the conclusion of the paper.

2. RELATED WORK

Several investigations have been conducted in the areas of congestion control and fault tolerance in WSNs. This section explains the algorithms existing in the literature in the above mentioned areas.

2.1. Congestion Control algorithms

In literature, many algorithms are proposed for controlling congestion in WSNs. A detailed study of various congestion control algorithms is presented in [2], [3]. Congestion control algorithms involve congestion detection, congestion notification and congestion mitigation. Congestion is detected based on factors like buffer occupancy, channel load, packet loss, packet service time, packet inter-arrival time. After detecting congestion, it is notified to the upstream nodes either implicitly or explicitly. Finally congestion is controlled by either traffic control or resource control techniques [2], [3].

The protocol proposed by Wan et al. [5] called COngestion Detection and Avoidance (CODA) detect congestion based on buffer occupancy and channel loading conditions. Once congestion is detected, explicit back pressure message is sent to the upstream node. Source node performs closed loop congestion control based on the acknowledgment received from the sink.

In [6], Wu, et al. proposed hop by hop control where each node detects congestion based on congestion degree and buffer occupancy. Both local and upstream nodes perform processing according to the feedback signal. Channel access priority and transmission rate are modified based on the node's local data and the data of downstream node.

In [7], Gholipour, et al. proposed a distributed traffic aware routing scheme for networks with multiple sinks. Next hop node is selected by considering normalized buffer size, congestion degree, depth of the node and average cumulative queue length of the nodes in the path to sink. Data transmission rate is adjusted according to buffer size and congestion degree of current and upstream nodes.

In ECODA [8], proposed by Tao, et al. congestion is detected based on dual buffer threshold and weighted buffer difference. Weighted fairness is achieved by a scheduler with the help of dynamic priority of the packets. Each upstream node receives the value of delay to reach the sink from the downstream nodes. Finally, delay based rate adjustment is done by the source node.

To address the QoS requirements of various applications and to provide fairness among various sensor nodes priority based congestion control techniques are proposed by Wang, et al. [9], Sridevi et al. [10], Rezaee, et al. [11]. Scheduling of packets from the queues and rate adjustment are done based on priority.

2.2. Fault diagnosis algorithms

A detailed classification of fault detection techniques is proposed by Mahapatro et al. [12]. The algorithm proposed by Xiao et al 2007 [13] identifies faulty sensor nodes by considering correlation between the readings and the trustworthiness of sensor nodes. A sensor node with a large number of similar and trustworthy neighbors is considered to be more trustworthy. Ranks are assigned to sensor nodes based on trustworthiness. Sensor nodes perform self -diagnosis by checking their current reading vector with the previous reading vector. If a sensor node finds unusual behavior in the comparison, then it gets the help from the neighbors to decide whether it is really faulty or not.

A distributed algorithm is developed to detect permanent, intermittent and transient faults. Readings of each sensor node is compared with that of its neighbors periodically. The algorithm discriminates faults based on the results of comparison in the successive tests [4].

2.3. Reinforcement learning algorithms

In WSNs, Q-learning algorithm is used in applications like power management, medium access control and cooperative communication [14]. In [15], Forster et al. proposed Q-learning based clustering protocol for WSNs. Liang at al. [16] and Kiani et al. [17] proposed a Q-learning based routing protocol for sensor networks. Next hop node is selected using reward function calculated based on factors like distance, energy and delay.

Tan et al. [18] proposed R-learning based congestion avoidance scheme for multi path routing in WSNs. Rewards are assigned based on energy efficiency and packet loss ratio. Congestion state is set by comparing the reward with the queue threshold values. In [19], a novel way of propagating negative rewards in addition to propagating positive rewards is presented. The negative reward approach is useful to avoid critical problematic situations in real time problems. In [20], the authors proposed a novel method to adjust the learning rate of the algorithm based on congestion detection method for many-core embedded systems.

3. PROPOSED ROUTING ALGORITHM

This paper proposes a Q-learning based fault tolerant routing with congestion control (QFTRCC) for multi-hop WSNs. The main aim of the proposed algorithm is to consider both node fault and congestion to improve the quality of the selected routing path.

3.1. Overview of Q-learning algorithm

Q-learning is a reinforcement learning technique proposed by Watkins and Dayan in 1992 [21]. In Q-learning, agents learn the nature of the environment at run time and perform specific actions accordingly. For every action, agents obtain rewards otherwise called as Q-values from the environment. These rewards indicate the goodness of the actions. From the list of available actions, the agent selects a particular action, executes it and receives reward. The Q-value is updated using the reward. The Q-learning parameters are defined as follows:

*State***:** Each sensor node S_i is represented as a state. The set of states in the WSN is represented as $S=(S_1, S_2, ..., S_n)$ where n is the number of sensor nodes.

Action: In each state S_i, an action $a(S_j|S_i)$ is executed to move from S_i to S_j. The action $a(S_j|S_i)$ represents that sensor S_i sends a packet to the sink through sensor S_j, provided that they are in the communication range of each other.

Reward: Reward function represents the cost of forwarding the data from the sensor S_i to S_j.

3.2. Proposed Reward function

In selecting the route, we have two different objectives namely fault tolerance and congestion control. To achieve fault tolerance in routing, the next hop node should be free from permanent fault and transient fault. Permanent fault occurs when the remaining energy of sensor node is fully exhausted. Due to transient fault, a sensor node may generate wrong readings. Transient fault is identified by measurement correlation. Link quality and node buffer account for congestion. In addition, the selected route should be the shortest one. Taking these objectives into consideration, the parameters used for calculating the reward are listed below:

Hop count: The hop count gives the number of hops between the selected next hop and the sink. The hop count accounts for delay in processing the data in the sensor node. By selecting a path with less number of hops, the data transfer delay can be minimized. Let $HC(S_i)$ represents the number of hops from sensor node S_i to sink.

Node lifetime: The lifetime of a sensor node depends on its remaining energy and average energy consumed per round as given by (1). When the remaining energy of the sensor node falls below a minimum

energy threshold, then the data transfer fails. The node has to retransmit the data again through some other node. A node with higher remaining energy should be selected as the next hop node.

$$LT_i = \frac{RE_i}{ECR_i} \tag{1}$$

where LT_i represent the lifetime of sensor S_i, RE_i represents the remaining energy of sensor S_i and ECR_i represents the average energy consumed in sensor S_i per round.

Average Correlation: The average correlation gives the measure of the correction of the measurement of the next hop node. A sensor node which has higher correlation with the neighbors should be selected, as it is free from transient fault. Two sensor nodes S_i and S_j are said to be neighbors if the distance (d_{ij}) between them is less than the communication range R. Let $x_i(t)$ be the reading of the sensor S_i measured at time 't'. To consider temporal correlation, the readings of the sensor node S_i measured in a time interval 't' is represented as a vector $r_i(t)$ as given by equation (2). The correlation (co_{ij}) between the readings of two sensors S_i and S_j is measured using Extended Jaccard Similarity as given by equation (3).

$$r_i(t) = x_i(t-(n-1)\Delta t) + \cdots + x_i(t-\Delta t) + x_i(t) \tag{2}$$

$$co_{ij} = \frac{r_i(t)\cdot r_j(t)}{\left\|r_i(t)\right\|_2^2 + \left\|r_j(t)\right\|_2^2 - r_i(t)\cdot r_j(t)} \tag{3}$$

where

$$\begin{aligned}\left\|r_i(t)\right\|_2^2 &= \left|x_i(t-(n-1)\Delta t)\right|^2 + \cdots \left|x_i(t-2\Delta t)\right|^2 \\ &+ \left|x_i(t-\Delta t)\right|^2 + \left|x_i(t)\right|^2\end{aligned} \tag{4}$$

Here we consider the sensor readings generated in a time interval Δt within a window of size n. The correlation varies from 0 to 1. When the readings of the two sensor nodes are different from each other, then co_{ij} is zero otherwise co_{ij} is one.

The average correlation of sensor S_i ($ACO(S_i)$) is calculated as the average value of the correlation of the sensor with all its neighbors.

$$ACO(S_i) = \frac{\sum_{j\in Nei(S_i)} co_{ij}}{N_{Nei(Si)}} \tag{5}$$

Where $Nei(S_i)$ represents the set of neighbors of sensor S_i and $N_{Nei(Si)}$ represents the number of neighbors of sensor S_i.

Link Quality: Every sensor node sends data to its neighbor through wireless communication link. Link quality between two nodes S_i and S_j($LQ(S_i,S_j)$) is obtained from the ratio of number of packets correctly sent from the given node to the total number of packets sent. To reduce the number of retransmitted packets, the link with highest link quality must be selected.

Free buffer: To avoid congestion, it is always advisable to select a node with higher free buffer capacity, so that when the arriving rate of the packets exceeds the service rate of the packets, extra packets can be stored in the free available buffer. $FB(S_i)$ represents the free buffer capacity of sensor S_i

Weight values are assigned to each of the above parameters to calculate the reward of the next hop node. A sensor node at state S_t receives a reward value $r(S_t, a_t)$ from the downstream node S_d after executing an action a_t.

$$r(s_t, a_t) = w_1 \frac{1}{HC(S_d)} + w_2 \times LT(S_d) + w_3 \times ACO(S_d) + w_4 \times LQ(S_t, S_d) + w_5 FB(S_d) \quad (6)$$

where $w_1 + w_2 + w_3 + w_4 + w_5 = 1$. The weight values w_i assigned to different metrics are selected as per the application requirement.

3.3. Q-value updation

After forwarding a packet, each sensor node receives a reward from the downstream neighbor to which it has forwarded a packet. The reward is piggybagged in the next data packet. The reward indicates the goodness of performing an action. Each sensor node receives a positive reward, when the packet forwarding is successful. To avoid congested and faulty nodes being selected as next hop, negative rewards are propagated which will be discussed in sections D, E and F. The sensor node calculates the new value for the action as follows:

$$Q_{new}(a_i) = Q_{old}(a_i) + \alpha \times (r(S_i, a_i) - Q_{old}(a_i)) \quad (7)$$

Here $Q_{old}(a_i)$ and $Q_{new}(a_i)$ represent the Q value before and after executing an action a_i, $r(S_i, a_i)$ represent the reward value and α is the learning rate of the algorithm. Instead of using a fixed learning rate, the algorithm uses a flexible learning rate depending upon congestion status.

3.4. Q-learning routing algorithm

During network initialization, each node set a timer for exchanging hello packet (HP) based on its remaining energy. Upon the expiry of the timer, sensor nodes broadcast hello packet in their transmission range, so that the packet is received by all the one hop neighbors of the node. The hello packet contains hop count, lifetime, average correlation, link quality and free buffer information of the node as discussed in section B. When sensor nodes receive the hello packet from the downstream neighbors, they update the Q- table with the Q-value calculated by (7). Sensor nodes select the neighbor with the highest Q-value as the next hop node and send data packet (DP) to that neighbor. After the completion of this action, they receive reward from the downstream neighbor and update the Q-value for that neighbor, if it is positive. If they receive control packet (CP) indicating congestion, then they reduces data rate and send congestion notification to upstream neighbor. The detection of local congestion by the nodes also initiate the transmission of congestion notification to the upstream neighbor. Algorithm 1 explains these steps.

Algorithm 1: Q-learning based routing algorithm

```
begin
Set timer for hello packet exchange
if (timer expires) then
    Broadcast hello packet to one hop neighbors
    Reset timer
end if
if (HP received from downstream neighbors) then
    Update Q-table
end if
if (DP received from upstream neighbors) then
```

```
        Send data to downstream neighbor with highest
                                        Q-value
        Receive reward from downstream neighbor
        Update Q-value of downstream neighbor
else if (CP received from downstream neighbor) then
        Reduce data rate
        Send congestion notification to upstream node
else if (Congestion detected locally) then
        Send congestion notification to upstream node
    end if
end
```

3.5. Congestion Detection

The main aim of the proposed algorithm is to improve the network lifetime by sending the packets from sensor nodes to the sink though less congested non-faulty sensor nodes. To achieve this, the Q-value should be updated dynamically based on current congestion level of the sensor nodes. Each sensor node monitors its buffer occupancy using Exponential Weighted Moving average method as given by (8).

$$\overline{BO_i^r} = (1-w) \times \overline{BO_i^{r-1}} + w \times BO_i^r \tag{8}$$

In the above equation, $\overline{BO_i^r}$ and $\overline{BO_i^{r-1}}$ represent the average buffer occupancy of S_i in round *r*, *r*-1 respectively. BO_i^r is the current buffer occupancy of node S_i in round r and w represent the weight value. The buffer status (BS_i^r) of the node in the current round is calculated as the difference between current buffer occupancy BO_i^r and average buffer occupancy $\overline{BO_i^r}$ as given by (9).

$$BS_i^r = BO_i^r - \overline{BO_i^r} \tag{9}$$

The current buffer status is compared with minimum and maximum buffer threshold values. The minimum buffer threshold and the maximum buffer threshold values are set to 25% and 70% of the total buffer size respectively.

$$BT_{\min} = 0.25 \times B_{total} \tag{10}$$

$$BT_{\max} = 0.70 \times B_{total} \tag{11}$$

If the buffer status BS_i^r is less than the minimum threshold, then the node is not congested and it is in safe zone. In this case, the learning rate is set to a small value of 0.1. If the buffer status is in between the minimum threshold and the maximum threshold, then the node is in probably congested state. In this case, the learning rate is set to a medium value of 0.5. If the buffer status is greater than the maximum threshold, then the node is in congested state and the learning rate is set to a larger value of 0.9. Algorithm 2 clearly explains the steps involved in congestion detection.

/*Algorithm 2: Congestion Detection */

begin

$\overline{BO_i^r} = (1-w) \times \overline{BO_i^{r-1}} + w \times BO_i^r$

$BS_i^r = BO_i^r - \overline{BO_i^r}$

if ($BS_i^r > BT_{\max}$) then

Node status = congested
Set learning rate = 0.9
Send negative Reward—$r(S_t, a_t)$
Send control packet

else if (($BS_i^r > BT_{\min}$ and $BS_i^r < BT_{\max}$)) then

Node status = probably_congested
Set learning rate = 0.5
Send negative Reward—$r(S_t, a_t)$
Send control packet

else

Node status = non congested
Set learning rate = 0.1
Send positive reward $r(S_t, a_t)$

end if

end if

end

3.6. Fault diagnosis

In fault diagnosis, sensor nodes performs self-diagnosis followed by neighbor diagnosis. Self-diagnosis process is based on the comparison of current reading vector with previous reading vector. When the difference between these two vectors is more than the threshold, then the node starts neighbor diagnosis. Sensor node initiating the neighbor diagnosis sends the diagnosis message. After receiving the diagnosis message, each of its neighbor sends the correlation between its reading and the received reading. If the sensor node receives a correlation of 1 from at least half of its neighbors, then it is a fault free node. Otherwise the node has some transient fault. Once a sensor node is suspected to be faulty, it sends negative reward to its upstream neighbors, so that it can be avoided being selected as the forwarding node. The steps involved in the fault diagnosis are shown in algorithm 3.

3.7. Negative rewards

Sensor nodes send positive rewards $r(S_t, a_t)$ when they are non-congested and non-faulty. When the sensor nodes detect themselves to be congested and faulty, then they send negative reward $-r(S_t, a_t)$ to avoid these nodes to be selected as next hop nodes. When a sensor S_i receives a negative reward from S_j, then it selects another node as next hop. The reward is calculated by (6).

//Algorithm 3:Fault Diagnosis algorithm

begin

if($|r_i(t) - r_i(t-1)| \geq \delta$) then

for all $Nei(S_i)$ do

Obtain the correlation from $Nei(S_i)$

$$ACO(S_i) = \frac{\sum_{j \in Nei(S_i)} co_{ij}}{N_{Nei(Si)}}$$

if ($ACO(S_i) \geq 0.5$) then

set status = non-faulty

send positive reward $r(S_t, a_t)$

else

set status= faulty

send negative reward to upstream neighbors

end if

end for

end if

end

4. SIMULATION RESULTS

The performance of the QFTRCC algorithm is evaluated by simulating the algorithm in Matlab R2011b. The simulations are conducted for a sensor network of area 200 m × 200 m. The simulation uses a static sink at the center of the field at the coordinate (100, 100). The simulations are conducted by varying the number of sensor nodes from 100 to 500. The simulations are performed 100 times, each time with 2000 rounds and the average value of readings are plotted in the graphs. The QFTRCC algorithm is compared with CODA algorithm. To evaluate the energy consumption of the proposed algorithm, the energy model

Table 1
Parameters and their values

Parameter	*Value*
Network area	200 m × 200 m
Number of nodes	100 to 500
Sink location	100, 100
Data Packet Size	4 Kb
Buffer Size	10 packets
ε_{fs}	10 pJ/bit/ m2
ε_{mp}	0.0013 pJ/bit/ m4
Eelec	50 nJ/bit
Eagg	5nJ/bit/signal

proposed by Heinzelman et al. as in [22] is used. Table 1 shows the parameters and their values used in the simulation. To prove the performance efficiency of the proposed algorithm packet drop ratio, energy consumption per packet and throughput are considered as the performance metrics.

4.1. Packet drop ratio

The packet drop ratio is the ratio of total number of lost packets to the total number of transmitted packets. From the figure 1, we can observe that QFTRCC algorithm achieves 1.994% lower packet drop ratio on average compared to CODA. The reduction in the packet drop ratio is due to that fact the proposed scheme selects routes based on both congestion level and the fault tolerance level. The best next hop node is chosen so that packets dropped due to congestion and faulty sensor nodes are minimized.

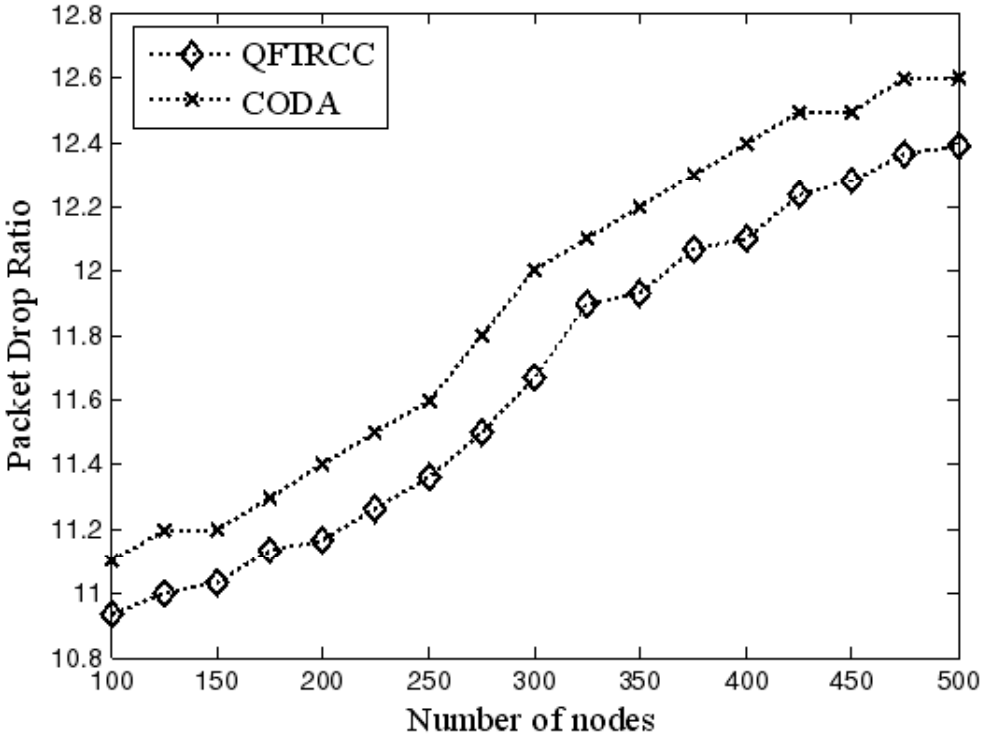

Figure 1: Packet Drop Ratio

4.2. Energy consumption per packet

The average energy consumption per packet in QFTRCC and CODA algorithms are compared with each other for different number of sensor nodes. From the figure 2, it is observed that CODA consumes 9.315% more energy than QFTRCC. In QFTRCC algorithm, since the number of dropped packets are reduced, the additional energy required for retransmission of the drop packets is reduced and hence the energy consumption per packet is reduced. The reduction in energy consumption per packet improves the useful lifetime of the sensor network.

4.3. Throughput

Throughput is measured by the cumulative sum of the packets received at the sink node till a particular round. Figure 3 shows the throughput comparison of QFTRCC and CODA algorithms. From the figure, it is clearly observed that QFTRCC algorithm achieves 15.8% improvement in throughput compared to CODA. The main reason for the improvement in throughput is the use of non-faulty nodes as the next hop in addition to selecting non-congested nodes.

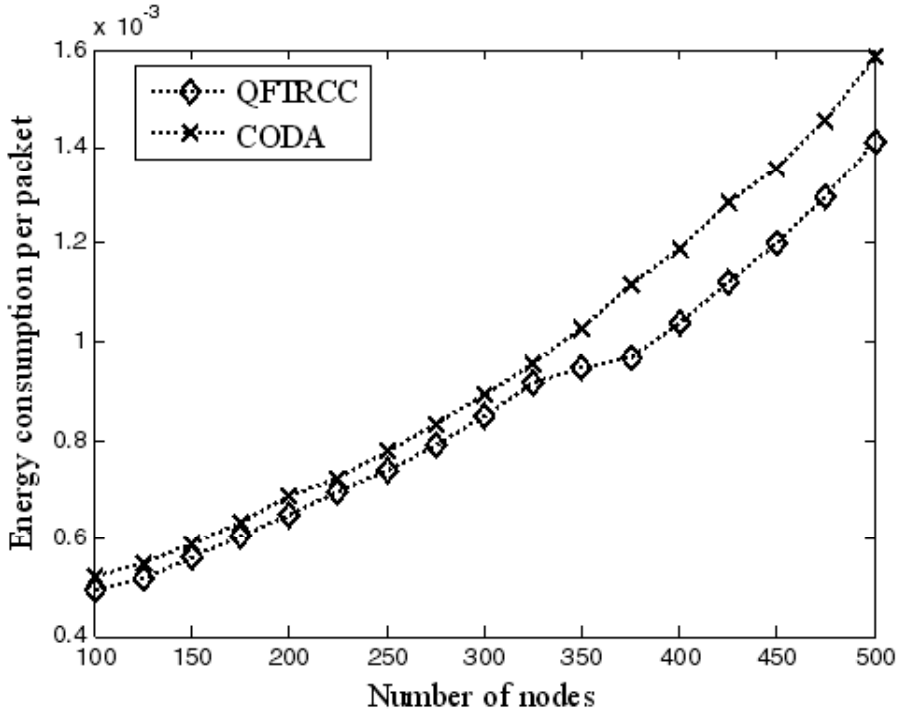

Figure 2: Energy consumption per packet

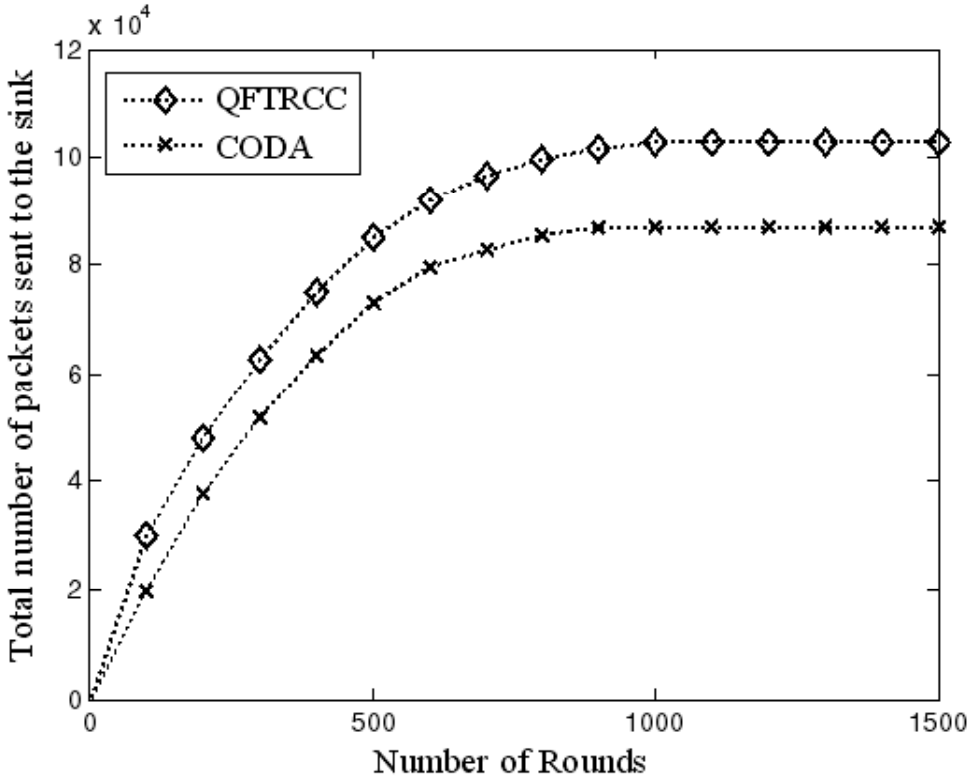

Figure 3: Throughput

5. CONCLUSION

Q-learning based routing algorithm with fault tolerance and congestion control is proposed in this paper. Positive rewards are assigned to next hop nodes based on node lifetime, average correlation, link quality, free buffer and hop count. Learning rate of the algorithm is set based on congestion. Dual threshold scheme detects congestion based on average buffer occupancy. Negative rewards are propagated to the

upstream nodes in the occurrence of congestion and fault. Simulation results show that QFTRCC algorithm achieves lower energy consumption, lower packet drop ratio and higher throughput compared to CODA algorithm.

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," Computer networks, vol. 38, pp. 393-422, 2002.

[2] M. A. Kafi, D. Djenouri, J. Ben-Othman, and N. Badache, "Congestion control protocols in wireless sensor networks: a survey," IEEE Communications Surveys & Tutorials, vol. 16, no. 3, pp. 1369-1390, 2014.

[3] A. Ghaffari, "Congestion control mechanisms in wireless sensor networks: A survey," Journal of Network and Computer Applications, vol. 52, pp. 101-115, 2015.

[4] A. Mahapatroa, and P. M. Khilar, "Transient fault tolerant wireless sensor networks," in Proc. Computer, Communication, Control and Information Technology Conference, pp. 97-101, 2012.

[5] C. Y. Wan, S. B. Eisenman, and A. T. Campbell, "CODA: congestion detection and avoidance in sensor networks," in Proc. 1st International Conference on Embedded networked sensor systems, pp. 266-279, 2003.

[6] G. Wu, F. Xia, L. Yao, Y. Zhang, and Y. Zhu, "A hop-by-hop cross-layer congestion control scheme for wireless sensor networks," Journal of Software, vol. 6, no. 12, pp. 2434-2440, 2011.

[7] M. Gholipour, A. T. Haghighat, and M. R. Meybodi, "Hop-by-hop traffic-aware routing to congestion control in wireless sensor networks," EURASIP Journal on Wireless Communications and Networking, vol. 1, 15, 2015.

[8] L. Q. Tao, and F. Q. Yu, "ECODA: enhanced congestion detection and avoidance for multiple class of traffic in sensor networks," IEEE transactions on consumer electronics, vol. 56, no. 3, pp. 1387-1394, 2010.

[9] C. Wang, B. Li, K. Sohraby, M. Daneshmand, and Y. Hu, "Upstream congestion control in wireless sensor networks through cross-layer optimization," IEEE journal on selected areas in communications, vol. 25, no. 4, pp. 786-795, May 2007.

[10] S. Sridevi, M. Usha, and G. P. A. Lithurin, "Priority Based Congestion Control For heterogeneous Traffic in Multipath Wireless Sensor Networks," in Proc. IEEE International Conference on Computer Communication and Informatics (ICCCI), pp. 1-5, 2012.

[11] A. A. Rezaee, M. H. Yaghmaee, A. M. Rahmani, and A. H. Mohajerzadeh, "HOCA: Healthcare Aware Optimized Congestion Avoidance and Control protocol for Wireless Sensor Networks," Journal of Network and Computer Applications, vol. 37, pp. 216-228, 2014.

[12] A. Mahapatro, and P. M. Khilar, "Fault diagnosis in wireless sensor networks: a survey," IEEE Communications Surveys & Tutorials, vol. 15, no. 4, pp. 2000-2026, 2013.

[13] X. Y. Xiao, W. C. Peng, C. C. Hung, and W. C. Lee, "Using sensor ranks for in-network detection of faulty readings in wireless sensor networks," in Proc. 6th ACM international workshop on Data engineering for wireless and mobile access, pp. 1-8, June 2007.

[14] K. L. A. Yau, P. Komisarczuk, and P. D. Teal, "Reinforcement learning for context awareness and intelligence in wireless networks: Review, new features and open issues," Journal of Network and Computer Applications, vol. 35, no. 1, pp. 253-267, 2012.

[15] A. Forster, and A. L. Murphy, "CLIQUE: Role-free clustering with Q-learning for Wireless Sensor Networks," in Proc. 29th IEEE International Conference on Distributed Computing Systems, ICDCS'09, pp. 441-449, June 2009.

[16] X. Liang, I. Balasingham, and S. S. Byun, "A reinforcement learning based routing protocol with QoS support for biomedical sensor networks," in Proc. IEEE First International Symposium on Applied Sciences on Biomedical and Communication Technologies, pp. 1-5, Oct. 2008.

[17] F. Kiani, E. Amiri, M. Zamani, T. Khodadadi, and A. A. Manaf, "Efficient intelligent energy routing protocol in wireless sensor networks," International Journal of Distributed Sensor Networks, 2015.

[18] H. Tan, L. Zhao, W. Liu, Y. Niu, and C. Zhao, "Adaptive congestion avoidance scheme based on reinforcement learning for wireless sensor network," in Proc. IET International Conference on Communication Technology and Application (ICCTA 2011) , pp. 228-232, 2011.

[19] T. Fuchida, K. T. Aung, and A. Sakuragi, "A study of Q-learning considering negative rewards," Artificial Life and Robotics, vol. 15, no. 3, pp. 351-354, 2010.

[20] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and J. Plosila, "Adaptive load balancing in learning-based approaches for many-core embedded systems," The Journal of Supercomputing, vol. 68, no. 3, pp. 1214-1234, 2014.

[21] C. J. Watkins, and P. Dayan, “Q-learning. Machine Learning,” vol. 8, pp. 279-292, 1992.

[22] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, “An Application-Specific Protocol Architecture for Wireless Micro sensor Networks,” IEEE Transactions on Wireless Communications, vol. 1, no. 4, pp. 660-670, 2002.