Distributed Fault Tolerance for WSNs with Routing Tree Overlays

Chilukuri Shanti and Anirudha Sahoo
Department of Computer Science and Engineering
Indian Institute of Technology, Bombay, Powai, Mumbai, India
Email: {shanti, sahoo}@cse.iitb.ac.in

***Abstract*—WSNs are inherently power constrained and are often deployed in harsh environments. As such, node death is a possibility that must be considered while designing protocols for such networks. Rerouting of data is generally necessary so that data from the descendant nodes of the dead node can reach the sink. Since slot allocation in TDMA MAC protocols is generally done based on the routing tree, all the nodes must switch to the new routing tree to avoid collisions. This necessitates disseminating the fault information to all the nodes reliably. We propose a flooding algorithm for disseminating fault info to the network reliably even in a lossy channel. Simulation results show that the proposed flooding scheme consumes lesser energy and converges faster than a simple flooding scheme.**

I. Introduction

Wireless Sensor Networks (WSNs) have a wide range of potential applications such as environment monitoring, earthquake detection, patient monitoring systems, forest fire detection etc. Sensor networks are also being deployed for many military applications, such as target tracking, surveillance and security management. Applications and protocols for WSNs should be carefully designed in an energy-efficient manner so that the lifetime of sensors can be longer, since they are generally battery-powered. If the sensor network is to be used for real time applications, the MAC protocol should provide QoS (e.g., delay) guarantee, in addition to being energy efficient. An efficiently designed TDMA-based MAC can save a lot of power compared to CSMA-based MACs, while transporting data with minimum delay as reported in [1], [2].

Due to the harsh environments in which WSNs are increasingly being deployed ([3], [2]), the possibility of nodes getting disconnected from the network is high. Failure of a node due to a dead battery is also possible. Such scenarios may lead to the network graph no longer being connected. If data has to be reported to the sink (base station) reliably even after node failure, node replacement or node reconfiguration is mandatory. In this paper, we focus on fault tolerance in WSNs using TDMA MACs, keeping in view applications that require delay guaranteed and reliable data delivery, with minimum energy expenditure.

TDMA MAC protocols have a schedule of time slots that the nodes follow to avoid collision. Slots are assigned to nodes so that the transmissions do not interfere with each other. [2] proposes a centralized schedule such that data reaches the sink in a single TDMA frame, while keeping the length of the TDMA frame minimum, by reusing slots beyond two-hop neighbors. [1] proposes a delay-guaranteed schedule that reuses slots beyond the interference range of a node. In this protocol, routing is automatically achieved together with MAC, as only the next-hop relaying node stays awake to listen to transmitted data. In any TDMA protocol with delay-guarantee, relaying nodes must be given enough slots to transmit their own sensed data, as well as data to be relayed by them. In centralised or distributed slot allocation, if some nodes do not receive the information (the new schedule in the centralized case and information about the dead node in the distributed case) the network will have nodes following inconsistent schedules. This may lead to collisions and loss of data, which is against the very design goal of a TDMA MAC. Hence, the focus of our study is to provide fault tolerant service in a network with a lossy channel, following a TDMA MAC with a routing tree overlay. We propose an energy efficient flooding algorithm that ensures that all the nodes receive the fault messages even in a lossy channel. The proposed algorithm takes much lesser time and energy compared to a simple flooding scheme.

The rest of the paper is organized as follows: In Section II, we discuss related work. Section III describes the network topology and the flooding algorithm proposed by us. Results of simulation and conclusions are given in Sections IV and V.

II. Related Work

Fault detection and reporting has been studied both independently and as part of the MAC protocol. [4] presents a survey of approaches to fault tolerance and detection techniques in WSNs. The authors provide a classification of faults and categorise the surveyed approaches based on their ability to detect and recover from faults. Faults can be detected by self-diagnosis based on the battery voltage as described in [5]. Another way to detect a fault is upon the absence of data from the neighboring nodes, as in [6]. However, this is not a correct way to detect a fault in event-driven networks, where data is sent only when there is an event of interest.

When a node detects a faulty node, it reports the fault to the sink or to other nodes that may be involved in fault recovery. These fault reports may be sent using CSMA as done in [6] or TDMA. Fault recovery is generally done by replication of the network elements that are prone to failure.

III. Network Topology

We represent the network as a routing tree with the sink as the root. The number of levels in the routing tree is $(H+1)$, with the sink at level *0*. The number of nodes at the i^{th} level is

978-1-4244-8953-4/11/$26.00

Notation	Definition
G	number of nodes in the network including the sink
$n_{i,j}$	j^{th} node at the i^{th} level of the routing tree
$p(n_{i,j})$	parent node of $n_{i,j}$
M_i	number of nodes at level i
$\alpha(n_{i,j})$	number of children of $n_{i,j}$
$c(n_{i,j}, k)$	k^{th} child of $n_{i,j}$
$S(n_{i,j})$	number of nodes in the subtree of $n_{i,j}$
$H+1$	total number of levels in the routing tree including the sink at level *0*
T	number of slots in the TDMA frame
ρ	duration of each TDMA slot (secs.)

TABLE I
EXPLANATION OF NOTATIONS

M_i. The j^{th} node (numbered from the left and starting index of 0) at level i is denoted by $n_{i,j}$. The parent of node $n_{i,j}$ is denoted by $p(n_{i,j})$. The number of children of $n_{i,j}$ is given by $\alpha(n_{i,j})$ and $c(n_{i,j}, k)$ denotes the k^{th} child of $n_{i,j}$, where $0 \leq k < \alpha(n_{i,j})$. Each node senses data and transmits it to the sink in one or more hops. For quick reference, the notations used in this section and the rest of the paper are given in Table I.

A. Fault Detection and Reporting

Detecting that a node is faulty can be done in several ways, as discussed in Section II. [6] proposes a MAC in which a node is considered dead if there is absence of data from a child or descendant node for two data gathering cycles. However, this may lead to detecting a fault even when a node is not dead, if it has no data to send. Hence a better way to detect a fault is to monitor a periodic control message that is sent in a control slot, irrespective of the event rate. A node $n_{i,j}$ is determined to be dead if the node's neighbor detects that it has not received some predefined number of consecutive control messages. Then this neighbor (say node X) creates a *fault report*. The fault report contains information about the id of the dead node and is sent to the sink in the next data slot of node X. A bit flag in the header of the data packet can indicate whether the packet received by the sink contains data or fault information.

B. Dissemination of Fault Information in the Network

Once the sink receives a fault report, in centralized slot allocation, it can create a new schedule with new parents for the children of the dead node. [6] proposes a fault tolerance mechanism in which the superframe length is increased by the number of extra slots needed for taking on the children of a dead node. This may be quite large for networks with many hops or nodes. This increase becomes quite substantial if the network has multiple faults. A large TDMA frame size is not desirable for delay constrained applications like fire monitoring ([3]), voice data ([2]) etc.

The increase in the TDMA frame length can be reduced (especially in the case of multiple faults) if the slot allocation is done with the changed network tree. However, this involves sending changed schedules (in case of centralized slot allocation) or sending information about the dead node (in case of a distributed slot allocation like that used in [1], where nodes calculate the new schedule based on the new topology) to all the nodes in the network. This must reach all the nodes in the network, otherwise some of them start following the new schedule while some don't. To ensure this in a wireless network which is prone to packet losses, we resort to flooding the network.

We propose a CSMA-based flooding scheme that ensures that all the nodes receive the fault information/new schedule, while conserving energy. When the sink receives a fault report, its sends a message to the nodes in a control slot of the next TDMA frame, directing them to enter the CSMA mode after the end of the current TDMA frame. In the CSMA mode, each node starts a beacon timer and stays in the listening mode. The sink originates a *fault info* packet with the new schedule/fault information. The format of this packet is *id, bitmap, TTL, seq.no., payload*. The $bitmap$ field contains a bitmap of G bits, where G is the number of nodes in the network, including the sink. The TTL field is the time to live field, which is set by the sink depending on the maximum number of hops in the network. The sequence number field carries the sequence number of the packet, which can be used to discard duplicate *fault info* packets. The TTL and sequence number fields help in restricting the number of packets in the flood.

We assume that nodes are aware of the network topology. This can be done by loading the nodes at the deployment phase or by an initial beacon exchange phase as described in [1] or [6]. Each bit in the $bitmap$ field corresponds to a node in the network. The j^{th} node of the i^{th} level is denoted by the k^{th} bit of the bitmap. Since M_i denotes the number of nodes at the i^{th} level, k is given by

$$k = \begin{cases} j + \sum_{l=0}^{i-1} M_l & \text{if } 0 < i \leq H \\ 0 & \text{if } i = 0 \end{cases} \quad (1)$$

Hence, k is in a way the node's unique index in the network. The sink's index is 0. If the k^{th} bit is 1, it denotes that the k^{th} node has received the *fault info* message. All the bits in the bitmap except for the zeroth bit are set to 0 by the sink. Each node maintains a list of the nodes that have received the *fault info* message by keeping the bitmap in its memory. In addition, each node maintains a table Ω. Each entry of the table Ω is a tuple $\{$*id, seq_no*$\}$.

The seq_no field is used to determine the freshness of a packet, so that old packets from a node can be dropped if there are fresher packets from the same node. Initially, each node knows its id and the number of nodes G in the network. The bitmap in the node's memory, $bitmap_{memory}$ is initialised to all 0s and the table Ω is empty. Every node (other than the sink) is in the listenting mode. The sink originates the fault information message with the TTL field set to MAX_TTL and the bitmap having all 0s except the 0^{th} bit which is set to 1. The $payload$ field contains the actual fault information. MAX_TTL is set to the maximum distance (in terms of hops) from the sink to any node in the network. This can be the new schedule when slot allocation is done in a centralized manner (by the sink) or the location of the dead node, when the nodes calculate their slots in a distributed manner with the knowledge of the network topology.

Each node maintains two timers: the *beacon timer* and the *silence timer*. The silence timer is used to wait for any further fault messages in the network. It expires when the node does

not hear any message for the duration of the timer. The beacon timer is used to broadcast the fault info message periodically, until all bits $bitmap_{memory}$ are 1, as explained below. When a node receives a fault info message, it checks the seq_no and TTL fields to check if this is a fresh packet, as shown in Figure 1. If the packet is fresh, it updates the seq_no for this node's entry in the table Ω. The node then updates the bitmap in its memory ($bitmap_{memory}$) as explained below:

- When a node first receives a message, it sets the bit that corresponds to itself to 1.
- When a node receives a *fault info* packet, it performs a simple bitwise *OR* of the bitmap in its memory and the received bitmap.

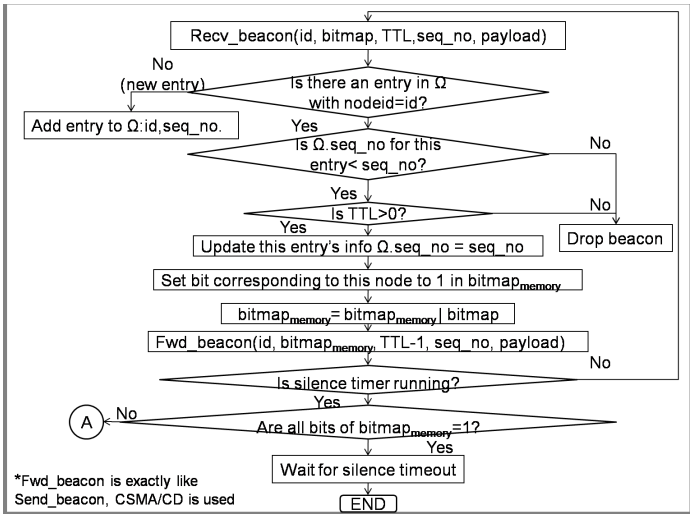

Fig. 1. Flowchart for receiving and forwarding fault info messages

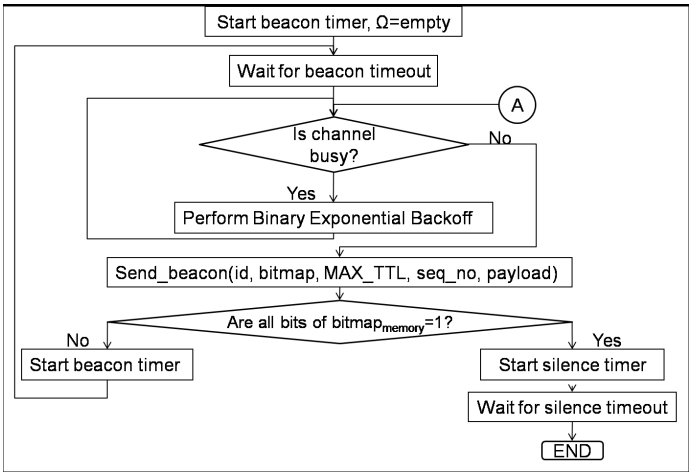

Fig. 2. Flowchart for sending fault info messages

Hence, the $bitmap$ field of the packet acts as an acknowledgement from all the nodes which have received the message. The bitmap in the node's memory acts as the state of the entire network (as perceived by the node) in terms of receipt of the *fault info* packets. The node places the modified bitmap from its memory and decremented TTL in the $bitmap$ and TTL fields respectively. It then forwards this message when its beacon timer expires. As long as all the bits in the bitmap in the node's memory are not 1, it means that some node in the network has not received the *fault info* packet and the node periodically keeps sending the *fault info* packet as shown in the flowchart presented in Figure 2. When a node receives a message, it checks if the $bitmap_{memory}$ has all 1s. If not, it keeps sending periodic messages. If the $bitmap_{memory}$ has all 1s, the silence timer is started. When the *silence timer* goes off, it means that none of the nodes are sending fault info messages. This means that the $bitmap_{memory}$ of every node has all 1s, which in-turn means that all the nodes have received the fault info message. Hence the node stops sending the *fault info* messages when the silence timer goes off.

The length of the bitmap field considered by us is 64 bytes, which is enough for 512 nodes. Practical deployments even in large areas have much lesser number of nodes. Even for large networks, since the bitmap reflects the state of the entire network, the algorithm converges quickly, making it more scalable (in terms of time and energy consumed for flooding) than the simple flooding algorithm, in which no bitmap field is necessary.

IV. Simulation Results

We set up simulation in ns2 to study the performance of the algorithms proposed by us. As an example, we consider the parameter values given in Table II. The values of the physical parameters of the nodes are taken from [7], which contains representative values for μamps sensor nodes. The μamps nodes are designed for transmitting data up to 1Mbps within a range of 100 m. Our simulation and analysis takes a data rate of 19.2 Kbps. We repeated all the experiments for a confidence interval of 97%.

Parameter	Value	Parameter	Value
ρ	27 ms	R	100m
P_{rx}	63mW	P_{tx}	30mW
P_{idle}	30mW	P_{sleep}	0.003mW
$TransitionPower$	30mW	$TransitionTime$	2.45ms
$InitialNodeEnergy$	54,000J		

TABLE II
Physical parameters used in simulation

Flooding the network with fault info messages is done to ensure that all nodes receive the messages even with a lossy channel. While flooding ensures this, the energy and time required to flood the network increase with the increase in channel losses. Figure 3 shows the time taken for bitmap flooding to converge in a network of radius $150m$ with 172 nodes. We considered the log-normal shadowing model of radio propagation as it accounts for path loss and the variation of received power. The path loss exponent is taken to be 4.0, 5.0 and 6.0 (values typical of obstructed closed space). For each path loss exponent value, we varied the shadowing deviation from 3.0 (typical for closed space, line-of-sight) to 12.0 (outdoor). It can be seen that as the path loss exponent increases more time is consumed by the network. Also, increasing shadowing deviation increases the time taken for flooding, except for the case of a low path loss exponent (4.0). For this case, the time consumed does not change much with the shadowing deviation, as the number of packets lost

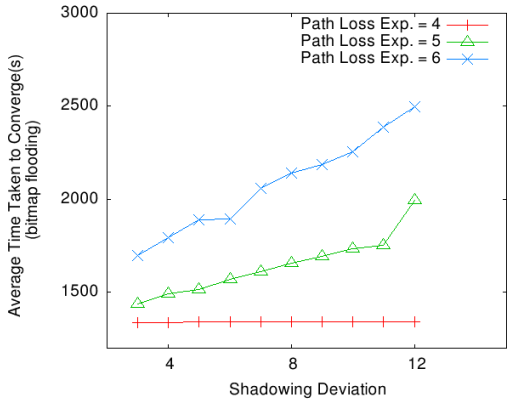

Fig. 3. Effect of Propagation Loss on Convergence Time of Flooding (network radius = 150m)

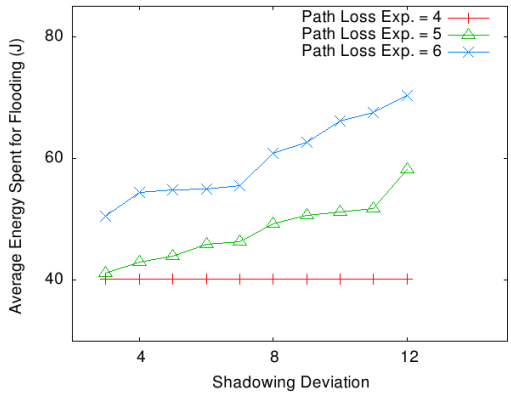

Fig. 4. Effect of Propagation Loss on Energy Spent for Flooding (network radius = 150m)

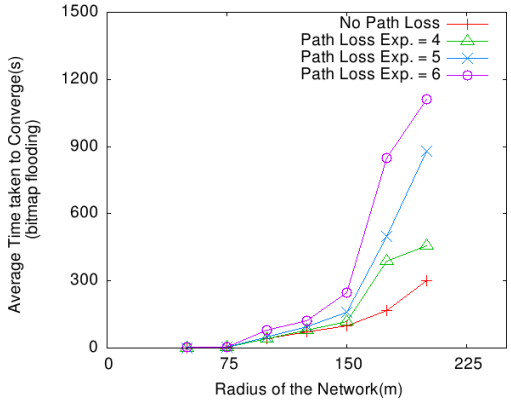

Fig. 5. Effect of Propagation Loss on Convergence Time of Flooding (standard deviation=8.0)

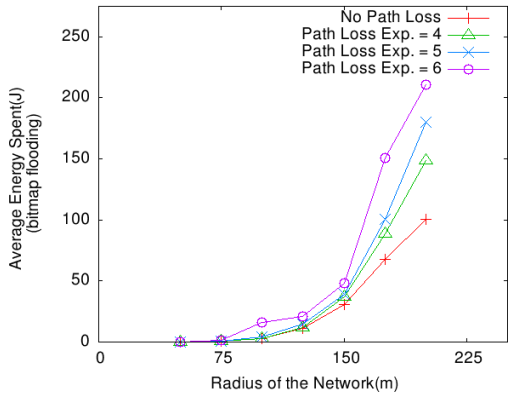

Fig. 6. Effect of Propagation Loss on Energy Spent for Flooding (standard deviation=8.0)

is low. All nodes received the fault info messages in all cases of path loss and deviation.

Figure 4 depicts the average energy spent by a node for bitmap flooding to converge in such a lossy network. It is is seen that the energy spent shows a similar trend as that of the time to converge. We note that the maximum energy spent (for path loss exponent=6.0, shadowing deviation=12.0) is 70.9J, which is 0.13% of the total node energy.

Figures 5 and 6 show the time taken for flooding to converge and the average energy spent by a node respectively, as the network radius increases form 50m to 275m. For these graphs, the deployment is of uniform node density of $0.0015 nodes/m^2$. The lossy nature of the channel is varied by considering the shadowing model, taking the path loss exponent to be 0.0 (no loss), 4.0, 5.0 and 6.0. The standard deviation is taken to be 8.0. As expected, the energy spent and time taken to converge increase as the network radius and the path loss exponent increase. Due to lack of space, simultion results for non-lossy channels and two heuristics that we proposed for parent assignment are not included here. They are available in [8].

V. Conclusions

We presented an energy-efficient and reliable bitmap-based flooding mechanism for WSNs with routing tree overlays. The proposed flooding scheme can be used to disseminate information necessary for rerouting upon a fault. Simulation results show that this scheme consumes much lesser energy and converges faster than a simple flooding scheme, even when the channel is lossy. Simulation also shows that the proposed flooding algorithm is more time and energy consuming for a lossy channel.

References

[1] C. Shanti and A. Sahoo, "Dgram: A delay guaranteed routing and mac protocol for wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, pp. 1407–1423, 2010.

[2] A. Rowe, R. Mangharam, and R. Rajkumar, "RT-Link: A global time-synchronized link protocol for sensor networks," *Ad Hoc Netw.*, vol. 6, no. 8, pp. 1201–1220, 2008.

[3] H. Xu, L. Huang, J. Wu, Y. Wang, B. Xu, J. Wang, and D. Wang, "Wireless fire monitoring system for ancient buildings," in *Proceedings of the 2nd international conference on Scalable information systems (InfoScale '07)*, 2007, pp. 1–4.

[4] H. Alwan and A. Agarwal, "A survey on fault tolerant routing techniques in wireless sensor networks," in *SENSORCOMM '09: Proceedings of the 2009 Third International Conference on Sensor Technologies and Applications*, 2009, pp. 366–371.

[5] L. Benini, G. Castelli, A. Macii, E. Macii, M. Poncino, and R. Scarsi, "A discrete-time battery model for high-level power estimation," in *DATE '00: Proceedings of the conference on Design, automation and test in Europe*, 2000, pp. 35–41.

[6] W. L. Lee, A. Datta, and R. Cardell-Oliver, "FlexiTP: A Flexible-Schedule-Based TDMA Protocol for Fault-Tolerant and Energy-Efficient Wireless Sensor Networks." *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 6, pp. 851–864, 2008.

[7] E.Shih,S.-H.Cho, N.Ickes, R.Min, A.Sinha, A.Wang and A.Chandrasekharan, "Physical layer driven algorithm and protocol design for energy-efficient wireless sensor networks." in *IEEE International conference on Mobile computing and networking (Mobicom)*, 2001, pp. 272–287.

[8] C. Shanti and A. Sahoo, "Distributed Fault Tolerance for WSNs with Routing Tree Overlays." IIT Bombay, Tech. Rep. TR-CSE-2010-33, December 2010.