Bitvector: Fault tolerant aggregation scheme for monitoring in nuclear power plants

Wenchen Wang, Daniel Mossé
Computer Science Department
University of Pittsburgh
{wangwenchen, mosse}@cs.pitt.edu

Daniel G Cole
Department of Mechanical Engineering and Materials Science
University of Pittsburgh
dgcole@pitt.edu

***Abstract*—Industrial wireless sensor networks (IWSNs) have become popular, given the lower cost of installation than wired networks and their flexibility. WSNs are applied in monitoring, measurement, and control. Nuclear power plants (NPPs), in particular, have not been eager to adopt WSNs due to reliability, electromagnetic and radio frequency interference (EMI/RFI) as well as security concerns. On the other hand, the two main benefits from adding WSNs to NPPs are *increased reliability* from using robust wireless monitoring as a backup network for the primary wired system and *reduce costs* from installation and maintenance (in comparison to wired networks).**

We propose a new fault model and a fault tolerant in-network aggregation scheme, *bitvector*. First, in order to mimic the interference environment in NPP, we define a new fault model with time-dependent faults. We propose the *fault duration* concept, a time span during which the link error rate is fixed, such as when EMI/RFI appears between nodes. Therefore, the fault duration is an indicator of the network stability. Second, based on the fault model, there are some network links whose quality are not good during a fault duration interval. Bitvector estimates neighbor link quality dynamically and orders primary and backup parents dynamically according to link quality estimation. In addition, since data rate is low in NPPs, bitvector is able to use redundancy to improve accuracy. In particular, for some faults, bitvector defers fault detection and fault recovery by one sensing interval. Experiments show that bitvector significantly improves performance in root mean square (RMS) error and correct message rate; benefits of bitvector are even larger for environments with high fault rates.

Keywords—nuclear power plant; condition monitoring; in-network aggregation; time-dependent fault model

I. Introduction

Industrial Wireless Sensor Networks (IWSNs) are a type of WSNs designed for industrial environment. The collaborative nature of IWSNs brings several advantages over traditional wired industrial monitoring and control systems, including self-organization, rapid deployment, flexibility, and inherent intelligent-processing capability [1]. The nuclear industry has, however, has been slower in implementing new technologies—wireless technologies are no exception [2]. Nuclear power plants are not willing to adopt WSNs due to three concerns [3]: reliability, EMI/RFI (electromagnetic and radio frequency interference) and security. Wireless condition monitoring system has attracted more and more attention in NPPs field. Condition monitoring in NPPs requires that suitable sensors be used, dedicated parameters be monitored, baseline (normal conditions) be defined and the trend of the data be captured to identify condition changes [4]. The authors in [5] propose a real time condition monitoring architecture of NPPs using WSNs. The author in [2] proposes a design of prototype condition monitoring system. They have a scenario where data from wireless sensors installed on plant equipment is integrated with existing wired process sensors. Data from both wired sensors and wireless sensors sends to data acquisition and analysis servers.

There are several special characteristics of WSNs in NPPs. For our paper, we focus on two of them. The first is high packet error rate because of EMI/RFI. The second is the low data rate in control systems of NPPs.

According to [3], EMI/RFI is a concern because the high levels of electromagnetic noise from high-powered devices and ionizing radiation sources create a harsh environment for many commercial wireless devices to perform reliably. In addition, the signal-to-noise-ratio (SNR) of hot-leg coolant temperature sensor in NPPs is from $10dB$ to $7dB$ [6], which means the environment is very noisy. Therefore, EMI/RFI leads to high packet error rate in WSNs.

Since wireless condition monitoring in NPPs is a backup network, data is not sent out very frequently as the general wireless sensor network. Therefore, the data rate is relatively low.

We present a wireless sensor network scheme, *bitvector* that meets the two afore mentioned characteristics of NPPs condition monitoring. We make three contributions as follows.

- We propose a new time-dependent fault model that includes the *fault duration* concept to simulate fault rate varying with time. This is needed because the interference in the wireless environments in NPPs is unpredictable and it could cause permanent, intermittent or transient faults.
- For the challenge of having high error rates, some links in WSNs are affected by NPP interference for a fault

duration time. Bitvector is able to estimate link quality dynamically, and to automatically select the parent-child pairs according to link quality.

- We take advantage of the low data rate and judiciously add little redundancy to improve accuracy.

We compare bivector with Ridesharing [7] and FOMA [8]. Through simulations, we show that bitvector can gain as much as 40% more accuracy than FOMA and 20% more accuracy than Ridesharing, while delivering lower message overhead.

The rest of the paper is organized as follows. In Section II, we review the related work about link quality estimation, in-network aggregation and fault model design. Section III presents the basic idea of the bitvector scheme. In Section IV, we discuss about our fault model design. In Section V, we explain how we setup our simulation and experiment results. Section VI summaries our work and presents conclusions.

II. BACKGROUND

In order to monitor advanced small modular reactors (SMRs) in NPP, we proposed a supervisory networked control system architecture, shown in Figure **1**. For each SMR, there are two control loops. First, a primary control loop (local controller -> local actuator -> plant -> local controller) is connected by a field bus. Second, a secondary control loop (sensor -> module-level supervisory controller -> local actuator -> plant -> sensor) is connected by a wireless sensor network. The secondary control loop is to monitor the condition of the reactor, acting as a backup for the primary control loop. If the local controller or the field bus fails, the module-level supervisory controller (MLSC) acts as a backup controller to control the reactor.

For module-level wireless sensor network, sensors attached to SMR sensed different parameters of SMR. For each parameter, there are several sensors monitoring the same parameters. In order to reduce network traffic, in-network data aggregation can be applied. Data aggregation (calculate average value) happens when sensor B receives data from sensor A and the received data of sensor A has the same category as the sensor B. Then sensor B aggregates the data of sensor A with its own data. In this way, the network traffic can be reduced. In our paper, we only consider sensors send out data to MLSC. MLSC acts as a base station.

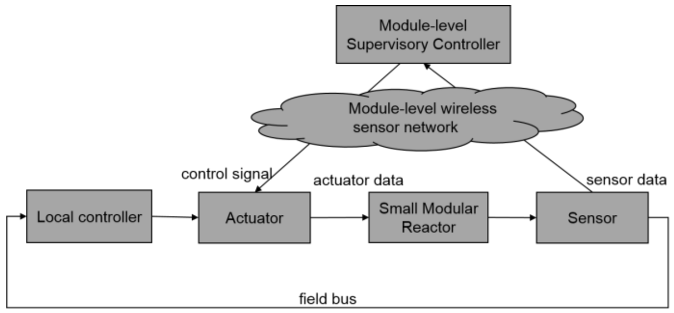

Figure 1 Supervisory networked control system for advanced SMRs

III. RELATED WORK

A. Time-Dependent Fault Model

Faults occur anytime and anywhere in WSNs. Many papers focus on designing fault models for WSNs. According to [20], the nature of the physical phenomenon constitutes the temporal correlation between each consecutive observation of a sensor node. The fault rate of the network varies based on the temporal behavior. So, designing a good fault model that close to the situation in real life is important.

However, as far as we know, there is not much research effort focusing on fault model design. Most of studies assign some constant link or node failure, determined before the network starts. In [21], [22], the authors mention that based on the temporal behavior of a fault, it can be considered as permanent, intermittent or transient faults. However, they just consider permanent faults in their study. The author in [23] designs a fault model by assigning faults to some nodes and also proposes a method for fault detection. Although it is able to localize fault region efficiently, the fault rate of the network is still fixed during network lifetime.

B. In-network data aggregation schemes

Much research in WSNs has been developed in the last decade. To save energy and bandwidth, *in-network data aggregation* schemes for WSNs attempt to aggregate as many sensed values as possible onto a single message not by simply consolidating messages, but in performing some transformation in the sensed values. There are two main types of in-network data aggregation schemes, namely tree-based and gossip-based schemes.

Gossip-based schemes [9], [10] do not create or maintain any specialized routing structure, but their disadvantage is that their actual aggregate computation is slow to converge [11]. We do not consider gossip-based schemes.

For many tree-based topology of in-network aggregation schemes [7], [8], [12], [13], [14], [15], the many parent-child relationships are determined before network starts. Some schemes consider only one parent-child pair for each node, which remains fixed during the entire network lifetime. However, due to failures, the network is typically not static and thus some schemes attempt to “fix” the tree by changing parent-child structure dynamically due to faults [15]. Our starting point is a tree with fixed structure, that is, fixed potential parent-child relationships.

If the quality of the link between child and parent is not good, the child’s message cannot be sent out with high success probability. Therefore, fixed parent-child relation could decrease the network reliability. The Ridesharing scheme [7] uses redundant paths in the WSN to deliver a correct aggregate result to the base station. In particular, Ridesharing organizes sensors in a track graph (see Figure 3 in next section), where one sensor may have multiple parents (one primary parent and one or more backup parents). Each sensor broadcasts its message to its parents. If a primary parent does not receive its child’s message, the backup parent of that child takes care of that message (it aggregates the sensed value in its own message). In this way, a message from one sensor is delivered with high reliability.

Later work [16] removes the assumption that the parent-child relation is fixed and describes how to consider link quality in the scheme, while studying how link quality estimation impacts different data aggregation schemes. For example, it is shown that the quality between tracks have more impact on the error rate than those with in the same track and that finding an optimal parent schedule is NP-hard. Still, link quality estimation mechanism is not embedded in the data aggregation scheme, and the parent-child relation is still fixed.

The OPAG scheme [14] proposes to carry out data aggregation at the multiple levels, not only at the parent level. However, data aggregation nodes are still assigned statically, that is, before the network starts. The FOMA scheme [8] is an improvement of OPAG scheme, where the parent-child relation does not change during network running. FOMA has four phases to collect data: (a) creating a routing tree structure, (b) data aggregation node (DAN) selection, (c) signature construction, and (d) data collection. Although FOMA improves reliability in high network density, it uses a lot control messages and still the parent-child relation is fixed.

C. *Link Quality Estimation for in-network data aggregation scheme*

Link quality estimator is often applied in WSNs. Link quality estimation in WSNs is a fundamental building block for several mechanisms and network protocols. For instance, routing protocols rely on link quality estimation to overcome low-power links' unreliability and maintain the correct network operation [17]. And link quality estimation involves link monitoring, link measurements and metric evaluation [17].

However, not much effort has been done to combine link quality estimation technique and aggregation schemes efficiently in WSNs. In [18], a three-level monitoring architecture is proposed to estimate link quality and then compute aggregates, where each level consists of a class of tools. However, this monitoring approach is energy-intensive. In [19], they present an algorithmic framework for link loss monitoring based on the recent modeling and computational methodology of factor graphs for the data aggregation communication paradigm in sensor networks. However, computing the link quality includes complex computation for link estimation and is very costly in dynamic situations.

IV. Fault Model Design

In the past, most WSNs have considered constant link reliabilities, or a certain distribution with a constant average. However, in real-world networks, especially in NPPs, the fault rate is time-varying as EMI/RFI appears and disappears. This implies that the fault rate for each link does not remain the same for the duration of the network lifetime.

We devised a time-dependent fault model to account for realistic faults in NPP networks. For example, if the EMI/RFI happens in a certain area of NPPs, each sensor pair is affected with the same interference during a specific time interval (duration of EMI/RFI). Each link error rate may stay the same for a while (we call this interval the *fault duration*) and then change to another value (could be higher or lower, depending on the network conditions, but it is usually lower—normal network conditions). Formally, we propose the concept of fault duration, which is a time span that error rate of all links in network remain the same. After that time span, the error rate changes and is fixed for the next fault duration. For the entire network running time, the fault rate changes as a function of the fault duration, which can be seen as an indicator of the network stability.

For example, Figure 2 shows that the link error rate varies with time, when the link error rate is randomly chosen from the range (0.0, 0.4). When fault duration is 1, link error rate varies every one time interval. When the fault duration is 2, link error rate varies within the same range (0.0, 0.4) every two time intervals.

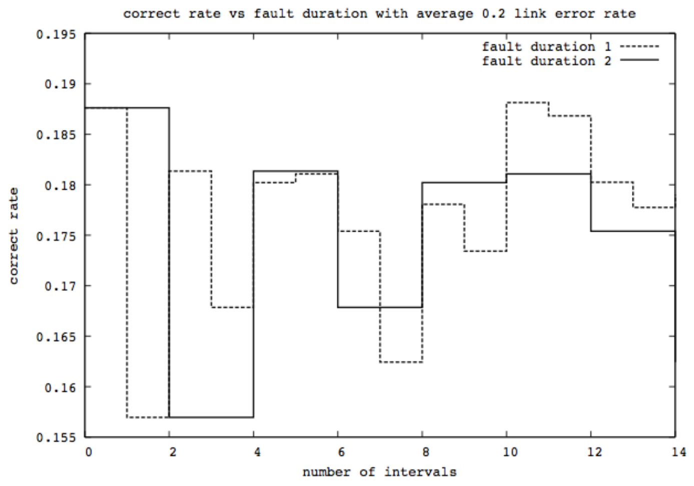

Figure 2 Fault model illustration with fault duration 1 and fault duration 2

V. Bitvector Fautl-Tolerant Aggregation

Our new bitvector scheme enhances/extends Ridesharing [7]. According to the fault model in section IV, we devise a *bit vector* for each node to record its neighbors' situation and the bit vector is appended to the message sent by the node, in addition to the aggregated value (message data/payload). The bitvector scheme has the following contributions beyond the Ridesharing scheme, as describe in the sections below:

- *Dynamic link quality estimation:* estimate neighborhood link quality of a node by efficiently keeping track of recent packet receptions, reflecting the state of the links dynamically.
- *Order primary and backup parents dynamically:* parent selection (primary, backups) for a child is obtained based on the dynamic link quality estimation.
- *Dynamic TDMA scheduling for each level:* the transmission order for nodes in a level is determined at first by node types. Primary parents have higher priority than backup parents.

- *Online fault detection and recovery:* both detection and recovery are achieved efficiently by taking advantage of natural broadcasting and overhearing messages.

Similar to Ridesharing scheme [7], sensors are organized in a track graph, as shown in Figure 3, the base station is in level 0, sensors one hop away from the data sink are in level 1, and so on. A directed edge from a child C to a parent P1 indicates that P1 and C are within each other's radio range and that P1 listens to C's communication. Each sensor has multiple parents, one primary parent and several backup parents.

We define *sensing interval* as the time of data sending from the highest level to the base station. And the sensor's data traffic is sending periodically (every sensing interval). Sensing interval and interval are used interchangeably in our paper.

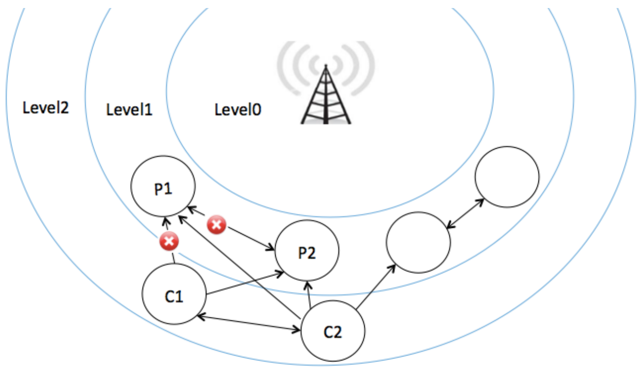

Figure 3 Track Graph Example Illustration

A. *Dynamic Link Quality Estimation*

Link quality is estimated using a bit vector that each node attaches to every data message it sends. One element of a bit vector is called *bit element* corresponding to one neighbor. A bit element contains two fields. The first field is *received bit* (*r-bit*), which indicates whether it has received a message from a certain neighbor for the current interval. If the r-bit is 0, it means the sensor did not receive message from that neighbor, and it is 1 otherwise. The second field is a 3-bit *probability value* (*p-value*) used to evaluate the quality of the link between a certain neighbor and itself. The p-value ranges from 0 (i.e., 000) to 7 (i.e., 111). When a node receives a message from one neighbor correctly, the p-value increases by 1 or remain 7 (if current p-value is already 7) for that neighbor. Otherwise, the p-value decreases by 1. In our protocol, if the p-value is -1, it means the link fails. To start with, the r-bit is initialized to 1 and p-value to 111. In practice, to save message overhead, each node only attaches parent and children bit elements of the node's bit vector to the data message.

Each node knows who their neighbors are in the network construction phase. They know they should hear from all their neighbors once at the end of each interval. If a node does not receive a message from a neighbor, it sets r-bit of the neighbor to be 0 and decreases the p-value by 1 at the end of that interval.

Figure 3 depicts an example. The tower is the base station. Node C_1 has two parents, P_1 and P_2. Node C_1 is located in level 2 and its parents are located one level above it. Assume that the link between C_1 and P_1 and the link between C_1 and P_2 fail. At interval 0, C_1 broadcasts its data message with its bit vector to P_1 and P_2. At the end of interval 0, P_1 realizes that it did not receive the message from C_1. It sets the r-bit of C_1 to be 0 and decreases the p-value by 1 to 110 (i.e., C_1's bit element in P_1's bit vector is 0110). P_2 receives the message from C_1, so the initial value of r-bit of C_1 remains as 1 and the p-value of C_1 as 111. When P_1 broadcasts its message, C_1 cannot overhear P_1's message (we assume links are all symmetric) and C_1 sets P_1's bit element to 0110 at the end of interval 0. The bit vectors of C_1, P_1 and P_2 at the end of interval 0 are shown in Table **1**.

Table 1 Example Result

	P_1 bit element	**P_2 bit element**	**C_1 bit element**
P_1 bit vector	1111	0110	0110
P_2 bit vector	0110	1111	1111
C_1 bit vector	0110	1111	1111

B. *Order primary and backup parents dynamically*

the primary parent with good quality link between the parent and its child has higher success probability of transmitting the child's message. Therefore, we select primary parent with highest link quality to reduce error (improve accuracy).

Data is sent level by level from the highest level to the lowest level. The level that sends out data is called transmission level. In order to let children estimate their parent link quality, children need to listen when their parents are sending out messages. Suppose level *i* is a transmission level, children in level *i+1* and parents in level *i-1* listen messages from level *i*.

In bitvector, since each node maintains a bit vector for its neighbors, a child has its parents' p-values. Each child orders its parents according to p-values. The parents with the highest (lowest) p-value is selected as the primary (last backup) parent. If two p-values are the same, we break ties by node id. With this protocol, we can make parent selection dynamic. For the previous example, please refer to Table **1**. Child C_1 should order its parents as P_2 and P_1 at the end of the interval 0, since P_2 has higher p-value (111) than P_1 (110).

When a parent node receives a message from its child, it checks the bit vector attached in the child's data message to see whether it is selected as the primary parent.

In conclusion, each child decides the order of its primary and backup parents. Each child makes decision at the end of each interval, according to calculated p-values of its parents during that interval. In this way, parents' order can be selected dynamically by their children. And parents receive the parent selection decisions from their children. A parent may act different roles for different children. In other words, a parent node may be a primary parent for a child and a backup parent for another child. For example, in Figure 3, children C_1 and C_2

have two common parents P_1 and P_2. P_1 may be primary parent of C_1, but backup parent of C_2.

C. Dynamic TDMA Scheduling for Each Level

In order to do message correction, the primary parent needs to be scheduled to send out messages before the backup parents. Since parents order is changed by their children dynamically, we need to specify the sending orders dynamically. But an optimal parent scheduling algorithm is NP-hard, which can be proven as in [16], by reducing the schedule selection problem to the minimum feedback arc set problem. We design an approximate algorithm to do parent scheduling for each level.

Without loss of generality, we describe our TDMA scheduling as designed for each level. For nodes at each level, we divide them into three *time partitions*, children partition, primary parent partition and backup parent partition. Nodes in children partition are scheduled to send messages before nodes in primary parent partition. And nodes in primary parent partition send message before nodes in backup parent partition. The reason that we define children partition is that a node can be both parent and sibling to a certain node in the same level. For example, in Figure 3, C_2 selects C_1, P_1 and P_2 as its parents. So C_1 is a parent and sibling of C_2. Children node should send before their parents. The children partition is for those children nodes located in the same level as their parents. For the previous example, for the level 2 TDMA scheduling, C_2 needs to send message before C1, because C_2 is child of C_1. C_2 is located in children partition. The primary parent partition contains the nodes that are the primary parents of any of their children. The backup parent partition contains the nodes that are only the backup parents of all of their children. Note that at a cursory examination one may believe that a conflict exists in case two children C_1 and C_2 select common parents P_1 as primary and backup, respectively, while P_2 is chosen as backup and primary, respectively, as Figure 3 shows. We note that in our protocol a parent is classified as backup parent partition only if that parent is not selected as primary parent for any sensors. That means P_1 and P_2 both are classified as primary parent partition. When a parent receives a message from a child, each node knows whether it is a primary parent according to its children's bit vector. If a parent does not receive a message from a child, the parent will use the previous parent order assigned by its child. Within each partition, the nodes are ordered by their ids in ascending order. Nodes with smaller ids send messages before nodes with larger ids.

In this way, we achieve dynamic TDMA scheduling for each level.

D. Online Fault Detection and Fault Recovery

There are three types of faults in our scheme, and each fault has a corresponding fault recovery method. The first fault condition is that a primary parent does not receive a child's message, but the side link between the primary and the backup parent does not fail. The second fault condition is that a primary parent does not receive a child's message and the side link between the primary and the backup parent fails. The third fault is caused by the conflict of TDMA scheduling that a child's backup parent sends message before its primary parent. The last two kinds of faults cannot be handled in Ridesharing [7].

1) ***Primary link failure:*** The backup parents who overhear the message can handle missing messages in current interval. This function works the same way as Ridesharing [7]. If backup parent overhears primary parent, it will check bit vector of their common child, if the received bit of the common child is 0 in primary parent bitvector list. The backup parent will aggregate the common child message.

2) ***Side link failure:*** Bitvector scheme handles side link failure by delaying recovery to the next interval. In interval i, when a side link fails, the backup parent does not know whether the parent with higher priority received the child message or not. In this case, the backup parent simply saves the child message until the end of next interval $i+1$. During the current interval i, the child will listen to their parent messages. (We assume that all links in the network are symmetric, that is, if a child does not overhear a parent's message, neither does the parent.) When the backup parent receives a message from its child in the next interval $i+1$, it checks the child's bit vector and knows whether the child's primary parent received last interval i's message, according to the r-bit of primary parent bit element. If the r-bit is 0, it means the child did not overhear the primary parent, so the primary parent did not receive the child's message from interval i. In this case, the backup parent will aggregate the saved child message from last interval i in the current interval $i+1$.

For example, in Figure 3, suppose the side link between P_1 and P_2 and the link between C_1 and P_1 fail, in interval i, the child C_1 sends out a message m_{i1}. P_1 does not receive m_{i1} and sends out its own data message m_{i2}. But due to the side link between P1 and P_2 failure, P_2 cannot overhear m_{i2} of P_1, so that P_2 does not know whether P_1 aggregated m_{i1}. P_2 saves m_{i1} in its message queue. At the same time, child C_1 overhears each of its parent message and update its bit vector (P_1: 0110, P_2: 1111). In the next interval i+1, child C_1 sends out another message $m_{(i+1)1}$ with bit vector P_2 1111 and P_1 0110 (C_1 reorders its parents and selects P_2 as its primary parent). P_2 received message $m_{(i+1)1}$ with bit vector from C_1 again. Since our links are all symmetric, according to C_1's r-bit of its parents, P_2 knows that C_1 did not overhear P_1, and that P_1 also did not receive m_{i1} in the last interval i. Therefore, P_2 aggregates m_{i1} and $m_{(i+1)1}$ in current interval $i+1$. In this way, P_2 in interval $i+1$ handles the missing message m_{i1} from interval i.

Since some faults are detected one interval after the faults really happen, we assign the message transmission rate to be twice that of the data rate of sensors. The impact of this fault recovery method is that more messages will be injected in the network and it consumes more energy. We evaluate the effects on the energy and performance in Section V.

*3) **Overhearing messages after sending out own messages:*** Although unusual, there is a slight chance that some backup parents may send messages before the primary parent. And backup parents overhear the primary parents' messages after sending out their own data messages. Their common child's data cannot be aggregated if primary parent of those children fails. We allow nodes in a transmission level to continue listening to their siblings after sending out their own data messages. In this way, backup parents saves common child messages in current interval and aggregates it in the next interval, when the primary parent is detected to have failed to aggregate the common child's message. This fault recovery method gains more network reliability by consuming more power. We evaluate the effects on the reliabiilty and energy in Section V.

VI. EVALUATION

In our evaluation, we compare bitvector scheme with Ridesharing [7] and FOMA [8] schemes. We use the following metrics to evaluate the performance of the three schemes:

- *Average relative RMS*: the average normalized root mean square error to the correct result, which is metric of network accuracy. In other words, this computes how far off the value received at the base station is from the actual results that should be received by the base station. RMS in our paper means RMS error.
- *Average correct message ratio*: the ratio of correct messages received by base station and total number of messages, averaged over the number of intervals.
- *Average energy per node per interval*: average energy consumed per node (total energy divided by number of nodes in the network) averaged over the number of intervals.
- *Average message overhead per interval*: that is, the total amount of overhead per message (counted by bytes) transmitted in network averaged over the number of intervals. This metric represents the message overhead for different schemes transmitted in network.

A. *Simulation Setup*

In our simulation, we put a number of sensors in a grid distribution. Each sensor is only able to hear messages from its eight direct neighbors. Similar to Ridesharing [7], the power consumption is 65 mW for transmission, 21.0 mW for listening and reception, and 0 mW in sleep mode. The network bandwidth is assumed to be 38.4 Kbps. For comparison purposes, we implemented bitvector, Ridesharing, and FOMA and our new fault model based on fault duration. Each simulation runs for 16 sensing intervals. The results are averaged over 50 runs, to ensure statistical significance of the results.

B. *Experimental Results*

*1) **Accuracy comparison:*** We vary the link error rate and the fault duration, when the total number of nodes is 49 (7x7 grid). We use two metrics, namely relative RMS and correct message rate, to evaluate network accuracy. Figure 4 shows a 3D graph of the relative RMS when varying link error rate and fault duration. Figure 5 shows a 3D graph of the correct message rate when varying link error rate and fault duration. The bitvector is more robust (that is, better in both metrics) than Ridesharing and FOMA schemes when link error rate increases. When link error rate is 0.4, the bitvector has about 20% improvement in RMS comparing to Ridesharing and 46% improvement in RMS comparing to FOMA. This is because when link error rate is high, each child in bitvector reorders their parents more frequently to select the highest quality link to send out data first and therefore becomes more tolerant to faults.

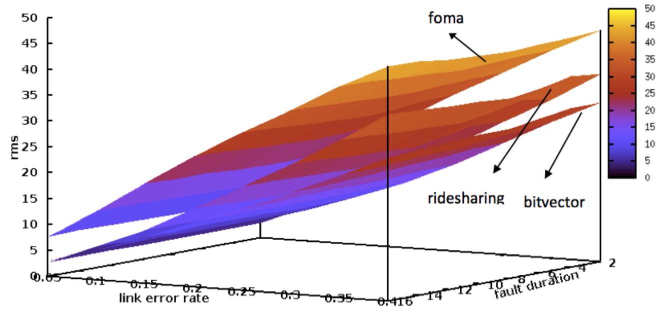

Figure 4 Relative RMS when varying link error rate and fault duration (the number of nodes is 49)

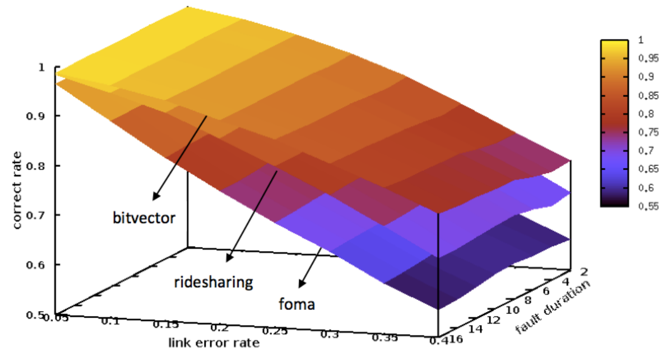

Figure 5 Correct message rate when varying link error rate and fault duration (the number of nodes is 49)

The reasoning of parent reorder was validated through an experiment that changes the parent reorder frequency (i.e., the number of times parent were reordered averaged over the number of intervals). Figure 6 shows the parent reorder frequency when varying link error rate and fault duration. As expected, we found some correlation between network accuracy and parent reorder frequency when link error rate is not very high (between 0 and 0.4). That is, the more frequent the parents being reordered, the more accuracy we can get. But when the link error rate is high (larger than 0.4), the parent reorder frequency decreases, because the network is so lossy that children cannot overhear their parents' messages very often, so that they cannot update their bitvectors and reorder their parents. We note that this case is rare, but it should still be studied.

Figure 7 shows that RMS of bitvector, FOMA and Ridesharing all decrease as fault duration increases. And bitvector decreases more in RMS error than FOMA and Ridesharing. It is because bitvector is able to estimate the link quality and reorder parents by link quality. When the fault duration gets bigger, parents with high quality is more likely being chosen, so that the RMS error decreases as fault duration increases. In addition, Figure 8 shows the parent reorder frequency versus fault duration when link error rate is 0.4. As expected, the parent reorder frequency decreases when fault duration increases. The reason is that when network is stable, the link error rate tends to be constant, so that the parents' order of each node is stable. Note that in Figure 6, there are more data points showing the same phenomenon for the other link error rate, but we can see the decrease is less steep for very low and very high link error rate. It is because for low link error rate, link quality is good and parent reordering happens infrequently for small fault duration. Parent reorder frequency does not increase obviously when fault duration increases. This is due to a similar phenomenon of that explaining of Figure 6, when the link error rate is larger than 0.4.

We also carried out a sensitivity analysis of the schemes to network size; for that, we varied the network size from 25 (5x5 grid) to 169 (13x13 grid) and changed link error rate to compare relative RMS. Figure 9 is a 3D graph showing the RMS while varying link error rate and network size. We found that all the three schemes perform better in smaller size network than larger size network. But bitvector scheme always has highest reliability. Note also from Figure 9 that the shape of the curves changes slightly with larger networks, making the RMS better/lower for very large or very small link error rate; this is due to a similar phenomenon of that explaining Figure 4.

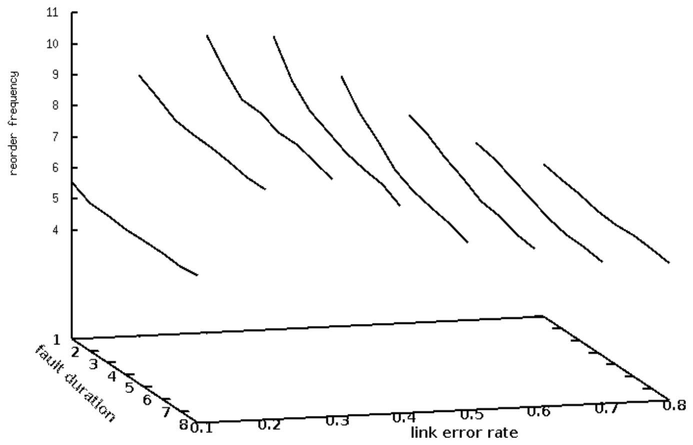

Figure 6 Parent reorder frequency when varying link error rate and fault duration (the number of nodes is 49)

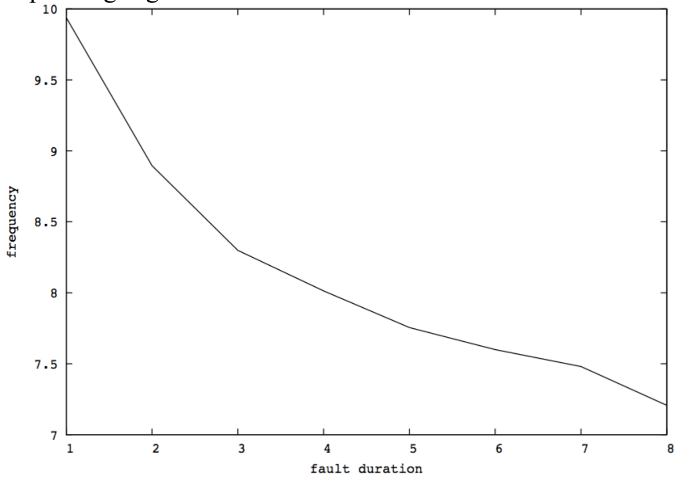

Figure 8 Parent reorder frequency vs fault duration for link error rate of 0.4

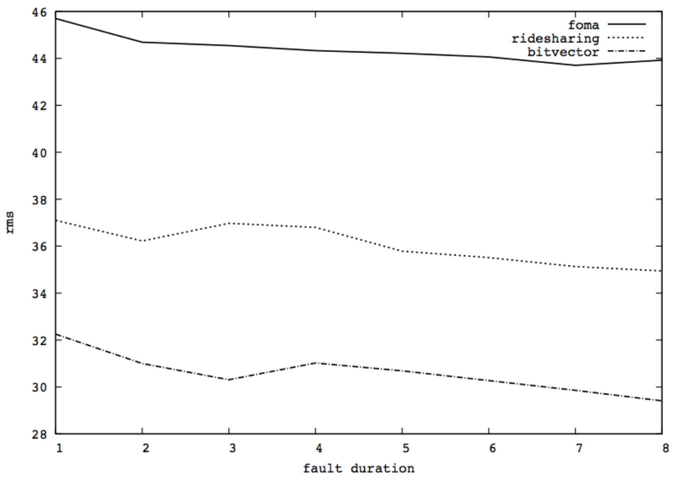

Figure 7 RMS vs fault duration (number of nodes is 49)

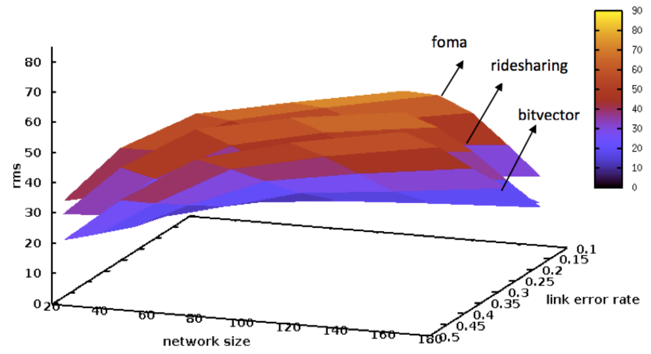

Figure 9 Network size while varying link error rate and RMS

*2) **Energy comparison:*** In Figure 10 we show the measured average energy consumption per node per interval for the three schemes tested. In the experiment, we vary link error rate when there are 49 sensors with fault duration = 4 (fixed for the duration of the experiment to isolate the effects of error rate). The result shows that bitvector consumes more energy than FOMA, because it allows nodes in two levels (i.e., parents and children) to listen to the nodes transmitting. FOMA also has extra costs of energy for sending out many control messages (more exactly, *3n* messages, where *n* is the number of sensors in the network). Ridesharing has the least energy consumptions because it only allows nodes to listen to their parents one level below, and each message has less overhead message than both bitvector and FOMA schemes (see below).

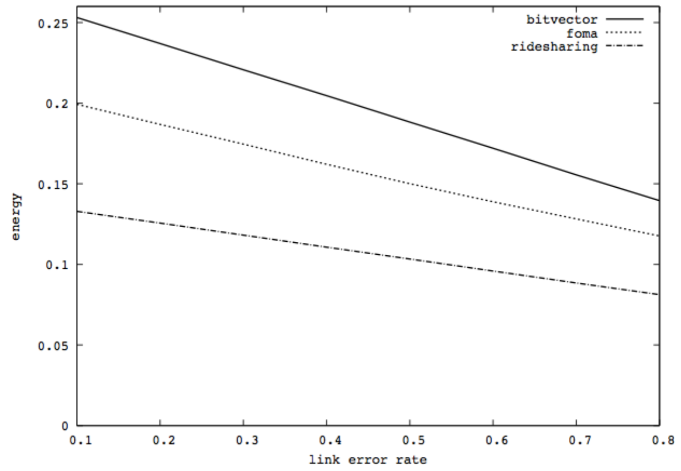

Figure 10 Energy vs link error rate with fault duration 1 (the number of nodes is 49)

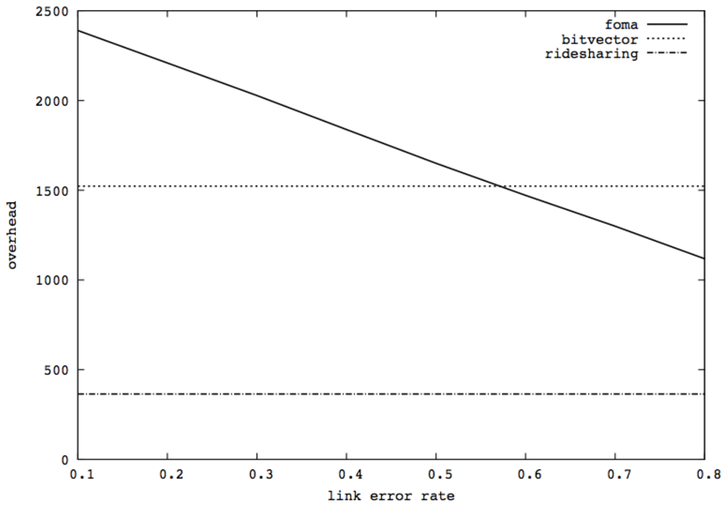

Figure 11 Overhead vs link error rate with fault duration 1 (the number of nodes is 49)

*3) **Message overhead comparison:*** In this experiment, we evaluate the message overhead in bytes send per interval. The result is shown in Figure 11. For Ridesharing, the message overhead for each node is 2 bits times the number of children. For bitvector scheme, each node needs to send out a bit vector with parent bit elements and children bit elements (4 bits per bit element). Therefore, bitvector has more overhead messages than Ridesharing. But the amount of overhead does not vary with link error rate for Ridesharing or for bitvector. In contrast, for FOMA, the overhead is as follows: the DAN (data aggregation node) selection phase needs to send out DAN messages from the root to all leaf nodes; in the signature construction phase, it needs to send out a signature message from each leaf node to the root; in the data collection phase, each node needs to send out data message including one aggregation signature and one or more partial results. For FOMA scheme, the amount of control messages decrease as the link quality gets worse, since fewer partial results need to be sent.

*4) **Network density**:* To evaluate how network density[1] affects the network reliability, we carried out an experiment that changes network density with fixed link error rate of 0.3, when there are 169 sensors in the network. The performance of all the schemes increases with the increase in network density as shown in Figure 12. For Ridesharing, that is because the probability that a node finds three parents decreases. For bitvector, there are two main reasons: (a) first, finding three parents is hard (see Ridesharing above), and (b) second, the probability to reorder the parents decrease due to lossy network. For FOMA, the success probability that a data message is sent to DAN node decreases.

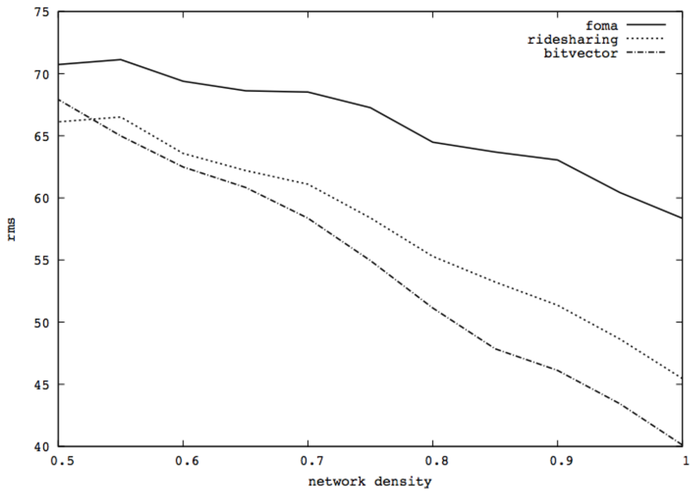

Figure 12 RMS for varying network density (The number of nodes is 169 and link error rate is 0.3)

VII. Conclusion

In this paper, we introduced a new realistic time-dependent fault model called fault duration that is ideal to model interference in nuclear power plant wireless monitoring environments, where interference in the network lasts for a fixed interval of time. We also devised new in-network aggregation scheme, bitvector, for monitoring nuclear power plants. Our bitvector scheme embeds a dynamic link quality estimation mechanism that selects the best parent-child links for child nodes to propagate their results/data. Our scheme shows a much lower root mean square error (RMS) comparing

[1] We changed the network density by creating holes in the network, that is, removing sensors (or making them "dead") from randomly selected locations in the grid, with a uniform distribution.

to state of the art (FOMA and Ridesharing), while it has lower message overhead, and it allows for fault detection and recovery, link quality estimation and parent reorder. In addition, the network accuracy improves as fault duration gets bigger. Bitvector also performs better for high network density.

ACKNOWLEDGMENT

We would like to thank all the anonymous reviewers for their suggestions. The work in this paper was supported by Department of Energy.

REFERENCES

[1] Gungor V C, Hancke G P. Industrial wireless sensor networks: Challenges, design principles, and technical approaches[J]. Industrial Electronics, IEEE Transactions on, 2009, 56(10): 4258-4265.

[2] Hashemian H M, Kiger C J, Morton G W, et al. Wireless sensor applications in nuclear power plants[J]. Nuclear Technology, 2011, 173(1): 8-16.

[3] Hashemian H M, Strasser W. What you need to know about sensor reliability before and after and accident[C]//IAEA TWG NPPIC Meeting, Austria. 2011.

[4] http://www.power-eng.com/articles/print/volume-115/issue-8/features/power-plant-condition-monitoring.html

[5] Lin R, Wang Z, Sun Y. Wireless sensor networks solutions for real time monitoring of nuclear power plant[C]//Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on. IEEE, 2004, 4: 3663-3667.

[6] Jin X, Guo Y, Sarkar S, et al. Anomaly detection in nuclear power plants via symbolic dynamic filtering[J]. Nuclear Science, IEEE Transactions on, 2011, 58(1): 277-288.

[7] Gobriel S, Khattab S, Mossé D, et al. Ridesharing: Fault tolerant aggregation in sensor networks using corrective actions[C]//Sensor and Ad Hoc Communications and Networks, 2006. SECON'06. 2006 3rd Annual IEEE Communications Society on. IEEE, 2006, 2: 595-604.

[8] Ashouri M, Yousefi H, Hemmatyar A M A, et al. FOMA: Flexible overlay multi-path data aggregation in wireless sensor networks[C]//Computers and Communications (ISCC), 2012 IEEE Symposium on. IEEE, 2012: 000508-000511.

[9] Boyd S, Ghosh A, Prabhakar B, et al. Randomized gossip algorithms[J]. Information Theory, IEEE Transactions on, 2006, 52(6): 2508-2530.

[10] Aysal T C, Yildiz M E, Sarwate A D, et al. Broadcast gossip algorithms for consensus[J]. Signal Processing, IEEE Transactions on, 2009, 57(7): 2748-2761.

[11] Hakoura B, Rabbat M G. Data aggregation in wireless sensor networks: A comparison of collection tree protocols and gossip algorithms[C]//Electrical & Computer Engineering (CCECE), 2012 25th IEEE Canadian Conference on. IEEE, 2012: 1-4.

[12] Madden S, Franklin M J, Hellerstein J M, et al. TAG: A tiny aggregation service for ad-hoc sensor networks[J]. ACM SIGOPS Operating Systems Review, 2002, 36(SI): 131-146.

[13] Manjhi A, Nath S, Gibbons P B. Tributaries and deltas: Efficient and robust aggregation in sensor network streams[C]//Proceedings of the 2005 ACM SIGMOD international conference on Management of data. ACM, 2005: 287-298.

[14] Chen Z, Shin K G. OPAG: Opportunistic data aggregation in wireless sensor networks[C]//Real-Time Systems Symposium, 2008. IEEE, 2008: 345-354.

[15] Berfield A, Chrysanthis P K, Mossé D. LSynD: Localized Synopsis Diffusion[C]//Object and Component-Oriented Real-Time Distributed Computing, 2007. ISORC'07. 10th IEEE International Symposium on. IEEE, 2007: 313-320.

[16] Gobriel S, Khattab S, Mossé D, et al. Considering link qualities in fault-tolerant aggregation in wireless sensor networks[C]//Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE. IEEE, 2009: 1-6.

[17] Baccour N, Koubaa A, Mottola L, et al. Radio link quality estimation in wireless sensor networks: a survey[J]. ACM Transactions on Sensor Networks (TOSN), 2012, 8(4): 34.

[18] Zhao J, Govindan R, Estrin D. Computing aggregates for monitoring wireless sensor networks[C]//Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on. IEEE, 2003: 139-148.

[19] Mao Y, Kschischang F R, Li B, et al. A factor graph approach to link loss monitoring in wireless sensor networks[J]. Selected Areas in Communications, IEEE Journal on, 2005, 23(4): 820-829.

[20] Vuran M C, Akan Ö B, Akyildiz I F. Spatio-temporal correlation: theory and applications for wireless sensor networks[J]. Computer Networks, 2004, 45(3): 245-259.

[21] Gupta G, Younis M. Fault-tolerant clustering of wireless sensor networks[C]//Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE. IEEE, 2003, 3: 1579-1584.

[22] Silva I, Guedes L A, Portugal P, et al. Reliability and availability evaluation of wireless sensor networks for industrial applications[J]. Sensors, 2012, 12(1): 806-838.

[23] Chen J, Kher S, Somani A. Distributed fault detection of wireless sensor networks[C]//Proceedings of the 2006 workshop on Dependability issues in wireless ad hoc networks and sensor networks. ACM, 2006: 65-7