FTDE: Distributed Fault Tolerance for WSN Data Collection and Compression Schemes

Azad Ali, Abdelmajid Khelil, Neeraj Suri
TU Darmstadt, Germany
{azad,khelil,suri}@deeds.informatik.tu-darmstadt.de

***Abstract*—Wireless Sensor Networks (WSNs), often employ data compression schemes to efficiently collect data at sink for processing due the constrained resources. However, various factors such as can compromise the core objective of accurate collection and delivery of the sensor data. However, most current compression schemes fail to consider the faults and the resulting data errors arising from low node reliability (node crashes), harsh operating environments and communication faults resulting into erroneous data being collected. This paper proposes a novel and efficient distributed model based fault-tolerance technique applicable to varied compression schemes. Being fully distributed, our scheme corrects data errors at the point of origin (faulty sensor nodes), and thus avoids costly transmissions of corrupt data, decreasing message cost, and enhances compression effectiveness by correcting the erroneous samples.**

I. Introduction and Contributions

Wireless Sensor Networks (WSNs), typically being built of low-cost, unreliable components and often deployed in harsh environments, encounter a wide variety of operational and environmental faults. The faults can be hardware, memory, sensor or transceiver defects that result in errors such as node crashes, message corruption, sampling errors, or communication failures such as message losses or network partitioning [1][2]. As the core objective of a WSN is to collect data of usable fidelity, hence effective fault handling needs address the issue of erroneous/corrupt data. Existing works have documented that a substantial portion of aggregated data (up to 49%) collected in actual WSN deployments can be faulty, with up to 3-60% incorrect data for a single sensor [3]. Such high data corruption rates undermine the usefulness of the data, thus motivating our approach to address data corruption.

A major challenge for finite energy battery-powered WSN deployments (and for recovery solutions) is the high cost of communication operations. Accordingly, our work proposes a communication-efficient distributed approach to (a) handle data errors as close to the data source as possible, and to (b) use representative data models instead of multiple instantaneous sensor readings, for error detection and correction. This not only helps us reduce the message costs but also allows to detect and repair data errors more efficiently. We highlight the direct adaptability of our approach to various existing data collection schemes. We also demonstrate how existing data compression schemes [4] can easily be extended to become fault tolerant using our proposed approach.

Our proposed approach implicitly addresses a large number of potential faults by constructing an abstract/composite data error class that takes advantage of observing (and making a model of) data patterns of recently sensed data, either locally or in the immediate proximity. Our proposed model based Fault-Tolerance for Data Errors (FTDE) scheme allows us to detect errors with minimal additional communication and computation overhead. Surprisingly, the effective message cost is often negative because (i) the message cost incurred by the proposed scheme is almost negligible, and (ii) our scheme detects the erroneous samples and discards them, which otherwise would have been classified as outliers and need to be transported to the sink, incurring considerable message costs. This reduction in message cost is validated by simulation results.

II. Related Work

Fault handling in WSN, i.e., detection, prevention, isolation, identification and recovery, has been extensively targeted given that WSNs are built using low-cost failure-prone components. WSN relevant faults[1] are typically classified into two broad classes: Functional faults and data errors where functional faults eventually lead to data errors. We present a brief synopsis to highlight the needed research gaps and refer the reader to [1][2] for comprehensive details.

a) Functional Faults in WSN: Functional faults span hardware faults (sensor, memory), software faults [5] (bit flips, buffer overflows), which may lead to compromised functionality of node failures (transient or permanent), communication failures, packet loss or link breakage. A common feature of the techniques addressing functional faults is that they are often driven by the network topology and less by sensed data content and semantics. Accordingly, they focus on topology anomalies such as corrupt packets [6], node crashes, faulty links. Most of approaches either fail to be effective across varied functionalities or across the varied classes of failures. Hence, our focus is on data errors that we discuss next.

b) Handling Data Errors in WSN: Data errors result from internal (functional faults) and external influences, e.g., environmental interference and noise. There are four primary classes of sensor data errors: Sporadic irregularities/spikes, noisy values, constant/stuck values and constant drifts from the phenomena value. Most of data error detection schemes rely on a basic assumption that the sensor readings from the same region should have similar values [7]. FIND [8] ranks event suspecting nodes based on their sensor readings as well as their physical distances from the event. Consensus-based approaches [9] abstract the fault pattern by achieving a data-based agreement among the nodes that have detected a sudden

Research Supported in part by TUD GRK 1362, CASED & EC-SPRIDE.

[1] A fault represents an anomalous condition; error is the observable deviance resulting from the activation of the fault, and failure represents loss of service.

reading change in order to detect an event on consensus basis. Some work propose to solve constant drifts through online calibration using a correction function [10][11][12]. Outlier detection [13] is an approach related to data error handling.

c) The Novelty of our Proposed Data Errors Approach: We focus on detection of data errors, implicitly accounting for the root causes as the correctness of the delivered/processed data is imperative than ascertaining the root cause. We use selective sampling from constructed data models, which help to filter the fluctuations and the model output can be reliably used to detect individual data errors. Our approach considers both event-based and continuous data collection designs to cover a wide spectrum of WSN applications.

III. Fault Abstraction and Data Errors

We refer to the environmental data that needs to be collected (e.g., temperature, pressure) as *reference* data. The data actually collected by the sensor nodes is the *sampled* data. The sampled data may not necessarily be same as the reference data because of the faults causing data errors. Hence, the sampled data is often not good enough to be used as the reference data. Moreover, the instantaneous (sampled) values are not suitable for fault detection because of the erroneous samples. Hence, we need a data approximating mechanism to approximate the reference data. Accordingly, we propose a scheme where we first construct a model out of the sampled data and use the model output to represent the reference data. The model output in conjunction with the sampled data is then used to detect and correct the erroneous samples. The model does not depend on a single instantaneous value, hence it is relatively less prone to the the faults and the resulting data errors. Hence, the model provides a good approximation of the reference data.

1) The Fault Abstraction: Various functional faults result into data errors. Examples include sensor faults generating wrong readings, bit flips in memory affecting the sensor data, non-recommended deployment conditions resulting in wrong calibrations. However, we do not detect the faults explicitly, rather we focus on the detection and correction of erroneous data produced by the underlying faults. Consequently, we do not require any specific knowledge of a given fault but still implicitly correct it by correcting sampling errors. Hence, in the rest of manuscript, we will be dealing with data errors instead of faults directly. We define an erroneous sample as:

Definition 1: An erroneous sample $v_e(t)$ deviates at least ξ units from the reference value $v(t)$, i.e., $|v_e(t) - v(t)| > \xi$.

In the absence of reference values, erroneous samples deviating less than ξ from the reference values are not discernible from the non-erroneous sampled data (as detailed in Sec. IV-B).

2) Data Error Types: Data error types in sampled data from WSNs has been extensively studied in [14]. Please note the term "data errors" in current work and "data faults" in [14] are equivalent. Authors in [14] classify the data errors as short duration high variation sample values (NOISE), sharp suddenly momentary changes between normal sampled values (SHORT), long duration beyond expected sample values and uncorrelated to the underlying physical phenomenon (CONSTANT or "Stuck-at"). SHORT are the transient errors, CONSTANT are the permanent errors and NOISE can be either a transient or permanent depending on their duration.

IV. Model-based Data-Fault-Tolerance

We now detail the proposed scheme.

A. Data Error Detection and Approximation Models

In the absence of reference data, sensor nodes cannot differentiate a correctly sampled value from a noisy sample. Hence, we use data modeling to approximate the sampled data, which in turn is used to generate reference data. The AR (Auto-regressive) model acts as low-pass filter and removes noisy samples. The filtered output from the model is used in conjunction with sampled data to detect data errors. The AR model serves dual purpose of approximating the sampled values for data compression and filtering by removing the noisy samples. For details on model construction please refer [15].

B. Sporadic Data Errors

Sporadic data errors refer to SHORT data errors and also NOISE when it lasts for less than one fourth the training queue length (T). For sporadic data errors we first propose a mechanism to detect the anomalies in the sampled data and then describe a mechanism to correct these anomalies. The detection process is divided into two stages, (a) local node level detection, and (b) cluster level detection. Most of the time series schemes such as PAQ [16] and ASTC [15] already have a clustering scheme implemented. Hence, for FTDE we do not have to additionally implement any clustering scheme and can directly utilize available clustering.

1) Node Level Detection: Each node initially detects sporadic data errors. However, as shown later, individual nodes can misclassify some legitimate samples as data errors. Hence, a second cluster-level stage is needed to scrutinize the classification in the first stage (node level) and make final classification.

Adaptive Modeling (AM) through Outlier Detection: Sensor nodes first construct a model out of the sampled data. Using the constructed model, the estimated reference data is calculated. The sampled values not appropriately approximated by the model within the user defined accuracy bounds (ϵ) are termed as *outliers*. Accordingly, if the difference between the sampled value and the model estimated value exceeds ϵ, it is generally classified as outlier. Non-erroneous samples do not significantly vary around mean, hence the outliers are expected to be within a specified valid range. Hence, as described in Def. 1, we have a threshold ξ to detect data errors, and if the approximation error is beyond ξ, it is classified as a data error instead of outlier. Nodes can determine an estimated value to be an outlier or error ($e(t) = \hat{v}(t) - v(t)$) using Eq. (1):

$$\hat{v}(t) = \begin{cases} \hat{v}(t), |e(t)| < \epsilon; \\ v(t), |e(t)| \geq \epsilon \ and \ |e(t)| < \xi; \\ \hat{v}(t), |e(t)| \geq \xi \end{cases} \tag{1}$$

Alg. 1 describes the process on a sensor node for the classification of sampled values into outliers and data errors. The set $\mathbb{V}$ refers to the approximated reference values $\hat{v}(t)$ as estimated by the model. Alg. 1 tracks data errors and outliers by constructing two sets, namely $\mathbb{O}(t)$, set of samples classified as outliers and $\mathbb{E}(t)$, set of sampled classified as data errors. Based on the outlier and data error threshold (ϵ and ξ), the estimated value is classified either as an outlier (Alg. 1, Line 5), or as a data error (Alg. 1, Line 7), or it is correctly estimated by the compression scheme. The lists of outliers (the set $\mathbb{O}$) and preliminary errors set (data errors, set $\mathbb{E}$) are then sent to the cluster head for further processing. The data errors are further processed to identify if any sample values originating from an event were incorrectly classified as data errors.

Algorithm 1 Error and Outliers Detection Algorithm

```
1: for $\hat{v}(t)$in$\mathbb{V}(t)$ do
2:     $e := |\hat{v}(t) - v(t)|$
3:     if $e > \epsilon$ and $e < \xi$ then
4:         $\hat{v}(t) \leftarrow v(t)$;
5:         Append $v(t)$ to $\mathbb{O}(t)$;
6:     else if $e \geq \xi$ then
7:         Append $v(t)$ to $\mathbb{E}(t)$;
8:     end if
9: end for
```

2) Cluster Level Detection: A physical event, i.e., sudden change in physical attribute such as temperature, pressure, may also exhibit behavior similar to sporadic errors and consequently be misclassified as a data error during stage 1. Hence, we consider the node level sporadic data error detection only as an initial/temporary classification, because a single node does not have enough information to determine if the classification of the sample as a data error is due to corruption of the sample by an underlying functional fault or it is a result of a physical event. Hence, we have devised a second stage cluster-level sporadic data error detection mechanism that can reclassify the sample values classified as data errors in the first stage to either data errors or events. FTDE requires simple 1-hop clusters that are typically available in different compression schemes [15] [16]. The cluster head collects the data errors from all the member nodes in the cluster, hence it can use this information to verify if the initial classification by a node was incorrect and then reclassify the incorrectly classified samples as an outliers

In order to differentiate a data error from a physical event, we develop a formal notation for data errors and then develop a logical basis for differentiating the data errors from physical events. We now define the terms related to the probabilistic random error events and physical events. The classification of a sample value as an error (hence, detection of a sporadic fault or an event) is a random event.

Definition 2: The random event the model approximated value is classified, using Alg. 1, as data error due to corruption by an underlying functional fault is defined as *data error event*.

Definition 3: The random the model approximated value is classified as an event arising due to a change (sudden, extreme) change in the phenomenon is defined as a *phenomenon event*.

Next, we discuss both events and analyze how they can be differentiated using the error set ($\mathbb{E}(t)$), obtained using Alg. 1, from multiple sensor nodes in the cluster.

Distinguishing Data Error Events from Phenomenon Events: We now develop the basis to differentiate the two events and then use it to classify them on the cluster head.

Axiom 1: Data error events taking place on two separate sensor nodes are independent.

Based on the discussion in Sec. IV-B, the underlying cause of the data errors are predominantly random hardware faults resulting in random data errors. These factors affect each sensor node independently. Moreover, faults of one node cannot cause errors on another node. Hence, a faulty sensor on Node 1 will not corrupt the samples of Node 2. Accordingly, if sporadic data errors on Nodes 1 and 2 are given by $E_1(t)$ and $E_2(t)$, then *data error events* occurring on two different nodes are independent from each other, i.e., $P(E_1(t) \wedge E_2(t)) = P(E_1(t))P(E_2(t))$.

As data error events are random events (Def. 2):

Theorem 1: Data error events occurring on two different sensor nodes are independent random variables.

Proof: A data error event detected on a sensor node is a random stochastic event. Accordingly, we assume R_1 and R_2 represent random variables for data error events on sensor node S_1 and S_2. For R_1 and R_2 to be independent, each data error event (random event) of R_1 must be independent of data error events of R_2 and vice versa. From Axiom 1, each data error event $E_1(t)$ and $E_2(t)$ belonging to random variables R_1 and R_2 is independent because of physically independent hardware faults that cannot influence the other sensor node, hence R_1 and R_2 are also independent random variables.

Corollary 1: The random variables for data error events on a set of n sensor nodes are independent.

Using Theorem 1, each pair of data error event random variables R_i and R_j belonging to sensor pair S_i and S_j, $i \neq j$ is independent. Hence, all data error event random variables for n sensor nodes are independent of each other.

Axiom 2: An environmental event detected separately by two sensor nodes at time t is not independent.

Unlike data error events, where each event is generated by a separate stochastic process (hardware fault), the environmental events are essentially replication of the same random variable, generated by a single stochastic process (e.g., suddenly change in temperature/pressure) that is sampled by multiple sensor nodes. Hence, an environmental event happening on two sensor nodes at time t is not independent.

Axiom 2 can be extended, similar to Axiom 1, to show that the environmental random variables (random variables related to environmental events) are not independent.

Corollary 1 and Axiom 2 set the foundation for differentiating data error events from environment events. Accordingly, the cluster head can determine if event random variables (initially reported by all the sensors as data error events, $\mathbb{E}(t)$, classified using Alg. 1) of two sensor nodes are independent then these are due to data error events, otherwise they are cause by environment events. However, cluster head needs to know the joint probabilities and the probabilities of the events to determine independence of the events, as given in Axiom 1. Moreover, sensor nodes have very limited computational, memory and energy resources. Given that we have shown that (a) the error events are independent, while (b) phenomenon events are not independent, and (c) the evaluation on the sensor nodes is not computationally feasible, we have devised a simpler measure to infer the independence of the events to detect the data error events.

Detection of Data Error and Phenomenon Events: The cluster head possesses a broader localized view on the events using the preliminary error sets collected from all the member nodes. The cluster head uses the results from Corollary 1 to detect data error events. However, as discussed earlier, using Corollary 1 is not practically feasible, as the cluster heads do not have the joint probabilities. Hence, we propose a simple two stage mechanism to detect data error events. The cluster head (S_H) constructs a cumulative preliminary error set ($\mathbb{E}_{S_H}(t)$, Alg. 2, Line 1) by merging preliminary error sets of its member nodes ($\mathbb{E}_{S_i}(t)$, where $\mathbb{E}_{S_i}(t)$ is the preliminary error set from sensor node S_i belonging to the cluster C_j). The cluster head detects the error events in two stages using the cumulative preliminary errors set.

In the first stage, the cluster head determines if more than one sensor node detected a data error event for a given time index (Alg. 2, Line 3). As the data error events are independent (Axiom 1), it is highly unlikely that more than one node will detect the data error events at the same time index. Even if they do detect data error events at the same time, they will still be independent. Whereas, for a phenomenon event we can easily observe more than one nodes reporting a data error event (due to mis-classifiction during in the preliminary error set), which actually will be indicative of a phenomenon event, rather than a data error. However, we cannot fully rule out the possibility of data error events taking place at more than one nodes at a given time index as a coincidence, depending on the cluster size and node density. Hence, if the cluster head detects a data error on more than one sensor node, it needs to process further to verify and finally classify the concerned sample either as a data error event or a phenomenon event.

In the second stage, the cluster head processes the values detected as data error events in the first stage. It computes the standard deviation of these error values for all cluster member nodes that reported these data error events (Alg. 2, Line 4). If the error values were generated by a phenomenon event, the sampled values collected by various nodes will be highly correlated (as they are not independent) and the standard deviation is expected to be low. Whereas, for error events, even if they happen to occur at the same time index, it is highly unlikely that the values sampled due to the data errors are correlated. Hence, we define a tolerance threshold for the standard deviation around the mean for the errors (ρ) to qualify as phenomenon event, i.e., if the standard deviation of the error values are beyond the defined threshold then they are classified as data error events (Alg. 2, Line 5), otherwise, they are classified as the phenomenon event (Alg. 2, Line 6-8). In Alg. 2, $\mathbb{I}(t)$, $\mathbb{P}(t)$ refer to finally classified data error events and phenomenon events sets respectively. The sample values related to final phenomenon event set $\mathbb{P}(t)$ are passed to the sink as outliers, so that the sink may appropriately approximate the phenomenon anomalies, whereas, the data errors classified as data error events are suppressed and not sent to the sink. Instead model is used for sample estimation .

V. Performance Evaluation

We now discuss the simulation results for proosed scheme.

Algorithm 2 Classification of preliminarily faults into fault and phenomenon events

```
 1: Construct $\mathbb{E}_{S_H}$ from $\mathbb{E}_{S_i}$ for $S_i \in C_j$
 2: for $t_i \in T$ do
 3:     if $COUNT_{S_j \in C_k}(E_{S_H}(S_j, t_i)) > 1$ then
 4:         if $SD_{S_j \in C_k}(E_{S_H}(S_j, t_i)) > \rho$ then
 5:             $\mathbb{I}(t) \leftarrow \mathbb{I}(t) \cup \left(\bigcup_{S_j \in C_k} E_{S_H}(S_j, t_i)\right)$;
 6:         else
 7:             $\mathbb{P}(t) \leftarrow \mathbb{P}(t) \cup \left(\bigcup_{S_j \in C_k} E_{S_H}(S_j, t_i)\right)$;
 8:         end if
 9:     else
10:         $\mathbb{I}(t) \leftarrow \mathbb{I}(t) \cup \left(\bigcup_{S_j \in C_k} E_{S_H}(S_j, t_i)\right)$;
11:     end if
12: end for
```

A. Simulation Settings

In order to conduct comprehensive evaluation simulations, we used the publicly available real-world data set [17] as reference data. We use TOSSIM to perform network simulations and MATLAB to reconstruct the signal at the sink. We implemented FTDE for PAQ and ASTC to evaluate the FTDE's adaptability to different schemes. We conduct simulations on humidity data as it varies steadily over time to further evaluate the adaptability of the proposed scheme to changing dynamics.

We selected third order AR models with user defined approximation threshold for compression scheme $\epsilon = 0.1$, inter-node error threshold $\rho = 0.5$, data error threshold $\xi = 4$ and a training length of 75. In order to intensively evaluate FTDE, we performed controlled corruption of the original data set at the rate of 5%, 10%, 20%, 30%, 40%, 50%, 60%, and 70% by adding noise. As discussed earlier, we know from the literature that the aggregated data in WSN can be corrupt up to 49% and up to 60% for an individual node [3]. We increased the corruption rate even further to 70%, as it allows us to evaluate the behavior of the proposed scheme when the amount of corrupt data is more than the real data. FTDE starts to degrade for error rates beyond 50%, as the models constructed out of majority of corrupted samples actually match closely to the corrupted samples than the reference data. The added data errors are the combination of SHORT, NOISE and CONSTANT at varying degrees, as various real world data from sensor nodes contain these data errors at varying degrees [3]. ξ is selected to be twice the standard deviation, which causes some of outliers to be misclassified as errors in stage 1 (Sec. IV-B1). However, it reduces the chances of the erroneous samples to be classified as outliers. Due to our multistage error classification mechanism the misclassified outliers samples in stage 1 (Sec. IV-B1) are later correctly classified back to outliers in stage 2 (Sec. IV-B2). Interestingly, we do not have to explicitly adapt to the rate of data corruption to detect the faults. If the faults rate is low and there are predominantly SHORT or short duration NOISE data errors, they are corrected by the transient error detection mechanism. However, if the transient error detection fails, the permanent error detection mechanisms runs in parallel to detect the CONSTANT and long duration NOISE data errors.

1) Methodology and Performance Metrics: Our performance evaluation is based on two key metrics: *Accuracy* of collected data and *message efficiency* as an implied measure of the energy consumption. The humidity data set [17] is considered as the reference. The corrupted data set is used in the simulations. Please note that in reality we will not have the reference data, rather only the sampled data. The corrupted data used for the simulations is equivalent to the sampled data. We collect the data models and outliers at the sink through PAQ and ASTC, both enhanced by FTDE to detect and correct the sample errors in the network. The collected data (approximations using the received models and outliers) is then compared with the reference data to evaluate the effectiveness of FTDE. We define accuracy as the mean square error (MSE) between the data collected assuming corrupted data and the reference data. The plots depict the aggregated MSE of all the sensor nodes over the complete duration of approximated data as accuracy measure. We measure the message efficiency as the message overhead created by the selected enhanced compression scheme, i.e., either PAQ with FTDE or ASTC with FTDE. The message cost accounts for all the stages of both compression scheme and FTDE, i.e., modeling, outliers, errors and events detection, reporting to the cluster head and finally transporting the models and the outliers to the sink. In the first simulation study, we compare the performances of PAQ, FTDE-PAQ (PAQ enhanced with FTDE) and FTDE-PAQ-CH_{EF} (PAQ enhanced with FTDE and where the data collected by the cluster head (CH) data is Error Free (EF)). In a second study, we compare the performance of ASTC to FTDE-ASTC (ASTC enhanced with FTDE).

B. Results

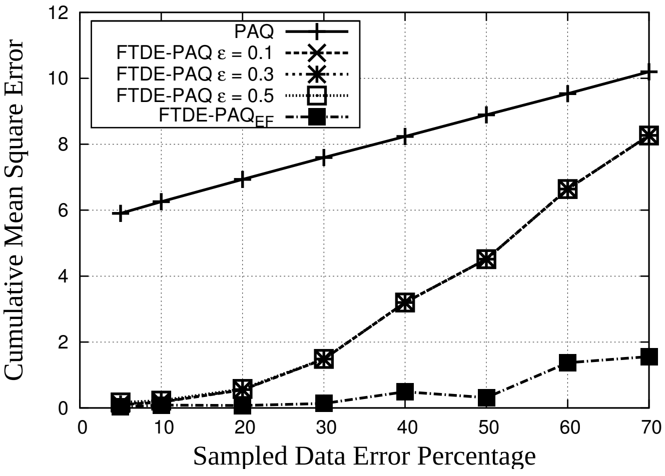

Fig. 1. Mean Approximation Error Relative to Reference Data

Fig. 1 depicts the approximation error relative to the reference data for PAQ, FTDE-PAQ and FTDE-PAQ-CH_{EF}. We can observe that PAQ cannot detect erroneous samples and blindly tries to fit to the corrupted data. Hence, relative to the reference data the resulting error is considerably larger than the error threshold $\epsilon = 0.1$. Whereas, FTDE can easily detect the erroneous samples and stays close to the user defined error threshold up to 10% of data corruption. The impact of error threshold is negligible on FTDE in terms of achieved cumulative MSE, hence the results for various error thresholds overlap indistinguishably on the graph However, performance of FTDE starts to degrade with increasing data corruptions. As the sampled data (here the corrupted data) is used for constructing the models, hence the models also get corrupted. Still, FTDE performance is consistently better than standard PAQ scheme. FTDE-PAQ-CH_{EF} depicts a hypothetical scenario, where cluster head data is error free resulting into error free model. Accordingly, FTDE can easily detect the erroneous samples and the resulting error is considerably reduced. It hints that the performance of FTDE can be significantly improved by constructing the model on nodes that have least corrupted data. This can be achieved either by exploiting an adaptive compression scheme like ASTC or a few high quality sensor nodes.

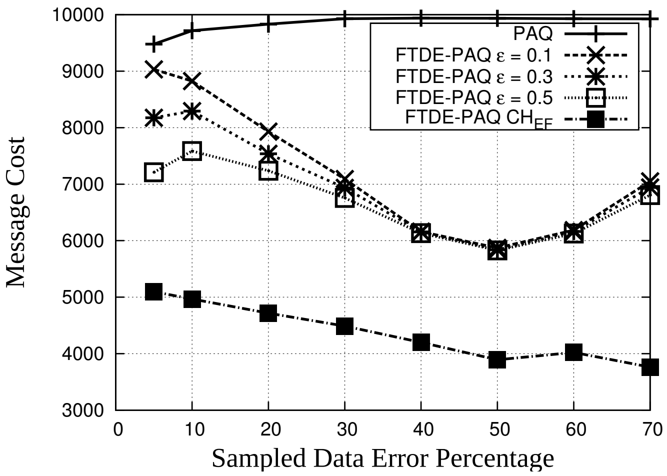

Fig. 2. FTDE-PAQ Message Cost

Fig. 2 depicts the message cost incurred by PAQ, FTDE-PAQ and FTDE-PAQ-CH_{EF}. As the errors and outliers are indifferent to PAQ, hence it classifies all as outliers and accordingly transport them to the sink to maintain the accuracy. The large number of outliers entail a high message cost. In contrast, FTDE-PAQ can detect the erroneous samples and discard them using the proposed two-stage model based scheme. Due to true distributed nature of the scheme, we are able to detect and discard the erroneous sample at the origin. Eliminating erroneous samples reduces the number of outliers to be reported, considerably reducing the message cost. We also observe that more the data is corrupted, more the samples are classified as erroneous and discarded and not transported to the sink, instead they are approximated through compression scheme models. This results in steady decline in message cost. Further simulations at $\epsilon = 0.3$ and $\epsilon = 0.5$ show that we do actually observe a slight increase in message cost initially when the data corruption is increased from 5% to 10% and then dropping until 50%. It is noteworthy that $\epsilon = 0.1$ represents a very high level of accuracy requirement for the given data set, as it requires the model approximation to be within 0.2% of the original data. Hence, (a) there are many outliers produced due to high accuracy requirement, and (b) the model is not accurate, as it was constructed from the corrupted data. However, as we increase the data corruption rate, it reduces outliers and instead introduces more erroneous samples (as more outlier samples are also replaced by the erroneous samples). Once, data corruption rate crosses 50%, the corrupted samples become more abundant than the actual data. The constructed model built from these corrupted sampled and tries to approximate the corrupted samples instead, which results in mis-classification of errors as outliers increasing message cost. Fig. 2 also depicts an interesting result. As the cost includes all the message exchanges, including the message overhead of FTDE, hence we can observe that the the effective message cost of FTDE is negative in comparison to the standard compression scheme. This is because the overhead introduced by the FTDE is negligible in comparison to the message overhead it reduces by detecting the erroneous samples and avoid to report them to the sink as outliers. Hence, FTDE not only increases the approximation accuracy but additionally reduces the message cost.

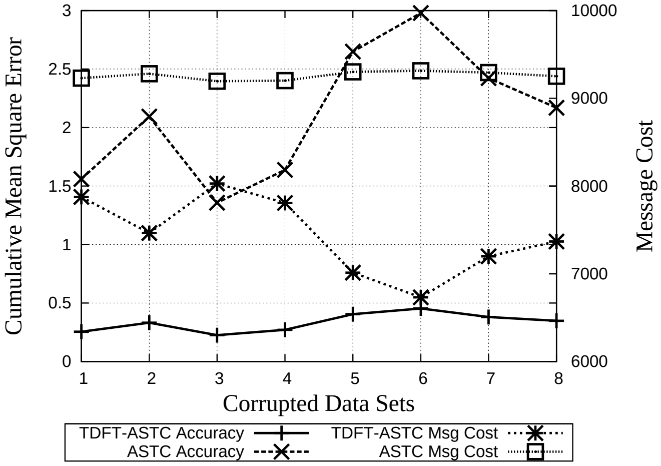

Fig. 3. FTDE-ASTC Accuracy and Message Cost

In order to test capability of FTDE to take advantage of ASTC's adaptive mechanism, the data corruption was carried out at different rates for different nodes, ranging from 10% to 50%, which corresponds to the natural distribution of data errors better in contrast to uniform error rate. In Fig. 3, we illustrate the performance results for FTDE-ASTC and ASTC w.r.t. accuracy and message efficiency. ASTC fairs better than PAQ and maintains low MSE as it can adaptively select the best model. However, it can not distinguish an outlier from erroneous sample, hence FTDE-ASTC achieves far better MSE results. Accordingly, we observe that FTDE-ASTC mostly maintains a MSE of maximum 0.4 in Fig. 3, because it can exploit ASTC adaptive model selection scheme to detect the erroneous values and discard them. Similarly, in Fig. 3 the message cost for ASTC is very high because it classifies all the errors as outliers and reports them to the sink. Whereas, FTDE-ASTC discards the erroneous sample avoiding unnecessary message transmissions. The aggressive message drop for data set 4 in Fig. 3 is achieved because one of the nodes participating in modeling construction had particularly low corruption rate, and FTDE-ASTC exploited the adaptive modeling to identify and select the specific node and used its model to discard the erroneous samples.

VI. Conclusion

We proposed FTDE, a fully distributed, fault-agnostic data error detection and correction scheme. Following the design objective, the proposed scheme is very lightweight in terms of memory, computation and message cost. The simulation results demonstrated that established time-series based compression schemes, when enhanced by FTDE, are able to isolate erroneous data and maintain high approximation accuracy relative to the non-enhanced schemes. In addition, FTDE actually reduced the message overhead since it allows to suppress the transmission of erroneous data. We demonstrated through the simulations that FTDE can adapt to different time-series based schemes and accordingly further improves its own performance by exploiting the specific enhancement of a given scheme. In future, we plan to port the proposed FTDE to other time-series based WSN operations such as data aggregation and data prediction with the aim of designing a generalized data fault tolerance scheme. We also plan to extend it to compression scheme other than time-series schemes.

References

[1] M. Cinque *et al.*, "A survey on resiliency assessment techniques for wireless sensor networks," in *Proc. of MOBIWAC'13*, 2013, pp. 73–80.

[2] L. M. S. de Souza *et al.*, *A survey on fault tolerance in wireless sensor networks*, 2007.

[3] A. B. Sharma *et al.*, "Sensor faults: Detection methods and prevalence in real-world datasets," *ACM Trans. on Sensor Networks*, vol. 6, no. 3, pp. 23:1–23:39, Jun. 2010.

[4] M. A. Razzaque *et al.*, "Compression in wireless sensor networks: A survey and comparative evaluation," *ACM Trans. on Sensor Networks*, vol. 10, no. 1, pp. 5:1–5:44, 2013.

[5] M. Khan *et al.*, "Troubleshooting interactive complexity bugs in wireless sensor networks using data mining techniques," *ACM Trans. on Sensor Networks*, vol. 10, no. 2, 2014.

[6] A. R. M. Kamal, C. Bleakley, and S. Dobson, "Packet-level attestation (pla): A framework for in-network sensor data reliability," *ACM Trans. Sen. Netw.*, vol. 9, no. 2, pp. 19:1–19:28, Apr. 2013.

[7] M. C. Vuran *et al.*, "Spatio-temporal correlation: theory and applications for wireless sensor networks," *Computer Net.: The Int. Journal of Computer and Telecomm. Netw.*, vol. 45, no. 3, pp. 245–259, 2004.

[8] S. Guo, H. Zhang, Z. Zhong, J. Chen, Q. Cao, and T. He, "Detecting faulty nodes with data errors for wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 10, no. 3, pp. 40:1–40:27, May 2014.

[9] T. Clouqueur, K. Saluja, and P. Ramanathan, "Fault tolerance in collaborative sensor networks for target detection," *Computers, IEEE Transactions on*, vol. 53, no. 3, pp. 320–333, Mar 2004.

[10] Bychkovskiy *et al.*, "A collaborative approach to in-place sensor calibration," in *Proc. of ACM/IEEE IPSN'03*, 2003, pp. 301–316.

[11] E. Miluzzo *et al.*, "CaliBree: A self-calibration system for mobile sensor networks," in *Proc. of IEEE DCOSS'08*, 2008, pp. 314–331.

[12] N. Ramanathan *et al.*, "Rapid Deployment with Confidence: Calibration and Fault Detection in Environmental Sensor Networks," in *Technical Report CENS-TR-62, Center for Embedded Networked Sensing*, 2006.

[13] Y. Zhang *et al.*, "Outlier detection techniques for wireless sensor networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 12, no. 2, pp. 159–170, 2010.

[14] K. Ni *et al.*, "Sensor network data fault types," *ACM Trans. Sen. Netw.*, vol. 5, no. 3, pp. 25:1–25:29, Jun. 2009.

[15] A. Ali *et al.*, "An adaptive and composite spatio-temporal data compression approach for wireless sensor networks," in *Proc. of ACM MSWiM'11*, 2011, pp. 67–76.

[16] D. Tulone and S. Madden, "PAQ: Time series forecasting for approximate query answering in sensor networks," in *EWSN'06*, 2006, pp. 21–37.

[17] S. Madden, "Intel lab data," http://db.csail.mit.edu/labdata/labdata.html, 2003.