CREDC Technical Report: Resilient Data Collection in Refinery Sensor Networks Under Large Scale Failures

Tianyuan Liu[1], Hongpeng Guo[1], King-Shan Lui[2], Haiming Jin[1], and Klara Nahrstedt[1]

[1]Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA
[2]Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong

Abstract

Wireless sensors and measurement devices are widely deployed in oil and gas refineries to monitor the health of the pipes. These sensors are deployed along the pipes in an open area and thus are subject to large scale failures due to cyber-physical attacks and hazardous environments. In this paper, we study the resilience issues in collecting data from a dense and large scale set of sensors deployed over the physical refinery pipe network. We construct a multi-tree sensor mesh network over the refinery sensors for data collection. The reporting messages within one of the trees, while passing along the tree, are protected by a secret key shared among all sensors on the tree. Our construction aims to minimize the data collection time and ensures that the information leakage probability of the secret key is bounded. To tolerate large scale failures, we present a distributed self-healing protocol, which enables a tree node to discover a secondary path when its parent fails. The simulation result shows that the self-healing protocol tolerates large scale failures with high probability and has small overhead in data collection time.

I. Introduction

To monitor the health of oil and gas refineries, wireless sensors and measurement devices are massively installed to collect various real-time measurements like temperature, pressure and corrosion [1]–[4]. Several standards such as ISA100.11a [5] and WirelessHART [6] are actively updated for industrial applications. These wireless devices form a wireless mesh network and pass the monitoring data through a gateway to the control center to facilitate robust monitoring and control. As the sensors are often deployed around pipes in an open area, they may be subject to different forms of attacks and damages, and thus become faulty after installation. For example, the sensors around pipes can be damaged in pipe fire or compromised by attackers.

Fast response to abnormal behaviors and situational awareness is very important to reduce the risk and scope of accidents. A reliable communication network is thus necessary to ensure that alert messages can be delivered in time. Therefore, the data collection infrastructure should be resilient in a way that sensors should be accessible even if some relays fail. Although both ISA 100.11a and WirelessHART introduce redundancy of gateways and paths as the reliability enhancement [7], [8], sensors can still be inaccessible under large scale failures.

In addition to network reliability, information security is also an important issue in the refinery sensor network. For the sensor network that uses a shared key, leakage of this shared key enables an attacker to impersonate or eavesdrop the sensing devices. Both ISA 100.11a and WirelessHART support a shared join key that a device can authenticate itself to the gateway [9]. Protecting the shared keys should be a major security concern of the system administrator.

In this paper, we design a resilient data collection framework which provides reliability under large scale failures and protection of shared security key. We consider the scenario where a large number of wireless measurement devices (MDs) are deployed in a dense refinery sensor network. Each MD is capable of communicating with nearby devices within a small range. A subset of MDs are selected as

access points (AP) and attached to the gateway. MDs that are not connected to the gateway directly relay their data through the mesh network. Fig. 1 shows an example topology with ten MDs connecting to three access points MD_1, MD_6 and MD_9. The arrows represent the uplinks of each MD. We show the unused communication links of MD_7 by dashed lines. Therefore in this example, MD_7 can communicate with MD_2, MD_6, MD_8 and MD_9 by short-range wireless communication.

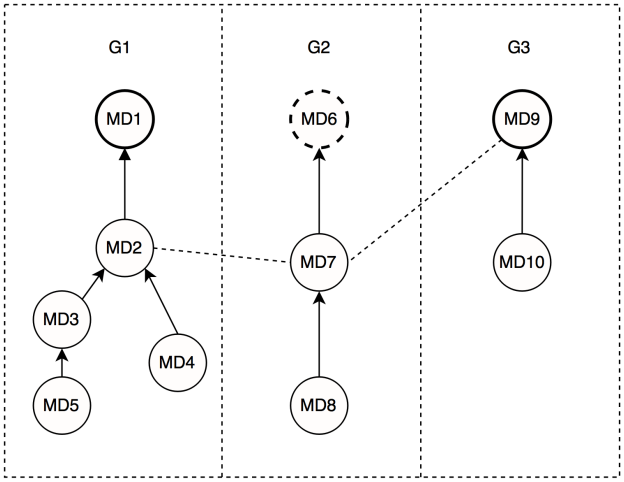

Fig. 1. An Example Topology

The multi-tree structure is easy to maintain as each node only needs to keep track of its parent and children. Besides, as the MDs are divided into disjoint trees rooted at different nodes, data traffic becomes isolated, and data protection becomes easier. For example, when we want to encrypt a command for distributing among the trees, different tree can use a different security key. This security key can then be selected according to the different situations of the tree. A tree with nodes that may be compromised easier can opt for using a longer key or refreshing the key more frequently. Moreover, in case a key of a certain tree is leaked, the other trees are still safe. In our formulation for tree construction, we will consider the leakage probability of the security key used.

Before the first data collection, the tree collection structure is computed by a system administrator (SA) based on the network topology. The SA then informs the MDs about their neighbors, parent, and children to build the data collection paths. The time to collect data on a certain tree is related to the height of the tree. Since each tree can collect data simultaneously, the amount of time needed to collect data from the forest is the maximum height of all trees. We formulate the Resilient Data Collection Forest Problem (RDCFP) using mixed-integer linear programming (MILP) to optimize the overall data collection time with the consideration of key security.

After the tree structure is setup, it will be used for multiple data collection rounds. Since the MDs are exposed to an insecure physical environment, an MD may fail and stop working. This failed MD cannot relay data for its descendant MDs anymore. At this point, tree re-construction is needed. In a dense network, an MD has an adequate number of neighbors within its communication range, so that it can find another path among its neighbors to connect with SA when its parent fails. Therefore, we adopt the self-healing approach that nodes affected would try to establish their data reporting paths locally through negotiating with neighbors. With proper heuristics in path selection, the data collection time after self-healing does not increase significantly compared with the optimal forest.

Our main contributions in this paper can be summarized as follows:

- We designed a resilient data collection framework for the refinery sensor network which has built-in reliability and security features.
- We proposed a distributed self-healing protocol to tolerate large scale simultaneous relay failures.
- We optimized the data collection time by formulating the forest construction problem using MILP.

- We evaluated the self-healing protocol on refinery pipeline simulator and compare the reliability of self-healing protocol to the approach adopted by WirelessHART.

II. Related Work

Tree-based data collection in sensor networks has been actively studied [10]–[12]. Most work aims at optimizing energy usage or sensor lifetime while reducing data reporting latency. Due to the massive number of sensors, a hierarchical data collection structure is usually adopted. Cluster heads are selected to collect data from sensors within its neighborhood, and then reports the data to the data sink. Some recent studies on how to select cluster heads to balance energy and latency can be found in [13], [14]. As energy harvesting has been proposed to prolong the lifetime of a sensor, some researchers study data collection with this emerging technology [15], [16]. To reduce traffic, compression techniques are studied to improve the data collection performance [17], [18].

A major class of fault tolerance and recovery techniques in wireless sensor networks exploit the link or path redundancy in the network. In a multi-path structure, packets are routed through disjoint backup paths when the primary path fails [19], [20]. The resilience of WirelessHART is achieved by a special multi-path structure [8], where each intermediate node on the path must have at least two neighbors to forward the traffic. However, this approach does not consider the shared key security and we will show that its performance under large scale failure is not satisfactory in section VI.

III. System Model

A. Network Model

We consider the network model as a set of MDs $\{\mathrm{MD}_1, \mathrm{MD}_2, \cdots, \mathrm{MD}_n\}$ densely placed in a 3-dimensional space. All the MDs have an identical short communication range R and thus form an underlying communication network. Two MDs are considered as *neighbors* if their distance is no larger than R and can communicate with each other directly.

A subset of r MDs $\{\mathrm{MD}_{i_1}, \mathrm{MD}_{i_2}, \cdots, \mathrm{MD}_{i_r}\}$ are given as roots (access points). Each non-root MD is connected to one of the roots in a multi-hop manner and thus forms a forest in the network. For the rest of the paper, we also use the term *group* referring to a set of MDs in the same tree.

B. Security Model

The data reported by each MD should be protected in terms of both integrity and confidentiality. The data should be read only by the SA, but not intermediate MDs which help relaying the data. On the other hand, the intermediate MDs should be able to authenticate the relayed message in order to defend data injection attacks [21], [22].

Similar to [23], we develop the secure tree-based data collection framework that allows MDs to report private data to SA while facilitating intermediate MDs to authenticate the messages. *Diffie-Hellman* key exchange is used to establish an encryption key for the data between SA and every MD. Integrity check along the tree path is supported by a *group key* which is known by all members in the tree.

If an adversary knows the group key, he can stealthily inject forged messages into the network. This immediately leads to vulnerabilities such as data injection attack and denial-of-service attack. A secure protocol should confine the risk of leaking a key to a certain acceptable bound. To model the group key leakage probability, we assume that each MD leaks the key with a certain probability. The key leakage probabilities of the MDs can then be used to develop the probability of leaking the group key to an adversary. Our protocol ensures this probability is not greater a predefined threshold. Different MDs may have different key leakage probabilities as the MDs in an oil refinery are subject to different risks according to the positions and functions.

A risk introduced by using the group key is that, if an adversary knows the group key, he can inject arbitrary messages without being detected by the integrity check. Although the SA can finally detect the

data carried in the message are not legitimate, network resource can be exhausted in transmitting the messages. As the MDs may be exposed in an open environment, there is a risk that the group key is leaked from a group member. We model the leakage probability of MD_i to be p_i, and control the risk of group key leakage under a security threshold P_{th}.

C. Resilience Model

Apart from security attacks, MDs along the pipes are subject to physical damages that they may fail and stop functioning. When an MD fails and stops, it can no longer report data or relay messages for other MDs. More precisely in a tree-based data collection scheme, a failed MD would lead to the loss of the data of its whole subtree. To avoid this from happening, the data collection structure has to be resilient so that when a certain node fails, the loss of data should be minimal. Our resilience strategy is to design a distributed self-healing protocol, where MDs are able to find a secondary data path to the SA when their parents fail.

We consider the *fail-stop model* for all MDs, i.e., once an MD fails, it does not recover. We consider at most $k-1$ simultaneous failures for each data collection round where k is the number of neighbors for each MD, and suppose that at least one root does not fail. Furthermore, we assume that failures are detected before a data collection round happens, i.e., the secondary paths should have been turned on when a data collection command arrives.

IV. Protocol Design

A. Protocol Overview

After the MDs are deployed, we first compute the initial multi-tree data collection structure. This is done by the SA which has the full topology information and the key leakage probability of each MD. After the data collection structure is developed, SA tells each root its tree structure. Each tree root can then distribute the relevant sub-tree structure to each of its children. The process continues until each node knows its parent and its children (if any).

To protect the data, security keys have to be used. Necessary keys can be established and distributed in the process of setting up the trees. In [23], we describe how to develop various keys on a tree to protect the data reported by each MD. Our resilient data collection protocol in this paper does not assume any particular cryptographic mechanism to be used or how the data are protected through the keys. Instead, we consider the generic situation that there is a group key shared among all the MDs on a tree and the risk of leaking this key by each MD is known. Our protocol would ensure the risk of leaking this group key is bounded in the data collection structure.

B. Self-healing

The self-healing protocol aims at recovering network connectivity after some MDs fail. Two properties must be held after the self-healing process: (1) the risk of leaking any group key is small; (2) the data collection time is short. To maintain these properties, two types of information, *accumulative leakage probability* and *subtree height*, must be spread through the group members.

a) Accumulative Leakage Probability: Sharing a group key introduces extra security concern because an attacker can steal the group key from any of the group members. We model the key leakage probability of MD_i as p_i, which is a pre-defined parameter while deploying the MD. Then, the *accumulative leakage probability* of a tree $\mathcal{T}$ can be expressed as

$$P_{\text{leak}}(\mathcal{T}) = 1 - \prod_{i \in \mathcal{T}} (1 - p_i).$$

In order to control the risk, we require the leakage probability on every tree to be smaller than a predefined threshold $P_{\text{leak}}(\mathcal{T}) \leq P_{\text{th}}$. Therefore, we have

$$P_{\text{leak}}(\mathcal{T}) = 1 - \prod_{i \in \mathcal{T}} (1 - p_i) \leq P_{\text{th}}. \tag{1}$$

As keys are distributed, we also notify each MD about the tree structure and the leakage probability of its group members. Therefore, MD_i can compute the accumulative probability of its tree $\mathcal{T}$ and check if the following constraint holds

$$\sum_j \log(1-p_j) \geq \log(1-P_{\text{th}}),\ \forall\, \text{MD}_j \in \mathcal{T}.$$

This can be derived by taking $\log(\cdot)$ for both sides in inequality (1). We will use the accumulative leakage probability in the log form in the rest of our interpretation.

This constraint implies that the number of members in a group is upper-bounded. A group cannot admit a new set of MDs if the accumulative leakage probability will exceed the security threshold. Therefore, we define *group capacity* as the difference between security threshold and the current accumulative leakage probability of the group. Suppose MD_j is in tree $\mathcal{T}$. The group capacity $C_\mathcal{T}$ can be expressed as

$$C_\mathcal{T} = \log(1-P_{\text{th}}) - \sum_k \log(1-p_k),\ \forall\, \text{MD}_k \in \mathcal{T}.$$

$C_\mathcal{T} \leq 0$ must hold for all groups.

As MD_i has the information about the structure of its subtree, it can compute the accumulative leakage probability of its subtree. If the group capacities for its neighbors are also given, it can locally determine which neighbor can be a valid candidate for a secondary path.

b) Extra Tree Height: If there are multiple candidates for the secondary parent, we should find the best secondary parent so that the self-healing would add to the least extra data collection time. The data collection time on a tree depends on the highest tree branch. We denote the height of a subtree rooted at MD_j as H_j. If MD_i joins the subtree rooted at MD_j, the extra tree height added to the whole tree is

$$\max(0,\ H_i - H_j + 1).$$

We use the example topology in Figure 1 for further explanation. Suppose MD_6 fails and MD_7 is looking for a secondary parent from MD_2 and MD_9. Assume both G_1 and G_3 will not break the group capacity constraint by accepting this request. In this case, the height of subtree at MD_2, MD_9 and MD_7 are 2, 1 and 1 correspondingly. If MD_7 joins G_3, it turns out that the data collection time of G_3 will increase from 1 to 2. However, by joining G_1, there is no extra height added to G_1 because G_1 already has a longer branch $\text{MD}_1 \leftarrow \text{MD}_2 \leftarrow \text{MD}_3 \leftarrow \text{MD}_5$.

c) Self-healing Protocol: Now we start describing the detail of the protocol. Once MD_i detects a parent failure, it broadcasts a *join* message signed by its private sign key. While deploying the MDs, we require each MD to store a list of public keys of its neighbors. Thus, the neighbors of MD_i can verify if the join message comes from a valid peer by verifying the signature. Once the join request is authenticated, each neighbor MD_j replies with a *join-ack* message. To facilitate the choice of secondary parent, the group capacity and subtree height of MD_j is included in this message.

After MD_i receives all the join-acks, it computes the accumulative leakage probability of its subtree and uses the group capacity to determine a set of neighbors whose group can admit the whole subtree. If there are multiple candidate, it selects one with the least extra tree height. A *join-proposal* message is then sent to the selected neighbor. This message includes the accumulative leakage probability of MD_i's subtree.

d) Competing Requests: When multiple failures happen simultaneously in the system, a group can receive several join request at the same time. If every group member approves these requests independently, it may result in the violation of the security constraint.

To resolve the conflict, we require MD_j to forward the join-proposal to its tree root and wait for the approval from the root. The tree root approve the request based on first-come-first-serve criteria, and replies with an approval or disapproval message to MD_j.

e) Group Update: As soon as approving the join-proposal, the root MD also broadcasts the updated membership list down to all the members in the tree. In this way, every member can be notified about the join of MD_i and update the membership list as well as the group capacity.

After joining the new group, the MD_i and all the MDs in its subtree need to obtain the new group key used in the tree of MD_j. After getting approval from the root, MD_j sends MD_i its group key encrypted using PKI. MD_i continues to forward the group key down to its subtree until every MD in the subtree gets the new group key.

f) Pruning: Sometimes MD_i may find that it is impossible to join any of its neighbors' trees. This is more likely to happen when the accumulative leakage probability of MD_i' subtree is large. In order to reconstruct the trees, MD_i must prune some branches on its subtree so that the remainder of the subtree is small enough. Once it decides which children to prune, it sends *prune* messages to all these children. These children then run the self-healing protocol by their own.

g) Self-healing Failure: Occasionally, the self-healing protocol can not successfully identify a feasible secondary path. This could happen if the group capacities of all the neighbors' tree have reached the security constraint. In such cases, the disconnected MD should notify the SA about the failure. Therefore, we allow the disconnected MD to broadcast a *disconnect* message with its signature to all the neighbors. This message is relayed to SA and SA should recompute the forest if there is a feasible solution.

V. Precomputing Data Collection Forest

In this section, we describe how data collection forest is constructed in the tree computation phase. We formulate the problem as a Resilient Data Collection Forest Problem (RDCFP) using mixed-integer linear programming (MILP) and solve it by the Gurobi Solver [24].

A. Problem Description

We assume that SA knows the locations of all MDs and selects a set of roots such that a possible forest exist. We denote the topology of MDs as a directed graph $\mathcal{G} = \langle \mathcal{R}, \mathcal{M}, \mathcal{E} \rangle$, where $\mathcal{R}$ is the set of root candidates, $\mathcal{M}$ is the set of MDs and $\mathcal{E}$ is the set of edges. If MD_i and MD_j can communicate directly without the help of any relay, two directed edges $e_{i,j}$ and $e_{j,i}$ are present in $\mathcal{E}$. The edges are unweighted so that the length of a path is equal to the hop count.

Our objective is to minimize the time to collect data from every tree root. Since the data on each tree can be collected simultaneously, the data collection time of the whole forest depends on the maximum height of all trees. The height of each tree can be calculated by the depth-first search (DFS) algorithm.

B. Problem Formulation

Before introducing the detail, we describe the general idea of the formulation. We address the objective of minimizing data collection time from a "path" perspective. Let $\mathcal{P}_{ij}^k$ be the k-th pre-computed path from MD_i to MD_j with minimum length $L_{i,j}$, and use $\mathcal{K}_{ij}$ as a set of indices of all these paths. Then, the time to collect data from a subtree rooted at MD_j depends on the longest path on the subtree

$$H_j = \max_{i \in \mathcal{M}} \sum_{k \in \mathcal{K}_{ij}} x_{ij}^k L_{ij}, \tag{2}$$

where x_{ij}^k is an indicator variable such that

$$x_{ij}^k = \begin{cases} 1, & \text{if } \mathcal{P}_{ij}^k \text{ is selected as a path from } MD_i \text{ to } MD_j; \\ 0, & \text{otherwise.} \end{cases}$$

Thus, the time to collect data from all the candidate roots can be expressed as $\max_{j \in \mathcal{R}} H_j$.

Now, we formally present the Resilient Data Collection Tree Problem (RDCTP) as follows:

$$\min \max x_{ij}^k L_{ij} \tag{3}$$

$$\text{s.t.} \sum_{j\in\mathcal{R},\, k\in\mathcal{K}_{ij}} x_{ij}^k = 1, \forall i \in \mathcal{M} \tag{4}$$

$$x_{ij}^k \le x_{i'j}^l \quad \text{if} \quad P_{i'j}^l \subset P_{ij}^k \tag{5}$$

$$\sum_{i\in\mathcal{M},\, k\in\mathcal{K}_{ij}} x_{ij}^k \log(1-p_i) \ge \log(1-P_{\text{th}}), \forall j \in \mathcal{R} \tag{6}$$

The intuition of these constraints can be interpreted as follows:

(4) Every MD has a valid primary path to some candidate roots.

(5) The selected paths should together form a forest of trees. This constraint ensures that if a path $\mathcal{P}_{ij}^k$ is selected, its subpath $\mathcal{P}_{i'j'}^l$ must be selected. Therefore, the computed graph will be a tree.

(6) The accumulative leakage probability of MDs in the same tree is upper-bounded by the security threshold.

VI. Simulation

In this section, we describe how we evaluate the self-healing protocol. First, we generate the network topologies for refinery sensor networks. Then, we inject large scale failures into the network and measure the success rate of self-healing as well as the data collection time. We use Gurobi Solver [24] to get the optimal solution for RDCFP for comparison. Unfortunately, for large networks that are used in our simulation, it takes too long for the solver to generate solutions. We thus only consider a subset of shortest paths (subset approach) instead of all shortest paths (full path approach) described in the problem formulation. We measure the difference in performance of the subset approach with the optimal solution on small networks that are around 100 nodes. In the 200 topologies we tested, the results of subset approach and full path approach are identical.

A. Topology Generation

The most important intuition we follow to generate our sensor topologies is that, the sensors are often attached around pipes. These pipes often have linear structure and they are usually deployed horizontally. Therefore, instead of placing MDs randomly in a 3D space, we first generate some pipes and then place the MDs along these pipes.

Specifically, we assume that the refinery is a 50m×50m×30m box, and that the pipes are placed on four horizontal planes with a height difference of 10m. On each plane, we randomly generate N_p horizontal lines of length L_p. These lines are parallel to either x-axis or y-axis. Then, we place different number of sensors along the lines. For the sensors on the same pipe, we distributed them with equal distance D so that the following relationship holds

$$N_{\text{nodes}} = 4 \times N_p \times \frac{L_p}{D}. \tag{7}$$

In our simulation, N_p ranges from 30 to 48 with a step of 6 and L_p ranges from 25m to 35m randomly.

We identically choose the communication range for all MDs to be 15m, and the distance between two adjacent MDs on the same pipe to be 12m. We generate four experiment groups with the number of nodes ranging from 360 to 576, where each node has 51.67 to 81.02 neighbors on average. For each group, we generate ten different topologies. All the metrics are measured as an average of all the ten topologies.

In each topology, we randomly select the key leakage probability of each MD uniformly in the range $[0.008, 0.012]$. Furthermore, we set the security threshold as $P_{\text{th}} = 0.3$ so that on average, each tree can have at most $\frac{N_{\text{nodes}}}{\log(1-0.3)/\log(1-0.01)} \approx \frac{N_{\text{nodes}}}{35.49}$ nodes. In order to have a feasible solution with high probability, we select m MDs as root candidates such that $m \ge \frac{N}{30}$. The m nodes with the most neighbors are selected as roots.

B. Failure Injection

To evaluate the performance of self-healing protocol, we inject some random failures into the network and measure the success rate and data collection time of the self-healing process. To simulate the real-world failure where sensors fail simultaneously due to some physical damages to the pipes, we randomly select a coordination in the 3-D space and fail 2% of the nodes around this location. Note that some failures may not be helpful in evaluating the self-healing protocol, we selectively discard some failures if one of the followings is true:

1) All failed MDs are leaf-nodes in the forest. In this case, no predecessors is disconnected and self-healing does not happen at all.
2) The network is disconnected by the failure. In this case, it is impossible to construct a data collection forest.

For each topology, we randomly inject $N_{\text{fail}} = 1,000$ failure instances using the above criteria.

C. Metrics and Baseline

Two metrics are measured in each failure instance. First, we count the number of instances N_{recover} where our self-healing protocol succeeds to recover the network. We refer the ratio of N_{recover} to N_{fail} as Recovery Success Ratio (RSR). The RSR indicates how likely our self-healing protocol can recover from a random large scale simultaneous failure. Second, we use the Maximum Tree Height (MTH) as an indication of the data collection time. Three measurements are made in the network. Before we inject any failures, we measure the MTH of the forest. Since the forest is constructed by Gurobi Solver, this measurement always has smallest MTH. After we inject a failure instance, we remove the failed nodes from the network and run the Solver again to recompute the optimal MTH of the remainder network. Then, we run our self-healing protocol on the remainder network, and measure the MTH after the recovery is done.

We choose the reliable graph routing algorithm of WirelessHART as the baseline [8]. The reliable graph routing of WirelessHART builds a broadcast graph where each node has at least two parents to forward its packets. Therefore, it is guaranteed that each node has at least two paths to the gateway. Note that WirelessHART does not have the concept of shared key security built in its heart, we relax this constraint in our simulation. Without the shared key security constraint, the maximum tree height is naturally smaller because there is no limit on the number of nodes on a tree. Hence, it is not fair to compare the maximum tree height of these protocols. Instead, we only focus on evaluating the reliability these protocols.

We construct the reliable graph routing paths and inject the same failure instances that are used to measure the RSR of self-healing. We count the number of instances $N_{\text{connected}}$ that every living node in the reliable graph routing is still connected to at least one root. Similar to RSR, we use the ratio of $N_{\text{connected}}$ to N_{fail} to measure how likely the reliable graph routing can survive a large scale failure.

D. Result and Discussion

TABLE I
RSR OF SELF-HEALING V.S. RELIABLE GRAPH ROUTING

N_{node}	Self-healing	Reliable Graph Routing
360	91.1%	33.8%
432	92.3%	34.3%
504	92.4%	36.2%
576	93.0%	36.9%

Table I shows that our self-healing protocol can successfully recover over 90% of the injected large scale failures With an increasing node density in the network, our protocol achieves better RSR. This is expected because the self-healing protocol has a better chance to succeed if each node has more neighbors.

However, reliable graph routing fails to connect some of the living nodes in most of these cases. This implies that the reliable graph routing approach is not suitable for dealing with large scale simultaneous failures.

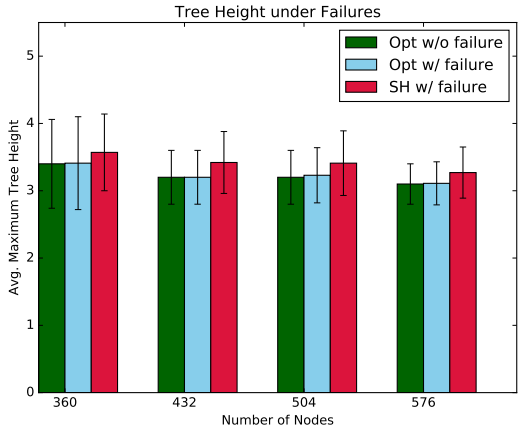

Fig. 2. Maximum Tree Height with and without Failure

We calculate the average of MTH over all the injected failure instances for each topology and show the three MTH measurements in Fig. 2. With a higher density of the network, the maximum tree height decreases consistently. In the worst case, the optimal forest reconstructed after failure introduces about 1% extra data collection time. The data collection time resulted from self-healing is about 7% higher than the optimal reconstruction. Considering that the computation of optimal forest is costly (it takes about 20 mins to construct the largest topology with 576 nodes), it is a reasonable tradeoff to recover connectivity with self-healing protocol.

VII. Conclusion

In this paper, we design a resilient data collection protocol in oil and gas refinery network. We propose a distributed self-healing mechanism that tolerates multiple simultaneous failures in a dense network. Through simulation on generated refinery network topologies, we show that the self-healing protocol can successfully recover most failures with a small amount of overhead on data collection time.

VIII. Acknowledgment

This material is based upon work supported by the Department of Energy under Award Number DE-OE0000780.

References

[1] T. Arampatzis, J. Lygeros, and S. Manesis, "A survey of applications of wireless sensors and wireless sensor networks," in *Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation Intelligent Control, 2005.* IEEE, 2005, pp. 719–724.

[2] J. Chow, J. Watson, L.-W. Chen, G. Paredes-Miranda, M.-C. Chang, D. Trimble, K. Fung, H. Zhang, and J. Zhen Yu, "Refining temperature measures in thermal/optical carbon analysis," *Atmospheric Chemistry and Physics*, vol. 5, no. 11, pp. 2961–2972, 2005.

[3] M. reza Akhondi, A. Talevski, S. Carlsen, and S. Petersen, "Applications of wireless sensor networks in the oil, gas and resources industries," in *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on.* IEEE, 2010, pp. 941–948.

[4] K. Wold and H. Jenssen, "Solutions for corrosion monitoring in refineries," 2014.

[5] ISA. Isa100, wireless systems for automation. [Online]. Available: https://www.isa.org/isa100/

[6] F. Group. Hart technology. [Online]. Available: https://fieldcommgroup.org/technologies/hart/hart-technology

[7] S. Petersen and S. Carlsen, "Wirelesshart versus isa100. 11a: The format war hits the factory floor," *IEEE Industrial Electronics Magazine*, vol. 5, no. 4, pp. 23–34, 2011.

[8] S. Han, X. Zhu, A. K. Mok, D. Chen, and M. Nixon, "Reliable and real-time communication in industrial wireless mesh networks," in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2011 17th IEEE*. IEEE, 2011, pp. 3–12.

[9] M. Nixon and T. Round Rock, "A comparison of wirelesshart and isa100. 11a," *Whitepaper, Emerson Process Management*, pp. 1–36, 2012.

[10] O. D. Incel, A. Ghosh, B. Krishnamachari, and K. Chintalapudi, "Fast data collection in tree-based wireless sensor networks," *IEEE Transactions on Mobile computing*, vol. 11, no. 1, pp. 86–99, 2012.

[11] C.-T. Cheng, N. Ganganath, and K.-Y. Fok, "Concurrent data collection trees for iot applications," *IEEE Transactions on Industrial Informatics*, 2016.

[12] J. Fei, H. Wu, and W. Y. Alghamdi, "Lifetime and latency aware data collection based on k-tree," in *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2015 IEEE Tenth International Conference on*. IEEE, 2015, pp. 1–6.

[13] R. Zhang, J. Pan, D. Xie, and F. Wang, "NDCMC: A hybrid data collection approach for large-scale wsns using mobile element and hierarchical clustering," *IEEE Internet of Things Journal*, to appear.

[14] Z. Xu, L. Chen, C. Chen, and X. Guan, "Joint clustering and routing design for reliable and efficient data collection in large-scale wireless sensor networks," *IEEE Internet of Things Journal*, to appear.

[15] A. Mehrabi and K. Kim, "Maximizing data collection throughput on a path in energy harvesting sensor networks using a mobile sink," *IEEE Transactions on Mobile Computing*, vol. 15, no. 3, March 2016.

[16] C. Wang, S. Guo, and Y. Yang, "An optimization framework for mobile data collection in energy-harvesting wireless sensor networks," *IEEE Transactions on Mobile Computing*, toappear.

[17] Y. Yao, Q. Cao, and A. V. Vasilakos, "Edal: An energy-efficient, delay-aware, and lifetime-balancing data collection protocol for heterogeneous wireless sensor networks," *Networking, IEEE/ACM Transactions on*, vol. 23, no. 3, pp. 810–823, 2015.

[18] X.-Y. Liu, Y. Zhu, L. Kong, C. Liu, Y. Gu, A. V. Vasilakos, and M.-Y. Wu, "Cdc: Compressive data collection for wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 8, August 2015.

[19] J. Kamato, L. Qian, W. Li, and Z. Han, "Biconnected tree for robust data collection in advanced metering infrastructure," in *Proc. of IEEE WCNC*, 2015.

[20] J. Silber, S. Sahu, J. Singh, and Z. Liu, "Augmenting overlay trees for failure resiliency," in *Proc. of IEEE Globecom*, 2004.

[21] Z. Yu and Y. Guan, "A dynamic en-route scheme for filtering false data injection in wireless sensor networks," in *SenSys*, vol. 5, 2005, pp. 294–295.

[22] X. Yang, J. Lin, W. Yu, P.-M. Moulema, X. Fu, and W. Zhao, "A novel en-route filtering scheme against false data injection attacks in cyber-physical networked systems," *IEEE Transactions on Computers*, vol. 64, no. 1, pp. 4–18, 2015.

[23] H. Jin, S. Uludag, K.-S. Lui, and K. Nahrstedt, "Secure data collection in constrained tree-based smart grid environments," in *Proc. of IEEE SmartGridComm*, 2014.

[24] Gurobi. Gurobi optimization - the best mathematical programming solver. [Online]. Available: http://www.gurobi.com/