Resilient Data Collection Protocol with In-Network Processing for Oil and Gas Refinery Networks

Hongpeng Guo*, King-Shan Lui*, Tianyuan Liu†, Klara Nahrstedt†
*Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong
†Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA

***Abstract*—To facilitate efficient control and monitoring, massive wireless sensors and measurement devices are deployed in Oil and Gas Refineries. These sensors are deployed along the pipes and data measured are correlated. In-network processing with enough data along a pipe allows abnormal events to be detected early by an analyzer deployed on-site. The data collection structure should then be carefully developed to facilitate each pipe to be monitored efficiently. On the other hand, as the sensors are deployed in harsh environment and subject to hardware damages, they may fail. Sensor failures may disrupt the on-site monitoring of pipes by analyzers. In this paper, we study the resilience issue in refinery sensor networks to facilitate the robustness of fast data collection and abnormal event monitoring even some devices fail. We present a quorum scheme to ensure every analyzer would get enough relevant data for analysis. We apply a multi-tree data collection structure to achieve fast data collection. To tolerate node failures, we present a novel distributed Refinery Resilient Protocol (RRP), which enables the affected nodes to discover an alternative path to relay data. The simulation results show that the RRP maintains efficient data collection and event monitoring even a significant portion of sensors have failed.**

I. Introduction

To monitor the health of oil and gas refineries, massive sensors or measurement devices are installed around pipelines [1] to collect real-time environment measurements and materials' chemical properties [2]. These sensors form an ad hoc network and report the monitoring data to the control center to facilitate robust monitoring and control [3]. As the sensors installed around the same pipeline measure the same chemical material and collect data of strong relations, appropriate in-network processing allows problems to be detected earlier. As a result, it would be desirable to develop a data collection structure that facilitates enough data of the same pipe to be accessed by some devices within the network for analysis. The sensors are deployed in an open area and subject to different kinds of damages such as hurricanes [4]. Unfortunately, due to the nature of the infrastructure, instant replacement may not be feasible [3]. Sensor failures not only make the data of a certain portion of the pipe unavailable but may also disrupt the on-site analyses due to insufficient amount of data. Therefore, it is very important to have a resilient data collection infrastructure so that enough data can still be reported in a timely fashion even if some devices fail.

In this paper, we consider the scenario where a large amount of measurement devices (MDs) are deployed around refinery pipelines and thus form a sensor network. Each MD generates data periodically to be reported to the Control Center (CC). Each MD is equipped with short-range wireless communication capability. A subset of MDs is connected to a long-distance wireless network so that they can directly communicate with the remote CC. MDs that are not equipped with long-distance communication ability have to rely on other MDs to relay their data in a hop-by-hop manner to the CC. We adopt the multi-tree (or forest) structure for data collection. That is, the MDs that can talk to the CC directly are tree roots. Each MD is associated with one and only one tree. The path from the MD to the root is the data reporting path.

In this paper, we want to facilitate a fast and efficient in-network processing so that pipe failures can be identified on-site. We assume that the root MDs can perform some preliminary analysis to the data on the same pipe to identify abnormal events happen on that pipe. We also assume that the root MD requires enough amount of data to perform the analysis. That is, a certain portion of the MDs on the same pipe have to report to the same root MD. Pipes can be dozens of meters long, and MDs on the same pipe cannot all communicate with each other directly. On the other hand, in the case that pipes align close to each other, an MD may have multiple neighbors on the other pipes. Hence, existing data collection schemes that assume in-network processing among neighbor nodes may not suffice the requirement that the analyzed data must come from the same pipe. This unique feature of refinery networks makes data collection a challenging problem which cannot be addressed directly by any conventional protocol investigated before. In addition, due to the harsh environment that the MDs are deployed, the data collection structure must be resilient. In this paper, we aim at developing a data collection structure for refinery networks so that monitoring can be done effectively even when some devices fail. The contributions of this paper are as follows:

- **C1** We incorporate the *quorum requirement* in our problem formulation to facilitate in-network processing and efficient on-site monitoring. The delay-optimal data reporting structure with quorum requirement is formulated as an Integer Linear Programming problem.
- **C2** We develop a distributed protocol that allows the MDs to work together to re-build the data collection structure while maintaining the quorum requirement when MDs fail.
- **C3** The simulation results suggest that our protocol outperforms the conventional methods on failure tolerance significantly, as well as maintaining nearly optimal data collection time.

II. Related Work

To the best of our knowledge, resilient data collection protocol providing in-network data analysis and on-site situation detection for disjoint trees has not been studied in the literature. Nevertheless, many works that study the following three problems independently can be identified. They are (1) Multi-sink data collection schemes in Wireless Sensor Networks (WSNs), (2) Quorum based protocol in WSNs, and (3) Fault tolerant techniques to maintain resilience.

Tree based multi-sink data collection in Wireless Sensor Networks (WSNs) has been actively studied. The major objectives of such designs [5–7] are optimizing energy consumption, reducing data collection delay and maintaining traffic balance. Data observed by each sensor will be relayed to the sink node along tree branches. Shallow trees are usually adopted in these data collection structures. The work in [8] makes use of two sinks to provide resilience that every node has to establish two node-disjoint paths to each sink. In order to achieve energy efficiency, data compression techniques are also widely studied to improve data collection performance [9].

Quorum based protocols are also widely used in WSNs aiming to reduce energy consumption. Quorum here refers to a minimal set of sensors to accomplish an operation. Quorum systems usually improve energy efficiency by reducing message transmissions and collisions [10]. [11] proposed a quorum-based wake-up scheduling scheme to save energy and prolong sensor's lifetime. [12, 13] used quorum to avoid message flooding and achieve energy-efficiency by reducing traffic. However, applying quorum to primary data analysis and event detection has not been studied in the literature.

A state of art class of fault tolerance techniques in WSNs is redundant path routing [14–17]. WirelessHART [18] standard keeps resilience by maintaining the multi-path routing structure that every intermediate node on a path must have at least two neighbors for traffic routing. However, such static resilient method does not perform well when massive continuous failures happen in the networks and such method also tends to break the quorum requirement while maintaining the network connections.

III. Protocol Overview

Many different tree structures can be used to connect the MDs together, but they vary in performance. Figure 1 presents an example of 11 MDs deployed on 3 pipes (l_1, l_2, and l_3). MD_5 and MD_{11} are equipped with long-distance communication capability and can directly communicate with the CC. Other MDs can only talk to their neighbors by short-range wireless communication. Neighbors are connected by lines in the figure. Suppose the quorum requirement is 75%, then at least 75% of the MDs on the same pipe must be on the same tree. Figure 2 shows a forest topology that satisfies the requirement. The tree edges are represented using thickened lines. Two trees are formed, one rooted at MD_5 and the other rooted at MD_{11}. All the three MDs on pipe l_1 are connected to MD_{11}. For pipes l_2 and l_3, at least three out of the four MDs on the pipe are associated with the same tree. It is worth noting that the quorum requirement may increase the height of the tree(s). In Figure 2, the height of tree MD_{11} is 3 because of MD_1. Although the shortest path from MD_1 to MD_5 is 2 and is shorter, MD_1 cannot join that tree because of the quorum requirement.

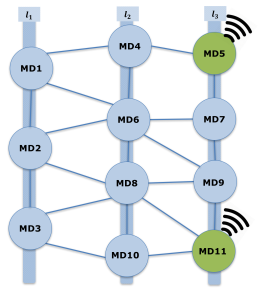


Fig. 1: Network Topology

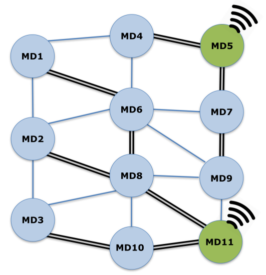


Fig. 2: Forest Topology

Therefore, the problem of identifying a minimum-height multi-tree data collection structure with quorum requirement is not trivial. Our goal is to *develop* and *maintain* a multi-tree structure that (1) each MD is connected to one and only one tree, (2) the quorum requiremet of each pipe is satisfied, and (3) the height of the tallest tree is minimized.

Our protocol has two phases: development phase and maintenance phase. The development phase is the first phase that constructs the initial multi-tree data collection structure before any data collection. The CC, who has the global topology information, computes the optimal structure and informs the MDs through tree roots in a hop by hop manner. Data collection can then be conducted through the trees. The protocol is now in the maintenance phase that the MDs work together to re-establish broken data reporting paths if a node fails. In this phase, new tree paths will be identified in a distributed and localized manner. The CC will be informed of the changes but does not have to participate in the computation process. This ensures a fast recovery to minimize the data loss and delay during the period that the paths are being re-established.

IV. Development Phase: Building Initial Forest

To support the quorum requirement, we first identify a data collection structure before any data are collected. The data collection structure is constructed by the CC with the global topology information of pipes and MDs. We formulate the Initial Trees Construction Problem (ITCP) using the Mixed Integer Linear Programming (MILP) and solve it using an optimal solver.

A. Problem Description

Suppose that we are given an Oil & Gas refinery sensor network consisting of M pipelines and N MDs. We denote the set of pipelines as $\mathcal{L} = \{l_1, l_2, ..., l_M\}$ and the set of MDs as $\mathcal{M} = \{\mathrm{MD}_1, \mathrm{MD}_2, ..., \mathrm{MD}_N\}$. The subset of MDs that can directly communicate with the CC is denoted as $\mathcal{R} \subset \mathcal{M}$. Two MDs are considered neighbors if they can talk to each other directly Each MD must be on one and only one pipe. The association between MDs and pipes is defined by a binary function $q(i, s)$ that $q(i, s) = 1$ if MD_i lies on pipe l_s.

Similar to [6] and [19], 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 among the trees. We identify the optimal data collection structure by selecting the appropriate path from each MD to a certain root. Let $\mathcal{P}_{i,j}^k$ be the k-th precomputed shortest path from MD$_i$ to MD$_j$ where MD$_j \in \mathcal{R}$. Denote $\mathcal{K}_{i,j}$ as a set of indices of all these shortest paths and their length is $L_{i,j}$. In Fig. 2, there are two shortest paths from MD$_8$ to MD$_{11}$ (MD$_8-$MD$_{10}-$MD$_{11}$ and MD$_8 -$ MD$_9 -$ MD$_{11}$) whose length is 2. Therefore, we have $L_{8,11} = 2$ and $\mathcal{K}_{8,11} = \{1, 2\}$. Let $x_{i,j}^k$ be a binary variable representing whether $\mathcal{P}_{i,j}^k$ is selected in the multi-tree structure. Thus,

$$x_{i,j}^k = \begin{cases} 1 & \text{if} \quad \mathcal{P}_{i,j}^k \text{ is selected} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The length of the longest path on the tree rooted at MD$_j \in \mathcal{R}$, also known as the height of tree rooted at MD$_j$, denoted as H_j, is thus

$$\max_i \sum_{k \in \mathcal{K}_{i,j}} x_{i,j}^k \cdot L_{i,j} \qquad \text{where} \qquad \text{MD}_i \in \mathcal{M} \tag{2}$$

The time to collect data from all the tree roots is $\max_j H_j,\ \text{MD}_j \in \mathcal{R}$.

We now explain how we model the quorum requirement. Let the number of MDs on pipe l_s be N_{l_s}. The quorum requirement is described using $\gamma_0 \in [0, 1]$. That is, at least $\gamma_0 \cdot N_{l_s}$ MDs should be included in the same tree. We assume γ_0 is the same for all pipes. The number of MDs on pipeline l_s that are connected to MD$_j \in \mathcal{R}$ is thus,

$$\sum_i \Big(\sum_{k \in \mathcal{K}_{i,j}} x_{i,j}^k \Big) \cdot q(i, s), \qquad \forall \text{MD}_i \in \mathcal{M}. \tag{3}$$

In order to keep some resilience space such that the quorum requirement would not be violated easily when MD failures happen, we use a larger threshold γ to construct the initial tree, where $\gamma_0 < \gamma \leq 1$.

Denote $\Delta(j, s)$ as a binary variable indicates whether the quorum for pipeline l_s is met by MD$_j \in \mathcal{R}$. $\Delta(j, s)$ can be modeled as a flooring function

$$\Delta(j, s) = \left\lfloor \frac{\sum_i (\sum_{k \in \mathcal{K}_{i,j}} x_{i,j}^k) \cdot q(i, s) + (1 - \gamma) N_{l_s}}{N_{l_s}} \right\rfloor \tag{4}$$

$\Delta(j, s) = 1$ if at least γN_{l_s} MDs on l_s reporting to MD$_j$, and $\Delta(j, s) = 0$ otherwise.

B. Mathematical Formulation

The ITCP is a mixed integer linear programming formulation as follows:

$$\min \max_j \quad H_j \qquad\qquad \forall \text{MD}_j \in \mathcal{R} \tag{5}$$

$$\text{s.t.} \sum_j \sum_{k \in \mathcal{K}_{i,j}} x_{i,j}^k = 1 \qquad \forall \text{MD}_i \in \mathcal{M}, \text{MD}_j \in \mathcal{R} \tag{6}$$

$$x_{i,j}^k \leq x_{i',j}^l \qquad\qquad \text{if} \quad P_{i',j}^l \subset P_{i,j}^k \tag{7}$$

$$\sum_j \Delta(j, s) \geq 1 \qquad\qquad \forall \text{MD}_j \in \mathcal{R}, l_s \in \mathcal{L} \tag{8}$$

The intuition of these constraints can be interpreted as follows:

(6) Every MD is connected to one and only one tree.
(7) The selected paths should together form a forest of trees. This constraint ensures that if a path $\mathcal{P}_{i,j}^k$ is selected, its sub-path $\mathcal{P}_{i',j}^l$ must be selected. Therefore, the computed graph will be a tree.
(8) For every $l_s \in \mathcal{L}$, there exists an MD$_j \in \mathcal{R}$ meeting the quorum requirement of l_s.

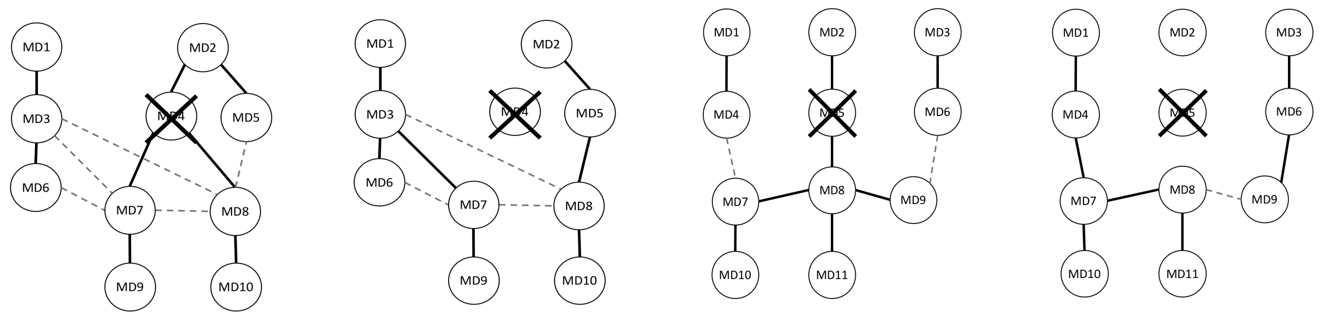
Although ITCP is an NP-hard MILP, we only need to compute the optimal multi-tree structure once for the initial construction in the CC, which should have enough computational resources. It is thus feasible to apply some optimal solver (i.e. GUROBI [20]) to develop the exact optimal solution. Note that when massive failure happens and the quorum requirement is thus violated, the CC will recompute the data collection forest for the MDs who are still alive. In the case that the CC cannot compute a feasible data collection structure, it indicates the surviving MDs no longer form a network with the desired quality in monitoring. The system administrator can then consider installing new MDs to restore the network connectivity.

V. Maintenance Phase: Resilient Protocol

After the forest is developed, the CC informs each root MD its own tree structure. The root can then inform its children to further establish the tree. The children inform their children and so on. MD$_i$ should keep the following connectivity and tree information:

1) $\mathcal{N}_i$: the set of MD$_i$'s neighbors that can directly talk to MD$_i$.
2) r_i: index of MD$_i$'s tree root.
3) p_i: index of MD$_i$'s parent.
4) $\mathcal{C}_i$: the set of MD$_i$'s children on the tree. $\mathcal{C}_i = \emptyset$ when MD$_i$ is a leaf node in the tree.
5) $\mathcal{T}_i$: The subtree structure rooted at MD$_i$. That is, MD$_i$ knows all its descendants on the tree. MD$_i$ also knows which pipelines they lie on. The height of $\mathcal{T}_i$ is denoted as $\mathcal{H}(\mathcal{T}_i)$.

Once all MDs know their tree neighbors (parent and children), they can report data through the tree according to the application requirement. In this work, we use the *fail-stop* failure model, the failed MDs will never recover. As each MD has only one data reporting path, it is very important to identify node failures as soon as possible so that an alternate reporting path can be established quickly. We assume the *HeartBeat Protocol* is used for failure detection. Every MD broadcasts its heart-beat to all its neighbors periodically. In this case, an MD can tell whether the tree parent has failed. If so, it should establish another tree path to report data. Details will be described in Sections V-A and V-B. The *HeartBeat Protocol* also allows a parent to detect whether a child has failed. The tree branch rooted at the failed child no longer belongs to the



(a) Identify an alternate path through non-tree neighbors

(b) Resultant forest after re-connected through neighbors

(c) Identify an alternate path through exploring descendants

(d) Resultant forest after seeking a path through descendants

Fig. 3: Examples of Tree Reconstruction Process.

tree. The parent should report this to the root. We will explain the details in Section V-C.

A. Find A New Parent From Neighbors.

When MD$_i$ finds the parent has failed, it would first consider to report data through its non-tree neighbors. Note that the non-tree neighbors of MD$_i$ refer to the set of MD$_i$'s neighbors who are not in the subtree $\mathcal{T}_i$. If a non-tree neighbor finds that it is possible (details will be described later) for MD$_i$ to join the tree it lies on, it acknowledges MD$_i$. MD$_i$ can then decide which neighbor it wants to be its new parent if several neighbors provide positive responses. We now describe the details of this process. We will explain how to handle the situation where no non-tree neighbor can take MD$_i$ as a child later.

1) **MD$_i$ sends a *join request* to all its non-tree neighbors.** In our mechanism, all non-tree neighbors would reply whether the join request can be accepted or not. Refer to the example network in Fig. 3a, there are 10 MDs in the figure represented by circles. Tree links are represented in solid lines and the dash lines denote some non-tree connectivities. There are two trees rooted at MD$_1$ and MD$_2$ in this example. Suppose MD$_4$ fails. Its children MD$_7$ and MD$_8$ detect the failure of the parent and send out *join request* to their non-tree neighbors. MD$_7$ sends a *join request* to MD$_3$, MD$_6$, and MD$_8$, while MD$_8$ sends the *join request* to MD$_3$, MD$_5$ and MD$_7$.
2) **MD$_j$ which receives the *join request* of MD$_i$ acquires information from its root to reply to the request.** In our protocol, we let MD$_i$ decide which tree to join. Therefore, when MD$_j$ receives a join request, it should reply positive as long as it is still connected to a tree. It should also provide the tree height information so that MD$_i$ can optimize the tree height in its decision. If MD$_j$ knows that it has also been disconnected, it should reply negative to the *join request* right away.
The tree root of MD$_j$ is MD$_{r_j}$. To acquire the height of $\mathcal{T}_{r_j}$, denoted as $\mathcal{H}(\mathcal{T}_{r_j})$, MD$_j$ sends the *root check* message to MD$_{r_j}$. Upon receiving the message, MD$_{r_j}$ replies height information $\mathcal{H}(\mathcal{T}_{r_j})$ and L_{j,r_j}, the path length between MD$_j$ and MD$_{r_j}$. The information allows MD$_i$ to compute the height of the tree if MD$_i$ joins $\mathcal{T}_{r_j}$. When MD$_j$ receives the information, it sends a positive reply to MD$_i$.
If the path from MD$_j$ to MD$_{r_j}$ is also broken, MD$_j$ would not receive a reply. We adopt the timeout mechanism for all messages that a reply is expected to avoid deadlocks. If MD$_j$ is no longer connected to a tree root, timeout would occur for the *root check* message, and it replies negative to the join request of MD$_i$.
In Fig. 3a, MD$_3$, MD$_5$, and MD$_7$ receive the *join request* message from MD$_8$. Since MD$_7$ has also known that its parent has failed and it is disconnected from the tree by the time that it receives the *join request* of MD$_8$, it sends a negative reply right away. MD$_3$ and MD$_5$, on the other hand, would send the *root check* messages to their corresponding tree roots. When MD$_5$'s *root check* is delivered to MD$_2$, MD$_2$ immediately replies the height information $\mathcal{H}(\mathcal{T}_2) = 1$ and $L_{5,2} = 1$. MD$_3$ gets the root response from MD$_1$ in a similar way. The *join request*s sent by MD$_7$ will be handled in a similar manner.
3) **MD$_i$ receives the replies from its non-tree neighbors and decides which tree to join.**
The non-tree neighbor MD$_j$ which replies positive becomes a candidate parent for MD$_i$ to join. MD$_i$ then selects a parent from these candidates. In order not to violate the quorum requirement, MD$_i$ should select a candidate that belongs to the same tree if possible. If there is no such candidate available or there are multiple candidates belonging to the same tree as MD$_i$, MD$_i$ computes the height of the tree for each candidate after MD$_i$ joins the tree. That is, if MD$_i$ selects the new parent to be MD$_j$, the longest branch from the tree root to any MD in $\mathcal{T}_i$ would be $L_{j,r_j} + \mathcal{H}(\mathcal{T}_i) + 1$, which may or may not increase the height of $\mathcal{T}_{r_j}$. Thus, the tree height increase upon MD$_i$ joining MD$_j$ is simply $\max\left(0, L_{j,r_j} + \mathcal{H}(\mathcal{T}_i) + 1 - \mathcal{H}(\mathcal{T}_{r_j})\right)$. MD$_i$ selects the tree with the minimum height increase.
Refer to Fig. 3a, MD$_7$ has two candidate parents, MD$_3$ and MD$_6$. They are both not connected to the original parent of MD$_7$. If MD$_7$'s subtree (MD$_7$ $-$ MD$_9$) joins at MD$_3$, the height of $\mathcal{T}_1$ will be increased by 1, which is smaller than the case where the subtree joins at MD$_6$. Therefore, MD$_7$ would select MD$_3$ as its new parent.

MD_8 has two candidates, MD_3 and MD_5. MD_5 belongs to the same tree that MD_8 is lying on. Therefore, MD_8 would join at MD_5. The final tree is illustrated in Fig 3b.

4) **MD_i informs the selected new parent MD_{j^*} and its descendants to update tree information.**
   MD_i must inform its descendants the new tree root they are now connected to. MD_i also tells MD_{j^*} that it has decided to be a child of MD_{j^*} and sends MD_{j^*} the sub-tree structure $\mathcal{T}_i$. The tree of MD_{j^*} is now changed. All nodes from MD_{j^*} to the root should be informed and update their sub-tree structure accordingly. When the tree root knows that there is a structural change, it should examine whether the change affects the quorum requirement of any pipe. The details will be discussed in Sec. V-C

B. Find A New Path Through the Descendants

In Sec V-A, we discussed the details about how to find a new parent from MD_i's non-tree neighbors. It is possible that all non-tree neighbors cannot help MD_i to connect back to the data collection structure. In this case, MD_i should explore to connect through its children. That is, if a child can identify an alternate tree path, MD_i can connect back to the tree as well. The following is the detailed procedure:

1) **MD_i tells its children to find new parents.**
   When MD_i found that it could not find a new parent from its non-tree neighbors, it will send a *find a new parent* message to all its children. Refer to Fig. 3c, suppose MD_5 fails. MD_8 does not have any non-tree neighbor. It thus tells its children (MD_7, MD_9 and MD_{11}) to find a new parent by themselves.
2) **MD_k, a child of MD_i, tries to find a new parent.**
   To identify a new parent, MD_k follows the procedure in Section V-A. There are two cases, MD_k could find a new parent and MD_k could not find a new parent.
   If a new parent is identified by MD_k and thus MD_k can reconnect to the data collection structure, MD_k will send *join invitation* message to invite its original parent. The *join invitation* message also contains the new tree height and tree root information of MD_k's data collection tree. MD_i can select the best former child as the new parent based on the information when multiple invitations are received.
   If no alternate parent can be identified by MD_k, MD_k will also report this issue to MD_i. MD_k will also tell its children to find a new parent by themselves by sending them a *find a new parent* message, which is the same process as MD_i tells its children. Note that this process can explore all of MD_i's descendants recursively.
   Refer to Fig. 3c. After MD_7, MD_9 and MD_{11} have received the *find a new parent* message from MD_8, they start to seek an alternate parent from their non-tree neighbors. MD_7 and MD_9 successfully identify their new parents (MD_4 and MD_6) and then send *join invitation* to MD_8. MD_{11}, on the other hand, cannot identify any alternate parent. It reports this issue to MD_8 and then tells its children to find a new parent on their own. Since MD_{11} has no children, MD_{11} then starts to wait for the join invitation from other MDs. MD_8 receives two invitations from MD_7 and MD_9. It will then select the new parent following the same rule described in Sec V-A(3). MD_7 is finally selected as smaller extra tree height will be introduced upon MD_8's joining.
   When MD_i confirms the *join invitation* from MD_k, it will directly establish the link to MD_k by passing its subtree structure $\mathcal{T}_i$ to MD_k. MD_k receives $\mathcal{T}_i$ and knows MD_i has accepted the invitation. MD_k will further relay the update message to the tree root to complete the join process of MD_i.
3) **MD_i helps its children if needed.**
   After MD_i has reconnected to the data collection structure, it will send *join invitation* to its children who have not sent *join invitation* to MD_i. This *join invitation* message also contains the tree height and tree root information. Because MD_i's child may also receive invitations from its children, the information can help it to select the best new parent. As shown in Fig. 3c, MD_{11} receives MD_8's *join invitation* and joins its tree. Thus the reconnection process is complete and the final topology is shown in Fig. 3d.

In the case when massive MDs fail, no descendant of MD_i can identify a new parent from its non-tree neighbors and the tree reconstruction thus fails. The reconstruction failure will always be known by MD_i's original tree root and reported to the CC. Details will be discussed in Sec. V-C. Our resilient protocol can also deal with the case when tree roots fail. The disconnected MDs can explore another tree to join using their connectivities as described in Sec. V-A and Sec. V-B. However, in the very rare case that all root MDs fail, no MD can contact with the CC anymore and the data data cannot be relayed to the CC. In this case, we have to restore the connectivity between the CC and some MDs, such as, extend the communication range of some non-root MDs so that they can talk to the CC directly. The details will be developed in our future work.

C. Structural Change Reporting

Tree structure changes can influence the quorum requirement of a pipe. That is, when a new subtree joins a tree, the tree may now satisfy the quorum requirement of more pipes. On the other hand, if a subtree is disconnected from a current tree, the quorum requirement of one or more pipes may be violated.

When a new sub-tree $\mathcal{T}$ joins the tree rooted at MD_r, MD_r should examine whether its tree now satisfies the quorum requirement of any new pipe that the nodes on $\mathcal{T}$ lie on. If so, it should inform the CC.

When an MD fails, the failure will be detected by its parent. The parent MD thus eliminates its subtree information and reports this failure to the tree root in a hop by hop manner. The tree root will then calculate whether this failure causes any quorum requirement violation for the pipes it is responsible for. When a violation happens, the tree root will then wait for some time in case the lost MDs may be reconnected back to the same tree. When the timeout occurs, the tree root will re-calculate the quorum requirement. If quorum violation still occurs (when many MDs are reconnected to another tree or reconstruction process fails), the tree root would report quorum change to the CC. The CC, knowing the quorum situation of all pipes and all root MDs, can decide whether there is a need to re-establish the whole data collection structure. The mechanism in Section IV can be used. If no solution can be found, it means the current network no longer supports the original services. The administrator should decide how to restore the network by installing new MDs or repairing failed ones.

Apart from the quorum change information, every root should also report its structural changes to CC every time a subtree join or is disconnected. CC will thus have a global knowledge of the forest real-time structure and can provide global solution to reconstruct the data collection forest when needed.

VI. Simulation

In this section, we model the refinery topology in a 3D space and inject random MD failures in each time slot. We evaluate the performance of our Refinery Resilient Protocol (RRP) on three metrics: (1) number of MDs that are still connected to the data structure, (2) quorum requirement of the pipes, and (3) height of the data collection forest. The three measurements will be taken after each time slot. We also compare RRP with two baseline protocols.

A. Topology Generation & Failure Injection

Instead of placing MDs randomly, we generate the MD topology following the intuition that MDs are usually placed around pipes to measure refineries' health condition and most pipes have a linear structure in the physical world. We assume the refinery as a $50m \times 50m \times 10m$ box. We first randomly generate M pipelines whose length ranges from $30m$ to $40m$ randomly. These pipes are parallel to either the x-axis or y-axis. We then place MDs along these pipelines spaced with same distance d. There are N MDs in total. Every MD in the same topology has the same short communication range r, where $r \geq d$. Optimal initial tree will be computed using the topology information.

To evaluate the performance of the resilient protocol. We introduce failures at discrete time. We divide the time line into T slots, and introduce k random independent MD failures in each time slot. The resilient protocol tries to get the affected MDs reconnected to the data collection forest in the same time slot. We take the three measurements at the end of each time slot to observe the network's condition with time going on.

Setting	N	M	γ_0	γ	r	d	k	T
I	594	54	50%	80%	6	3	2	30
II	309	36	50%	70%	8	4	1	50

TABLE I: Simulation settings

The parameter settings in our simulation are given in Table I. Setting I is a dense network with 594 MDs and 54 pipelines, while Setting II network contains 309 MDs and 36 pipelines. There are 30 time slots with two random failures occur in each time slot in Setting I,and there will be 60 MDs fail in the end, which is 10.10% of the total number of MDs. In Setting II, there are 50 time slots and one MD fails in each time slot, which causes 16.18% of total MDs fail at the end. We repeat these two simulation settings each for 100 times and present the average values and standard deviations in Fig. 4.

B. Two Baseline Protocols

We compare our protocol with two other protocols. All three protocols start with the same initial forest configuration which satisfies the quorum requirement with minimized forest height. We only measure the resilience behaviors against failures of the three protocols. The first one is Biconnected Tree Protocol (BTP) [15, 16]. BTP assigns every non-root MD a secondary parent from its uncle or cousin nodes which can lead a secondary path to the same tree root. When MD failure happens, the affected MDs try to connect to the original tree through their secondary parent. However, the MD who lost both its original and secondary parents will not get reconnected to the data collection trees again. The second baseline protocol is None Resilient Protocol (NRP). NRP is a data collection protocol without resilient property. When failure happens, NRP will not try to get the disconnected MDs reconnect to the data collection trees.

C. Simulation Results & Discussion

In Fig. 4a and Fig. 4d, we show the number of MDs that are still connected to the data collection trees after the continuous failures happen of the three protocols. It can be observed that the Refinery Resilient Protocol (RRP) keeps much more MDs connected in the network than BTP and NRP under both Settings I and II.

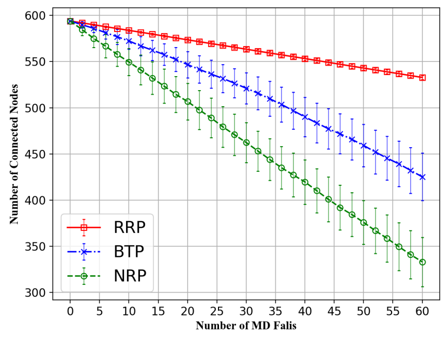
In Fig. 4b and Fig. 4e, we show the number of pipes that have at least γ_0 (50%) of MDs on the pipe connecting to the same tree when applying the above three protocols. As the figures show, RRP can keep nearly every pipe preserving this property through the whole time line, which is far better than BTP and NRP under both settings.

Fig. 4c and Fig. 4f show the tallest tree height while failures happen. As NRP has no resilient re-connection in the whole process, MD's failure will only prune the trees and never cause tree height increase. As RRP has a relatively complicated resilient process, it may explore every possible path to get the disconnected MDs back to the data collection structure and cause many tree structural changes. The results show that even the tree height would increase after re-construction, the increase is less than 1 in most situations, which is very small, or even still optimal. It is also worth noting that the decrease in height of BTP and NRP means some nodes have been disconnected as the original tree height should be minimal.

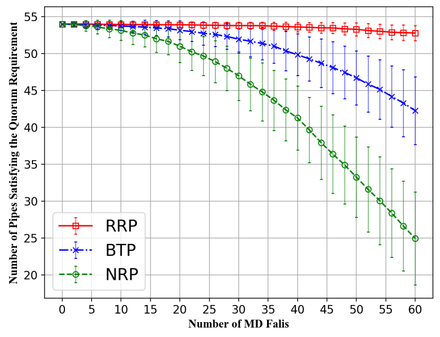
To conclude the observation from the above experiment, RRP performs much better than the other two baseline protocols to minimize the affected MDs and pipelines when continuous failures happen.

VII. Conclusion

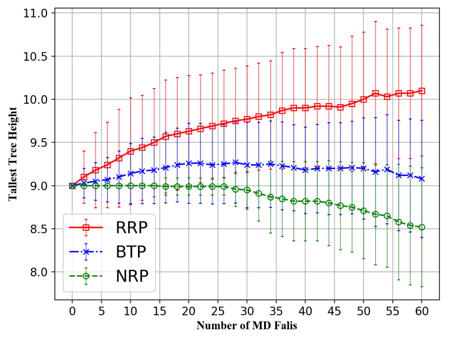
In this paper, we identify the unique feature of oil and gas refinery sensor networks of in-network processing and develop



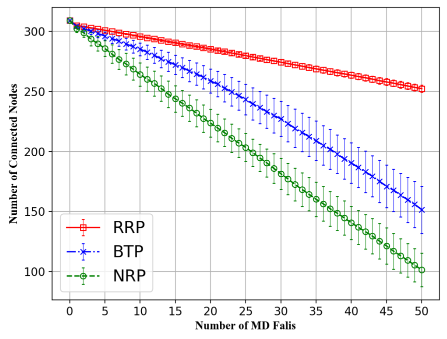
(a) Node Count (Setting I)



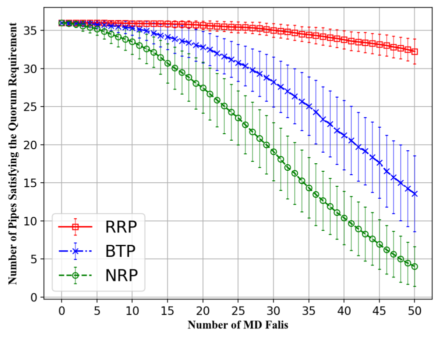
(b) Pipe Count (Setting I)



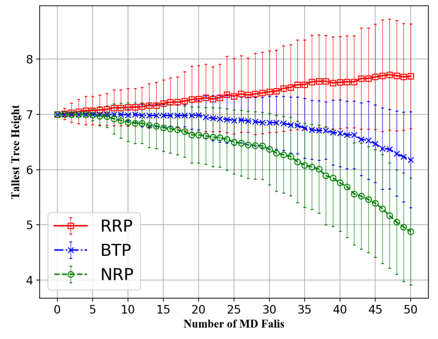
(c) Tallest Tree Height (Setting I)



(d) Node Count (Setting II)



(e) Pipe Count (Setting II)



(f) Tallest Tree Height (Setting II)

Fig. 4: Experimental Results

the quorum requirement. We then develop a resilient protocol to establish an efficient data collection structure that allows fast recovery upon device failures.

VIII. Acknowledgment

This material is based upon work supported by the Department of Energy under Award Number DE-OE0000780[1] and the University of Hong Kong Internal Grant Number 104003580.

References

[1] S. Savazzi, S. Guardiano, and U. Spagnolini, "Wireless sensor network modeling and deployment challenges in oil and gas refinery plants," *International Journal of Distributed Sensor Networks*, 2013.
[2] T. Arampatzis, J. Lygeros, and S. Manesis, "A survey of applications of wireless sensors and wireless sensor networks," in *IEEE International Symposium on, Mediterrean Conference on Control and Automation*, 2005.
[3] N. A. Pantazis, S. A. Nikolidakis, and D. D. Vergados, "Energy-efficient routing protocols in wireless sensor networks: A survey," *IEEE Communications surveys & tutorials*, 2013.
[4] Forbes. The impact of hurricanes harvey and irma on energy operations. [Online]. Available: https://www.forbes.com/sites/tortoiseinvest/2017/09/11/hurricanes-harvey-and-irmas-impact-on-energy-operations/#5c2e63d9341f
[5] O. D. Incel, A. Ghosh, B. Krishnamachari, and K. Chintalapudi, "Fast data collection in tree-based wireless sensor networks," *IEEE Transactions on Mobile computing (TMC)*, 2012.
[6] H. Jin, S. Uludag, K.-S. Lui, and K. Nahrstedt, "Secure data collection in constrained tree-based smart grid environments," in *IEEE SmartGridComm*, 2014.
[7] Y. K. Sia, H. G. Goh, S.-Y. Liew, and M.-L. Gan, "Spanning multi-tree algorithm for node and traffic balancing in multi-sink wireless sensor networks," in *IEEE FSKD*, 2015.
[8] P. Thulasiraman, S. Ramasubramanian, and M. Krunz, "Disjoint multipath routing to tow distinct drains in a multi-drain sensor network," in *IEEE INFOCOM*, 2007.
[9] 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," *IEEE/ACM Transactions on Networking (TON)*, 2015.
[10] D. Tulone and E. D. Demaine, "Revising quorum systems for energy conservation in sensor networks," in *IEEE International Conference on Wireless Algorithms, Systems and Applications (WASA)*, 2007.
[11] S. Lai, B. Ravindran, and H. Cho, "Heterogenous quorum-based wake-up scheduling in wireless sensor networks," *IEEE Transactions on Computers*, 2010.
[12] K. Kweon, H. Ghim, J. Hong, and H. Yoon, "Grid-based energy-efficient routing from multiple sources to multiple mobile sinks in wireless sensor networks," in *IEEE International symposium on wireless pervasive computing (ISWPC)*, 2009.
[13] J. Lee, J.-Y. Jang, E. Lee, S.-H. Kim, and M. Gerla, "Efficient sink location service for prolonging the network lifetime in wireless sensor networks," in *IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 2016.
[14] S. Han, X. Zhu, A. K. Mok, D. Chen, and M. Nixon, "Reliable and real-time communication in industrial wireless mesh networks," in *IEEE RTAS*, 2011.
[15] J. Kamto, L. Qian, W. Li, and Z. Han, "Biconnected tree for robust data collection in advanced metering infrastructure," in *IEEE WCNC*, 2015.
[16] J. Silber, S. Sahu, J. Sing, and Z. Liu, "Augmenting overlay trees for failure resiliency," in *IEEE GLOBECOM*, 2004.
[17] C. Sergiou, V. Vassiliou, C. Georgiou, C. Ioannou, N. Temene, and A. Paphitis, "Competition: Dynamic alternative path selection in wireless sensor networks," in *International Conference on Embedded Wireless Systems and Networks (EWSN)*, 2017.
[18] F. Group. Hart technology. [Online]. Available: https://fieldcommgroup.org/technologies/hart/hart-technology
[19] T. Liu, H. Guo, K.-S. Lui, H. Jin, and K. Nahrstedt. Resilient data collection in refinery sensor networks under large scale failures. [Online]. Available: https://www.ideals.illinois.edu/bitstream/handle/2142/97951/CREDC_tianyuan_klara.pdf?sequence=2&isAllowed=y
[20] Gurobi. Gurobi optimization - the best mathematical programming solver. [Online]. Available: http://www.gurobi.com/

[1]Disclaimer: The material was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.