Reliable and Real-time Communication in Industrial Wireless Mesh Networks

Song Han
University of Texas at Austin
shan@cs.utexas.edu

Deji Chen
Emerson Process Management
deji.chen@emerson.com

Aloysius K. Mok
University of Texas at Austin
mok@cs.utexas.edu

Mark Nixon
Emerson Process Management
mark.nixon@emerson.com

***Abstract*— Industrial wireless mesh networks are deployed in harsh and noisy environments for process measurement and control applications. Compared with wireless community networks, they have more stringent requirements on communication reliability and real-time performance. Missing or delaying of the process data by the network may severely degrade the overall control performance. In this paper, we abstract the primary reliability requirements in typical wireless industrial process control applications and define three types of reliable routing graphs for different communication purposes. We present efficient algorithms to construct them and describe the recovery mechanisms. Data link layer communication schedules between devices are further generated based on the routing graphs to achieve end-to-end real-time performance. We have built a complete WirelessHART communication system and integrated our solutions into its Network Manager. We have also conducted extensive simulations to evaluate the performance of our algorithms in large-scale networks. The experimental results show that our algorithms can achieve highly reliable routing, improved communication latency and stable real-time communication at the cost of modest overheads in device configuration.**

I. Introduction

Wireless process control has been increasingly recognized as an important technology in the field of industrial process management [1], [2], [3], [4], [5], [6], [7], [8]. Several industrial organizations such as ISA [9], HART [10] and ZigBee [11] have been actively pushing the application of wireless technologies in industrial automation. However, compared with wireless community networks, the industrial control environment is harsher and noisier and thus has more stringent requirements on reliable and real-time communication. Missing or delaying the process data may severely degrade the quality of control. The shifting wireless signal strength with time and location, the mobility of the control devices and power limitation due to battery usage make the problem even worse. Accordingly, network management techniques adapted for industrial wireless mesh are critical.

WirelessHART [12] is the first open wireless communication standard specifically designed for process measurement and control applications. Unlike the decentralized control adopted by wireless ad-hoc or peer-to-peer networks, it advocates explicit and centralized network management. The standard pushes the complexity of ensuring reliable and real-time communication to a centralized Network Manager, but it provides little guidance on how to meet the demanding design goals. This paper attempts to bridge the gap and is the result of a collaboration between Emerson Process Management and the University of Texas at Austin. Our previous efforts include implementing the full-blown WirelessHART protocol stack [3], [13] and developing the testing suite (Wi-HTest [14], [15]) for verifying the compliance and interoperatibility of WirelessHART devices. In this paper, we explore efficient approaches for forming a WirelessHART network, managing reliable graph routing, allocating network resources and constructing communication schedules.

In a typical WirelessHART network, each device has a designated sample rate to publish its process data to the Gateway through multi-hop transmissions. In the other direction, the Network Manager sends the control data back to the devices periodically. To help relay different types of data, the standard defines three types of communication graphs. The network shares one broadcast graph for propagating common control messages and one uplink graph for devices to publish process data. Each device further has a unique downlink graph from the Network Manager for forwarding specific control messages to itself. To achieve reliable communication, strict reliability requirements are enforced on these graphs. We abstract these requirements and introduce the concept of (k, m)-reliability in packet routing. Informally, the reliable graph requires that each intermediate device in the graph to be configured with k receive links from its parent and m send links to forward the packet to the destination. We shall present efficient algorithms to construct these reliable graphs and describe the recovery mechanisms. These algorithms are designed to maintain the maximum number of reliable nodes in the graphs while achieving good network latency.

Based on these routing graphs, the communication schedule in the mesh is further constructed. Our approach allows multiple devices to compete for the retry link to the same device, and split the traffic from one device among all its successors by reducing the bandwidth allocation on each of them. By designing the communication schedules on the successors so that their combination has the same communication pattern as the original device, the global communication schedule is then spliced into sub-schedules and distributed to the corresponding devices. These sub-schedules work together to guarantee that the periodic process/control data between devices and the Gateway can be forwarded through multi-hops in a timely manner.

We have conducted extensive simulations to evaluate the performance of the algorithms. We have also built a complete WirelessHART communication system including the Network Manager, Gateway, Access Points and a wireless mesh formed by multiple WirelessHART devices. We have integrated our algorithms into the Network Manager and are deploying the system to a manufacturing factory to collect sensor data from more than 200 devices.

The remainder of this paper is organized as follows. Section II briefly describes the WirelessHART network architecture. Section III presents the fundamental synchronization mechanism applied in WirelessHART networks. The details of reliable graph routing and communication schedule construction in WirelessHART are described in Section IV and Section V respectively. Section VI presents our design and implementation of the complete WirelessHART communication system. Section VII summarizes our simulation results. We conclude the paper and discuss the future works in Section VIII.

II. WirelessHART Architecture

Traditional wireless standards for office and manufacturing automation such as ZigBee [11] and Bluetooth [16] are not designed to meet the stringent timing and security requirements of industrial control. The WirelessHART standard is specifically targeted to solve these problems.

At the bottom of its communication stack, WirelessHART adopts IEEE 802.15.4-2006 [17] as the physical layer. On top of that, WirelessHART defines its own time-synchronized data link layer. Some notable features of WirelessHART data link layer include strict 10 ms timeslot, network-wide time synchronization, channel hopping, channel blacklisting, and industry-standard AES-128 ciphers and keys. The network layer supports self-organizing and self-healing mesh networking techniques and uses source routing and graph routing. In this way, messages can be routed around interferences and obstacles and greatly improve the network performance in noisy and harsh environments. WirelessHART distinguishes itself from other public standards by maintaining a central Network Manager. The Network Manager is responsible for maintaining up-to-date routes and communication schedules for the network, thus guaranteeing the reliable and real-time network communications.

Fig. 1 is a typical topology of a WirelessHART mesh network. All WirelessHART nodes support the basic mesh node functionalities, including routing capability. The basic node types of a WirelessHART network are:

- **Network Manager:** It is responsible for configuring the network, scheduling and managing communication among WirelessHART devices. It is implemented in software that resides in the Gateway or the Host.
- **Gateway:** It connects Host applications with the field devices. There is one Gateway per mesh.
- **Access Point:** It is attached to the Gateway and provides redundant paths between the wireless network and the Gateway.
- **Router:** It is deployed in the network to improve network coverage and connectivity.
- **Field Device:** It is attached to the process plant and collects data. It could be a sensor or an actuator.
- **Handheld:** It is a portable WirelessHART-enabled computer used to configure devices, run diagnostics, and perform calibrations.
- **Adapter:** It is a bridge device between the wireless mesh network and traditional wired HART devices.

III. Time Synchronization in WirelessHART Networks

WirelessHART is a TDMA-based network protocol and every communication in a WirelessHART network is time-synchronized. The basic time unit of communication activity is a fixed-length timeslot that is commonly shared by all network devices. The timeslot provides the time base for scheduling process data transmission. The duration of a timeslot defined in WirelessHART is 10 ms which is sufficient for sending or receiving one packet per channel and the accompanying acknowledgement, including guard-band times for network-wide synchronization. The specific timing requirement inside a WirelessHART timeslot is depicted in Figure 2. Precise time synchronization is critical to the operation of networks based on time division multiplexing. Since all communication happens in timeslots, the network devices must have the same notion of when each timeslot begins and ends, with minimal variation. Several mechanisms

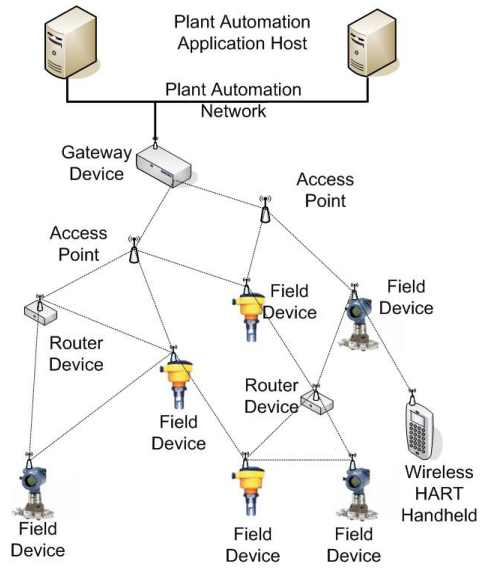

Fig. 1. A typical topology of a WirelessHART mesh network

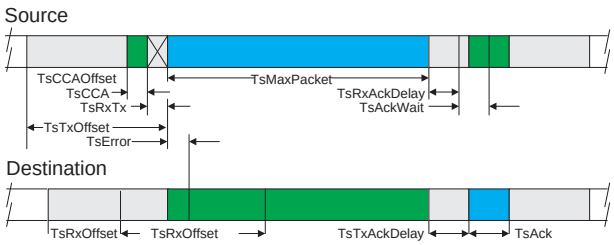

Fig. 2. Timing of a WirelessHART timeslot

are applied in WirelessHART for time synchronization. In a WirelessHART network, time propagates outwards from the Gateway.

When a new device joins a WirelessHART network initially, it has no idea what the current time is. For each incoming MAC layer packet, the device records T_a, the time when the packet's first bit arrives. Because of the strict timeslot structure, the device can derive the start of the next timeslot, T, from the packet's arrival time according to the following formula where TsTxOffset is the offset in the slot to start the preamble transmission.

$$T = T_a + 10\text{ms} - \text{TsTxOffset}$$

Synchronization happens not only in the device join process, but also during a node's normal operation. A receiving node always compares the start time of the incoming MAC layer packet and the expected arrival time measured on its own clock. The difference is the drift between their clocks. The receiver includes the difference in the time adjustment field of the corresponding ACK packet. Each node is designated a time source node. Whenever a node receives an ACK from its time source, it will adjust its clock based on the time adjustment field. If the sender is the time source of the receiver, the receiver adjusts its clock directly from the time difference value. Together, these adjustments provide the network-wide time synchronization in WirelessHART mesh networks.

IV. Reliable Graph Routing

In this section, we presents the details how we define and achieve the reliable routing in a typical wireless industrial mesh network like WirelessHART. We first describe the primary

routing approaches adopted in WirelessHART in Section IV-A. In Section IV-B, we summarize the notations we will use in the following discussion. Section IV-C abstracts the reliability requirements on packet routing, defines three types of communication graphs, and describe their properties. We discuss the difficulties in achieving complete reliable routing graphs in Section IV-D. The algorithmic details to construct these routing graphs with maximum number of reliable nodes and good latency are presented in Section IV-E, Section IV-G and Section IV-F.

A. Source Routing and Graph Routing

Two primary routing approaches are defined in the WirelessHART standard: graph routing and source routing. When using graph routing, a network device sends packets with a graph id in the network layer header along a path to the destination. All devices on the way to the destination must be pre-configured with graph information that specifies the neighbors to which the packets may be forwarded.

With source routing, to send a packet to its destination, the source includes in the network layer header an ordered list of devices through which the packet must travel. As the packet is routed, each routing device utilizes the next network device address from the packet header to determine the next hop to use. Since packets may go to a destination without explicit setup of intermediate devices, source routing requires knowledge of the complete network topology.

In industrial wireless networks, the process and control data must be transmitted in a reliable and timely manner. Any loss or delay in the data transmission may result in serious consequences. Since the source routing approach only establishes a fixed single path between the source and destination, any link/node failure will cut off their communication. For this reason, source routing is mostly used for diagnostics purposes. In this paper, we focus on the graph routing approach and investigate how to construct the reliable communication graphs among devices. Before we describe the detailed reliability requirements, we first summarize three types of routing graphs defined in a WirelessHART network. An example of these graphs is illustrated in Figure 3. A typical WirelessHART network topology includes a Gateway, multiple Access Points and many network devices. These devices are organized in a mesh and connected to the Gateway through the Access Points.

Uplink graph is a graph connecting all devices upward to the Gateway. It is used by the devices to propagate their process data periodically to the Gateway. Different devices may have different sample rates.

Broadcast graph is a graph connecting the Gateway downward to all devices. It is used by the Gateway to broadcast common configuration and control messages to the entire network.

Downlink graph is one per device. It is the graph from the Gateway to each individual device. The unicast messages from the Gateway and the Network Manager to each device traverse through this graph.

Based on these graphs, the Network Manager can further generate the corresponding sub-routes for each device. Only after the routes are constructed and downloaded to every device, can the network communication schedule be conducted, which we shall elaborate in Section V. When devices initially join into the network, they carry with them a list of neighbor entries including the received signal strength information. The Network Manager uses this information and the periodic health reports from the devices to construct and maintain the global network topology.

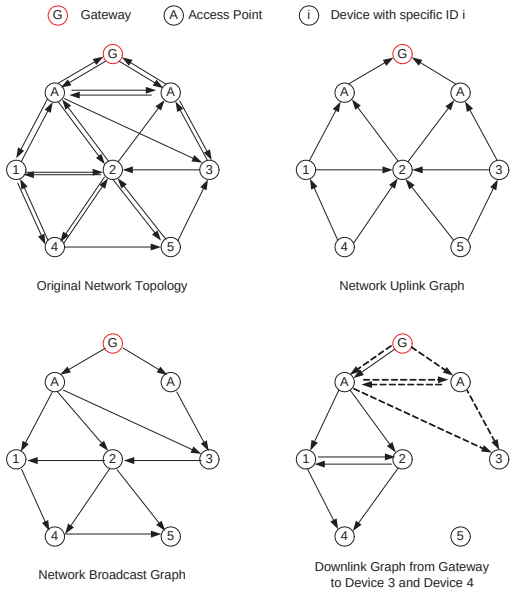

Fig. 3. An examples of different routing graphs

B. Notations

In this section, we summarize the notations to be used in the remainder of the paper. Given the original network topology $G(V, E)$, we use g to denote the Gateway, V_{AP} to denote the set of Access Points and V_D to denote the set of devices. We have $\{g\} \cup V_{AP} \cup V_D = V$. For each node $i \in V_D \cup V_{AP}$, we use δ_i^+ and δ_i^- to denote its outgoing and incoming degree. We use e_i^+ and e_i^- to denote its set of outgoing edges and incoming edges. Given a graph $G(V, E)$, we use $G_B(V_B, E_B)$ and $G_U(V_U, E_U)$ to represent its reliable broadcast graph and uplink graph respectively. The reliable downlink graph for each individual node $v \in V$ is denoted by $G_v(V_v, E_v)$.

C. Reliability Requirement and Reliable Graphs

Compared with wireless community networks, industrial wireless mesh networks have a much higher demand on the routing reliability to tolerate node and link failures. In this section, we summarize the reliability requirements defined by the WirelessHART standard and abstract them using the concept of (k, m)-*reliability* in packet routing. Notice that in this paper, we assume that the Gateway and Access Points are all connected through wire and reliable, so in the following of the paper, the reliability requirements will only apply to the wireless devices.

Definition IV.1: In a given directed graph $G(V, E)$, a node v satisfies the (k, m)-*reliability* if and only if v has no less than k incoming edges and no less than m outgoing edges. There is no constraint on the minimum number of incoming or outgoing edges on v if $k = 0$ or $m = 0$.

Based on this definition, we now give the definitions of the aforementioned reliable routing graphs and present their important properties.

Definition IV.2: In a given directed graph $G(V, E)$, a directed acyclic graph $G_B(V_B, E_B)$ where $V_B = V$ and $E_B \subseteq E$, is a reliable broadcast graph if the $(2, 0)$-reliability holds on every node in $V - \{g\} - V_{AP}$.

According to the definition, G_B requires that each wireless device has at least two parents in the graph from which it can receive the broadcast messages. This mechanism significantly

increases the chance for the broadcast messages to be propagated to the entire network. G_B has the following property.

Property IV.1: There are at least two paths from g to each device in G_B.

Proof: According to the definition of G_B, $\forall v, v \in V - \{g\} - V_{AP}$, it has two different parent nodes. There are two cases on v's parent node u. Case 1: u is an Access Point. In this case, it is obvious that there exists one path $g \rightarrow v \rightarrow u$. Case 2: u is a device. In this case, we conduct the same analysis on u and identify its parents. As G_B is acyclic, this process can be repeated and terminates when it reaches an Access Point. Thus there exists a path $g \rightarrow \cdots \rightarrow u \rightarrow v$. Because v has two different parent nodes, there are at least two paths from g to v in G_B. □

Different from the broadcast graph, the uplink graph is used by the devices to forward their process data to the Gateway with a certain fixed sample rate. An uplink graph is considered reliable if and only if for each device in the network except the Access Points, it has two children to forward its packet to the Gateway. In case when the communication between the device and one of its children is disabled, the process data can still be delivered to the Gateway through the alternative child.

Definition IV.3: In a given directed graph $G(V, E)$, a directed acyclic graph $G_U(V_U, E_U)$ where $V_U = V$ and $E_U \subseteq E$, is a reliable uplink graph if the $(0, 2)$-reliability holds on every node in $V - \{g\} - V_{AP}$.

Property IV.2: There are at least two paths from each device to g in G_U.

Proof: The proof is similar to the proof for Property IV.1.□

Property IV.3: G_B has no less than 2 Access Points.

Proof: Assume that there is only one Access Point p in either G_B. Let v be the node having a direct incoming edge from p in G_B. As p is the only Access Point, node u, the other parent node of v in G_B is a device. We repeat this analysis on u and we have a similar conclusion that one of u's parent node in G_B is still a device. This process can be continued infinitely and form a cycle because the number of devices in the network is finite. This is a contradiction with the definition of G_B. Our assumption is incorrect and G_B has no less than 2 Access Points. □

Property IV.4: G_U has no less than 2 Access Points.

Proof: The proof is similar to the proof for Property IV.3 □

Each WirelessHART network has only one broadcast graph for broadcasting message and one uplink graph for forwarding process data. To support the transmission of configuration and control messages from the Network Manager to a specific device, for example, to modify a device's communication tables, a unique downlink graph for each device is needed.

Definition IV.4: In a given directed graph $G(V, E)$, a directed graph $G_v(V_v, E_v)$ where $V_v \subseteq V$ and $E_v \subseteq E$, is a reliable downlink graph from g to v if 1) v is the only sink and g is the only source in G_v; 2) the $(0, 2)$-reliability holds on every node in $V_v - \{g\} - v$ and 3) there is only one cycle C of length 2 in G_v, and each node on the cycle has direct edges to v.

The definition of G_v is more complicated compared with G_B and G_U because it only involves part of the nodes in the original graph and it contains a cycle.

Property IV.5: $\forall v, v \in V - \{g\} - V_{AP}$, there exists at least one directed cycle in $G_v(V_v, E_v)$.

Proof: Assume that there is no cycle in G_v. Consider a node u who has a direct edge to v in G_v. u's other children are not v. Repeat this analysis on one of u's non-v children, it must also have a non-v child. This process can be repeated infinitely and finally form a cycle. This is a contradiction against our assumption. □

Property IV.5 states the existence of directed cycles in G_v. However, to guarantee the prompt delivery of the configuration and control messages, we must avoid arbitrary cycles in G_v which will generate infinite loops in packet forwarding. The definition IV.4 provides the guarantee by restricting the length of the cycle to 2 and requires that every node on the cycle must have a direct edge to the destination. Once the packet reaches such nodes, it will be directly forwarded to the sink and thus avoid the potential loop and unnecessary transmission delay.

D. Difficulties in Achieving Complete Reliable Graphs

The major barrier to construct reliable routing graphs is the underlying network connectivity. Better network connectivity will obviously lead to a higher chance for constructing completely reliable graphs. In this section, we evaluate the relationship between the network connectivity and the success ratio to construct these reliable graphs. In our experiments, we vary the network connectivity by changing the edge success probability p and the network density d. We assume open field, line-of-sight experimental scenarios. The simulation area is 450 m × 450 m, and the nominal communication distance of the device is 100 meters with a 0 dBm transmitter. To model the network connectivity, we assume that there is no edge between two nodes if their distance is larger than the communication distance. Otherwise, the edge exists with the probability p.

Figure. 4 illustrates the effect of varying the edge success probability on the success ratio of constructing reliable graphs. We observe that with 150 devices in the simulation, the success ratio drops quickly along with the decrease of p. When p is 0.8, the success ratio is around 80% for downlink graphs and above 95% for both G_B and G_U. However, when p drops to 0.5, we can barely construct reliable downlink graphs and the success ratios for constructing complete G_B and G_U are around 40%.

Under the same experimental settings, Figure. 5 shows the percentage of reliable nodes in the incomplete reliable graphs. We observe that the percentage of reliable nodes in incomplete uplink and broadcast graphs are always above 95% and this percentage for incomplete downlink graph is also larger than 75% even when the edge success probability drops to 0.5. Figure. 6 further evaluates the impact of the network density on the success ratio. We vary the size of the network from 75 to 150 and fix the edge success probability to 0.8. The results show that the network density has a great effect on network connectivity, and lower network density will lead to poor success ratio.

Based on these experimental results, we conclude that the success ratio for constructing reliable routing graphs is tightly related to the underlying network connectivity, and under some scenarios, it is impossible to achieve the complete reliable graphs. For this reason, we shall allow here violations of the reliability requirements in the routing graphs and focus on designing algorithms to construct routing graphs with the maximum number of reliable nodes. In the following of the paper, we shall still use G_B, G_U and U_v to represent the constructed broadcast graph,

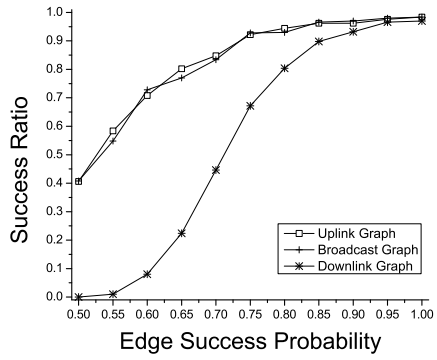

Fig. 4. Success ratio vs. Edge success probability

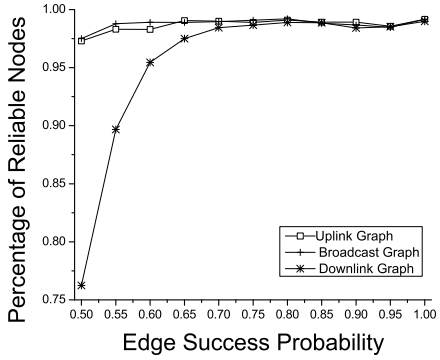

Fig. 5. Percentage of reliable nodes

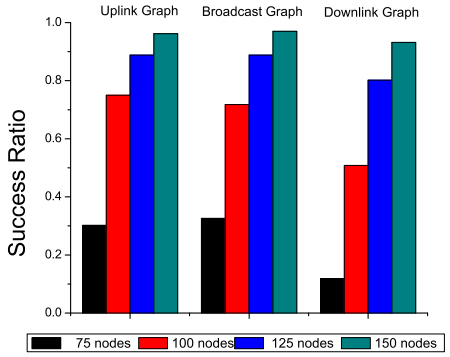

Fig. 6. Success ratio vs. Network density

uplink graph and downlink graph for node v even though they may not be completely reliable.

E. Constructing Reliable Broadcast Graph

In a broadcast graph, we say that a node i is reliable if and only if $\delta_i^- \geq 2$ where δ_i^- is the number of its incoming edges. Let $S_B = \{i \,|\, \delta_i^- \geq 2, i \in V\}$. We want to maximize $|S_B|$ when we construct the reliable broadcast graph G_B. Furthermore, to reduce the energy consumption in propagating broadcast messages to the whole network and to improve network latency, we also hope to minimize the average number of hops from the Network Manager to each node. For node i, we denote this average number of hops by $\bar{h}_i$ and use P_i to represent the set of its parent nodes in G_B. We have:

$$\bar{h}_i = \frac{\sum_{k \in P_i} \bar{h}_k}{|P_i|} + 1 \tag{1}$$

In this section, we present a greedy algorithm (Alg. 1) to achieve these two goals in constructing G_B. We first construct a broadcast tree by applying breadth-first search on the original graph $G(V, E)$ (Line 2). After removing all edges in the broadcast tree from G (Line 7), we incrementally add edges from G to G_B to improve the reliability.

We maintain a set S to keep record of the explored nodes in the algorithm and S is initialized as $\{g\} \cup V_{AP}$ (Line 13). We incrementally select one node v from $V_B - S$ and add it to S. In each loop, we identify S' from $V_B - S$, the set of nodes with at least one edge from S (Line 15). For each node v in S', we sort its incoming edges from S in G according to their average number of hops to the Gateway in ascending order (Line 18). We choose the first edge and update $\bar{h}_v$ according to Eq. 1 (Line 20). We also calculate n_v, the number of v's outgoing edges to $V_B - S$ in G (Line 21). In each round, we choose the node in S' with the minimum $\bar{h}_v$. We add it to S and add the corresponding edge to G_B. Ties are broken using n_v, because node with larger n_v can increase the chance to find reliable nodes in the next round (Line 23). This process continues until either all nodes in V_B are explored or no more reliable node can be identified in $V_B - S$. Notice that if the original network topology is disconnected, an error will be reported (Line 10). This will trigger the Network Manager to execute the appropriate recovery actions.

F. Constructing Reliable Uplink Graph

The construction of reliable uplink graph $G_U(V_U, E_U)$ is similar to that of $G_B(V_B, E_B)$. Essentially, we only need to reverse all edges in the original network topology $G(V, E)$, construct G_B and then reverse all its edges back. We define $G^R(V, E^R)$ to be the reversed graph of $G(V, E)$, and the greedy algorithm to construct $G_U(V_U, E_U)$ is summarized in Alg. 2.

Alg 1 Constructing Reliable Broadcast Graph $G_B(V_B, E_B)$

```
1: // G(V, E) is the original network topology
2: G_B(V_B, E_B) = BFS(G)
3: if V_B = V then
4:     for all node v ∈ V_B do
5:         Initialize h̄_v as v's height in G_B
6:     end for
7:     Remove E_B from G(V, E)
8: else
9:     // the underlying network topology is disconnected
10:    return FAIL;
11: end if
12:
13: Let S be the set of explored nodes and initially S = g ∪ V_AP
14: while S ≠ V_B do
15:    Identify S' ⊆ V_B − S: ∀v ∈ S', v has at least one edge from S
16:    if S' ≠ ∅ then
17:        for all node v ∈ S' do
18:            Sort its edges e_{u,v} from S in G according to h̄_u
19:            Choose the edge e_{u,v} with minimum h̄_u
20:            h̄_v = (h̄_v − 1 + h̄_u)/2 + 1
21:            Calculate n_v, the number of v's out edges to V_B − S in G
22:        end for
23:        Choose the node v with minimum h̄_v, break tie using n_v
24:        Add v to S and add e_{u,v} to E_B
25:    else
26:        break;
27:    end if
28: end while
29: return SUCCESS;
```

G. Constructing Reliable Downlink Graph

The construction of the reliable downlink graph $G_v(V_v, E_v)$ for a given node v in $G(V, E)$, is different from the constructions of G_B and G_U. It only involves part of the nodes in $G(V, E)$ and it is more complex because of the existence of cycles as shown in Property IV.5. Furthermore, according to Definition IV.4, we only want to have exact one cycle in G_v with length of 2 and restrict it to be between the two parents of v. Our optimization goals in constructing G_v are similar to that of G_B and G_U. We hope to maximize the number of nodes in the network to have reliable downlink graphs and for each downlink graph, we want to minimize its average number of hops from the Gateway.

Alg. 3 summarizes the framework of our approach. In the algorithm, given the original graph G, we construct the reliable downlink graph for each node in the network. For the Access Point, its downlink graph consists of the Gateway g, itself and

Alg 2 Constructing Reliable Uplink Graph $G_U(V_U, E_U)$

1: // $G(V, E)$ is the original graph, $G^R(V, E^R)$ is the reversed graph
2: Construct $G^R(V, E^R)$
3: Construct $G_B(V_B, E_B)$ from $G^R(V, E^R)$ by applying Alg. 1
4:
5: **if** $V_B = V$ **then**
6: // Constructing G_U by reversing all edges in G_B
7: $G_U(V_U, E_U) = G_B^R(V_B, E_B^R)$
8: **else**
9: // the network topology is disconnected
10: return FAIL;
11: **end if**
12: return SUCCESS;

Alg 3 Constructing Reliable Downlink Graphs in $G(V, E)$

1: Let S be the set of nodes with downlink graphs constructed
2: Initially $S = g \cup V_{AP}$ and $G_g = (\{g\}, \varnothing)$
3: Initially for each AP i in S, set $G_i = (\{g \cup i\}, \{e_{g,i}\})$
4:
5: **while** $S \neq V$ **do**
6: Identify $S' \subseteq V - S$: $\forall v \in S'$, v has at least two edges from S
7: // S_r is the reliable node set in S', initially $S_r = \varnothing$
8: **if** $S' \neq \varnothing$ **then**
9: **for all** node $v \in S'$ **do**
10: **for all** edge pair $(e_{u_1,v}, e_{u_2,v})$ from S **do**
11: **if** C 1 $\wedge$ C 2 $\wedge$ C 3 **then**
12: $S_r = S_r \cup \{v\}$
13: **end if**
14: $\bar{h}_{u_1,u_2} = (\bar{h}_{u_1} + \bar{h}_{u_2})/2$
15: **end for**
16: Choose the edge pair $(e_{u_1,v}, e_{u_2,v})$ with minimum $\bar{h}_{u_1,u_2}$
17: $\bar{h}_v = \bar{h}_{u_1,u_2} + 1$
18: **end for**
19: **if** $S_r \neq \varnothing$ **then**
20: Choose node v in S_r with minimum $\bar{h}_v$ and add it to S
21: **else**
22: Choose node v in S' with minimum $\bar{h}_v$ and add it to S
23: **end if**
24: // construct G_v: $\bar{h}_{u_1,u_2}$ is the min among all edge pairs to v
25: **ConstructDG**(G, G_{u_1}, G_{u_2}, v);
26: **else**
27: Identify $S'' \subseteq V - S$ and $\forall v \in S''$, v has one edge $e_{u,v}$ from S
28: **if** $S'' \neq \varnothing$ **then**
29: **for all** node $v \in S''$ **do**
30: $\bar{h}_v = \bar{h}_u + 1$
31: Calculate n_v, the number of its outgoing edges to $V - S$
32: **end for**
33: Add node v to S with maximum n_v, break tie using $\bar{h}_v$
34: **ConstructDG**(G, G_{u_1}, null, v);
35: **else**
36: return FAIL;
37: **end if**
38: **end if**
39: **end while**
40: return SUCCESS;

the edge from g to itself. We maintain S, a set of nodes whose reliable downlink graphs have already been constructed (Line 1). We incrementally identify an eligible node v in $V-S$ to construct G_v where three constraints are applied and v has the minimum $\bar{h}_v$ as calculated in Line 17. Constraint C1 and C2 are to satisfy the reliability requirements in G_v and Constraint C3 is to make sure that we can remove the internal cycles in the constructed G_v. If such an eligible node cannot be identified, we will instead choose the node that has two parents from S with the minimum average latency to the Gateway (Line 20). If every node in $V - S$ only has one parent from S, we choose the one with the minimum average latency (Line 27 - 37).

C1: v has at least two incoming edges from u_1 and u_2 in S.
C2: u_1 and u_2 form a cycle.
C3: There exists at least one edge in G from the cycle in G_{u_1} to u_2 and one edge from the cycle in G_{u_2} to u_1.

Alg. 4 describes how we construct G_v based on its parents' reliable downlink graphs G_{u_1} and G_{u_2}. In the algorithm, G_v is constructed by first merging G_{u_1}, G_{u_2}, v and edges among u_1, u_2 and v together (Line 4). We maintain S, the set of explored nodes in G_v and initialize S as $\{g, v, u_1, u_2\}$. We construct G_v in a bottom-up manner by incrementally selecting a node $i \in V_v - S$ which has two outgoing edges to S in G and has the minimum $\bar{h}_i$ (Line 6-30). This process continues until either all nodes in V_v are explored or V_{AP} has two outgoing edges to S (Line 7 - 10). Finally, we remove all nodes in $V_v - S$ and their corresponding edges from G_v (Line 32 - 34). If there is no node available to have two outgoing edges to S in G, we choose the node with the minimum $\bar{h}_i$ (Line 20 - 29).

H. Maintaining Reliable Routing Graphs under Churns

The algorithms presented in the previous subsections construct the reliable routing graphs in ideal scenarios where network devices join into the network sequentially and work properly after the join process. Although the industrial wireless mesh is usually quite stable after deployment, network devices may experience various failures and need to be reset in the real world. Wireless links can also be blocked by interference and become temporarily or permanently unavailable. All these scenarios require the Network Manager to recover the routing graphs to maintain the reliability requirements. Furthermore, corresponding adjustments on the communication schedules are also necessary along with these routing graph modifications.

In WirelessHART networks, network abnormalities and statistics are reported to the Network Manager through a set of network maintenance commands. These commands are summarized in Table I. Command 779 summaries the communication statistics of a specific device; Command 780 and 787 report a device's neighbor signal strengths; Command 788, 789 and 790 are triggered once a path failure or routing failure is detected in the network. These commands are carried in normal messages and published to the Network Manager. Based on these information, the Network Manager will update the network topology, adjust the routing graphs and communication schedules if necessary to reach a good balance between the reliability and recovery cost.

Our current heuristics to recover the reliable broadcast graph consists of two steps. We first reconstruct the broadcast tree by incrementally attaching a disconnected node to the tree as close to the Gateway as possible. In the second step, we repeat Phase 2 in Alg. 1 (Line 13-28). The only difference here is that to minimize the recovery cost, when we identify a node $v \in S'$ with already two incoming edges from S in G_B, we directly add it to S without comparing with other candidates. The mechanism to reconstruct the reliable uplink graph is similar. Designing efficient algorithms for reconstructing reliable downlink graphs is more challenging and will be addressed in our future works.

V. Communication Schedule and Channel Management

Many wireless industrial process control applications take the approach that devices specify their requirements in communication bandwidth and the Network Manager allocates the necessary resources such as timeslots, to maintain the periodic sensing-

Alg 4 ConstructDG ($G(V,E)$, $G_{u_1}(V_{u_1},E_{u_1})$, $G_{u_2}(V_{u_2},E_{u_2})$, v)

Let S contain explored nodes in $G_v(V_v,E_v)$: $S=\{g,v,u_1,u_2\}$

// Construct G_v by merging G_{u_1}, G_{u_2}, v, and edges among v, u_1, u_2
$G_v(V_v,E_v)=G_v(V_{u_1}\cup V_{u_2}\cup\{v\},E_{u_1}\cup E_{u_2}\cup\{e_{u_1,v},e_{u_2,v},e_{u_1,u_2},e_{u_2,u_1}\})$

while $S\neq V_v$ **do**
  if V_{AP} has two outgoing edges to S in G **then**
    $S=S\cup V_{AP}$
    break;
  end if
  for all node $i\in V_v-S$ **do**
    Sort i's out edges to S in G in descending order of $\bar{h}_i$
  end for

  Identify $S'\subseteq V_v-S$: $\forall v\in S'$, v has at least two edges to S
  if $S'\neq\emptyset$ **then**
    Choose node i with minimum $\bar{h}_i$
    Add first two edges from i to S in G to G_v if they don't exist
    Remove all other edges from E_v if they exist and add i to S
  else
    Identify $S''\subseteq V_v-S$: $\forall v\in S''$, v has one edge to S
    if $S''\neq\emptyset$ **then**
      Choose node i with minimum $\bar{h}_i$
      Add the edge from i to S in G to G_v if it doesn't exist
      add i to S
    else
      return FAIL;
    end if
  end if
end while

for all node $i\in V_v-S$ **do**
  Remove i from V_v and corresponding edges from E_v
end for
return SUCCESS;

control loop between the Network Manger and devices. In the sensing phase, the devices publish their process data to the Gateway through the uplink graph based on their specific sample rates; In the control phase, the Network Manager generates the corresponding control messages periodically. These control messages are sent back to each individual device on its downlink graph. The Network Manager maintains a global communication schedule for transmitting these process and control data and distributes the sub-schedule to each effected device.

The construction of the communication schedule is subject to several practical constraints in WirelessHART networks:

- The maximum number of concurrent active channels is 16.
- Each device can only be scheduled to TX/RX once in a slot.
- Multiple devices can compete to transmit to the same device simultaneously (in shared timeslot).
- On a multi-hop path, early hops must be scheduled first.
- The supported sample rates are defined as 2^n sec ($-2\leq n\leq 9$) from 250 ms (2^{-2} sec) to 8 min and 32 sec (2^9 sec).

Our design philosophy for constructing the communication schedule is to spread out the channel usage in the network as much as possible and to apply the fastest sample rate first policy (*FRF*) to schedule the devices' periodic publishing and control data.

We use the concept of superframe to group a sequence of consecutive timeslots and represent the communication pattern for a given sample rate. For example, if the sample rate is 1 min and 4 sec and the timeslot is 10 ms by default, then there are 6400 timeslots in this superframe. We define two types of superframes: data superframe and management superframe. The data superframe is used to support data transmissions between the devices and the Gateway while the management superframe is used to support exchanging network management messages. The number of data superframes is decided by the number of different sample rates existing in the network. Notice that there can be multiple devices having the same sample rate, thus a data superframe represents the periodic behavior of multiple devices sharing the same sample rate.

Command	Functionality
Command 779	Report device communication statistics
Command 780	Report neighbor health list
Command 787	Report neighbor signal levels
Command 788	Path down alarm
Command 789	Source route failure alarm
Command 790	Graph route failure alarm

TABLE I

Summary of network maintenance commands

We maintain a global matrix $\mathcal{M}$ to keep track of the current slot/channel usages in the network. Each entry in the matrix, $\mathcal{M}_{i,j}$ represents the slot usage at timeslot i on channel j and its type is one of the following four categories: unused, exclusive, shared and reserved. A unused entry can be allocated to any pair of devices if there is no communication conflict. An exclusive entry is one occupied by two devices for dedicated communication and cannot be used again. Reserved entries are managed by the Gateway or the Network Manager for maintenance purposes. Finally a shared entry allows multiple devices to compete for transmitting to the same device simultaneously. In our implementation, we allow 5 simultaneous transmissions conducted on the same shared timeslot. We also maintain several other important data structures for constructing the communication schedule. These include one data superframe $\mathcal{F}_i$ per sample rate r_i and the management superframe $\mathcal{F}_m$. Here we use l_i to denote the length of $\mathcal{F}_i$. For each node v, we maintain a schedule $\mathcal{S}_v$ to record its own slot and channel usage. The length of $\mathcal{M}$ and $\mathcal{S}_v$ are both equal to the maximum length among the existing superframes. These schedules will be finally installed back to the devices to achieve reliable real-time communication.

Alg 5 Constructing Data Communication Schedule

Sort device sample rates in ascending order: $r_1<r_2<\ldots<r_k$.
Identify the set of nodes with each sample rate: $N_1,N_2,\ldots,N_k$.
Initialize the schedule for each node as $\emptyset$

for all r_i from r_1 to r_k **do**
  Generate the data superframe $\mathcal{F}_i$
  for all node $v\in N_i$ **do**
    // Schedule publishing data and their retry
    ScheduleLinks(v, g, G_U, $\mathcal{F}_i$, 0, Exclusive);
    ScheduleLinks(v, g, G_U, $\mathcal{F}_i$, $\frac{l_i}{4}$, Shared);

    // Schedule control data and their retry
    ScheduleLinks(g, v, G_v, $\mathcal{F}_i$, $\frac{l_i}{2}$, Exclusive);
    ScheduleLinks(g, v, G_v, $\mathcal{F}_i$, $\frac{3l_i}{4}$, Shared);

    if all link assignments are successfully **then**
      continue;
    else
      // Defer bandwidth request from node v
      return FAIL;
    end if
  end for
end for
return SUCCESS;

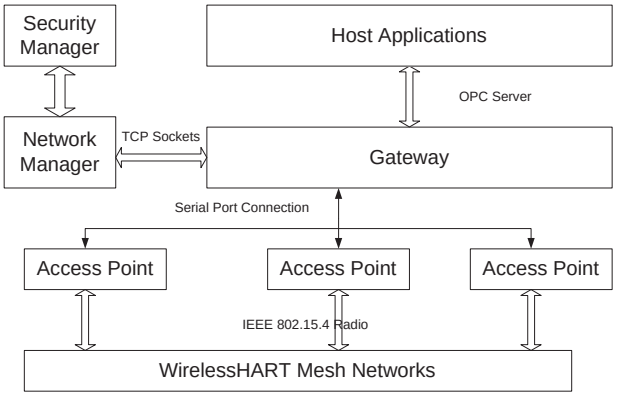

Fig. 7. Architecture of the complete WirelessHART communication system

We present our framework of constructing the data communication schedule in Alg. 5. The construction of the management schedule follows the same approach and is omitted for lack of space. In the algorithm, we apply the fastest sample rate first policy (*FRF*) in scheduling the data transmission. The construction is based on the reliable graphs we introduced in Section IV. For each device v, in its sensing phase, it allocates the primary and retry links along the uplink graph G_U to the Gateway (Line 9 - 10). In the control phase, the Gateway sends the control information back and allocates the primary and the retry links along the downlink graph G_v (Line 13 - 14). The ScheduleLinks(u, v, G, $\mathcal{F}$, t, o) function is summarized in Alg. 6. It allocates every link on the paths from u to v on graph G one by one in a depth-first manner. It allocates the earliest available timeslot t_i from t for each link and updates $\mathcal{M}$, $\mathcal{F}$ and each effected node's schedule accordingly. If we cannot find a slot in $[t, l_{\mathcal{F}}]$ to accommodate all the allocations, the Network Manager will defer the bandwidth request from the corresponding device until enough bandwidth resources are available (Line 19 - 20 in Alg. 5).

Notice that a device v is typically multi-hop away from the Gateway, and it has multiple paths to the Gateway due to the property of reliable graph routing. However, if we allocate the required communication bandwidth for device v on each hop along all its paths to the Gateway, most of the allocated links will be wasted because in each end-to-end transmission, only one path will be picked. This will severely degrade the schedulability of the network schedule. To address this problem, as shown in Alg. 6 (Line 17 - 33), when the device has two successors to forward the messages, we reduce the transmission rate between v and each of its successor to half of the original sample rate, and schedule the links on the corresponding superframe $\mathcal{F}'(l_{\mathcal{F}'} = 2 \cdot l_{\mathcal{F}})$. We determine the timeslot offset of these links in $\mathcal{F}'$ to make sure that their combinations will form a communication pattern the same as the original sample rate.

VI. System Implementation

We have built a complete WirelessHART communication system to verify the correctness and efficiency of our network management techniques. We are deploying the system to a manufacturing factory, collecting sensor data from more than 200 devices. Figure 7 depicts the general architecture of our system which has four major components: the WirelessHART mesh network, Gateway, Access Point and Network Manager. These four components are shown in Figure 8, and their details will be presented in the following sections.

Alg 6 ScheduleLinks(u, v, G, $\mathcal{F}$, t, o)

```
1: // u and v are the source and destination of the communication
2: // G is the routing graph and F is the superframe
3: // t is the earliest slot to be allocated and o is the link option
4:
5: Identify data superframe F' with l_F' = 2 l_F
6: for all node i ∈ Successor(u) do
7:     Identify the schedule S_u and S_i for node u and i
8:     if i is the only successor of u then
9:         Identify the earliest slot from t with a channel c to:
10:        Allocate entries M_{k·l_F+t_i,c}(k = 0, 1, ...) on M
11:        Allocate the slots k · l_F + t_i on S_u and S_i
12:        Allocate slot t_i on F
13:
14:        if All allocations are successful then
15:            ScheduleLink(i, v, G, F, t_i, o);
16:        end if
17:    else
18:        if i is the first successor then
19:            Identify the earliest slot from t with a channel c to:
20:            Allocate entries M_{k·l_F'+t_i,c} on M
21:            Allocate slots k · l_F' + t_i on S_u and S_i
22:            Allocate slot t_i on F'
23:        else
24:            Identify the earliest slot from t in M with a channel c to:
25:            Allocate entries M_{k·l_F'+l_F+t_i,c} on M
26:            Allocate slots k · l_F' + l_F + t_i on S_u and S_i
27:            Allocate slot l_F + t_i on F'
28:        end if
29:
30:        if All allocations are successful then
31:            ScheduleLink(i, v, G, F', t_i, o);
32:        end if
33:    end if
34:    if No feasible allocations available then
35:        return FAIL;
36:    end if
37: end for
38: return SUCCESS;
```

A. WirelessHART Devices

Our WirelessHART mesh network is formed by two types of devices. Rosemount [18] sensors and the devices that are developed by ourselves [3]. All these devices comply to the WirelessHART standards, thus there is no problem for them to interoperate with each other. These devices join into the network through the normal join procedure sequentially. For the details of the join procedure, readers are referred to [14]. The Network Manager organizes these devices into a multi-hop reliable mesh and configure them with the corresponding routing information and communication schedules. Once the devices are correctly configured, the Network Manager/Gateway and devices can begin to exchange management and data messages.

B. Gateway Design

The Gateway works as a server responsible for communicating with the Network Manager, processing the requests from the Host applications, collecting and maintaining cached data from all devices in the network. The overall architecture of the Gateway is illustrated in Figure 9 which is consisted of the following major components:

Physical Connections: The Gateway provides one serial port connection to each attached Access Point. A mapping between the Access Point address and the corresponding serial port number is maintained for forwarding the messages to the designated Access Point. The Gateway talks with the Network Manager

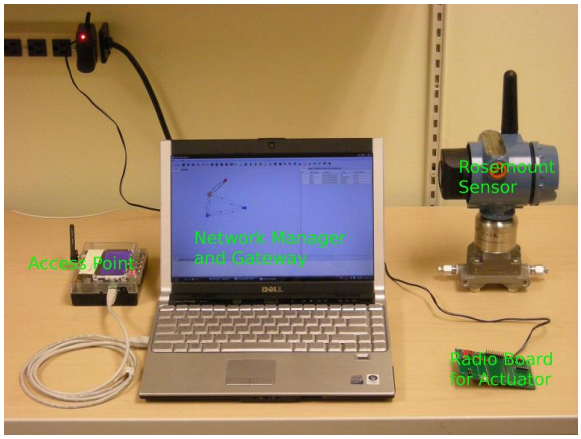

Fig. 8. The major components in our WirelessHART communication system

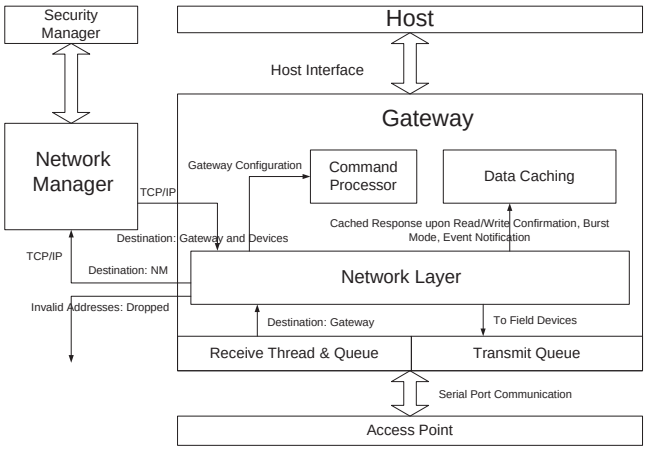

Fig. 9. The architecture of the Gateway

through a socket connection for message exchange. The Gateway also provides one or more Host Interfaces to backbone networks (e.g., the plant automation network) to receive the queries and send the responses back.

Real-time Database and Query Processor: The core parts of the Gateway are a real-time database and a query processor. The database provides data caching for burst mode, event notification, and common HART command responses. The query processor processes the queries from Host applications. If the requested data are already cached and still valid, they are returned immediately to the Host applications. This reduces network traffic and improves the Host application's responsiveness. Otherwise, the query processor will generate the request messages and send them to the corresponding devices. The return response data are cached in the Gateway and sent back to the Host applications.

Time Source: The Gateway maintains a time source module for maintaining network-wide time synchronization. It will notify all the devices in the network and let them synchronize with the Gateway through the approaches discussed in Section III. In our system, the actual time sources are the Access Points instead of the Gateway. The Access Point will periodically update the accurate time to the Gateway and Network Manager. The frequency of the update depends on how accurate the time is needed in Gateway and Network Manager.

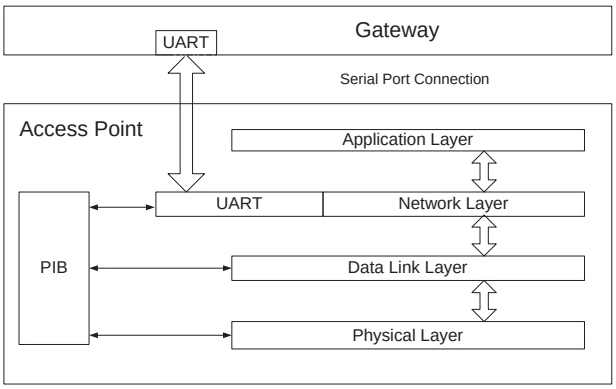

Fig. 10. The architecture of the Access Point

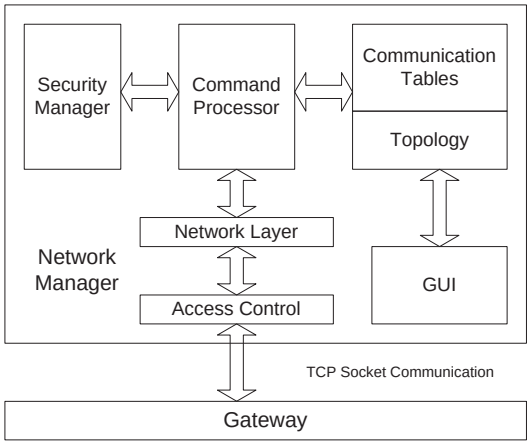

Fig. 11. The architecture of the Network Manager

C. Access Point Design

The Access Point is a bridge between the mesh network and the Gateway. There could be multiple Access Points attached to the Gateway providing load balancing and reliable graph routing. Each Access Point goes through the same join procedure as a normal device to authenticate itself and establish a secured connection with the Network Manager. The communication stack on the Access Point is extended from that of the device by adding an extra UART module. The messages received from the mesh will be forwarded to the UART module and sent to the Gateway. In the other direction, the messages from Gateway/Network Manager will be sent through the serial port and put into the network layer queue in the Access Point.

D. Network Manager Design

The core of a WirelessHART mesh network is the Network Manager. It is responsible for authenticating the devices, forming the network, allocating network resources and scheduling process data transmissions. We have described the detailed algorithmic issues in Section IV and Section V for generating routing graphs and constructing communication schedules. In this section, we present our implementation of the Network Manager and how we integrate our network management solutions into it. Figure 11 shows the architecture of the Network Manager which has four major components:

Command Processor: The application layer of the WirelessHART standard is command-oriented. The WirelessHART devices and the Network Manager interact by exchanging command requests and command responses. The command processor

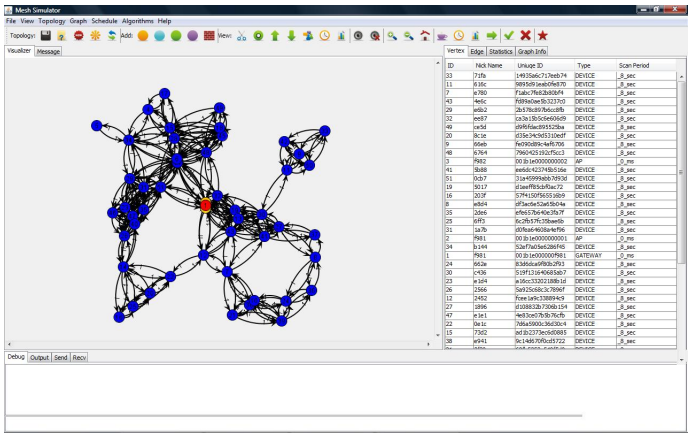

Fig. 12. Network topology in the visualizer

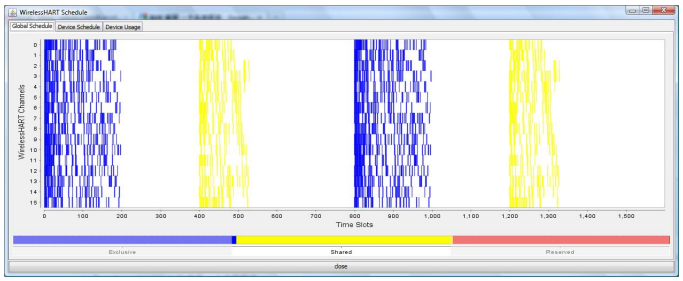

Fig. 13. Network global schedule in the visualizer

in the Network Manager processes the commands from the devices, updates the network topology and triggers the algorithms if necessary to reconstruct the routing graphs and communication schedules. The command processor is also responsible for generating the command responses and send them back to the devices to reconfigure them.

Topology and Communication Tables: In the Network Manager, the network topology is maintained in a directed graph structure. All the algorithms for constructing routing graphs and allocating network resources are conducted in the graph and the results are maintained in a set of communication tables. These tables include the session tables, graph tables, route tables, superframe and link tables. Interested readers are referred to [3] for their details.

Security Manager and Access Control: WirelessHART is a secure wireless communication protocol and it provides encryption and authentication in both the data link layer and network layer. The main task of the security manager is to manage various key information for the devices. It takes charge of the device join authentication and updates the key information in the network periodically for protection purpose. The access control module maintains a list of pre-approved devices together with their valid join keys. Only the devices on the list can be admitted into the network by providing the correct join keys.

Visualizer: Our visualizer is implemented based on the JUNG library [19]. It provides the user a straight-forward way to observe the network topology, the routing graphs, the device communication schedules, and the messages being exchanged between the Network Manager and devices in the mesh. As an example, Figure 12 presents a topology of a WirelessHART network with two Access Points and 50 field devices. By setting every device's sample rate at 8 sec, Figure 13 and Figure 14 show

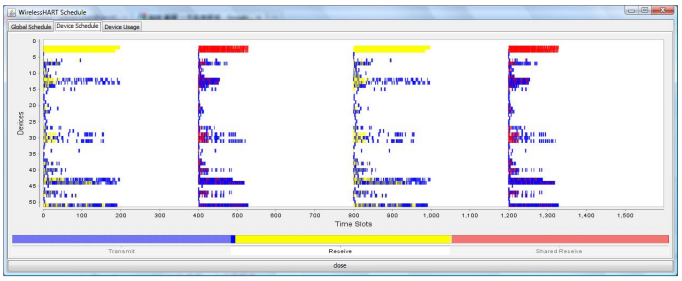

Fig. 14. Device schedules in the visualizer

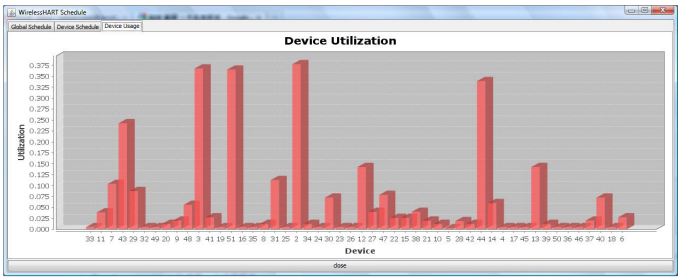

Fig. 15. Device usages in the visualizer

the global communication schedule for publishing these data to the Gateway and each device's individual schedule respectively. Figure 15 further depicts the bandwidth usage for each device in the network. The bandwidth is consumed either for publishing its own data or helping relay other devices' data to the Gateway. Any update on the network topology, routing graphs or the communication schedules will be reflected in the visualizer in real-time. With the visualizer, users can identify problematic network topology and bottlenecks limiting network throughput and perform appropriate adjustments.

We note that our Network Manager design is not only for the WirelessHART communication systems. It is also a generic simulator for wireless mesh networks and allows the users specify any network topology either through reading in a topology file or configuring it manually. It provides the user with a platform to design their algorithms, install and execute them on the specified topology and evaluate their performance.

VII. Performance Evaluation

This section summarizes the major results from our simulations to evaluate the performance of our algorithms in large-scale network. Our simulation model and parameter settings are described in Section VII-A. Section VII-B compares our algorithms in constructing routing graphs to traditional approaches. Section VII-C evaluates the performance of our approach for constructing communication schedules. The results show that our approaches can achieve higher routing success rate, better end-to-end communication latency while incurring only modest configuration overheads on devices.

A. Simulation Model and Parameters

In the simulations, we assume open field, line-of-sight experimental scenarios. The simulation area is fixed at 450 m $\times$ 450 m and the device communication distance is 100 meters with a 0 dBm transmitter. We assume that there is no edge between a pair of nodes if they are not in each other's communication range. Otherwise, an edge exists with an edge success probability p that is varied from 0.0 to 1.0. The size of the network is varied from 50 to 150 to evaluate the effect of network density on the

algorithm's performance. We disable a given portion of links in the network to evaluate the reliability of the constructed routing graphs and this percentage is varied from 0% to 100%.

B. Performance of Reliable Routing Graphs

We conducted a series of experiments to evaluate the performance of reliable broadcast graph G_B, reliable uplink graph G_U and reliable downlink graph G_v for each individual node v. As essentially G_U is the reversed version of G_B, its performance is similar to that of G_B. For this reason, the experimental results for G_U are omitted here.

We evaluate our approach for constructing G_B by comparing it with two baseline methods. The first method constructs a single broadcast tree using breadth-first search and the second method generates the max-reliable broadcast graph. In this method, when a node is chosen to be added to the broadcast graph, all its incoming edges from the current broadcast graph are also added. Our approach is different from this method because we only choose the first two incoming edges of the chosen node with minimum latency to achieve a good balance between the routing reliability and the configuration cost on the device. The configuration cost is defined as the average number of links to be configured per node. It is an important performance metric because wireless devices' memory is limited and larger number of links in the network will severely hurt the schedulability of the communication schedule. The first set of our experiments compare the configuration cost introduced by these three approaches. In the experiments, we vary the size of the network from 50 to 150 nodes and evaluate its effect. Figure 16 summaries our results. In the figure, we observe that the cost of the max-reliable approach is much higher than the other two and it increases linearly along with the increase of the network density. However, the configuration cost in our approach and the tree solution is low and stable. The cost in our approach is always below 2 links per node, and it is closer to the performance of the tree solution when the network density is low. This observation is mainly because when the network density is low, many nodes are difficult to find two parents in the network thus has only one link in the broadcast graph.

In the second set of experiments, we first construct the broadcast graphs based on these three approaches with 100 nodes in the network. We then gradually increase the percentage of failed links in the network from 0% to 95%. We measure the reliability of these three approaches and then apply the recovery mechanisms we discussed in Section IV-H on them. We compare their recovery cost in terms of number of changed links. Figure 17 shows that along with the increased percentage of failed links in the network, the reliability of the tree solution drops quickly and when half of the links die, only around 25% nodes are reachable from the Gateway. Our approach performs much better. With the same percentage of failed links in the network, around 55% nodes are still connected to the Gateway. Among all three approaches, the max-reliable broadcast graph has the best performance as a tradeoff of its poor scalability and much higher configuration cost. In the figure, we also show a curve of the reachability for the broadcast graphs after the recovery. As the recovery mechanisms are all based on the same underlying network topology, all three approaches have the same reachability after reconstruction. This in turn verifies the correctness of our recovery approaches.

Figure 18 and Figure 19 show the number of changed links in the recovery process. Figure 18 shows the number of necessary changes to resume the connectivity of the broadcast graphs while Figure 19 shows the number of necessary links to recover their reliability properties as much as possible. We observe from Figure 18 that the tree solution always has the heaviest recovery cost. On the other hand, as the max-reliable broadcast tree has the best reliability, its recovery cost is the minimum. The performance of our approach sits between them. However, Figure 19 shows that to recover the reliability property, our approach needs to add more links in the broadcast graph than both of the tree solution and the max-reliable broadcast graph. The reason for this observation is because the tree solution has no reliability requirement but just to maintain the connectivity; while the max-reliable approach has added most of the links in the construction stage thus the recovery cost is also relatively smaller.

To evaluate our approach for constructing reliable downlink graph, we compare it with another two methods. The first method constructs a single shortest path between the Gateway and each individual node. The second method constructs two node-disjoint paths between the Gateway and each node to achieve reliable downlink routing to some extent. Figure 20 demonstrates the comparison of the reliability among these three approaches. It clearly shows that our approach maintains the best reliability and always outperforms the two node-disjoint path method more than 30%. As a tradeoff, as shown in Figure 21 and Figure 22, the average number of nodes in our reliable downlink graphs is larger than the relatively simple solutions. The average number of nodes in our reliable downlink graph is around two times larger than that of the single path routing and 1.2 times larger than that of the two node-disjoint path routing. Furthermore, as each node in the reliable downlink graph has two outgoing edges to forward the messages, the average number of links in the downlink graph is also larger than the other two approaches. To address this drawback of the reliable downlink graph, we are working on extending the downlink graph routing approach defined in the standard to make it more scalable.

C. Construction of Communication Schedules

Our approach for constructing the network communication schedule has two unique features. First, we split the traffic from a device among all its successors by reducing the bandwidth requirement on each successor. The communication schedules on the successors are carefully designed so that their combination has the same patten as the original device. As the second feature, we use the concept of shared timeslot to allow multiple devices to compete for communicating with the same device simultaneously. This is especially useful for the links that are allocated for retry purpose and it can significantly improve the network throughput.

In this section, we evaluate the performance of these two introduced features by comparing our approach with three baseline methods. The basic methods either lack one of the features or both of them. For simplicity, we only show our experimental results on scheduling process data from devices to the Gateway on the uplink graph. Scheduling control data on the other direction is similar, thus is omitted here. We define two performance metrics for this set of experiments. The first metric is the scheduling success ratio which measures the percentage of nodes that can successfully allocate the required bandwidth along its paths to the Gateway; The second metric is the network utilization which measures the percentage of entries in matrix $\mathcal{M}$ that are already allocated for communication. Our results are summarized in Figure 23 and Figure 24 respectively.

In Figure 23, we compare the scheduling success ratio by

deploying 50 nodes in the network and varying the device sample rate from 250 ms to 4 min and 16 sec (each device has the same sample rate). We observe that by halving the bandwidth requirement on a device's successors (if it has two successors), the success ratio to schedule the end-to-end communications can be greatly improved. The improvement is more than 25% when the sample rate is 2 sec and is even higher when the sampling is faster. Figure 23 also shows that by applying the shared timeslot, the success ratio can be increased by 5% and this improvement is consistently shown in our experimental results until the sample rate is low enough that the scheduling success ratio approaches 100%. Figure 24 shows that when the approaches has a close scheduling success ratio, our approach has a much lower network utilization, and this will further help include more devices into the network. When the sample rate is fast, our approach has a higher network utilization because in these scenarios, the success ratio for other approaches is so poor that very limited number of devices can successfully allocate their required bandwidth along its path to the Gateway.

VIII. Conclusions and Future Work

In this paper, we study the problem how to achieve reliable and real-time communication in industrial wireless mesh networks. Taking WirelessHART network as an example, we abstract the reliability requirements in typical wireless industrial process control applications and present the algorithms for constructing three types of reliable routing graphs for different communication purposes. Based on these routing graphs, we describe how we construct the communication schedule in the network and highlight our approach's unique features. We present the architecture of a complete WirelessHART communication system that we have built and we have performed extensive simulations to evaluate the performance of our algorithms.

In ongoing and future work, we are deploying our system in a manufacturing factory with more than 200 sensor devices, so that we can evaluate the performance of our network management techniques in real industrial environments. We shall continue to look for more efficient approaches for constructing reliable routing graphs, especially for the reliable downlink graphs and their corresponding recovery mechanisms.

References

[1] Dick Caro, *Wireless Networks for Industrial Automation*, ISA Press, 2004.

[2] J. Song, A. K. Mok, D. Chen, M. Nixon, T. Blevins, and W. Wojsznis, "Improving pid control with unreliable communications," in *ISA EXPO Technical Conference*, 2006.

[3] J. Song, S. Han, A. K. Mok, D. Chen, M. Lucas, M. Nixon, and W. Pratt, "Wirelesshart: Applying wireless technology in real-time industrial process control," in *RTAS*, 2008.

[4] Rajeev Alur, Alessandro D'Innocenzo, Karl H. Johansson, George J. Pappas, and Gera Weiss, "Modeling and analysis of multi-hop control networks," in *RTAS*, 2009.

[5] Gera Weiss, Rajeev Alur, Alf J. Isaksson, and Karl H. Johansson, "Scalable scheduling algorithms for wireless networked control systems," in *CASE*, 2009.

[6] Joonas Pesonen, Haibo Zhang, Pablo Soldati, and Mikael Johansson, "Methodology and tools for controller-networking codesign in wirelesshart," in *ETFA*, 2009.

[7] Shahid Raza, Adriaan Slabbert, Thiemo Voigt, and Krister Landernäs, "Security considerations for the wireless hart protocol," in *ETFA*, 2009.

[8] Gabriella Fiore, Valeria Ercoli, Alf J. Isaksson, Krister Landernäs, and Maria Domenica Di Benedetto, "Multihop multi-channel scheduling for wireless control in wirelesshart networks," in *ETFA*, 2009.

[9] "ISA," `http://www.isa.org/`.

[10] "HART communication," `http://www.hartcomm.org`.

[11] "ZigBee Alliance," `http://www.zigbee.org`.

[12] "WirelessHART," `http://www.hartcomm.org/protocol/wihart/wireless_technology.html`.

[13] J. Song, S. Han, X. Zhu, A. K. Mok, D. Chen, and M. Nixon, "Demonstration of a complete wirelesshart network," in *SenSys (demo)*, 2008.

[14] S. Han, J. Song, X. Zhu, A. K. Mok, D. Chen, M. Nixon, W. Pratt, and V. Gondhalekar, "Wi-HTest: testing suite for diagnosing wirelesshart devices and networks," *RTAS*, 2009.

[15] S. Han, J. Song, X. Zhu, A. K. Mok, D. Chen, M. Nixon, W. Pratt, and V. Gondhalekar, "HTest-W: Testing suite for diagnosing wirelesshart devices and networks," in *SenSys (poster)*, 2008.

[16] "Bluetooth," `www.bluetooth.com/bluetooth`.

[17] "IEEE 802.15.4 WPAN Task Group," `www.ieee802.org/15/pub/TG4.html`.

[18] "Rosemount," `http://www.emersonprocess.com/Rosemount/`.

[19] "Java Universal Network/Graph Framework," `jung.sourceforge.net/`.

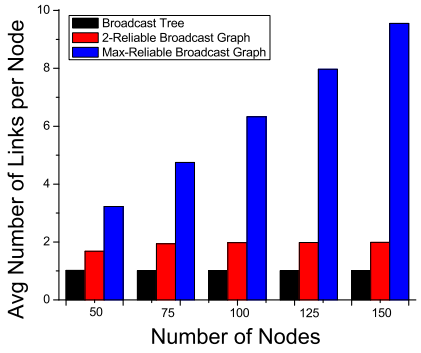

Fig. 16. Configuration costs in broadcast graphs

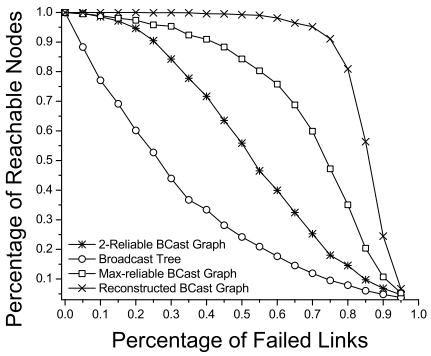

Fig. 17. Reachability in broadcast graphs

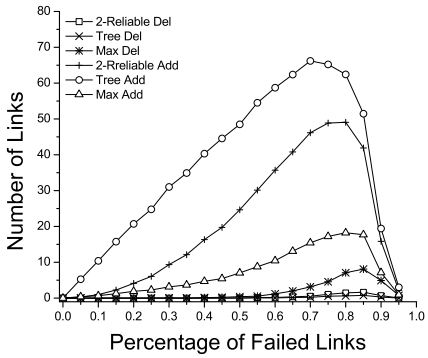

Fig. 18. Recovery cost to regain connectivity in G_B

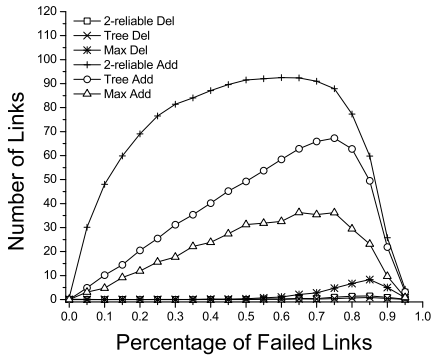

Fig. 19. Recovery cost to regain reliability in G_B

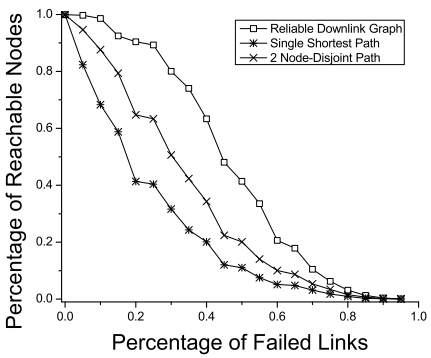

Fig. 20. Reachability in downlink graph

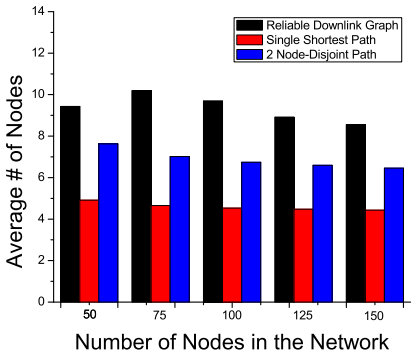

Fig. 21. Average # of nodes per downlink graph

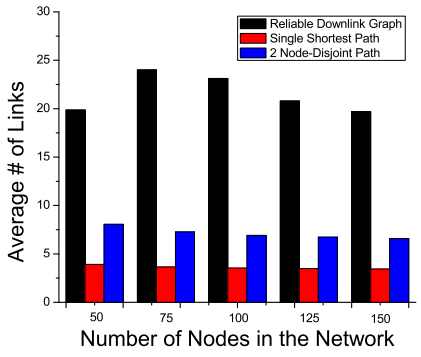

Fig. 22. Average # of edges per downlink graph

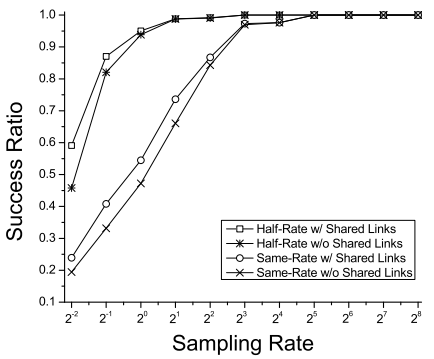

Fig. 23. Success ratio vs. Sample rate

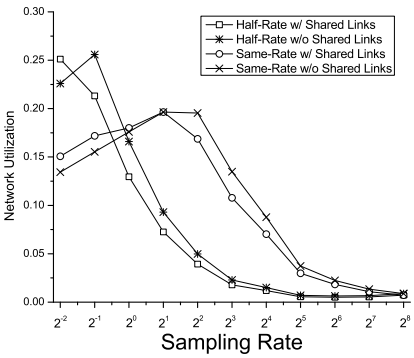

Fig. 24. Network utilization vs. Sample rate