Report Number: WUCSE-2015-004

2015-09-29

Maximizing Network Lifetime of Wireless Sensor-Actuator Networks under Graph Routing

Authors: Chengjie Wu, Dolvara Gunatilaka, Abusayeed Saifullah, Mo Sha, Paras Tiwari, Chenyang Lu, and Yixin Chen

Process industries are adopting wireless sensor-actuator networks (WSANs) as the communication infrastructure. The dynamics of industrial environments and stringent reliability requirements necessitate high degrees of fault tolerance in routing. WirelessHART is an open industrial standard for WSANs that have seen world-wide deployments. WirelessHART employs graph routing schemes to achieve network reliability through multiple paths. Since many industrial devices operate on batteries in harsh environments where changing batteries are prohibitively labor-intensive, WSANs need to achieve long network lifetime. To meet industrial demand for long-term reliable communication, this paper studies the problem of maximizing network lifetime for WSANs under graph routing. We formulate the network lifetime maximization problem for WirelessHART networks under graph routing. Then, we propose the optimal algorithm and two more efficient algorithms to prolong the network lifetime of WSANs. Experiments in a physical testbed and simulations show our linear programming relaxation and greedy heuristics can improve the network lifetime by up to 50% while preserving the reliability benefits of graph routing.
... **Read complete abstract on page 2.**

Maximizing Network Lifetime of Wireless Sensor-Actuator Networks under Graph Routing

Complete Abstract:

Process industries are adopting wireless sensor-actuator networks (WSANs) as the communication infrastructure. The dynamics of industrial environments and stringent reliability requirements necessitate high degrees of fault tolerance in routing. WirelessHART is an open industrial standard for WSANs that have seen world-wide deployments. WirelessHART employs graph routing schemes to achieve network reliability through multiple paths. Since many industrial devices operate on batteries in harsh environments where changing batteries are prohibitively labor-intensive, WSANs need to achieve long network lifetime. To meet industrial demand for long-term reliable communication, this paper studies the problem of maximizing network lifetime for WSANs under graph routing. We formulate the network lifetime maximization problem for WirelessHART networks under graph routing. Then, we propose the optimal algorithm and two more efficient algorithms to prolong the network lifetime of WSANs. Experiments in a physical testbed and simulations show our linear programming relaxation and greedy heuristics can improve the network lifetime by up to 50% while preserving the reliability benefits of graph routing.

Maximizing Network Lifetime of WirelessHART Networks under Graph Routing

Chengjie Wu, Dolvara Gunatilaka, Abusayeed Saifullah*, Mo Sha†,
Paras Babu Tiwari, Chenyang Lu, Yixin Chen
Department of Computer Science & Engineering, Washington University in St. Louis
* Department of Computer Science, Missouri University of Science & Technology
† Department of Computer Science, State University of New York at Binghamton

***Abstract*—Industrial Wireless Sensor-Actuator Networks (WSANs) enable Internet of Things (IoT) to be incorporated in industrial plants. The dynamics of industrial environments and stringent reliability requirements necessitate high degrees of fault tolerance. WirelessHART is an important industrial standard for WSANs that have seen world-wide deployments. WirelessHART employs graph routing to enhance network reliability through multiple paths. Since many industrial devices operate on batteries in harsh environments where changing batteries is prohibitively labor-intensive, WirelessHART networks need to achieve a long network lifetime. To meet industrial demand for long-term reliable communication, this paper studies the problem of maximizing network lifetime for WirelessHART networks under graph routing. We first formulate the network lifetime maximization problem and prove it is NP-hard. Then, we propose an optimal algorithm based on integer programming, a linear programming relaxation algorithm and a greedy heuristic algorithm to prolong the network lifetime of WirelessHART networks. Experiments in a physical testbed and simulations show our algorithms can improve the network lifetime by up to 60% while preserving the reliability benefits of graph routing.**

***Index Terms*—WirelessHART, industrial wireless sensor-actuator networks, graph routing, network lifetime maximization.**

I. Introduction

With the emergence of industrial standards such as WirelessHART [1] and ISA100 [2], process industries are embracing IoT technology based on low-power wireless mesh networks for process automation [3]. The process industry has installed more than 24 thousand WirelessHART networks around the world, with more than 5 billion operating hours in the field [4].

The limited energy supply of IoT devices necessitates the efficient utilization of battery power. Energy consumption is closely coupled with route selection. Selecting a routing path that optimizes energy efficiency can lead to a longer network lifetime. In industrial environments, changing batteries can be dramatically expensive and difficult, e.g., oil fields spanning large areas under harsh environmental conditions. Thus, maximizing the lifetime of the network is an important problem that needs to be tackled.

Although the problem of energy efficient routing has been extensively studied for traditional wireless networks, the strict reliability requirements in industrial WSANs bring new challenges. To support reliable communication over wireless mesh networks, the WirelessHART standard adopts a graph routing approach. A graph route consists of a primary path and multiple backup paths. For each intermediate node on the primary path, a backup path is generated to handle link or node failure on the primary path. Moreover, the energy consumption of network nodes is highly coupled with the (re)transmission scheduling policy adopted by industrial standards. Graph routing introduces unique challenges in energy-efficient routing that has not been investigated in earlier research on energy-efficient routing for wireless sensor networks.

This paper addresses the network lifetime maximization problem of WirelessHART networks under graph routing. Specifically, our contributions are five-fold:

- Formulation of the network lifetime maximization problem under graph routing and proof of its NP-hardness.
- An optimal network lifetime maximization algorithm based on integer programming.
- An approximation algorithm through linear programming relaxation of the integer programming algorithm.
- An efficient greedy heuristic with lower computational complexity.
- Implementation and evaluation of the proposed algorithms on a physical WSAN testbed, as well as in simulations.

Our evaluation shows that our algorithms can improve the network lifetime by up to 60%, and the greedy heuristic is more efficient than the linear programming relaxation approach.

The rest of the paper is organized as follows. Section II reviews related works. Section III describes the network model. Section IV formulates the lifetime maximization problem and proves its NP-hardness. Section V presents our lifetime maximization graph routing algorithms. Section VI evaluates the graph routing algorithms in experiments and simulations. Section VII concludes the paper.

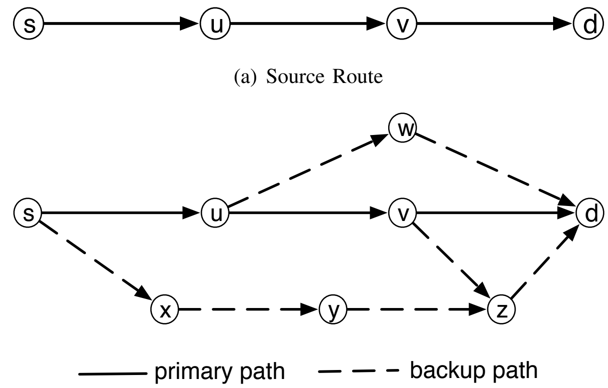
II. Related Work

Energy-aware routing for wireless sensor and ad hoc networks has received significant attention [5]. Stojmenovic and Lin [6] proposed a protocol to minimize total power consumption and extend network lifetime. Chang and Tassiulas maximized network lifetime by balancing network traffic among the nodes in proportion to their residual energy [7], [8]. Wu et al. [9] proposed a routing algorithm to improve the lifetime and reliability of the network based on local topology information. Li et al. [10] proposed a routing protocol that combines the benefits of selecting the path with minimum power consumption and the path that maximizes residual power in the nodes. Doshi et at. [11] implemented a minimum energy routing version of the DSR protocol in a network simulator. Kalpakis et al. [12] studied the lifetime maximization problem for tree topology networks. Despite considerable results on the general problem of network lifetime optimization, none of the aforementioned works address graph routing. Note the path diversity provided by graph routing is a key technique that the WirelessHART standard used to achieve reliable communication in industrial settings [13].

WirelessHART networks have attracted a lot of attention in the research community [14]–[23]. Previous literature studied real-time transmission scheduling [15], [16], [22], communication delay analysis [19], [21], rate selection [20], and system performance optimization [23]. All these works assumed that routes of the flows are already given, and did not provide any routing protocol.

There has been increasing interest in developing new routing approaches for WirelessHART networks. Zhao et al. proposed a routing algorithm called ELHFR [24]. Gao et al. proposed a multipath graph routing algorithm with subgraphs called ORMGR [25]. Han et al. proposed routing algorithms [26] to construct reliable routing graphs. However, in the aforementioned works, hop count is the only criterion when choosing the links. Network lifetime is not considered when making the routing decision. Wu et al. [27] studied real-time routing for WirelessHART networks, which did not consider network lifetime. Our work is motivated by an earlier experimental study of WirelessHART routing protocols [13] that showed graph routing achieved higher reliability at higher energy cost, and hence it is essential to develop energy-efficient graph routing protocols.

To improve energy efficiency in WirelessHART networks, Wang et al. proposed a routing algorithm called DHEIRP [28], which chooses the next hop node by comparing the residual energy of neighbors. Memon et al. proposed a load-balanced routing algorithm [29] that chooses the next-hop node by comparing the communication loads of neighbors. The JRMNL algorithm [30] chooses the next hop according to node communication load, node residual energy and link transmission energy consumption. Zhang et al. proposed a routing algorithm [31] to select next hop by taking into account the remaining energy, the quality of the link and the number of hops. However, all works above take an approach sitting between the source routing and graph routing. After building a graph, a source route is used to deliver a single packet, although different packets may use different source routes. As a result, a packet will not benefit from path diversity to improve reliability. As path diversity and graph routing are crucial for industrial applications (especially control applications) to meet their stringent reliability requirements, we investigate the open problem of network lifetime maximization under graph routing in WirelessHART networks in this paper.



(a) Source Route

(b) Graph Route

Fig. 1: Source and Graph Routing

III. Network Models

A WirelessHART network [1] consists of a gateway, multiple access points, and a set of field devices (sensors and actuators). The access points and field devices are equipped with half-duplex radio transceivers compatible with the IEEE 802.15.4 standard [32], and form a wireless mesh network. The access points are wired to the gateway and serve as bridges between the gateway and field devices.

WirelessHART adopts a centralized network management architecture. The network manager (a software module running on the gateway or a host connected to the gateway) manages all devices in the network. The network manager gathers the network topology information, generates and disseminates the routes and transmission schedule to all network devices. This centralized network management architecture enhances the predictability and visibility of network operations at the cost of scalability.

WirelessHART adopts a Time Division Multiple Access (TDMA) MAC layer protocol on top of the IEEE 802.15.4 physical layer. All devices in the network are time synchronized. Time is divided into 10 ms slots, and each slot can accommodate one data packet transmission and its acknowledgment. WirelessHART supports multi-channel communication using up to 16 channels specified in the IEEE 802.15.4 standard. In a slot, only one transmission is scheduled on each channel across the entire network to avoid collision.

A. Routing Model

WirelessHART supports both source routing and graph routing. Under *source routing*, a single path from the source to the destination is generated for each data flow as shown in Figure 1(a).

Under *graph routing*, redundant paths are provided to handle link failures. As shown in Figure 1(b), a single path is generated as a primary path (solid arrows) and a backup path is generated for each device along the primary path except the destination d. For instance, a backup path $u \to w \to d$ is generated for node u and it is used when the transmission on $u \rightarrow v$ fails.

A WirelessHART network can be defined as $G = (V, E)$, where V denotes a set of devices and E denotes a set of bidirectional links[1] between devices. A link in E can be a link between two field devices or a link between an access point and a field device. We define a graph route as below:

Definition 1. *Given a source device s and a destination device d, a graph route $R = \{\phi_0, \phi_1, \cdots, \phi_{|\phi_0|}\}$ is a set of paths from s to d, where ϕ_0 is the primary path and $|\phi_0|$ denotes the number of links in ϕ_0. Each device v_i on the primary path ϕ_0, except the destination d, has a backup path ϕ_i from itself to the destination, which does not include v_i's outgoing link on the primary path.*

A WirelessHART network can support multiple data flows in the network. Two graph routes are generated for each data flow: an uplink graph route and a downlink graph route. The uplink graph route starts from the sensor and ends at the access points. A downlink route starts from an access point and ends at the actuator. As the data flows are usually generated by process monitoring or control applications, they usually generate packets periodically.

B. Transmission Scheduling Model

In WirelessHART networks, a time slot can be a *dedicated* slot or a *shared* slot. In each channel, only one transmission is scheduled in a dedicated slot, while multiple transmissions may compete for a shared slot in a CSMA/CA fashion.

Only dedicated slots are used for source routing. A transmission and a retransmission are scheduled in dedicated slots for each link under the source routing.

Both dedicated slots and shared slots are used for graph routing. For each device on the primary path, the network manager allocates two dedicated slots for a transmission and a retransmission on its outgoing link along the primary path, and also assigns a third shared slot on its outgoing link along its backup path. Therefore, each link on the primary path is assigned two dedicated slots and each link on backup paths is assigned a shared slot. Since WirelessHART networks usually only employ high-quality links, shared slots are assigned to backup paths to reduce delay and enhance bandwidth.

C. Energy Consumption Model

We model the energy consumption under graph routing in this subsection. We only consider the energy consumption of the radio which is related to packet transmission and reception. The energy consumption of microprocessors, sensors, and other parts is out of the scope of this paper. For a single packet, we calculate the energy consumption of each device on both the primary and backup paths. Since the scheduling policies for transmissions on the primary path and backup path are different, we calculate the energy consumption for them separately. For each transmission along the primary path, two dedicated slots are assigned. If the first transmission succeeds, the retransmission will not occur and both sender and receiver will turn off their radios at the second slot. Otherwise, a retransmission will occur in the second time slot. If both the transmission and retransmission along the primary path fail, there will be a second retransmission along the backup path.

Figure 2 shows the timing of a transmission in a time slot. The top of the timing diagram shows the operation of the sender and the bottom shows that of the receiver. When a shared slot is assigned, the sender will perform Clear Channel Assessment (CCA) before transmitting the packet. We use *TsMaxPacket* to denote the maximum time to transmit a packet. When scheduled as the transmission's receiver, the receiver must enter receive mode. The receiver must keep the radio on to listen to potential packet transmission. We denote the minimum time to wait for the start of a message as *TsRxWait*. If a transmission is detected, the receiver keeps receiving until it receives the entire packet. Otherwise, the receiver will turn off the radio after the receive window expires. We denote the power of transmitting and receiving a packet as P_t and P_r respectively.

Assume v_i is a device on the primary path, which is scheduled to send one packet to device v_j. We use α to denote the Packet Reception Ratio (PRR) for this link. Then the probability that it successfully transmits a packet to its receiver v_j on the first try is α. The probability that it fails in the first attempt and needs to retransmit the packet to v_j is $1 - \alpha$. So the expected time length that the sender keeps its radio on is

$$\alpha \times \textit{TsMaxPacket} + 2(1-\alpha)\textit{TsMaxPacket} = (2-\alpha)\textit{TsMaxPacket}$$

In the case of checksum error, the receiver needs to keep the radio on for *TsMaxPacket*, so the receiver on the primary path has the same expected time length keeping its radio on as the sender. By incorporating the power, we get the expected energy consumption of a device as a sender or a receiver for delivering one packet on the primary path. We denote E_t as the expected energy consumption of device v_i to transmit a packet to v_j on a primary path, thus

$$E_t = (2 - \alpha) P_t \times \textit{TsMaxPacket} \tag{1}$$

The expected energy consumption of device j to receive a packet from i on a primary path is:

$$E_r = (2 - \alpha) P_r \times \textit{TsMaxPacket} \tag{2}$$

Since transmission on a backup path only happens when the two previous attempts fail, the chance that there is an actual packet transmission on a backup path is $(1 - \alpha)^2$, e.g., less than 0.01 if we use a PRR threshold of 0.9. However, as long as a transmission is scheduled on a link, the receiver needs to turn on the radio and listen for *TsRxWait* time to check whether there is an incoming packet. Then the expected energy consumption of device i on a backup path to transmit a packet is

$$E_{tb} = (1 - \alpha)^2 P_t \times \textit{TsMaxPacket} \tag{3}$$

[1] WirelessHART only uses bidirectional links for packet transmission and acknowledgement.

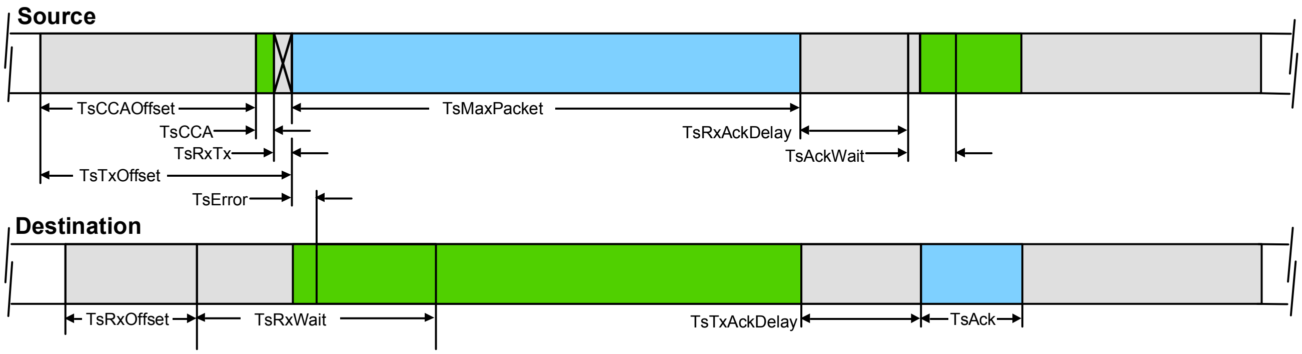


Fig. 2: Transaction timing in one time slot [1]

The expected energy consumption of device i to receive a packet on backup path is:

$$E_{rb} = (1-\alpha)^2 P_r \times \mathit{TsMaxPacket} + (1-(1-\alpha)^2) P_r \times \mathit{TsRxWait} \tag{4}$$

Table I summarizes the transmission and reception power of the CC2420 radio chip [33], which is compatible with the IEEE 802.15.4 standard. Table I also shows the timing parameters of packet transmission specified in the WirelessHART standard [1]. Based on Table I, we obtain the expected energy consumptions in Table II, assuming a PRR of 90%, a typical threshold used for blacklisting links in WirelessHART networks.

Parameter	Value	Unit
P_t	52.2	mW
P_r	59.1	mW
TsMaxPacket	4256	μs
TsRxWait	2200	μs

TABLE I: Representative Radio Parameters

Variable	Value	Unit
E_t	277	μJ
E_r	244	μJ
E_{tb}	2.2	μJ
E_{rb}	131	μJ

TABLE II: Expected energy consumption of devices to transmit or receive a packet

Since the expected energy consumption of transmitting a packet through a link along a backup path is two order magnitude less than the other three expected energy consumptions, we ignore E_{tb} in the routing algorithm for simplicity.

IV. Graph Route Lifetime Maximization Problem

In this section, we formulate the *Graph Route Lifetime Maximization* (GRLM) problem. Our objective is to maximize the network lifetime, which is the time interval before the first field device exhausts its battery. This definition is well accepted by previous literatures.

In terms of lifetime optimization, the most significant difference between WirelessHART networks and traditional wireless sensor networks is path diversity. Instead of scheduling transmissions on only one path, WirelessHART networks schedule transmissions on both the primary path and backup paths.

Definition 2. *In a GRLM problem, we are given a graph $G = (V, E)$ with battery capacity B_i for each device v_i, and a set of flows $F = \{f_1, f_2, \cdots, f_N\}$. Each flow f_k has a source s_k, a destination d_k, and a data rate r_k. The GRLM problem is to find graph routes for all flows to maximize the network lifetime.*

The GRLM problem is NP-hard because even the source routing version of the problem is NP-hard as shown below.

Proof. To prove the SRLM problem is NP-hard, we prove its decision version is NP-complete. The decision version of SRLM is: given a network lifetime T for a network, can this lifetime be satisfied by the network?

Clearly, the decision problem of SRLM is NP. Given a solution with source routes, we can verify whether the network can satisfy the lifetime T by checking the lifetime of each device. We calculate the expected energy consumption rate of each device by taking account of data rates of flows which pass this device and expected energy consumption per packet shown in equations (1) and (2). The time complexity is $O(|V|N)$.

To prove the decision problem of SRLM is NP-complete, we use a well known NP-complete problem. Fortune et al. [34] proved the *Maximum Edge-Disjoint Paths* problem (MEDP) is NP-hard. In MEDP, we are given a graph $G = (V, E)$, and a set of N device pairs $\Theta = \{(s_k, t_k) : k = 1, \cdots, N\}$. The goal is to find the maximum subset of pairs from Θ , along with a path for each chosen pair, so that no two paths share the same link. The decision problem of MEDP is whether a given set of device pairs Θ have link-disjoint paths. The decision problem of MEDP is NP-complete.

We reduce the decision problem of MEDP to the decision problem of SRLM. The reduction algorithm takes an instance of the decision problem of MEDP problem as input. Given a graph G, we construct an auxiliary graph G' in the following

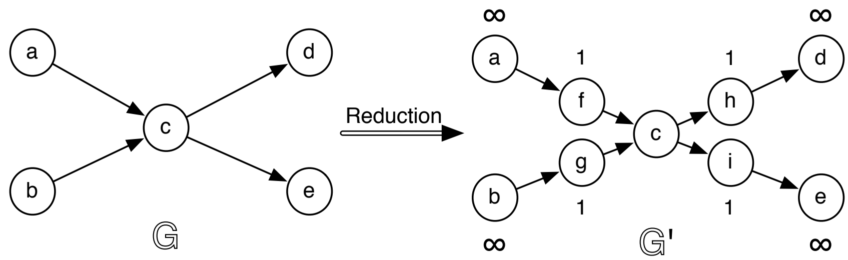


Fig. 3: Reduction

manner. For each link e in G (i.e., $a \to c$ in Figure 3), we break it into two links ($a \to f$ and $f \to c$) and add a new link-device (f) to connect these two links (Figure 2). All devices in the original graph are assigned battery capacity as $+\infty$, and all newly added link-devices are assigned unit battery capacity 1. For each device pair (s_k, t_k) in Θ, we create a flow F_i in G' with source s_k, destination t_k, and unit rate 1. The targeted lifetime of the network is $T = \frac{1}{E_t + E_r}$. Note that $\frac{1}{E_t + E_r}$ is the lifetime of a link-device if only one flow goes through it. To complete the proof, we show that all pairs in Θ have link-disjoint paths if and only if the network lifetime of G' is no less than T.

$\leftarrow$ If all device pairs have link-disjoint paths in G, then the reduced paths in G' can have a network lifetime no less than T. Since at most one reduced flow goes though each link-device, the lifetime of each link-device is no less than the network lifetime target T. We prove that the network lifetime of G' is no less than T.

$\rightarrow$ If the network lifetime of G' is no less than T, then there are link-disjoint paths for all device pairs in Θ. Since the battery of each link-device can support exactly one flow, only one path will go through each link-device, which indicates those paths are edge-disjoint paths. Then we get link-disjoint paths in the original graph G.

Because the reduction takes polynomial time and an instance of the decision problem of MEDP is true if and only if the reduced instance of the decision problem of SRML is true, the decision problem of SRML is NP-complete. $\square$

V. Lifetime Maximization Graph Routing Algorithms

In this section, we propose an optimal solution based on integer programming, followed by more efficient solutions based on linear programming relaxation and greedy heuristic. The efficiency of the routing algorithms are important because the network manager needs to recompute routes as network topology and channel condition change in real-world environments.

A. Integer Programming

In this subsection, we formulate the GRLM problem as an integer program based on our energy consumption model. All the field devices are powered by batteries, while the access points and the gateway are connected to wired power sources. We define the *load* of a field device as its expected energy consumption rate, which depends on the rates of flows passing it. Then the lifetime of a field device is modeled as the initial battery divided by load. Here we denote the initial battery capacity of a device v_i as B_i, and the load as L_i. We use γ_i to denote the *normalized load* of v_i, defined as $\frac{L_i}{B_i}$. For access points and the gateway, batteries are set to be infinity. Our goal is to maximize the minimum lifetime among all devices, which is expressed as $\max \min_i \frac{B_i}{L_i}$. This objective function can be transformed to minimize the maximum normalized load $\gamma_i = \frac{L_i}{B_i}$. Hence the GRLM problem can be formulated as $\min \max_i \gamma_i$. We use Γ to denote the upper bound of γ_i in (5f) below, and the objective function becomes $\min \Gamma$.

$$\text{Objective:} \quad \text{minimize} \quad \Gamma$$

$$\sum_{\overrightarrow{s_k j} \in E} x^k_{s_k,j} = 1 \tag{5a}$$

$$\sum_{\overrightarrow{ji} \in E} x^k_{j,i} + \delta_{i,s_k} = \sum_{\overrightarrow{ij} \in E} x^k_{i,j} + \delta_{i,d_k}, \forall i \in V \tag{5b}$$

$$\sum_{\overrightarrow{ji} \in E} y^k_{j,i} + \sum_{\overrightarrow{ij} \in E} x^k_{i,j} = \sum_{\overrightarrow{ij} \in E} y^k_{i,j}, \forall i \in V \setminus \{d_k\} \tag{5c}$$

$$\sum_{\overrightarrow{ip} \in E \text{ and } p != j} y^k_{i,p} >= x^k_{i,j}, \quad \forall \overrightarrow{ij} \in E \tag{5d}$$

$$\gamma_i = \sum_k \frac{r_k}{B_i} \Big(\sum_{\overrightarrow{ij} \in E} x^k_{i,j} E_t + \sum_{\overrightarrow{ji} \in E} x^k_{j,i} E_r + \sum_{\overrightarrow{ji} \in E} y^k_{j,i} E_{rb} \Big) \tag{5e}$$

$$\gamma_i \le \Gamma, \forall i \in V \tag{5f}$$

$$x^k_{i,j} \in \{0,1\}, y^k_{i,j} \in \mathbb{Z}_{\ge 0}, \forall \overrightarrow{ij} \in E \tag{5g}$$

We formulate the integer program as follows. The primary path variable $x^k_{i,j}$ is a binary variable. If a link $\overrightarrow{ij}$ is used in the primary path for flow k, then $x^k_{i,j}$ equals 1, otherwise, it equals 0. The same rule is applied to backup path variable $y^k_{i,j}$. However, since multiple backup paths may share a same link, the backup path variable $y^k_{i,j}$ is an non-negative integer variable, which could be larger than 1.

First, there is only one link used in the primary path among all outgoing links of the source s_k (5a). Then the conservation constraint (5b) says the sum of outgoing primary path variables equals the sum of incoming primary path variables at every device except the source s_k and the destination d_k, where $\delta_{i,j}$ is the Kronecker delta function [35]. Here $\delta_{i,j}$ equals 1 if i and j are the same, and 0 otherwise.

The conservation constraint for backup path variables is different from the constraint for primary path variables because the backup paths do not start from the source of the flow. They start from devices on the primary path. For backup paths, there are two cases. For a device on a backup path but not on the primary path (e.g. network device z in Figure 1(b)), it follows the same conservation constraint as the primary path variables, which means the sum of outgoing backup path variables equals the sum of incoming backup path variables. For a network device which is on both the backup path and primary path (e.g. u in Figure 1(b)), it does not have any incoming backup path. However, it still has an outgoing backup path, and the amount of backup path variables equals the amount of outgoing primary path variables. To incorporate both cases, we formulate this requirement in constraint (5c), which specifies that the sum of outgoing backup path variables from a device equals the sum of incoming backup path variables plus the sum of outgoing primary path variables.

Since the backup link should not coincide with the primary link for the same packet, constraint (5d) is added to make sure that the backup path of a link on the primary path does not use this link. Constraint (5e) calculates the normalized load γ_i of each device i. And constraint (5f) guarantees that normalized loads of all network devices are no larger than Γ. The objective is to minimize the maximum normalized load, which is equivalent to minimizing Γ.

B. Linear Programming Relaxation

For large scale networks, an integer programming based solution does not scale well. We use a linear programming relaxation approach to speed up the route calculation. We solve the problem in two phases. In the first phase we focus on the primary path variables. In the beginning, we relax each primary path variables $x_{i,j}^k$ from binary to real number within $[0, 1]$, and relax each backup path variable $y_{i,j}^k$ from non-negative integer to non-negative real number. Then we solve the problem and obtain the solution. We round the variables to 1 if they are above a threshold θ, otherwise round it to 0. For each flow f_k, we want to find the highest threshold θ_k for primary path variables such that there exists a path from the source to the destination. We use a gradient based algorithm to find this threshold. The step size is 0.05. The initial threshold is 0.5. If a path is found, then we increase the threshold by one step. Otherwise, we decrease the threshold by one step. The algorithm terminates if no higher threshold can be found. We repeat this algorithm for each flow and will get a primary path for each flow.

After the first phase, we obtain primary paths for all flows. In the second phase, we keep primary path variables fixed and relax backup path variables to non-negative real numbers. After we get the results with non-negative backup path variables, we round them to 1 following a similar approach in the first phase. For each flow, starting from the first backup path (whose source is the source of the flow) to the last backup path (whose source is the last hop of the flow destination in the primary path), we use the gradient based algorithm to find the highest threshold that allows a path from the source to the destination. We use the GNU Linear Programming Kit (GLPK) [36] to solve the integer program and its linear programming relaxation.

C. Greedy Heuristic

To further speed up the routing process, we introduce an efficient greedy heuristic. When selecting a graph route, our greedy heuristic selects the graph route with small normalized load, which is the expected energy consumption rate divided by the initial battery capacity. The basic idea is to let the devices with higher battery capacity carry more network traffic under the graph routing setup. To solve this problem more efficiently, we use an algorithm inspired by Dijkstra's shortest path algorithm [37]. For each flow, starting from the destination, we gradually update each device's normalized load. Each time we select a device with the smallest normalized load and update the normalized loads of its neighbors. The normalized load is the key concept in our algorithm.

Our greedy heuristic runs iteratively. In each iteration, we select graph routes for flows from the highest rate to the lowest rate. For each flow, we pick up a graph route with minimum normalized load. Our iterative algorithm stops if the maximum normalized load increases or the decrease of maximum normalized load is less than a threshold Γ_{th}, which is set to $\frac{\min_k r_k E_{rb}}{\max_i B_i}$ in our current implementation. For each flow, the Minimum Load Graph Route (MLGR) function in Algorithm 1 is called to find a graph route. We use an algorithm similar to Dijkstra's shortest path algorithm, where normalized load is used like the edge weight in Dijkstra's algorithm. Within MLGR, we use λ to denote temporary normalized loads for devices in the network. After MLGR return a graph route, the related devices on the graph route will update their normalized load γ with temporary normalized load λ. We maintain a queue Q which includes all network devices with their updated normalized load. We use a map H to track last hop devices.

At each step, a device u with minimum temporary normalized load λ_u is picked up from the queue. If its temporary normalized load λ_u equals ∞, then the remaining devices cannot be added to the primary path. Then MLGR function fails to find a graph route for current flow and returns ∞. If u is the source, then the MLGR function adds it to the primary path and returns its temporary normalized load λ_u. We can obtain the primary path by tracing back through last hop map H.

If none of above case is true, we will check u's neighbors one by one to see whether they can be added into the primary path. For each neighbor v, we use the Minimum Load Source Route (MLSR) function in Algorithm 2 to check whether there is a path from v to the destination d in the graph $G' = (V, E \setminus \{\overrightarrow{vu}\})$ and return the one with the minimum normalized load.

Algorithm 1: Minimum Load Graph Route

Function *MLGR*(G, s, d, r, γ, B)

Input : A graph $G(V, E)$, source s, destination d, flow rate r, normalized load vector γ, battery vector B

Variable: Last hop vector H, Backup Paths P, temporary normalized load λ

Output : Normalized load of the graph route picked up by the algorithm (∞ if no graph route is found)

for ***each*** *vertex* $v \in V$ **do**
- $\lambda_v = \infty$;
- $H_v = NULL$;
- $P_v = \emptyset$;
- add v to Q;

$\lambda_d = \gamma_d + \frac{rE_r}{B_d}$;

while Q ***is not*** *empty* **do**
- $u = v \in Q$ with minimum λ_v;
- remove u from Q;
- **if** $\lambda_u == \infty$ **then**
  - return ∞;
- **if** $u ==$ *source* **then**
  - return λ_u;
- **for** ***each*** *neighbor* v *of* u *within* Q **do**
  - Graph $G' = (V, E \setminus \{\overrightarrow{vu}\})$;
  - $\gamma_{backup} = MLSR(G', v, d, P_v, r, \gamma, B)$;
  - **if** $\gamma_{backup} \neq \infty$ **then**
    - $alt = max(\lambda_u, \gamma_v + \frac{rE_t + rE_r}{B_v}, \gamma_{backup})$;
    - **if** $alt < \gamma_v$ **then**
      - $\lambda_v =$ alt;
      - $H_v = u$;

Algorithm 2: Minimum Load Source Route

Function *MLSR*($G, s, d, P_s, r, \gamma, B$)

Input : A graph $G(V, E)$, source s, destination d, flow rate r, normalized load vector γ, battery vector B

Variable: Last hop vector H, temporary normalized load λ

Output : Normalized load of the source route picked up by the algorithm (∞ if no graph route is found)

for ***each*** *vertex* $v \in V$ **do**
- $\lambda_v = \infty$;
- $H_v = NULL$;
- add v to Q;

$\lambda_d = \gamma_d + \frac{rE_{rb}}{B_d}$;

while Q ***is not*** *empty* **do**
- $u = v \in Q$ with minimum λ_v;
- remove u from Q;
- **if** $\lambda_u == \infty$ **then**
  - return ∞;
- **if** $u ==$ *source* **then**
  - return λ_u;
- **for** ***each*** *neighbor* v *of* u *within* Q **do**
  - $alt = max(\lambda_u, \gamma_v + \frac{rE_{rb}}{B_v})$;
  - **if** $alt < \lambda_v$ **then**
    - $\lambda_v =$ alt;
    - $H_v = u$;

We update the temporary normalized load of device v based on its new normalized load $\gamma_v + \frac{rE_t + rE_r}{B_v}$, its parent u's temporary normalized load λ_u and the normalized load of the backup path γ_{backup}.

Here the MLSR function is a single path version of MLGR. At each step, it picks up the device u with minimum temporary normalized load λ_u. If λ_u equals ∞, then the source s cannot be connected to the destination d, and MLSR function returns ∞. If the source s is picked up with a temporary normalized load λ_s less than ∞, then s is connected with the destination d, and MLSR function returns λ_s. The MLSR function can obtain the path from the last hop trace. If none of above case is true, the MLSR function will check device u's neighbors and update their temporary normalized loads according to u's temporary normalized load λ_u.

Since MLSR takes the form of the Dijkstra's algorithm, its time complexity is $O(|E| + |V|log|V|)$. MLGR is a nested version of MLSR, its time complexity is $O(|E|(|E| + |V|log|V|) + |V|log|V|) = O(|E|^2 + |E||V|log|V|)$. The time complexity of each iteration is $O(N|E|^2 + N|E||V|log|V|)$, given there are N flows. The number of iterations is bounded as the upper bound of normalized load $\Gamma_{up} = \frac{\sum_k r_k(E_t + E_r)}{\min_i B_i}$ divided by the threshold of normalized load change $\Gamma_{th} = \frac{\min_k r_k E_{rb}}{\max_i B_i}$. The time complexity of our greedy heuristic is $O(\frac{\Gamma_{up}}{\Gamma_{th}} N|E|^2 + \frac{\Gamma_{up}}{\Gamma_{th}} N|E||V|log|V|)$

VI. Evaluation

We evaluate our routing algorithms through both experiments on a physical wireless sensor-actuator network (WSAN) testbed and simulations. We compare our Integer Programming approach (IP), Linear Programming approximation (LP), and Greedy Heuristic algorithm (GH) with the reliable and real-time routing (RRC) approach that Han et al. proposed in [26] and Dijkstra's shortest path algorithm (SP) [37]. RRC builds uplink and downlink routing graphs for all flows based on hop count. We build a graph route on top of RRC's routing graph by selecting one path as the primary path and using available alternative paths as backup paths. Because RRC does not fully explore the network to find backup paths, some network devices on the primary path do not have backup paths. In SP, we first run Dijkstra's algorithm to get the primary path with the shortest hop count, then run the same algorithm to select

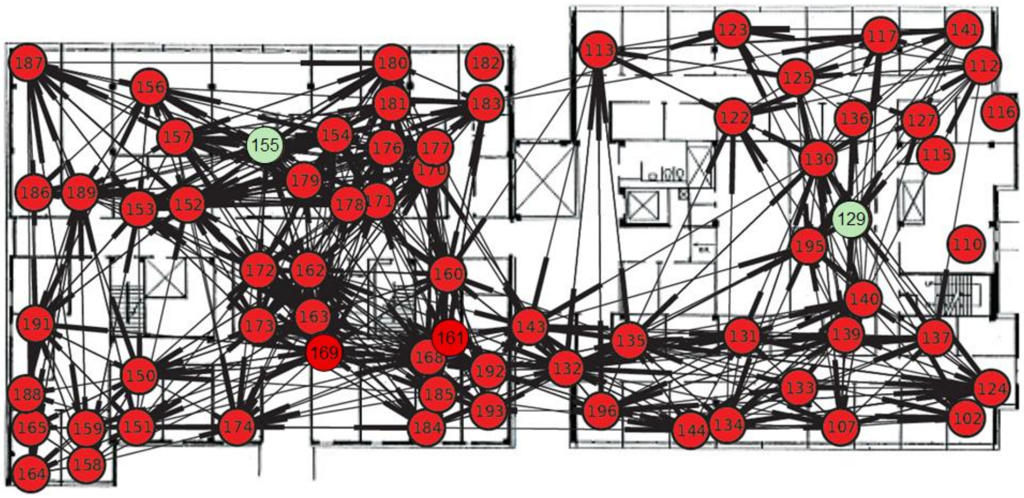


Fig. 4: Topology of the WSAN Testbed

backup paths for each network device on the primary path while avoiding outgoing link on the primary path.

A. Experiments on a WSAN Testbed

We evaluate our routing designs on an indoor WSAN testbed consisting of 63 TelosB motes equipped with TI CC2420 radio. The testbed is located on the fifth floors of two adjacent buildings on the Washington University campus. Each mote in the testbed is connected to a wired backbone network that helps facilitate the experiments and measurements without interrupting the wireless communication. Each mote in the testbed runs the WirelessHART protocol stack presented in [13]. The protocol stack is implemented in TinyOS 2.1.2 on top of the CC2420x radio driver, which is compatible with the IEEE 802.15.4 standard. The protocol stack supports the key WirelessHART network features including a multi-channel TDMA MAC protocol and source and graph routing protocols. Field devices are time synchronized using the Flooding Time Synchronization Protocol (FTSP) [38].

Figure 4 shows the topology of our testbed. We select motes 129 and 155 (green circles) as the access points, which are physically connected to a root server (gateway). The other motes act as field devices (red circles). The network manager is a software running on the root server. For each link in the testbed, we measured its *packet reception ratio (PRR)* by counting the number of received packets among 250 packets transmitted over the link. Following the practice of industrial deployment, we only add links with PRR higher than 90% in all channels used to the topology of the testbed. To avoid channels occupied by the campus Wi-Fi, we use IEEE 802.15.4 channels 11 to 15 in our experiments.

We generate 8 flows in our experiment. The period of each flow is picked up from the range of $2^{0 \sim 7}$ seconds, which are typical periods used in the process industry as specified in the WirelessHART standard [1]. The length of the hyper-period is 128 seconds. The relative deadline of each flow is equal to its period. We run the experiments for 100 rounds of hyper-

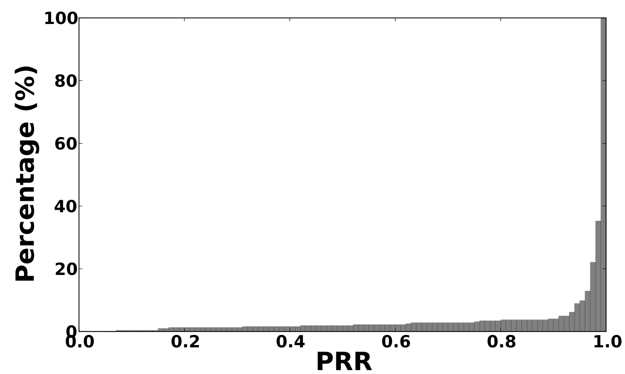


Fig. 5: Histogram of Link Qualities

period (around 3 hours) to collect at least 100 periods of data traces for each flow. Based on the data traces we collected, we evaluate our proposed approaches in terms of *delivery ratio* and *expected network lifetime*. The delivery ratio of a flow is defined as the fraction of packets that are successfully delivered to the destination.

The expected network lifetime is calculated based on the collected traces. Because TelosB motes in the testbed are wire powered, we assign virtual battery capacity for each mote randomly from the range of $8000J$ to $9000J$, where $8640J$ is the typical capacity of two AA batteries. We analyze the collected data traces from the experiments to obtain the energy consumption of each network device in 100 rounds of hyper-period. Based on that, we project the expected network lifetime.

To study reliability, we first measure the link qualities. Figure 5 shows the histogram of link qualities (PRR) of 327 links we used in our experiments. We collect the PRR of each link on all 4 channels. Although our link selection process only selects links with PRR higher than 90%, we find some links have much lower PRR than the 90% threshold at run time. For example, link $\overrightarrow{158\ 156}$ under channel 12 has the lowest PRR

Routing	Algorithm	Flow Index							
		1	2	3	4	5	6	7	8
Source	SP	0.993	0.874	0.898	1.0	1.0	1.0	1.0	1.0
	RRC	0.992	0.760	0.833	0.996	0.989	0.994	1.0	1.0
	GH	0.996	0.886	0.897	0.997	0.998	1.0	1.0	1.0
	LP	0.997	0.827	0.896	0.998	0.989	1.0	1.0	1.0
Graph	SP	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	RRC	0.996	0.990	0.988	1.0	1.0	1.0	1.0	1.0
	GH	0.998	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	LP	0.998	1.0	1.0	1.0	1.0	1.0	1.0	1.0

TABLE III: Delivery Ratios of Flows

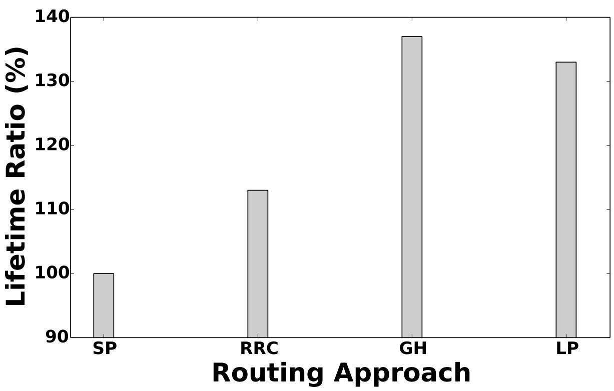


Fig. 6: Expected Network Lifetime relative to SP

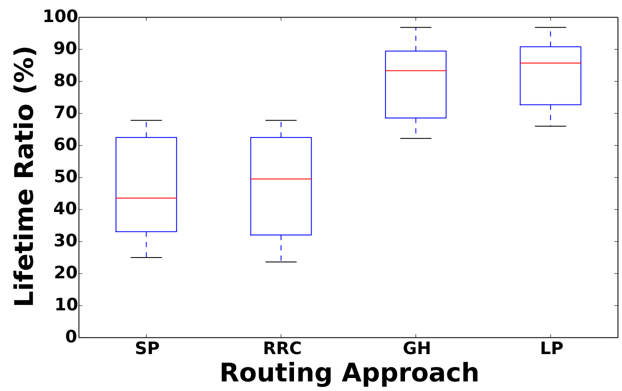


Fig. 7: Expected Lifetime Relative to Optimal Solution

of 7%. The dynamics of wireless links suggest it is necessary to have path diversity.

Table III shows the delivery ratios of all 8 flows under both graph routing and source routing. We use the primary path of the graph route as the source route of each flow. Our results show that graph routing provides a better delivery ratio than source routing. For example, the delivery ratios of all four routing algorithms for flow 2 under source routing are below 0.9, which can be unacceptable for industrial applications. In comparison, their delivery ratios under graph routing are at least 0.99. Our results demonstrate the effectiveness of redundant routes in improving reliability. We also found in RRC's graph routes for flow 1, 2, and 3, 50% of the links on the primary paths do not have backup paths. The lack of backup paths makes RRC vulnerable to link dynamics.

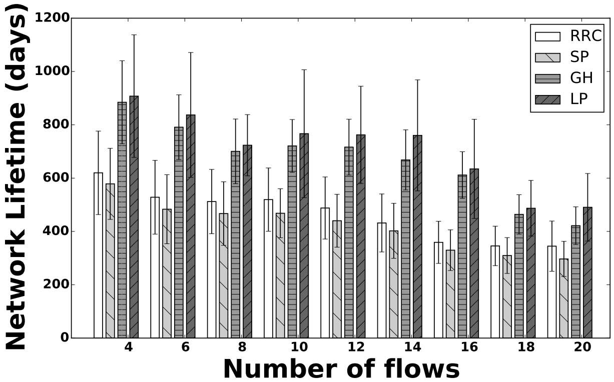
Figure 6 presents the expected lifetimes of different routing approaches normalized to that under SP. Because it takes too long to compute routes for the testbed topology under the IP approach, we do not have the results of IP. The results show SP has the shortest expected lifetime and GH has the longest expected lifetime. GH's expected lifetime is 37% longer than SP, and LP's expected lifetime is 33% longer than SP. RRC achieves a lifetime longer than SP and shorter than LP. Our results show GH and LP enhance the expected network lifetime compared to SP and RRC.

B. Simulations

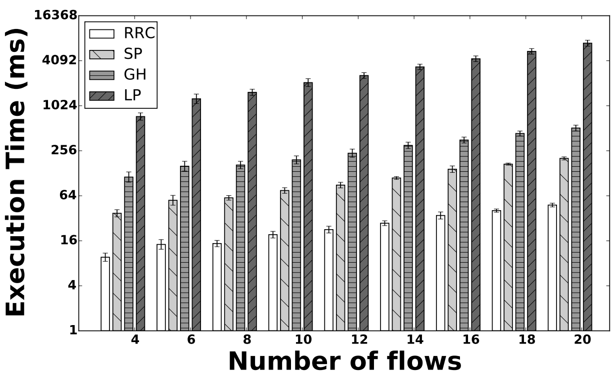
We compare different routing algorithms through simulation in this subsection. The simulator is written in C++ and follows the design of our testbed. All simulations are performed on a MacBook Pro laptop with 2.4 GHz Intel Core 2 Duo processor. We use a trace driven simulation. All data collected in experiments are imported into the simulator. Similar to the experiments, we use links with PRR higher than 90%. The period of each flow is randomly generated from the range of $2^{0 \sim 7}$ seconds. And we use IEEE 802.15.4 channels 11 to 15 in our simulations. In the simulation, when a packet is transmitted on a link, the simulator uses a data point from the traces collected in experiments. We generate different results by randomly assigning network device battery capacities from $8000J$ to $9000J$.

Because the IP approach is computationally expensive, which requires more than 24 hours to complete its execution in simulations based on our testbed topology, we evaluate all five routing algorithms on a small topology consisting of 10 motes and 20 links from our testbed. Figure 7 shows the lifetime ratios of SP, RRC, GH, and LP relative to IP. The median of GH and LP are 83% and 85% of the optimal lifetime under IP. Compared with IP, SP and RRC have 44% and 47% median lifetime ratios. The figure shows that GH and LP outperform SP and RRC in terms of the expected lifetime.

We further test our algorithms with a large number of flows



(a) Network Lifetime



(b) Execution Time (in log scale)

Fig. 8: Simulation Results on Testbed Topology

in simulation on the entire testbed topology. We evaluate our routing designs on different numbers of flows by increasing the number of source and destination pairs. We randomly select motes as sources and destinations. We compare four routing designs in terms of *network lifetime* and *execution time*.

Figure 8(a) shows the expected network lifetime of different routing designs on the entire testbed topology. In general, network lifetime decreases as the number of flows increases, because more flows bring more energy consumption to network devices. Furthermore, results show SP consistently has the shortest network lifetime. RRC's network lifetime is longer than SP but shorter than GH and LP. GH and LP provide longer network lifetime than the other two. The figure shows GH and LP can improve the network lifetime over RRC by up to 63% and 76%.

The computational complexity of the four routing algorithms are presented in Figure 8(b). The figure compares execution times of four algorithms in log scale. The results show LP is much slower than the other three algorithms. This happens because linear programming solver in general is slower than straightforward routing algorithms such as SP and GH. Besides LP, GH has the highest time complexity. However, the maximum execution time of GH in our simulation is approximately 0.35 seconds, which is acceptable to WirelessHART networks that need to reconfigure a network only in response to topology change.

VII. Conclusion

As IoT starts gaining adoption in industrial applications, industrial WSANs provide critical communication infrastructure for industrial automation. Industrial WSANs face significant challenges in achieving long-term reliable communication in harsh environments. While the WirelessHART standard adopts graph routing to enhance network reliability, the problem of maximizing network lifetime for graph routing becomes a critical open problem. This paper introduces and formulates the network lifetime maximization problem for graph routing. We present an optimal graph routing algorithm based on integer programming, and two efficient algorithms based on linear programming relaxation and greedy heuristic, respectively. We have implemented our graph routing algorithms on a physical WSAN network testbed. Experimental results on the testbed and in simulations show the linear relaxation and greedy heuristic can improve the network lifetime by up to 60% when compared to an existing graph routing algorithm. Moreover, the greedy heuristic requires significantly lower computation time, making it particularly suitable for WirelessHART networks that may compute graph routes frequently when facing network changes in open environments.

Acknowledgments

This work is supported, in part, by NSF through grant 1320921 (NeTS) and the Fullgraf Foundation.

References

[1] "WirelessHART specification," 2007, http://www.hartcomm2.org.

[2] "ISA100: Wireless Systems for Automation," https://www.isa.org/isa100/.

[3] J. Song, S. Han, A. K. Mok, D. Chen, M. Lucas, and M. Nixon, "WirelessHART: Applying Wireless Technology in Real-Time Industrial Process Control," in *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'08)*, April 2008.

[4] "Emerson's WirelessHART report," http://www2.emersonprocess.com/en-us/plantweb/wireless/pages/wirelesshomepage-flash.aspx.

[5] S. Singh, M. Woo, and C. S. Raghavendra, "Power-aware routing in mobile ad hoc networks," in *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'98)*, 1998.

[6] I. Stojmenovic and X. Lin, "Power-Aware Localized Routing in Wireless Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 11, 2001.

[7] J.-H. Chang and L. Tassiulas, "Energy conserving routing in wireless ad-hoc networks," in *IEEE International Conference on Computer Communications*, 2000.

[8] ——, "Maximum Lifetime Routing in Wireless Sensor Networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 4, 2004.

[9] C. Wu, R. Yuan, and H. Zhou, "A Novel Load Balanced and Lifetime Maximization Routing Protocol in Wireless Sensor Networks," in *IEEE Vehicular Technology Conference (VTC-Spring'08)*, May 2008.

[10] Q. Li, J. Aslam, and D. Rus, "Online Power-Aware Routing in Wireless Ad-hoc Networks," in *ACM International Conference on Mobile Computing and Networking (MobiCom'01)*, 2001.

[11] S. Doshi, S. Bhandare, and T. X. Brown, "An On-demand Minimum Energy Routing Protocol for a Wireless Ad Hoc Network," *Mobile Computing and Communications Review*, vol. 6, no. 3, 2002.

[12] K. Kalpakis, K. Dasgupta, and P. Namjoshi, "Efficient Algorithms for Maximum Lifetime Data Gathering and Aggregation in Wireless Sensor Networks," *Computer Networks*, vol. 42, pp. 697 –716, 2003.

[13] M. Sha, D. Gunatilaka, C. Wu, and C. Lu, "Implementation and Experimentation of Industrial Wireless Sensor-Actuator Network Protocols," in *European Conference on Wireless Sensor Networks (EWSN'15)*, February 2015.

[14] C. Lu, A. Saifullah, B. Li, M. Sha, H. Gonzalez, D. Gunatilaka, C. Wu, L. Nie, and Y. Chen, "Real-Time Wireless Sensor-Actuator Networks for Industrial Cyber-Physical Systems," *Proceedings of the IEEE*, 2016.

[15] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "Real-Time Scheduling for WirelessHART Networks," in *IEEE Real-Time Systems Symposium (RTSS'10)*, December 2010.

[16] O. Chipara, C. Wu, C. Lu, and W. Griswold, "Interference-Aware Real-Time Flow Scheduling for Wireless Sensor Networks," in *Euromicro Conference on Real-Time Systems (ECRTS'11)*, July 2011.

[17] C. Wu, Y. Xu, Y. Chen, and C. Lu, "Submodular Game for Distributed Application Allocation in Shared Sensor Networks," in *IEEE Conference on Computer Communications (INFOCOM'12)*, March 2012.

[18] B. Li, Z. Sun, K. Mechitov, C. Lu, S. Dyke, G. Agha, and B. Spencer, "Realistic Case Studies of Wireless Structural Control," in *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS'13)*, April 2013.

[19] C. Wu, M. Sha, D. Gunatilaka, A. Saifullah, C. Lu, and Y. Chen, "Analysis of EDF Scheduling for Wireless Sensor-Actuator Networks," in *IEEE/ACM Symposium on Quality of Service (IWQoS'14)*, May 2014.

[20] A. Saifullah, C. Wu, P. B. Tiwari, Y. Xu, Y. Fu, C. Lu, and Y. Chen, "Near Optimal Rate Selection for Wireless Control Systems," *ACM Transactions on Embedded Computing Systems (TECS'14)*, April 2014.

[21] A. Saifullah, D. Gunatilaka, P. Tiwari, M. Sha, C. Lu, B. Li, C. Wu, and Y. Chen, "Schedulability Analysis under Graph Routing for WirelessHART Networks," in *IEEE Real-Time Systems Symposium (RTSS'15)*, December 2015.

[22] B. Li, L. Nie, C. Wu, H. Gonzalez, and C. Lu, "Incorporating Emergency Alarms in Reliable Wireless Process Control," in *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS'15)*, April 2015.

[23] B. Li, Y. Ma, T. Westenbroek, C. Wu, H. Gonzalez, and C. Lu, "Wireless Routing and Control: a Cyber-Physical Case Study," in *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS'16)*, April 2016.

[24] J. Zhao, Z. Liang, and Y. Zhao, "ELHFR: A graph routing in industrial wireless mesh network," in *International Conference on Information and Automation (ICIA'09)*, June 2009.

[25] G. Gao, H. Zhang, and L. Li, "A Reliable Multipath Routing Strategy for WirelessHART Mesh Networks Using Subgraph Routing," *Journal of Computational Information Systems*, vol. 9, 2013.

[26] S. Han, X. Zhu, A. K. Mok, D. Chen, and M. Nixon, "Reliable and Real-time Communication in Industrial Wireless Mesh Networks," in *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'11)*, April 2011.

[27] C. Wu, D. Gunatilaka, M. Sha, and C. Lu, "Conflict-Aware Real-Time Routing for Industrial Wireless Sensor-Actuator Networks," Washington University in St. Louis, Tech. Rep. WUCSE-2015-005 (2015), 2015, http://openscholarship.wustl.edu/cse_research/507/.

[28] Y. Wang, S. Zhang, and X. Lin, "Distributed low-power dissipation routing algorithm based on WirelessHART," *Modern Electronics Technique*, pp. 60–64, 2013.

[29] A. A. Memon and S. H. Hong, "Minimum-Hop Load-Balancing Graph Routing Algorithm for Wireless HART," *International Journal of Information and Electronics Engineering*, vol. 3, pp. 221–225, 2013.

[30] S. Zhang, A. Yan, and T. Ma, "Energy-Balanced Routing for Maximizing Network Lifetime in WirelessHART," *International Journal of Distributed Sensor Networks*, pp. 1–8, 2013.

[31] Q. Zhang, F. Li, L. Ju, Z. Jia, and Z. Zhang, "Reliable and Energy Efficient Routing Algorithm for WirelessHART," in *Algorithms and Architectures for Parallel Processing*, ser. Lecture Notes in Computer Science, 2014, vol. 8630, pp. 192–203.

[32] "IEEE 15.4 standard," 2011, https://standards.ieee.org/about/get/802/802.15.html.

[33] "CC2420 documentation," http://www.ti.com/lit/ds/symlink/cc2420.pdf.

[34] S. Fortune, J. Hopcroft, and J. Wyllie, "The directed subgraph homeomorphism problem," *Theoretical Computer Science*, vol. 10, no. 2, pp. 111 – 121, 1980.

[35] "Kronecker delta function," http://en.wikipedia.org/wiki/Kronecker_delta.

[36] "GNU Linear Programming Kit," http://www.gnu.org/software/glpk/.

[37] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*, 2nd ed. McGraw-Hill Higher Education, 2001.

[38] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi, "The Flooding Time Synchronization Protocol," in *ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*, November 2004.