Wireless Netw (2018) 24:3205–3219
https://doi.org/10.1007/s11276-017-1530-1

Convergecast scheduling and cost optimization for industrial wireless sensor networks with multiple radio interfaces

Xi Jin[1] · Huiting Xu[3] · Changqing Xia[1] · Jintao Wang[1] · Peng Zeng[1,2]

Published online: 26 May 2017

Abstract Industrial wireless sensor networks have been widely deployed in many industrial systems. The main communication paradigm of such systems, known as convergecast, is to converge sensing data to a centralized manager. The rapid and reliable data convergecast is essential to the industrial production. Multiple radio interfaces on a network device and convergecast scheduling algorithms can effectively reduce convergecast delay. Existing works confine to the convergecast based on linear- and tree-based routing. Compared to the two routing schemes, graph routing is more reliable. Although the graph routing gains more popularity in industrial networks due to its better reliability, few works have addressed its temporality performance. On the other hand, the number of radio interfaces also impacts on the convergecast delay. In this paper, we present a holistic framework to solve how to use multiple radio interfaces to converge data. First, we propose a convergecast scheduling algorithm for industrial wireless sensor networks with multiple radio interfaces. Second, based on our proposed scheduling algorithm, we propose an optimal algorithm and a fast heuristic algorithm to minimize the number of radio interfaces under the temporality constraint of industrial production. Evaluations show that all our algorithms perform closely to the optimal solution.

Keywords Industrial wireless sensor networks · Multiple radio interfaces · Raw data convergecast scheduling · Cost minimizing

1 Introduction

Wireless sensor networks have been widely used in industrial applications, e.g., monitoring systems in industrial production [1–3], collaborative location and tracking systems [4, 5]. In such systems, monitoring data generated by each sensor device are sent to a centralized manager via the wireless network. The many-to-one communication paradigm is known as *convergecast*. Industrial applications usually have stringent requirement on the performance of wireless sensor networks. Delayed convergecast may degrade the control performance and even cause catastrophic consequences. Figure 1 shows a wireless sensor network in a cement factory. The temperature of the rotary kiln is very important for cement manufacture. The temperature message must be sent to the control room as soon as possible. If a data packet with high-temperature message is delayed, workers cannot take measures in time and thus the defective product rate increases and the kiln may explode.

✉ Peng Zeng
zp@sia.cn

Xi Jin
jinxi@sia.cn

Huiting Xu
xuht_neu@hotmail.com

Changqing Xia
xiachangqing@sia.cn

Jintao Wang
wangjintao@sia.cn

[1] Laboratory of Networked Control Systems, Shenyang Institute of Automation, Chinese Academy of Science, Shenyang 110016, Liaoning, China

[2] Shenyang Institute of Automation, Guangzhou, Chinese Academy of Sciences, Guangzhou 511458, Guangdong, China

[3] Northeastern University, Shenyang 110819, China

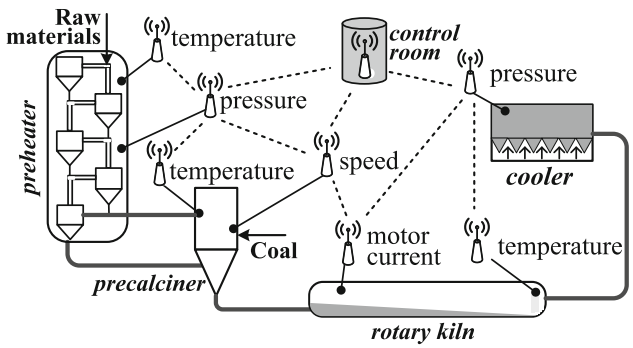


Fig. 1 A wireless network in a cement factory

To reduce the delay of data convergecast, multiple radio interfaces on a network device and novel convergecast scheduling methods have to be used. Two kinds of convergecast scheduling methods have been proposed. The first one is aggregated convergecast [6–8], in which sensing data are aggregated at intermediate nodes to reduce the number of packets. The less packets lead to less delay, but the aggregated convergecast may discard some information required by the industrial control. So we focus on the second type, in which the raw data are delivered from the source device to the sink without modification.

Some networks about using multiple radio interfaces to converge raw data have been proposed, e.g., only the sink has multiple radio interfaces [9–11] and all network nodes are equipped with multiple radio interfaces [12, 13]. However, all of these works focus on linear- [14] and tree-based [15] routing. While in industrial wireless protocols, the graph routing scheme has been more widely used than other routing schemes [16–18]. The main difference between the three types of routing paths is the number of predecessors and successors of forwarding nodes. In linear-based routing, each forwarding node only has a predecessor and a successor; in tree-based routing, a forwarding node has one successor and multiple predecessors; in graph-based routing, a node can have multiple successors and multiple predecessors. So the graph routing path can improve system reliability by routing packets around fault nodes and external interference. But to the best of our knowledge there are no previous works about the convergecast method based on the graph routing. When the routing path of a flow is a graph, part of the flow can be transmitted in parallel and the other part must be serial. The series-parallel hybrid feature do not exist in linear- and tree-based routing. Due to this feature, the existing approaches cannot be applied to industrial networks. On the other hand, the number of radio interfaces is a main factor influencing the convergecast delay. A radio interface cannot serve more than one packet simultaneously. Therefore, the number of radio interfaces equipped on a node indicates the parallelism degree of the node. If radio interfaces on a node cannot serve all waiting packets simultaneously, the extra delay will be introduced. So a small number of radio interfaces cannot converge data in time while excessive radio interfaces cause cost waste. In this paper, we present a holistic solution to solve how to apply multiple radio interfaces to converge data in industrial wireless sensor networks. We list our contributions as follows.

First, we study the convergecast scheduling problem in the wireless sensor network with multiple radio interfaces. In this problem, the number of radio interfaces is considered as a given parameter and will be determined in the second problem our paper focuses on. We translate the scheduling problem to the shortest path problem, and then design a method based on the spanning tree to optimize the scheduling. Evaluations show that our method outperforms existing ones, and the different between our method and the optimal solution is less than 10%.

Second, we find the minimal number of radio interfaces required to guarantee the given convergecast delay. To reduce the solution space, we analyze the upper and lower bounds of the number of radio interfaces for each node. Then based on the result of the first problem, we propose an optimal branch and bound algorithm and a fast heuristic algorithm to search the reduced solution space. Evaluations show that our heuristic algorithm is close to optimal, and the different between them is less than 3%.

The rest of paper is organized as follows. Section 2 introduces the related work. Section 3 describes our system model and problem statement. Section 4 presents our convergecast scheduling algorithm. Section 5 presents the cost optimization algorithm. Section 6 shows evaluations. Section 7 concludes this paper.

2 Related work

Multi-radio wireless networks have been widely studied in previous works. Generally, multiple homogeneous radio interfaces are used to improve the network throughput, e.g., [19–23]. On the other hand, some works use heterogeneous radio interfaces to achieve the tradeoff between energy efficiency and performance, e.g., [12, 24–26]. However, these previous works do not focus on the TDMA convergecast scheduling.

For the raw convergecast scheduling, the work in [27] shows the decision version of raw convergecast scheduling on single channel is NP-complete in a weak sense. Similarly, the work in [28] also focus on single channel and provide a distributed scheduling algorithm for tree networks. Then some works consider the multi-channel network, the work in [29] evaluates the impacts of power control on the number of time slots required by raw

convergecast scheduling. The work in [30] proposes a greedy heuristic approach to minimize the end-to-end delay for tree-based wireless sensor networks. The work in [1] presents a time-optimal method for binary-tree networks. And the work in [31] proposes a time- and channel-optimal method for linear networks. Based on the above two works, the work in [32] takes the impact of packet copying into account to enhance the channel utilization. The work in [33] proposes two scheduling algorithms for real-time flows with different periods and multiple linear paths. The same authors in [34] propose a fixed priority assignment algorithm for the same system model. Though the two works do not address the convergecast scheduling, their problem model can be reduced to the convergecast scheduling problem when all flows have the same period and are from sensors to the sink. The work in [35] investigates interference through experimental studies and implement a method in a practical system to improve the network performance. The work in [36, 37] uses cooperative communication technique to optimize the network throughput. However, all these above convergecast scheduling works are based on single radio interface. The authors of [9–11] consider how to use multiple radio interfaces to improve the network performance. In [9], their method MODESA is centralized. In [10], they extend the method MODESA to an adaptive strategy. In [11], they propose a distributed scheduling algorithm. Additionally, the work in [13] consider how to use dual-radio wireless sensor networks to collect data. But these works address linear- and tree-based routing path and are not suitable for reliable industrial networks.

The problem of minimizing the cost of wireless networks has been studied as a part of the network planning. For example, the work in [38] minimizes the total installation cost for wireless indoor networks. The work in [39] minimizes the number of sensor nodes in the oil pipeline monitoring system. The work in [40] minimizes the cost of installing access point box and radio interfaces. But the previous works do not consider how to minimize the number of radio interfaces under the convergecast delay constraint.

3 System overview and problem statement

The wireless protocols WirelessHART [17, 41] and WIA–PA [18] are widely used in industrial applications. To improve the reliability and temporality of the industrial systems, they all support the following techniques: Time Division Multiple Access (TDMA), multi-channel, centralized manager and graph routing scheme. According to these techniques, we present our system model in the following. Note that our system model is fully compatible with the IEEE 802.15.4e MAC behavior mode TSCH (Time Slotted Channel Hopping) [42]. Therefore, it can be used in networks that support the TSCH mode, such as all WirelessHART networks and part of WIA–PA networks.

3.1 Network model

We consider an industrial wireless sensor network characterized by $W = <N, L>$:

- A network consists of sensor nodes and a sink node, which is connected to a centralized network manager. The node set is denoted by $N = \{n_1, n_2, \ldots\}$.
- Matrix $L : N \times N$ is the set of links. If the nodes n_i and n_j can directly communicate with each other, the element l_{ij} in the matrix L is equal to 1; otherwise, $l_{ij} = 0$. The set L presents the network topology.

The number of radio interfaces equipped on the node n_i is r_i, i.e., the node n_i can serve at most r_i packets simultaneously. The radio interface set is $R = \{r_1, r_2, \ldots\}$. We define the cost of radio interfaces as the sum of all r_i, i.e., $C = \sum_{\forall r_i \in R} r_i$.

The network is based on TDMA scheme. Each time slot allows an one-hop packet transmission and its acknowledgement to be transmitted. For simplicity, we call one-hop packet transmission and its acknowledgement as a *transmission*. If a transmission waits to be scheduled on a node, it is called as *released* transmission. Our network model supports 16 non-overlapping channels, which are defined in the IEEE 802.15.4e protocol and support by the WirelessHART protocol and the WIA–PA protocol. But not all of them can always be accessed due to external interference. We use M $(1 \leq M \leq 16)$ to denote the number of available channels. Each channel supports a transmission at a time slot.

3.2 Data flow model

The data flow set is denoted by $F = \{f_1, f_2, \ldots\}$. All flows generate packets on their source nodes at the same time, which is represented as the first time slot, then these packets are transmitted to the sink via their routing graphs. If the last packet arrives at the sink on the z-th time slot, the convergecast delay is z. The routing graph π_i of the flow f_i is constituted by links (as shown in Fig. 2-(a)). Every node along the routing graph has at most two links from itself to other nodes and attempts to send a packet at most three times (including the initial transmission and two retransmissions) [16, 17]. Each transmission solely occupies a time slot at the given channel. The initial transmission and the first retransmission are on the same link, whose type is defined as $L1$. The second retransmission is on another link

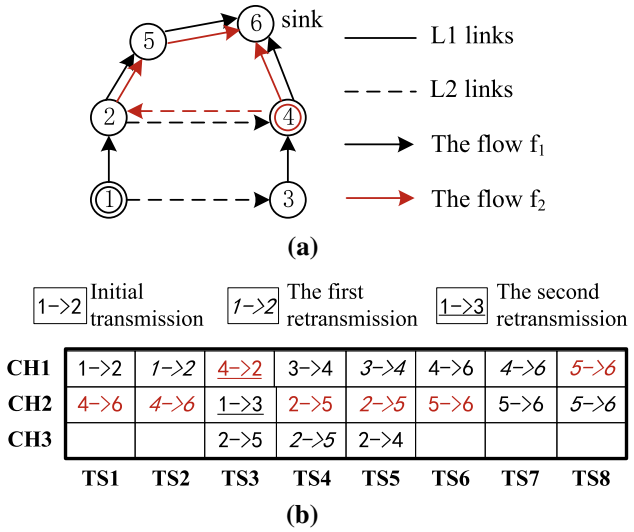


Fig. 2 A WirelessHART network and its convergecast scheduling. **a** Graph routing, **b** a convergecast scheduling

if there exists, whose type is *L2*. A node may receive the same packet from different paths, e.g. for the flow f_1, the node n_4 receives the same packet from the sub-paths $\{n_1, n_2, n_4\}$ and $\{n_1, n_3, n_4\}$. In this case, the time slots assigned to the node n_4 for sending the packet are after the time slots where the node n_4 receives all packets. Since a packet is delivered via multiple sub-paths, the graph routing can effectively improve the system reliability. For example, if the node n_5 breaks down and the link l_{25} is invalid, the packets of the flow f_1 can also be delivered to the sink via the paths $\{n_1, n_2, n_4, n_6\}$ and $\{n_1, n_3, n_4, n_6\}$.

3.3 Scheduling rule

The centralized network manager assigns a time slot and a channel to each transmission. Several transmissions constitute a packet transmitting, i.e., a packet is transmitted from its sources node to the sink at the assigned time slots and channels. These assignments are called as *convergecast scheduling*. The convergecast scheduling for the example in Fig. 2-(a) is shown in Fig. 2-(b), where each node is equipped with two radio interfaces, and *TS* and *CH* is the abbreviation of *Time Slot* and *CHannel*, respectively. The convergecast delay z is equal to 8. In our paper, the schedulable resource at one time slot on one channel is called as *sub-slot*. Each sub-slot corresponds to a block in Fig. 2-(b). The sub-slot at the i-th time slot and the j-th channel is called as the $((i-1) \times M + j)$-th sub-slot. When the network manager generates the convergecast scheduling, two types of conflicts must be avoided. The first one is the *scheduling conflict*, in which any two transmissions cannot be assigned the same sub-slot. If two transmissions are sent on the same channel simultaneously, their signals overlap and they cannot be distinguished. The second one is the *node conflict*. Since a radio interface only serves a transmission at a time slot, the number of transmissions served by a node at a time slot must be not larger than the number of radio interfaces equipped on the node.

3.4 Problem statement

In this paper, we solve the following two problems about using multiple radio interfaces to implement convergecast.

1. Given the network model W, the radio interface set R, the number of available channels M, the data flow set F and their routing graphs, our objective is to minimize the convergecast delay z under the conflict constraints (shown in Sect. 4).
2. Given the network model W, the number of available channels M, the given upper bound of convergecast delay $\bar{z}$, the data flow F and their graph routings, our objective is to minimize the radio interface cost under the conflict constraints such that all packets can be converged in the duration of $\bar{z}$ time slots (shown in Sect. 5).

4 Convergecast scheduling

There are complex relationships between the transmissions. Due to the releasing order and the resource conflict, some of the transmissions must be scheduled serially. But the others can be scheduled in parallel on multiple channels. Therefore, first, we use the unfolding graph to represent the relationships between them. Based on the unfolding graph, we propose a 0–1 Integer Linear Programming and a heuristic algorithm to solve the scheduling problem.

4.1 Unfolding graph

Algorithm 1 constructs the unfolding graph $G = <V, D, U>$ according to the data flow set F and their routing graphs. Each transmission corresponds to an element v_a in the vertex set V. And each vertex contains the source and destination nodes of the transmission (denoted by $v_a = <\alpha_a, \beta_a>$). The directed edge set D represents the releasing order. If there exists an element $d_{ab} \in D$, the vertex v_b can be released after the vertex v_a is scheduled. If several directed edges can compose a path from v_a to v_b, then the vertex v_a is the ancestor of the vertex v_b, denoted by $v_a \prec v_b$. The undirected edges in the set U denote that the releases of the two vertices v_a and v_b are unordered. For two vertices v_a and v_b, if $v_a \not\prec v_b$ and $v_b \not\prec v_a$, then $u_{ab} \in U$. Note that if $v_a \prec v_b$ and there does not exist $d_{ab} \in D$, then the edge from v_a to v_b does not appear in the graph G. This design reduces the graph size, but does not impact on the solution.

Algorithm 1 Construct the Unfolding Graph

Require: F, $\forall \pi_i$
Ensure: $G =< V, D, U >$
1: $a = 1$;
2: **for** each $f_k \in F$ **do**
3: $\forall n_i \in N$, $\Omega_i \leftarrow \emptyset$;
4: $Q.enqueue$(the source node of the flow f_k); // Q **is a FIFO queue**
5: **while** $!Q.isEmpty()$ **do**
6: $n_i = Q.dequeue()$;
7: **if** $\exists l_{ij} \in \pi_k$ and the type of l_{ij} is $L1$ **then**
8: $V \leftarrow V + \{v_a, v_{a+1}\}$; $D \leftarrow D + \{d_{a,a+1}\}$;
9: $\forall v_b \in \Omega_i, D \leftarrow D + \{d_{ba}\}$;
10: $\Omega_j \leftarrow \Omega_j + \{v_{a+1}\}$; $a+ = 2$;
11: **else if** $\exists l_{ij} \in \pi_k$ and the type of l_{ij} is $L2$ **then**
12: $V \leftarrow V + \{v_a\}$; $D \leftarrow D + \{d_{a-1,a}\}$;
13: $\Omega_j \leftarrow \Omega_j + \{v_a\}$; $a++$;
14: **if** $|\Omega_j| == I_{kj}$ **then**
15: $Q.enqueue(n_j)$;
16: **for** each $v_a \in V$ **do**
17: **for** each $v_b \in V \backslash \{v_g | v_g \prec v_a$ or $v_a \prec v_g\}$ **do**
18: $U \leftarrow U + \{u_{ab}\}$;
19: **return** $< V, D, U >$;

In Algorithm 1, first, the algorithm constructs the directed graph $< V, D >$ (lines between 2 and 15), and then generates the set U to describe the resource conflict between two transmissions (lines between 16 and 18). The symbol Ω_i ($\Omega_i \subseteq V$) denotes the set of vertices that have been added to the set V and whose destination nodes are the node n_i. The symbol I_{ki} denotes the in-degree of the node n_i in the routing graph π_k. When $|\Omega_i|$ is equal to I_{ki}, the packet has been transmitted to the node n_i from all links and the transmissions from the node n_i to other nodes can be released (lines between 14 and 15). For each flow, we traverse its routing graph. The transmissions and their releasing order are added to the directed graph one by one (lines 8, 9 and 12). Figure 3(b) shows the unfolding graph of the network shown in Fig. 3(a). In Fig. 3(a), there are two flows f_1 (in black) and f_2 (in red). Algorithm 1 first constructs two graphs for the two flows, respectively. For f_1, seven black vertices $\{v_1, \ldots, v_7\}$ are generated and connected in order of releasing. In the same way, the red graph of f_2 is constructed. Finally, the edges that indicate the resource conflict are added between the two graphs. The number of iterations of the **for** loop in line 2, the **while** loop in line 5 and the **forall** in line 9 is $O(|F|)$, $O(|N|)$ and $O(|N|)$, respectively. So the time complexity of Algorithm 1 is $O(|F||N|^2)$.

4.2 0–1 integer linear programming

Based on the unfolding graph, we formulate our problem as the following 0–1 ILP problem. The network manager assigns the m_a-th channel and the s_a-th time slot to the vertex v_a, and the objective of these assignments is to converge all data in the minimum number z of time slots. We define several binary variables to represent them:

- $m_{ax} = 1$ if $m_a = x$, otherwise $m_{ax} = 0$.
- $s_{ax} = 1$ if $s_a = x$, otherwise $s_{ax} = 0$.
- $z_x = 1$ if $z = x$, otherwise $z_x = 0$.

So the optimization objective is represented as

$$\min \sum_{\forall x \in [1,|V|]} x \cdot z_x \tag{1}$$

where the range of x is from 1 to the number of elements in the set V. It is because that in the worst case there is only one vertex scheduled at every time slot and thus scheduling all vertices needs at most $|V|$ time slots. The programming must respect the following constraints.

- *Uniqueness constraint* Each vertex can be scheduled at only one time slot and on only one channel. Similarly, the variable z is equal to one value.

$$\forall v_a \in V, \sum_{\forall x \in [1,|V|]} s_{ax} = 1 \tag{2}$$

$$\forall v_a \in V, \sum_{\forall x \in [1,M]} m_{ax} = 1 \tag{3}$$

$$\sum_{\forall x \in [1,|V|]} z_x = 1 \tag{4}$$

- *Time slot constraint* Any transmission cannot be scheduled after the last time slot.

$$\forall v_a \in V, 1 \leq \sum_{\forall x \in [1,|V|]} x \cdot s_{ax} \leq \sum_{\forall y \in [1,|V|]} y \cdot z_y \tag{5}$$

- *Channel constraint* At most M channels can be used.

$$\forall v_a \in V, 1 \leq \sum_{\forall x \in [1,M]} x \cdot m_{ax} \leq M \tag{6}$$

- *Releasing order constraint* If there exists the edge d_{ab}, the vertex v_b is scheduled after the vertex v_a.

$$\forall d_{ab} \in D, \sum_{\forall x \in [1,|V|]} x \cdot s_{ax} < \sum_{\forall y \in [1,|V|]} y \cdot s_{by} \tag{7}$$

- *Conflict constraint* The node conflict constrains that at any time slot the number of vertices served by a node is not larger than the number of radio interfaces equipped on the node.

$$\forall x \in [1, |V|], \forall n_i \in N, \sum_{\forall v_b \in V_i} s_{bx} \leq r_i \tag{8}$$

where V_i denotes the set of the vertices that use the node n_i. The scheduling conflict constrains that any two vertices cannot be scheduled at the same time slot and on the same channel, i.e., both variables of two vertices cannot be equal to one.

$$\forall v_a \in V, \forall v_b \in V \backslash \{v_a\}, \forall x \in [1, |V|], \forall y \in [1, M],$$
$$s_{ax} + s_{bx} + m_{ay} + m_{by} < 4 \tag{9}$$

According to the above formulation (1)–(9), ILP solvers can solve this problem.

4.3 A heuristic algorithm based on minimum spanning trees

Using ILP solvers can obtain the optimal solutions. But the running time of ILP solvers may become unacceptable as the network size grows. So we design a fast heuristic algorithm for the scheduling problem.

The vertex v_a is scheduled at the Λ_a-th sub-slot. We call Λ_a as the absolute distance of the vertex v_a. We use the array $Sch[1, \ldots, |V|]$ to denote the scheduling order generated by our algorithm. If $v_a = Sch[c]$ and $v_b = Sch[c+1]$, the vertex v_b is scheduled after $(\Lambda_b - \Lambda_a)$ sub-slots of scheduling the vertex v_a and there is not any vertex scheduled between them. Thus, the number of time slots that is used to schedule all vertices is $\left\lceil \frac{\Lambda_{Sch[|V|]}}{M} \right\rceil$.

Our problem is to schedule all vertices in the minimum number of time slots. In other words, we want to find the minimum absolute distance of the last vertex in the array Sch, which visits each vertex exactly once. The problem is similar to the shortest Hamiltonian path problem. Between our problem and the general shortest Hamiltonian path problem, there are two differences. The first one is the releasing order constraint. In the general shortest path problem, vertices are visited in any order as long as the path is the shortest. But in our problem, the releasing order must be constrained by the set D. So in our algorithm we need to verify whether these constraints hold. The second one is that the distance between two vertices is fixed in the general shortest path problem. While in our problem, the distance is unknown before the schedules are generated. In the following, we introduce how to calculate the distance in our problem. The first vertex is scheduled at the 1st sub-slot, so $\Lambda_{Sch[1]} = 1$. We assume that the absolute distance $\Lambda_{Sch[c]}$ and the scheduled array from the first vertex to the c-th vertex (denoted by $Sch_c[]$) are known.

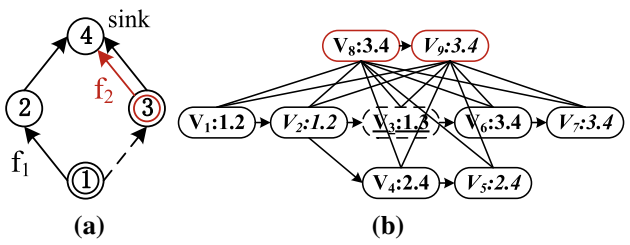


Fig. 3 An unfolding graph. **a** Tow flows, **b** the unfolding graph

Then if $v_a = Sch[c]$ and the vertex v_b is scheduled after the vertex v_a, we have that

$$\Lambda_b(a) = \Lambda_a + \min\left\{ MaxD(b, a, Sch_c), \left\lceil \frac{\Lambda_a}{M} \right\rceil \times M - \Lambda_a + 1 \right\} \tag{10}$$

where

$$\begin{aligned} &MaxD(b, a, Sch_c) \\ &= \max\left\{1, M \times \max\left\{ \left\lfloor \frac{|A(b, a, Sch_c)|}{r_{\alpha_b}} \right\rfloor, \left\lfloor \frac{|B(b, a, Sch_c)|}{r_{\beta_b}} \right\rfloor \right\}\right\} \\ &= \left\{ v_g | \forall Sch_c[g], \left\lceil \frac{\Lambda_{Sch[g]}}{M} \right\rceil = \left\lceil \frac{\Lambda_a + 1}{M} \right\rceil, \alpha_b \in v_g \right\} \\ &= \left\{ v_g | \forall Sch_c[g], \left\lceil \frac{\Lambda_{Sch[g]}}{M} \right\rceil = \left\lceil \frac{\Lambda_a + 1}{M} \right\rceil, \beta_b \in v_g \right\} \end{aligned}$$

If the time slot used by the vertex v_a contains idle sub-slots and the vertex v_b does not have node conflicts with the vertices scheduled at this time slot, the vertex v_b can be scheduled at this time slot, i.e., $\Lambda_b(a) = \Lambda_a + 1$. Otherwise, the vertex v_b is scheduled at the first sub-slot of the next time slot, i.e., $\Lambda_b(a) = \Lambda_a + (\lceil \frac{\Lambda_a}{M} \rceil \times M - \Lambda_a + 1)$. The function $MaxD(\,)$ determines whether the vertex v_b has node conflicts with the vertices scheduled at this time slot. If there is no node conflict, $MaxD() = 1$ and the vertex v_b can be scheduled at this time slot; otherwise, $MaxD(\,)$ is the possible longest distance M, i.e., the absolute distance Λ_b is determined by $\lceil \frac{\Lambda_a}{M} \rceil \times M - \Lambda_a + 1$. The set $A(\,)$ (or $B(\,)$) contain the vertices that are scheduled at this time slot and use the source node (or the destination node) of the vertex v_b.

Our Convergecast Scheduling for Multiple radio interfaces (CSM) (shown in Algorithm 2) is based on the spanning tree algorithm (shown in Algorithm 3). In Algorithm 2, the first vertices of all flows are released before the schedule starts. The released vertex that has the least laxity time is selected to be firstly scheduled (line 1). Then Algorithm 2 finds a shortest path that starts from the selected node (line 2) and visits each vertex once (lines between 3 and 8). The method of finding the shortest path is based on minimum spanning trees. The symbol V' denotes the unscheduled vertices. The variable c denotes how many vertices have been added into the array Sch. In each iteration, a spanning tree is generated by the function GST() (line 4). The branch that satisfies the releasing order constraint and has the minimum density is selected as the scheduled vertex sequence (line 5). The branch B_g contains the vertices from the root to the vertex v_g. The *density* is equal to the quotient of Λ and $|B|$ and denotes how many sub-slots every vertex needs. So the less density denotes the less time slots required by this scheduling array. In the set V' all ancestors of the vertex v_g are denoted by $Ance(V', v_g) = \{v_a | \forall v_a \in V', v_a \prec v_g\}$. If $Ance(V', v_g)$

$\subseteq B_g$, i.e., all ancestors of the vertex v_g have been scheduled before itself, the branch B_g can satisfy the releasing order constraint. Finally the set V', the array Sch_c and the variable c are updated according to the selected branch (lines between 6 and 8). The time slot where the last vertex is scheduled is the convergecast delay z (line 9).

Algorithm 2 Convergecast Scheduling for Multiple radio interfaces (***CSM***)

Require: G, F, $\forall\pi_i$ and R
Ensure: z
1: find the first scheduled vertex v_a that has the highest LLF priority.
2: $V' \leftarrow V - \{v_a\}$; $Sch[1] = v_a$; $c = 1$;
3: **while** $V' \neq \emptyset$ **do**
4: $Tree \leftarrow$***GST***(V', Sch_c);
5: $v_k = \arg\min\{\frac{\Lambda_g - \Lambda_{Sch[j]}}{|B_g|-1} | \forall v_g \in Tree, Ance(V', v_g) \subseteq B_g\}$;
6: $V' \leftarrow V' - \{v_g | \forall v_g \in B_k\}$;
7: update Sch_j according to the set B_k;
8: $j+ = |B_k| - 1$;
9: **return** $z = \left\lceil \frac{\Lambda_{Sch[|V|]}}{M} \right\rceil$;

Algorithm 3 ***GST***(V', Sch_c)

Require: V', Sch_c and $\forall r_i$
Ensure: the spanning tree $Tree$, $\forall\Lambda_a$
1: $Tree \leftarrow \{Sch[c]\}$; $Y \leftarrow V'$;
2: $R' \leftarrow$ the released vertices in V';
3: **while** $Y \neq \emptyset$ **do**
4: $E = \{d_{ab} | v_a \in Tree, v_b \in R', d_{ab} \in D\} + \{u_{ab} | v_a \in Tree, v_b \in R', u_{ab} \in U\}$;
5: $e_{gk} = \arg\min\{\frac{\Lambda_b(a) - \Lambda_{Sch[j]}}{|B_a|} | \forall e_{ab} \in E\}$;
6: $Tree \leftarrow Tree + \{v_k, e_{gk}\}$;
7: $Y \leftarrow Y - \{v_k\}$;
8: update R according to the current Y;
9: **return** the spanning tree $Tree$, $\forall\Lambda_i$;

Our spanning tree algorithm is shown in Algorithm 3. First, the last vertex in the array Sch_c is the root of the tree (line 1). The symbol Y denotes the set of vertices that have not been added to the tree. The set R' contains the released vertices (line 2). In each iteration, one vertex that is in the set R' and has an edge from the tree to itself (line 4), is selected to add into the tree (line 6). After adding the vertex, the density of the new branch must be minimal (line 5). Then the sets R' and Y are updated (lines between 7 and 8).

In Algorithm 3, the number of iterations of the **while** loop in line 3 is $O(|V|)$. The time complexity of line 4, line 5 is $O(|V|)$ and $O(|E|)$, respectively. The time complexity of Algorithm 3 is $O(|V|(|V| + |E|))$. In Algorithm 2, the number of iterations of the **while** loop in line 3 is $O(|V|)$. So the time complexity of Algorithm 2 is $O(|V|^2(|V| + |E|))$.

5 Optimizing the number of radio interfaces

The rapid convergecast can improve the control performance. While in some industrial applications, as long as the convergecast scheduling provides a guarantee on end-to-end delay, the requirements of industrial control can be met [43]. If the service supplied by radio interfaces exceeds the requirement of convergecast scheduling, some radio interfaces may be unused. The price of a radio interface is similar to that of the embedded microcontroller. So to reduce the cost of industrial wireless networks, the unused radio interfaces must be removed. In this section, we minimize the cost of radio interfaces according to the given upper bound of convergecast delay. The proposed methods are also based on the unfolding graph. First, we formulate the problem as a mathematical problem to make it clearer, and then analyze the upper bound and the lower bound of the number of radio interfaces. Finally, based on the analysis, we design two algorithms to solve the problem.

5.1 Problem formulation

The objective of this paper is to minimize the number of radio interfaces. It is represented as

$$\min \sum_{\forall n_i \in N} r_i \tag{11}$$

Multiple radio interfaces make the network schedulable. So the constraints [Eqs. (2)–(9)] in Sect. 4.2 also are satisfied in this problem. Additionally, the given upper bound of convergecast delay $\bar{z}$ cannot be exceeded. Thus, the following constraint needs to be satisfied:

$$\sum_{\forall x \in [1,|V|]} x \cdot z_x \leq \bar{z} \tag{12}$$

5.2 Upper and lower bounds analysis

In this section, we want to determine the number of radio interfaces of each node. To reduce the search space, we analyze the lower and upper bounds of the number of radio interfaces of each node. First, we introduce some symbols. The unfolding graph of a flow contains multiple paths. For example, the graph of the flow f_1 in Fig. 3(a) has two paths from the source node to the sink $\pi_{1,1} = \{v_1, v_2, v_3, v_6, v_7\}$, $\pi_{1,2} = \{v_1, v_2, v_4, v_5\}$ and $\Pi_1 = \{\pi_{1,1}, \pi_{1,2}\}$. We define h_a^{post} as the number of vertices from the vertex v_a to the sink. And the vertex v_a is the h_a^{pre}-th hop from the flow's source node. If a vertex is contained in multiple paths, its h_a^{pre} is the minimum value among these paths and h_a^{post} is the maximum value. For example, $\{h_1^{pre}, h_1^{post}\} = \{1, 4\}$ and $\{h_2^{pre}, h_2^{post}\} = \{2, 3\}$. So the earliest released time γ_a^1

of the vertex v_a is equal to h_a^{pre}, and its latest scheduled time γ_a^2 is $\bar{z} - h_a^{post}$. The duration from γ_a^1 to γ_a^2 is the lifetime of the vertex v_a.

For the vertex $v_a = <n_i, n_j>$, if λ vertices that use the node n_i must be scheduled in the lifetime of the vertex v_a, then the utilization of the node n_i is equal to $\frac{\lambda}{r_i \cdot (\gamma_a^2 - \gamma_a^1 + 1)}$. Note that the λ vertices contain the vertex v_a. The utilization of a node is not larger than one. So in the lifetime of the vertex v_a, r_i must be larger than or equal to $\frac{\lambda}{\gamma_a^2 - \gamma_a^1 + 1}$. If the vertex v_b uses the node n_i and $[\gamma_b^1, \gamma_b^2] \subseteq [\gamma_a^1, \gamma_a^2]$, then the vertex v_b must be scheduled within the duration of $[\gamma_a^1, \gamma_a^2]$. The set of this kind of nodes is denoted as

$$\mathcal{V}_{i,[\gamma_a^1,\gamma_a^2]} = \{v_b | \forall v_b \in V_i, [\gamma_b^1, \gamma_b^2] \subseteq [\gamma_a^1, \gamma_a^2]\}$$

i.e., $\lambda = |\mathcal{V}_{i,[\gamma_a^1,\gamma_a^2]}|$. Recall that the set V_i denotes the set of vertices that use the node n_i. Thus, we can get that in the duration of $[\gamma_a^1, \gamma_a^2]$, r_i is not less than $\frac{|\mathcal{V}_{i,[\gamma_a^2,\gamma_a^1]}|}{\gamma_a^2 - \gamma_a^1 + 1}$. So the lower bound of the number of radio interfaces is

$$\forall n_i \in N, \mathcal{L}_i = \left\lceil \max\left\{ \frac{|\mathcal{V}_{i,[\gamma_a^2,\gamma_a^1]}|}{\gamma_a^2 - \gamma_a^1 + 1} | \forall v_a \in V_i \right\} \right\rceil \tag{13}$$

Theorem 1 proves the correctness of the lower bound.

Theorem 1 If $\exists n_i \in N$,

$$r_i < \left\lceil \max\left\{ \frac{|\mathcal{V}_{i,[\gamma_a^2,\gamma_a^1]}|}{\gamma_a^2 - \gamma_a^1 + 1} | \forall v_a \in V_i \right\} \right\rceil \tag{14}$$

then not all of packets can be converged to the sink within the given upper bound of convergecast delay $\bar{z}$.

Proof We assume there exists the node n_i. According to the definition of $\mathcal{V}_{i,[\gamma_a^2,\gamma_a^1]}$, we know that if Equation (14) holds, then there must exist some vertices that cannot be serviced by the node n_i within their lifetimes. If the vertex v_a that uses the node n_i is not scheduled within its lifetime, there are no enough time slots to scheduled the descendants of the node n_i before the time slot $\bar{z}$ because of the definition of the lifetime. □

The upper bound of the number of radio interfaces is determined by three factors. First, due to the limitation of implementation, a sensor node can be equipped with at most $\mathcal{R}_i$ radio interfaces. The constant $\mathcal{R}_i$ is given by node designers. Second, for a node, the number of its radio interfaces cannot exceed M, which denotes the number of channels. Because the M channels support at most M vertices to be scheduled simultaneously. Even if there exist more radio interfaces, they cannot be used. Third, the node n_i serves multiple paths. In the worst case, these pathes pass the node n_i simultaneously. So the number of its radio interfaces cannot exceed the number of these paths. These paths are contained in the set $\mathcal{F}_i = \{\pi_{kg} | n_i \in \pi_{kg}, \forall f_k \in F, \forall \pi_{kg} \in \Pi_k\}$. So,

$$\forall n_i \in N, \mathcal{U}_i = \min\{\mathcal{R}_i, M, |\mathcal{F}_i|\} \tag{15}$$

If $\exists n_i \in N, \mathcal{L}_i > \mathcal{U}_i$, the network cannot be scheduled within the given upper bound of convergecast delay.

5.3 Optimal branch-and-bound algorithm

In this subsection, we propose an optimal algorithm based on the classical Branch-and-Bound (BB) to minimize the cost of radio interfaces. This algorithm has two steps. First, we find a solution, which is used to distinguish infeasible solutions. Second, we search the optimal solution based on the BB method Algorithm 4. They are as follows.

Step 1 we construct an initial solution. The cost of the initial solution $\bar{C}$ is the initial upper bound for discarding infeasible branches in Algorithm 4. We modify Algorithm 2 and Algorithm 3 to obtain the initial solution. The symbol r_i' to denote the current number of radio interfaces. Before the algorithm begins, the number of usable radio interfaces of each node is equal to its lower bound $\mathcal{L}_i$. Then if the current number r_i' can be increased and increasing r_i' can reduce the distance, then a radio interface is added to the node n_i. The detailed modifications are as follows.

(1) Before Algorithm 2 begins, $\forall n_i \in N$, $r_i' = \mathcal{L}_i$.

(2) We modify Algorithm 3. Recall that we use B_g to denote the branch from the root to the vertex v_g. In a spanning tree, the distance of the vertex v_k is affected by its ancestors. If the node n_i is used by two vertices v_g and v_l($v_g \not\prec v_l$ and $v_l \not\prec v_g$), then the value of r_i' on branches B_g and B_l can be different, since the schedules on the two branches are different. So we use $r_{i,g}'$ and $r_{i,l}'$ to distinguish them. Before the tree is generated, $\forall n_i \in N, r_{i,*}' = r_i'$. The absolute distance $\Lambda_b(a)$ is redefined as follows.

$$\Lambda_b(a, p_\alpha, p_\beta) = \Lambda_a + \min\left\{ MaxD(b, a, Sch_c, p_\alpha, p_\beta), \left\lceil \frac{\Lambda_a}{M} \right\rceil \times M - \Lambda_a + 1 \right\}$$

where p_α and p_β denote the added number of radio interfaces in the source and destination of the vertex v_b, and

$$MaxD(b, a, Sch_c, p_\alpha, p_\beta) = \max\left\{ 1, M \times \max\left\{ \left\lfloor \frac{|A(b, a, Sch_c)|}{r'_{\alpha_b, b} + p_\alpha} \right\rfloor, \left\lfloor \frac{|B(b, a, Sch_c)|}{r'_{\beta_b, b} + p_\beta} \right\rfloor \right\} \right\}$$

Thus, when the vertex v_b is scheduled after the vertex v_a, if $r'_{\alpha_b,b}$ and $r'_{\beta_b,b}$ are less than their upper bound, the vertex v_b has four optional distances $\Lambda_b(a,0,0)$, $\Lambda_b(a,0,1)$, $\Lambda_b(a,1,0)$ and $\Lambda_b(a,1,1)$. $\Lambda_b(a,p_\alpha,p_\beta)$ denotes the absolute distance in the schedule Sch when the node n_{α_b} and n_{β_b} has $r'_{\alpha_b,b}+p_\alpha$ and $r'_{\beta_b,b}+p_\beta$ usable radio interfaces, respectively. If the current number of radio interfaces is equal to the upper bound, the parameter p_* is only 0. The selected edge e_{gk} has the minimal density among $\{\frac{\Lambda_b(a,p_\alpha,p_\beta)-\Lambda_{Sch[c]}}{|B_a|}\,|\forall e_{ab}\in E,\forall p_\alpha\in\{0,1\},p_\alpha\le\mathcal{U}_{\alpha_b}-r'_{\alpha_b,b},\forall p_\beta\in\{0,1\},p_\beta\le\mathcal{U}_{\beta_b}-r'_{\beta_b,b}\}$, and the minimal sum of p_α and p_β if some densities equal. The values of $r'_{\alpha_b,b}$ and $r'_{\beta_b,b}$ are updated according to the selected p_α and p_β. In this way, we can find a feasible solution as far as possible, and use radio interfaces as less as possible.

(3) In Algorithm 2, after the branch B_k is selected, we update every r'_i as $r'_{i,k}$. After Algorithm 2 calculated the convergecast delay z, if $z\le\bar{z}$, then $\bar{C}=\sum_{\forall n_i\in N} r'_i$ and the feasible solution is $R_{init}=\{r'_1,\ldots,r'_{|N|}\}$. Otherwise, $\bar{C}=\sum_{\forall n_i\in N}\mathcal{U}_i$ and $R_{init}=\emptyset$. We find that even though the network with the cost C cannot be scheduled within the given convergecast delay, the network with the cost $C-1$ may be scheduled (shown in Property 1 and Fig. 4). So there is no need to check whether $\forall n_i\in N, r_i=\mathcal{U}_i$ is a feasible solution or not. It only denotes that if we do not find a feasible solution in Step 1, the entire solution space between the lower bound and the upper bound must be searched by Algorithm 4.

Property 1 *Not all of increasing radio interfaces can reduce the convergecast delay.*

Intuitively, the more radio interfaces can supply more services and make convergecast delay less. However, even if the number of radio interfaces becomes greater, the changing of radio interfaces may lead to the increasing of node conflicts that can delay convergecast. We show an example to illustrate Property 1. Figure 4(a) shows a simple network and four flows. And Fig. 4(b) shows three convergecasts under different the number of radio interfaces. Note that for brevity Fig. 4(b) ignores the retransmissions. First, every node has only one radio interface. The convergecast delay is 7. Then the node n_4 is equipped with two radio interfaces. The convergecast delay reduces to 6, since they can support more services. In the third configuration the node n_2 is equipped with two radio interfaces. Due to the conflicts on the node n_4, only one

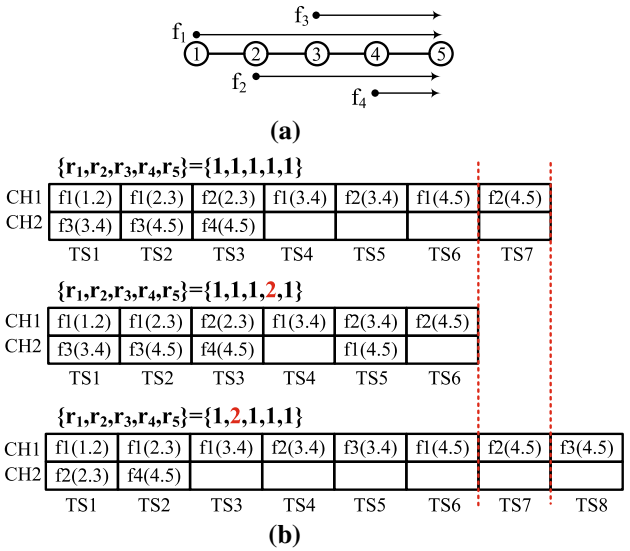


Fig. 4 An example for Property 1. **a** A network and four flows, **b** convergecast scheduling under different numbers of radio interfaces

transmission can be scheduled at each time slot from *TS*3 to *TS*8. Thus, its delay is larger than that of other two configurations, even though it has more radio interfaces than the first configuration.

Step 2 The method BB adopts a search tree, where each node corresponds to an array of the number of radio interfaces. The network nodes are sorted according to the decreasing order of their utilizations, and the network node $\mathbf{n}_1$ has the largest utilization. Note that the network node $\mathbf{n}_i$ does not corresponds to the network node n_i. The element $\mathbf{r}_i$ in the array denotes the number of radio interfaces in the network node $\mathbf{n}_i$.

The root of the search tree is the lower bound of the number of radio interfaces $\mathcal{L}_i$ on every network node. And the root is at level 1. It creates its children at level 2 and then these children create new nodes at the next level. For the new nodes created by the node of level ξ, the value of their element $\mathbf{r}'_\xi$ is set as from $\mathcal{L}_\xi$ to $\mathcal{U}_\xi$ and other elements are not changed. An example of the search tree is shown in Fig. 5.

Our BB algorithm is shown in Algorithm 4. It wants to find the minimal cost C_{min} and a feasible solution $\mathbf{R}$. The

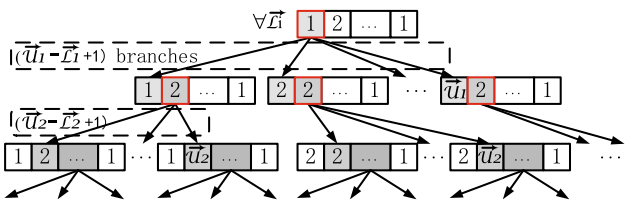


Fig. 5 An example of the search tree

cost $\bar{C}$ and the solution R_{init} obtained by *Step 1* are the initial solution (line 1). Then the algorithm adopts a FIFO queue Q to construct the search tree. First, the root is added into the queue Q (line 2). The tuple $<\mathbf{R}', \xi>$ denotes the array $\mathbf{R}'$ is at level ξ. Then when the new nodes are created, their tuples are added into the queue. These tuples are removed in order of FIFO (line 4). For each tuple, if its cost is less than the current minimal cost or there is no feasible solution (line 5), Algorithm 2 is invoked to obtain its delay z (line 6) and its children are created at level $\xi + 1$ (lines between 9 and 12). If the delay z of the current tuple is less than the upper bound of delay (line 7), the minimal cost and the feasible solution are updated (line 8). Finally, if the algorithm finds a feasible solution, then the minimal cost is also feasible (lines between 13 and 16).

Algorithm 4 Our BB method

Require: $\bar{C}$, R_{init}, $\forall\vec{\mathcal{L}}_i$, $\forall\vec{\mathcal{U}}_i$ and $\bar{z}$
Ensure: C_{min} and $\vec{R}$
1: $C_{min} = \bar{C}$; $\vec{R} \leftarrow R_{init}$;
2: $Q.enqueue(<\{\vec{\mathcal{L}}_1, ..., \vec{\mathcal{L}}_{|N|}\}, 1>)$;
3: **while** $!Q.isEmpty()$ **do**
4: $<\vec{R}', \xi> \leftarrow Q.dequeue()$;
5: **if** $(\sum_{\forall\vec{r}'_i \in \vec{R}'} \vec{r}'_i < C_{min})$ or $((\sum_{\forall\vec{r}'_i \in \vec{R}'} \vec{r}'_i == C_{min}) \&\& (\vec{R} == \emptyset))$ **then**
6: $z = \boldsymbol{CSM}(\vec{R}')$;
7: **if** $z \leq \bar{z}$ **then**
8: $C_{min} = \sum_{\forall\vec{r}'_i \in \vec{R}'} \vec{r}'_i$; $\vec{R} \leftarrow \vec{R}'$;
9: **if** $\xi < N$ **then**
10: **for** $\forall j \in [\vec{\mathcal{L}}_\xi, \vec{\mathcal{U}}_\xi]$ **do**
11: $\vec{r}'_\xi = j$;
12: $Q.enqueue(<\vec{R}', \xi + 1>)$;
13: **if** $\vec{R}! = \emptyset$ **then**
14: **return** C_{min} and $\vec{R}$;
15: **else**
16: **return** FAIL;

In the worst case, when no branches are discarded, the Branch-and-Bound algorithm traverses the entire solution space and can find the optimal solution. In Algorithm 4, the current minimal cost is the bound for discarding infeasible branches. If the optimal solution exists, it cannot be discard, since it is not less than the current minimal solution. So Algorithm 4 can find the optimal solution. But for some networks the execution time of the method BB may be unacceptable. So in the next subsection, we propose a fast and efficient heuristic algorithm to solve this problem.

5.4 A fast heuristic algorithm

In the example of Fig. 4, when all nodes have the same number of radio interfaces, the utilization of the node n_4 is the largest. Increasing its radio interfaces can reduce its utilization and node conflicts. The utilization is the main factor impacting on the convergecast delay. Our algorithm MinRI assigns radio interfaces to each node according to its utilization, such that the cost of radio interfaces in the entire network is minimal. When all nodes are equipped with one radio interface, the utilization of the node n_i is equal to $\max\{\frac{|\mathcal{V}_{i,[\gamma_a^2,\gamma_a^1]}|}{\gamma_a^2-\gamma_a^1+1} | \forall v_a \in V_i\}$. We define this utilization as the base utilization ρ_i. Then in our algorithm MinRI, we adjust the number of radio interfaces of each nodes according to its base utilization.

Algorithm 5 MinRI

Require: $\forall\mathcal{L}_i$, $\forall\mathcal{U}_i$, $\forall\rho_i$ and Δ
Ensure: C_{min} and R
1: $\forall n_i \in N$, calculate its density ρ_i;
2: **loop**
3: $\forall n_i \in N, r_i = \min\{\mathcal{U}_i, \max\{\mathcal{L}_i, \lceil\rho_i\rceil\}\}$;
4: $z = \boldsymbol{CSM}(R)$;
5: **if** $z \leq \bar{z}$ **then**
6: **return** $C_{min} = \sum_{\forall r_i \in R} r_i$ and R;
7: **else if** $\forall r_i \in R, r_i == \mathcal{U}_i$ **then**
8: **return** FAIL;
9: $\forall n_i \in N, \rho_i = \rho_i/\Delta$;

Our algorithm MinRI is shown in Algorithm 5. First, the algorithm calculates the base utilization for each node (line 1). Then, we use the given coefficient $\Delta(0<\Delta<1)$ to adjust the base utilization (line 9). The number of radio interfaces cannot be less than the utilization (Theorem 1). Otherwise, the flow set is unschedulable. So the number of radio interfaces is the ceiling of the utilization and is bounded by the upper and lower bounds (line 3). For each adjustment, Algorithm 2 is invoked to obtain the convergecast delay (line 4). If the delay is not larger than the given delay, the current cost is the result solved by our algorithm (lines between 5 and 6). With the adjustment of utilizations, the number of radio interfaces increases. When all the number of radio interfaces are the upper bound and the obtained delay is still larger than the given delay, the algorithm does not find a feasible solution (lines between 7 and 8). The complexity of this algorithm is pseudo-polynomial, since the number of iterations of the **loop** in line 2 is determined by the given coefficient $\Delta(0<\Delta<1)$ and the base utilization.

6 Evaluations

In this section, two evaluations are presented to demonstrate the effectiveness of our methods. First, we show that our convergecast scheduling algorithm has less convergecast delay than the classical scheduling policy. Then, based on our convergecast scheduling algorithm we present that our algorithm MinRI can effectively reduce the number of radio interfaces.

6.1 Convergecast scheduling algorithm

We compare our scheduling algorithm CSM against the classical scheduling policy LLF (Least Laxity First, which firstly schedules the transmissions whose packet has the least laxity time) [44]. Other classical policies, e.g., RM, DM and EDF [44], are not suitable for convergecast scheduling problem, because they must be used in the flow set with different periods or different deadlines. While the policy LLF supports the same period and deadline. All algorithms are implemented in C language. These programs run on a Windows machine with 3.4GHz CPU and 4GB memory.

The test cases used in our evaluations are generated randomly. In each test case, the network with the given number of nodes is generated in the square area A. The sink is placed at the center and other nodes are placed randomly. According to the suggestion in the work of [45], the number of nodes n and the square area A should satisfy $\frac{n}{A} = \frac{2\pi}{d^2\sqrt{27}}$, where the transmitting range d is set as **0 m. Then each node is connected to the nodes that are in its transmitting range and have been connected to the sink. If some nodes cannot connect to the sink, their locations are generated randomly again. There are $n-1$ flows in a network, and the source nodes of there flows are different. We randomly select some shortest paths to combine routing graphs.

We use the ILOG CPLEX solver to solve the 0-1 ILP problem [Eqs. (1)–(9)]. But sometimes the ILOG CPLEX cannot find the feasible solution in an acceptable time. So we use the lower bound of optimal solution as the comparison baseline. The lower bound LC is defined as

$$LC = \max\left\{\left\lceil\frac{|V|}{M}\right\rceil, \max\left\{\left\lceil\frac{|V_i|}{r_i}\right\rceil \middle| \forall n_i \in N\right\}\right\}$$

When we only consider the scheduling conflict, the optimal solution is not less than $\left\lceil\frac{|V|}{M}\right\rceil$. When we only consider the node conflict, the optimal solution is not less than the number of time slots used by any node. In order to make the problem solvable by the ILOG CPLEX, we set $n = 9$, and the number of radio interfaces in each node is randomly selected from the integers 1 and 2. We will extend these parameters in the next evaluation. Figure 6 shows that the normalized number of time slots (with LC being used as the baseline). For each configuration, 50 test cases that can be solved by the ILOG CPEX in an acceptable time are used to evaluate the effectiveness of the other two algorithms. The policy LLF and our algorithm CSM needs, on average, 2.3% and 1.5% more time slots than the optimal solutions, respectively. And the lower bound LC is tight.

In the following evaluations, the ILOG CPLEX cannot find a feasible solution in an acceptable time, we only compare the algorithms LLF and CSM with the lower bound LC. Figure 7 shows the normalized convergecast delay with LC being used as the baseline. For each configuration, 100 test cases are generated randomly. Considering the requirement of real applications and the implementation limitation, the sink has 3 radio interfaces and the number of radio interfaces in other nodes is randomly selected from the integers 1 to 3. These subfigures use the same scale and show the results under different numbers of channels. From the subfigures, we can know that

- No matter what the configuration is, our algorithm is more effective than the classical policy LLF. Comparing with the lower bound LC, in the worst case, our algorithm CSM introduces 10% extra delay. The optimal solution is larger than or equal to the baseline LC, so our algorithm can obtain the close to optimal solutions.
- When the number of channels ranges from 4 to 7, the more the channel resources, the more the time slots. It is because that the more channels cause the longer distance [Eq. (10)].
- When the number of channels is larger than 7, the number of time slots decreases with the increase of the channel resource. Since if the channel resource is abundant, the scheduling conflict almost disappear.

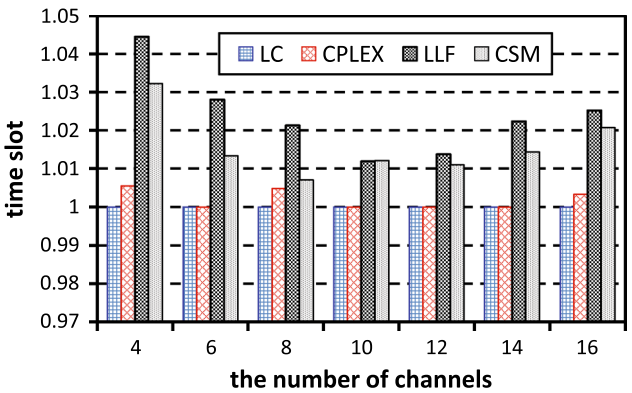


Fig. 6 The delay comparison among LC, LLF, our algorithm and CPLEX

Additionally, for the policy LLF the number of time slots increases with the increase of the network size. It is because that when the network size becomes large, the number of vertices increases sharply due to the graph routing scheme. In this case, the node conflict becomes more severe. While our algorithm CSM is not affected by it.

6.2 Minimizing cost

For the problem of minimizing cost, our BB algorithm can find the optimal solution. But its execution time increases dramatically as the number of nodes increases. When the number of nodes is 20, its execution time is about 30 seconds. When the number of nodes is 50, its execution time is about 40 minutes. So we set the sum of the lower bound $\mathcal{L}_i$ as the baseline, i.e., the lower bound of optimal solution is denoted as

$$LM = \sum_{\forall n_i \in N} \mathcal{L}_i$$

In the following evaluations, when the number of nodes is larger than 50, there are no the BB algorithm since its execution time is significantly long.

Besides the BB algorithm and *LM*, we consider the following comparison methods: (1) C+M: the convergecast scheduling algorithm is our proposed CSM and the minimizing cost algorithm is our algorithm MinRI. (2) L+M: the policy LLF and our algorithm MinRI. (3) C+H: the scheduling algorithm is CSM, and all network nodes are Homogeneous and have the same number of radio interfaces. Usually, the homogeneous nodes are adopted in real applications. For the method in the homogeneous network, first, we check the network whose all nodes have one radio interfaces to determine whether it can be scheduled within the given delay bound. If it cannot be scheduled, we increase the number of radio interfaces and check it again. Until the network can be scheduled. (4) L+H: the policy LLF and homogenous nodes. Additionally, in our algorithm MinRI, the given coefficient Δ

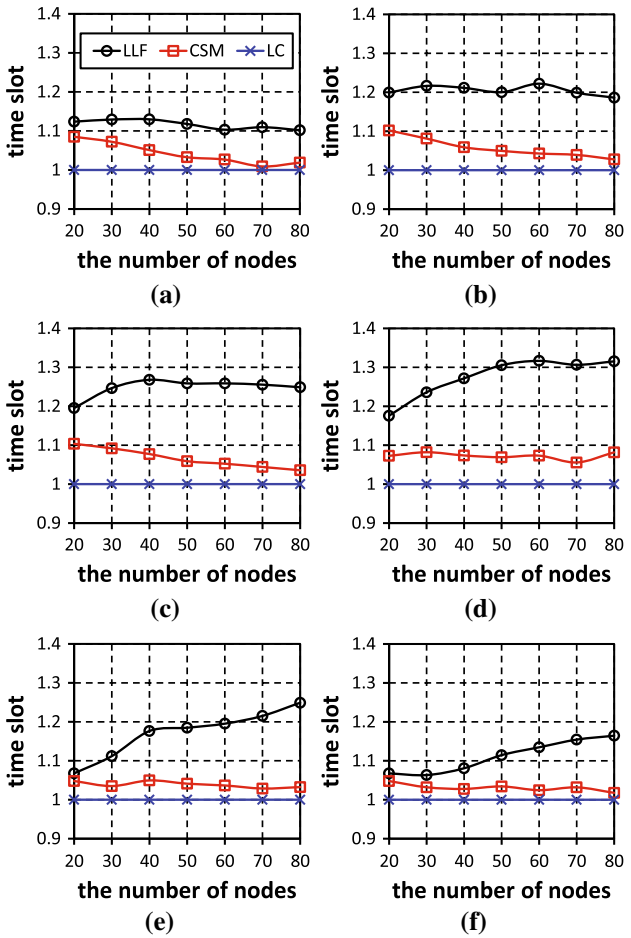


Fig. 7 The delay comparison among LLF, our algorithm and LC. **a** $M = 4$, **b** $M = 5$, **c** $M = 6$, **d** $M = 7$, **e** $M = 12$, **f** $M = 16$

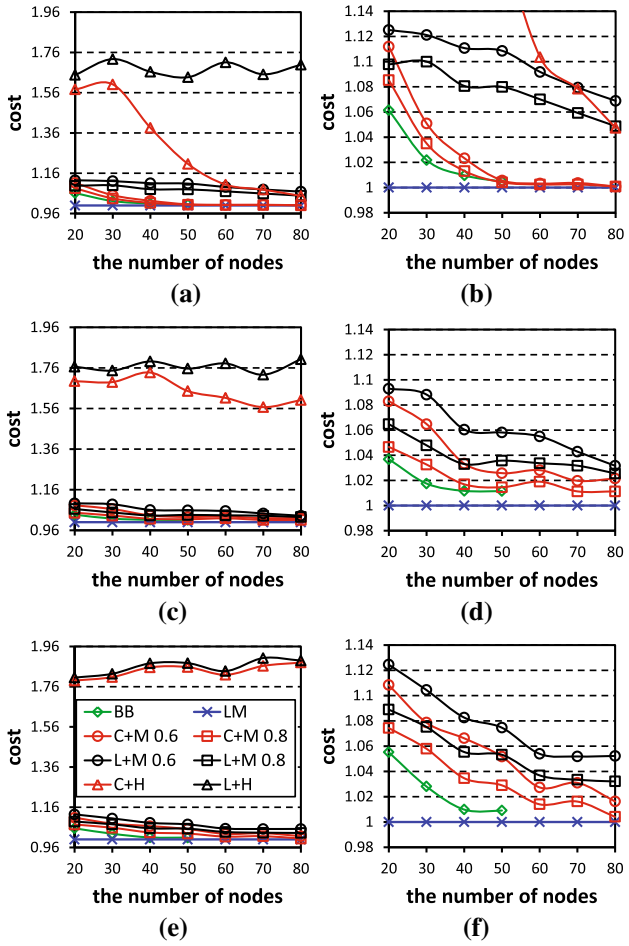


Fig. 8 Cost comparison. **a** $M = 4$, **b** A part of (**a**), **c** $M = 10$, **d** A part of (**c**), **e** $M = 16$, **f** A part of (**e**)

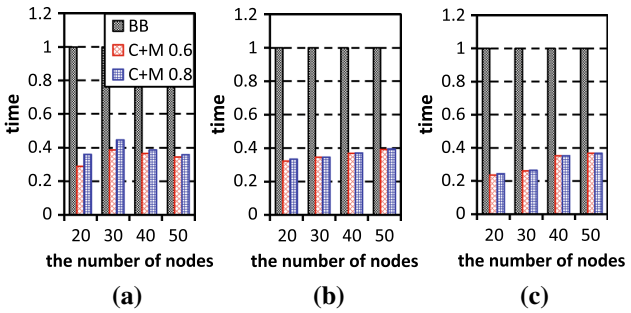


Fig. 9 The execution time comparison. **a** $M = 4$, **b** $M = 10$, **c** $M = 16$

influences the performance and execution time. So we set the coefficient Δ as 0.6 and 0.8, respectively.

Figure 8 shows the normalized cost with *LM* being used as the baseline. Each node can be equipped with at most 3 radio interfaces. And the delay $\bar{z}$ is the convergecast delay that is obtained when the network applies the method L+H and all nodes are equipped with two radio interfaces. Other parameters are the same with those of Fig. 7. To show the comparison clearly, the subfigures (b), (d) and (f) are the zoom-in of the subfigures (a), (c) and (e), respectively. From the figures, we know that

- Our algorithm MinRI significantly outperforms the homogeneous method. And its convergecast delay is close to the optimal solution. The different between our algorithm MinRI and the optimal solutions is less than 3%. Moreover, when $M = 4$ and $n \geq 50$, our delay is almost equal to the optimal solution.
- The larger the coefficient Δ, the algorithm MinRI is closer to the optimal solution. Figure 9 shows the execution time comparison among the algorithms BB and C+M. Though our algorithm MinRI introduces small extra delay, the execution time of the algorithm MinRI is less than 45% of that of the algorithm BB no matter what the configuration is. In the best case our algorithm MinRI only need 23% of the execution time of the algorithm BB.
- For the same mark symbol, the red lines denote our proposed scheduling algorithm CSM, and the black lines is the policy LLF. These evaluations also show our algorithm CSM performs better than the policy LLF.
- When the number of channels is 4 and the number of nodes is larger than 50, the method C+H is close to the method L+M. It is because that our algorithm CSM significantly outperforms the policy LLF. Even using the homogeneous nodes the cost obtained by our algorithm is also very small.
- The extra cost decrease with the increase of nodes. The large network contains more vertices due to the graph routing scheme. Thus, every node is used more frequently. When the algorithm adds one radio interface on a node, the increased radio interface eliminates more node conflicts than that in the small network. So a little extra cost is introduced in the large network.

7 Conclusion

This paper focuses on how to apply multiple radio interfaces to converge data. First, based on the shortest path problem we design a convergecast scheduling algorithm for multi-radio interface networks. Then, we propose two algorithms to minimize the number of radio interfaces. The two algorithms invoke our proposed scheduling algorithm to check the schedulability of networks with different the number of radio interfaces and find the optimized solution. Evaluations show that our proposed algorithms are close to optimal. In future work, we will extend our study to the flow set that contains flows with different periods and deadlines.

Acknowledgements This work was partially supported by the National Natural Science Foundation of China (61502474, 61501447 and 61233007) and the Youth Innovation Promotion Association of the Chinese Academy of Sciences.

Compliance with ethical standards

Conflict of interest The authors declare that there is no conflict of interest regarding the publication of this paper.

References

1. Soldati, P., Zhang, H., & Johansson, M. (2009). Deadline-constrained transmission scheduling and data evacuation in wirelesshart networks. In *European control conference*.
2. Lin, Z., Du, C., Meng, D., Yang, Z., & Wang, X. (2016). Rang-based distributed recruit scheduling in WSNs. *Information and Control*, *45*(6), 684–690.
3. Jin, X., Xia, C., Xu, H., Wang, J., & Zeng, P. (2016). Mixed criticality scheduling for industrial wireless sensor networks. *Sensors*, *2016*(16), 1–20.
4. Zhu, X., Huang, P.-C., Meng, J., Han, S., Mok, A. K., Chen, D., et al. (2014). Colloc: A collaborative location and tracking system on wirelesshart. *ACM Transactions on Embedded Computing Systems*, *13*(4), 125–148.
5. Sleep, S. R., Dadej, A., & Lee, I. (2016). Representing arbitrary sensor observations for target tracking in wireless sensor networks. *Computers & Electrical Engineering*.
6. Bagaa, M., Challal, Y., Ksentini, A., Derhab, A., & Badache, N. (2014). Data aggregation scheduling algorithms in wireless sensor networks: Solutions and challenges. *IEEE Communications Surveys & Tutorials*, *16*(3), 1339–1368.
7. Fasolo, E., Rossi, M., Widmer, J., & Zorzi, M. (2007). In-network aggregation techniques for wireless sensor networks: A survey. *IEEE Wireless Communications*, *14*(2), 70–87.

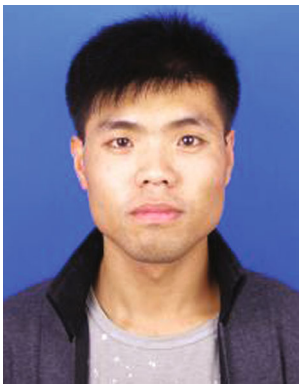
8. Liu, X. Y., Zhu, Y., Kong, L., Liu, C., Gu, Y., Vasilakos, A. V., et al. (2015). Cdc: Compressive data collection for wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, *26*(8), 2188–2197.
9. Soua, R., Minet, P., & Livolant, E. (2012). Modesa: An optimized multichannel slot assignment for raw data convergecast in wireless sensor networks. In *International performance computing and communications conference* (pp. 91–100).
10. Soua, R., Livolant, E., & Minet, P. (2013). An adaptive strategy for an optimized collision-free slot assignment in multichannel wireless sensor networks. *Journal of sensor and actuator networks*, *2*(3), 449–485.
11. Soua, R., Minet, P., & Livolant, E. (2014). A distributed joint channel and slot assignment for convergecast in wireless sensor networks. In *International conference on new technologies, mobility and security* (pp. 1–5).
12. Xu, X., Liang, W., & Xu, Z. (2013). Minimizing remote monitoring cost of wireless sensor networks. In *Wireless communications and networking conference* (pp. 1476–1481), IEEE.
13. Ji, S., Cai, Z., Li, Y., & Jia, X. (2012). Continuous data collection capacity of dual-radio multichannel wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, *23*(10), 1844–1855.
14. Lu, T., Liu, G., & Chang, S. (2016). Energy-efficient data sensing and routing in unreliable energy-harvesting wireless sensor network. *Wireless Networks*, *1–15*, doi:10.1007/s11276-016-1360-6.
15. Wadhwa, L. K., Deshpande, R. S., & Priye, V. (2016). Extened shortcut tree routing for ZigBee based wireless sensor network. *Ad Hoc Networks*, *37*(2), 295–300.
16. Dang, K., Shen, J.-Z., Dong, L.-D., & Xia, Y.-X. (2013). A graph route-based superframe scheduling scheme in wirelesshart mesh networks for high robustness. *Wireless personal communications*, *71*(4), 2431–2444.
17. IEC. (2009). IEC 62591: Industrial communication networks–wireless communication network and communication profiles–wirelesshart.
18. Liang, W., Zhang, X., Xiao, Y., Wang, F., Zeng, P., & Yu, H. (2011). Survey and experiments of WIA-PA specification of industrial wireless network. *Wireless Communications and Mobile Computing*, *11*(8), 1197–1212.
19. Lin, T.-Y., Hsieh, K.-C., & Huang, H.-C. (2012). Applying genetic algorithms for multiradio wireless mesh network planning. *IEEE Transactions on Vehicular Technology*, *61*(5), 2256–2270.
20. Chieochan, S., & Hossain, E. (2013). Channel assignment for throughput optimization in multichannel multiradio wireless mesh networks using network coding. *IEEE Transactions on Mobile Computing*, *12*(1), 118–135.
21. Xie, K., Wang, X., Liu, X., Wen, J., & Cao, J. (2015). Interference-aware cooperative communication in multi-radio multi-channel wireless networks. *IEEE Transactions on Computers*, *65*, 1–14.
22. Lin, T.-Y., Wu, K.-R., & Yin, G.-C. (2015). Channel-hopping scheme and channel-diverse routing in static multi-radio multi-hop wireless networks. *IEEE Transactions on Computers*, *64*(1), 71–86.
23. Ji, S., Li, Y., & Jia, X. (2011). Capacity of dual-radio multi-channel wireless sensor networks for continuous data collection. In *International Conference on Computer and Communications*, IEEE, (pp. 1062–1070).
24. Anastasi, G., Conti, M., Di Francesco, M., & Passarella, A. (2009). Energy conservation in wireless sensor networks: A survey. *Ad Hoc Networks*, *7*(3), 537–568.
25. Al Islam, A. A., Hossain, M. S., Raghunathan, V., & Hu, Y. C. (2014). Backpacking: Energy-efficient deployment of heterogeneous radios in multi-radio high-data-rate wireless sensor networks. *IEEE Access*, *2*, 1281–1306.
26. Stathopoulos, T., Lukac, M., Mclntire, D., Heidemann, J., Estrin, D., & Kaiser, W. J. (2007). End-to-end routing for dual-radio sensor networks. In *International conference on computer communications* (pp. 2252–2260). IEEE.
27. Choi, H., Wang, J., & Hughes, E. A. (2009). Scheduling for information gathering on sensor network. *Wireless Networks*, *15*(1), 127–140.
28. Gandham, S., Zhang, Y., & Huang, Q. (2008). Distributed time-optimal scheduling for convergecast in wireless sensor networks. *Computer Networks*, *52*(3), 610–629.
29. Incel, Ö. D., Ghosh, A., Krishnamachari, B., & Chintalapudi, K. (2012). Fast data collection in tree-based wireless sensor networks. *IEEE Transactions on Mobile Computing*, *11*(1), 86–99.
30. Kim, Y. G., Wang, Y., Park, B., & Choi, H. H. (2016). A heuristic resource scheduling scheme in time-constrained networks. *Computers & Electrical Engineering*, *54*, 1–15.
31. Zhang, H., Soldati, P., & Johansson, M. (2009). Optimal link scheduling and channel assignment for convergecast in linear wirelesshart networks. In *International symposium on modeling and optimization in mobile, ad hoc, and wireless networks* (pp. 1–8).
32. Zhang, H., Osterlind, F., Soldati, P., Voigt, T., & Johansson, M. (2010). Rapid convergecast on commodity hardware: Performance limits and optimal policies. In *IEEE Communications society conference on sensor mesh and Ad Hoc communications and networks* (pp. 1–9).
33. Saifullah, A., Xu, Y., Lu, C., & Chen, Y. (2010). Real-time scheduling for wirelesshart networks. In *Real-time systems symposium* (pp. 150–159).
34. Saifullah, A., Xu, Y., Lu, C., & Chen, Y. (2011). Priority assignment for real-time flows in wirelesshart networks. In *Euromicro conference on real-time systems* (pp. 35–44).
35. Cao, B., Ge, Y., Kim, C., Feng, G., Tan, H., & Li, Y. (2013). An experimental study for inter-user interference mitigation in wireless body sensor networks. *IEEE Sensors Journal*, *13*(10), 3585–3595.
36. Cao, B., Feng, G., Li, Y., & Wang, C. (2014). Cooperative media access control with optimal relay selection in error-prone wireless networks. *IEEE Transactions on Vehicular Technology*, *63*(1), 252–265.
37. Cao, B., Li, Y., Wang, C., & Feng, G. (2015). Dynamic cooperative media access control for wireless networks. *Wireless Communications and Mobile Computing*, *15*(13), 1759–1772.
38. Liu, N., Plets, D., Vanhecke, K., Martens, L., & Joseph, W. (2015). Wireless indoor network planning for advanced exposure and installation cost minimization. *EURASIP Journal on Wireless Communications and Networking*, *2015*(1), 1–14.
39. Xia, C., Liu, W., & Deng, Q. (2015). Cost minimization of wireless sensor networks with unlimited-lifetime energy for monitoring oil pipelines. *IEEE/CAA Journal of Automatica Sinica*, *2*(3), 290–295.
40. Zhang, J., Jia, X., Zheng, Z., & Zhou, Y. (2011). Minimizing cost of placement of multi-radio and multi-power-level access points with rate adaptation in indoor environment. *IEEE Transactions on Wireless Communications*, *10*(7), 2186–2195.
41. Jin, X., Kong, F., Kong, L., Liu, W., & Zeng, P. (2017). Reliability and temporality optimization for multiple coexisting WirelessHART networks in industrial environments. *IEEE Transactions on Industrial Electronics*,. doi:10.1109/TIE.2017.2682005.
42. IEEE Computer Society (2012). IEEE std. 802.15.4e, Part. 15.4: Low-rate wireless personal area networks (LR-WPANs) Amendament 1: MAC subplayer.

43. Song, J., Han, S., Mok, A. K., Chen, D., Lucas, M., & Nixon, M. (2008). Wirelesshart: Applying wireless technology in real-time industrial process control. In *Real-Time and embedded technology and applications symposium* (pp. 377–386).
44. Liu, J. W. S. (2000). *Real-Time Systems*. New Jersey: Prentice Hall.
45. Camilo, T., Silva, J. S., Rodrigues, A., & Boavida, F. (2007). Gensen: A topology generator for real wireless sensor networks deployment. In *Software technologies for embedded and ubiquitous systems* (pp. 436–445).

Xi Jin received her Ph.D. degree in Computer Science from Northeastern University, China, in 2013. She is currently an Associate Professor with the Shenyang Institute of Automation, Chinese Academy of Sciences. Her research interests include industrial networks, wireless sensor networks, real-time systems and cyber-physical systems.

Huiting Xu received her Master degree in Computer Science from Northeastern University, China, on 2010. She started her Ph.D. from 2011. She is currently an Ph.D. candidate in Computer Science and Engineer department of Northeastern University, China. Her research interests include data center, wireless sensor networks, Internet of things and cyber-physical systems.

Changqing Xia received the Ph.D. degree from Northeastern University, China in 2015. He is currently an assistant professor at Shenyang Institute of Automation, Chinese Academy of Sciences. His research interests include wireless sensor networks and real-time systems, especially the real-time scheduling algorithms, and smart energy systems.



Jintao Wang received the M.Sc. degree from Northeastern University, China, in 2012. He is currently a Ph.D. candidate at Shenyang Institute of Automation, Chinese Academy of Sciences. His research interests include industrial Ethernet and wireless sensor networks.

Peng Zeng received the B.S. degree in Computer Science from Shandong University, Shandong, China, in 1998, and his Ph.D. degree in Mechatronic Engineering from Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, China, in 2005. He is presently Professor with the Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, China. His current research interests include wireless sensor networks for industrial automation, smart grids, and demand response.