Efficient Link Scheduling and Channel Hopping for Convergecast in WirelessHART Networks

Haibo Zhang, Pablo Soldati and Mikael Johansson
School of Electrical Engineering, KTH

Convergecast, in which data from a set of sources is routed towards a data sink, is a critical functionality for wireless networks deployed for industrial monitoring and control. We address the joint link scheduling and channel hopping problem for convergecast in networks operating according to the recent WirelessHART standard. For networks with line and balanced tree routing topologies, we present jointly time- and channel-optimal scheduling policies taking into account different packet buffering capabilities. For networks with general tree routing topology, we first present time-optimal scheduling policies requiring only single-packet buffering. Then we establish the lower bounds on the number of channels for time-optimal convergecast under different packet buffering capabilities, and present a heuristic algorithm for time- and channel-optimal convergecast scheduling. We further demonstrate that, given any fixed number of channels, our scheduling policies are able to generate efficient schedules in terms of minimizing the convergecast time, thereby enabling to explore the tradeoffs between the number of time slots and channels needed to complete convergecast. Finally, we show how the results can be generalized to convergecast problems with pure relay nodes and to support multiple scan rates. Simulation and experimental results confirm that our schemes can provide very fast convergecast in WirelessHART networks.

Categories and Subject Descriptors: C.2.2 [**Computer-communication networks**]: Network Protocols

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Convergecast, TDMA, multi-channel, link scheduling, WirelessHART, channel hopping

1. INTRODUCTION

There is a strong current interest in migrating substantial parts of the traditionally wired industrial infrastructure to wireless technologies to improve flexibility, scalability, and efficiency. However, concerns about network latency, reliability and security along with the lack of device interoperability have hampered the deployment rate. To address these concerns, WirelessHART [HARTCOMM 2007], the first open and interoperable wireless communication standard especially designed to address the needs of real-world industrial applications, has recently been approved and released. WirelessHART is designed to be an easy-to-use wireless mesh networking protocol, leveraging on advanced techniques such as time diversity, frequency diversity, path diversity and power diversity to achieve the level of reliability and latency required to support advanced process monitoring and control applications [Gutierrez 2008].

Reliable real-time data delivery is instrumental to guarantee the performance of closed-loop controllers. To this end, WirelessHART combines Time Division

Multiple Access (TDMA) with per-transaction (packet+acknowledgement) channel hopping to control access to the network. TDMA enables deterministic communications with predictable delays, whereas channel hopping provides frequency diversity for avoiding interferers and reducing multi-path fading effects. Contrary to the philosophy of decentralization advocated in wireless ad-hoc networks, the WirelessHART standard "pushes" the complexity of ensuring reliable and expedited data transfer to a centralized entity, the network manager, responsible for constructing the global transmission schedule. Although the standard offers some guidelines for transmission scheduling, designing optimal scheduling policies to meet the stringent requirements of control applications still remains a major challenge.

Supervisory control with wireless technologies typically involves three phases: retrieving data from sensors to the gateway, computing the control actions (typically in a computer with fast wired access to the gateway), and disseminating the control commands from the gateway to actuators. Collecting data from multiple sources to the gateway is a many-to-one communication paradigm with corresponding networking primitive called *convergecast*. In WirelessHART, the transmission schedule is computed and optimized centrally at the network manager. After extracting the subschedule for each field device, the network manager needs to disseminate the subschedules to the corresponding devices. Within one control loop, the controller may generate different control commands for different actuators. In [Pesonen et al. 2009], we show that multiple commands dissemination can also be performed by reverse-convergecast (i.e., first generate the convergecast schedule from actuators to the gateway, and then use the reverse schedule for commands dissemination). Thus, efficient policies for convergecast scheduling are instrumental for industrial control applications of WirelessHART technology.

Link scheduling for convergecast in multi-hop wireless sensor/ad-hoc networks has been a topic of intense recent research. Most TDMA-based convergecast schemes (e.g., [Upadhyayula et al. 2003; Tseng and Pan 2006; Song et al. 2007; Gandham et al. 2008]) are based on single-channel communication, and focus on making clever use of spatial-reuse to decrease convergecast latency. Only a few studies consider multi-channel TDMA convergecast, see e.g. [Özlem Durmaz Incel and Krishnamachari 2008; Wu et al. 2008], in which channels are initially (and statically) assigned to link/nodes based on interference graph, and link scheduling is performed in a separate stage. Since interference could potentially be removed by scheduling interfering links on different time slots, these approaches may underperform dynamic scheduling and channel hopping. Moreover, optimal channel assignment is NP-hard in general [Wu et al. 2008; Özlem Durmaz Incel et al. 2008]. In this paper, we develop theory and algorithms for optimal convergecast scheduling with per-transaction channel hopping in WirelessHART networks, aiming at minimizing the convergecast latency while making economic use of the available channels and the buffer space at nodes. To the best of our knowledge, there is no existing work on convergecast scheduling taking into account these special features of WirelessHART. Our main contributions are:

—For networks with line routing topology, we prove that the minimum time to complete convergecast is $2N - 1$ time slots where N is the number of field devices in the network (Theorem 1), and the minimum number of channels required for this

operation is $\lceil N/2 \rceil$(Theorem 2) if each node can buffer only one packet at a time slot. When the field devices are allowed to buffer multiple packets, the optimal convergecast time remains the same while the number of required channels can be reduced to $\lceil N - \sqrt{N(N-1)/2} \rceil$(Theorem 3). For both cases, we present jointly time- and channel-optimal scheduling policies with time complexity $O(N^2)$.

—For networks with general tree routing topologies, we demonstrate that the minimum convergecast time is $\max\{2n_1 - 1, N\}$ slots, where n_1 is the maximum number of field devices in a subtree (Theorem 4). We present time-optimal scheduling policies with time complexity $O(DN)$ where D is the depth of the tree and prove that the optimal schedule needs at most D channels and requires only single-packet buffering capability(Corollary 3 and Theorem 5). We also establish lower bounds on the number of channels required for time-optimal convergecast under different packet buffering capabilities(Theorem 7 and Theorem 8) and propose a heuristic algorithm for jointly time- and channel-optimal convergecast scheduling.

—We investigate the tradeoff between the number of available channels and the convergecast time, and demonstrate that our schemes can also be employed to efficiently solve the channel-constrained time-optimal convergecast scheduling problem in which the number of available channels is less than the lower bound required for time-optimal convergecast (Corollary 4).

—We evaluate our schemes through both simulations and experiments on real hardware. Experimental results show that our schemes can provide very fast convergecast operation in WirelessHART networks.

We would like to emphasize that this paper focuses on fundamental performance limits and tradeoffs between convergecast latency, channel utilization, and buffer requirements under the assumption of reliable link-level transmissions. Clearly, these performance limits are also valid when links are unreliable (maintaining high reliability will require longer latency, more parallel transmission, more buffer space, or possibly all). Although we know how to compute and improve the reliability of our schedules [Pesonen et al. 2009], reliability is not covered in this publication.

This paper is organized as follows. Section 2 briefly introduces WirelessHART, and Section 3 presents our model and problem formulations. Section 4 and Section 5 develop theory and algorithms for time- and channel-optimal convergecast scheduling in networks with line and tree routing topologies, respectively. Section 6 discusses the channel-constrained time-optimal convergecast scheduling problem, and Section 7 presents algorithm for sub-schedule extraction and channel hopping. Section 8 discusses possible extensions and reviews related work. Section 9 shows simulation and experimental results and Section 10 concludes the paper. For ease of readability, the proofs of most theoretical results are collected in the Appendix.

2. A BRIEF OVERVIEW OF WIRELESSHART

WirelessHART is an extension of wired HART, a transaction-oriented communication protocol for process control applications. WirelessHART is a complete wireless mesh networking protocol supporting the full range of process monitoring and control applications, including equipment and process monitoring, advanced diagnostics and closed-loop control. As illustrated in Figure 1, the basic elements of a WirelessHART network include:

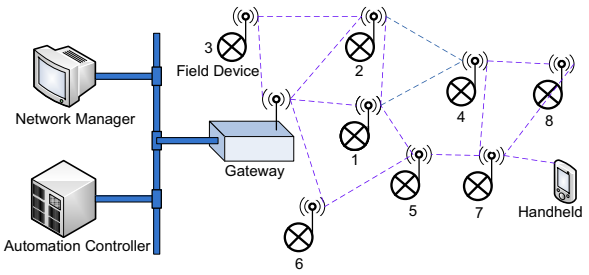


Fig. 1. Example of wirelessHART network infrastructure.

—**Field Devices** connected to the process equipment. All field devices are able to source, sink and forward packets on behalf of other devices in the network.

—**Gateways** enabling communication between host applications and field devices that are members of the WirelesHART network.

—A **Network Manager** responsible for configuring the network, health monitoring, managing routing tables and scheduling communication between devices.

WirelessHART networks may also include adapters for connecting to existing HART-compatible devices and handhelds to configure, maintain or control plant assets.

In contrast to existing standards and protocols for wireless sensor/ad-hoc networks, WirelessHART has some special features described as below:

Network-wide time synchronization and TDMA: WirelessHART has several mechanisms (e.g., pair-wise time synchronization) to promote network-wide clock synchronization within 1ms accuracy – a capability not supported in many other protocols. To meet the requirements for control applications, WirelessHART uses TDMA technology to arbitrate and coordinate communications between network devices. The standardized time slot length is 10ms.

Per-transaction channel hopping: WirelessHART supports per transaction (packet+acknowledgement) channel hopping to provide frequency diversity, avoiding external interferers and reducing multi-path effects. More details about per-transaction channel hopping is given in Section 7.

Limited channel resources: WirelessHART operates based on radios compliant with the IEEE 802.15.4-2006 physical layer standard, supporting only 16 physical channels in the 2.4 GHz ISM band. Channel blacklisting is employed to avoid bad channels with consistently high interference levels (e.g., due to the co-existence with 802.11). In practice, some channels may also be blacklisted to protect wireless services that share a fixed portion of the ISM band with the WirelessHART network. Thus the number of available channels for WirelessHART might be less than 16. Hence efficient channel utilization is instrumental.

Centralized network management and scheduling: To guarantee timely and reliable data delivery, routing topology and transmission schedule are centrally computed at the network manager (which has global knowledge of the network state), and then disseminated to all devices in the network.

Fore more details on WirelessHART, see e.g., [Kim et al. 2008; Gutierrez 2008].

3. PROBLEM FORMULATION

We model a WirelessHART network as a graph $G = (V, E)$ where the vertices in $V = \{v_0, v_1,...,v_N\}$ represent the network devices and the edges in E denote the device pairs that can sustain reliable communication. There is only one gateway (GW) denoted by v_0 and N field devices $v_1...v_N$. Time is synchronized and slotted with standardized length which allows transmitting one data packet and its associated acknowledgement. Each device is equipped with a half-duplex radio transceiver, implying that devices cannot transmit and receive at the same time slot. In our model, we assume that all data transmissions are scheduled in dedicated time slots[1], and parallel transmissions in the same dedicated time slot are scheduled on different channels. Based on this model, we investigate the fundamental performance limits for convergecast operation in WirelessHART networks.

In the first stage of design and analysis, we use a simplified convergecast model in which each field device initially generates exactly one packet destined to the GW, and pre-loads the packet in its data queue before convergecast starts. In Section 8, we will discuss how to extend the solutions to networks with multiple scan rates and pure relay devices. The convergecast messages are routed along a spanning tree rooted at the GW, denoted by $T = (V, E')$ with $E' \subset E$. We refer to T as the routing topology of the network, to stress that this could potentially be very different from the actual physical placement of devices. For every device v_i, f_{v_i} denotes its parent and C_{v_i} represents the set of its children in T. Let $s_t(v_i, v_j)$ denote the state of the directed link $(v_i, v_j) \in E'$ at time slot t with $s_t(v_i, v_j) = 1$ if device v_i transmits a packet to device v_j at time slot t, and $s_t(v_i, v_j) = 0$ otherwise. We use $p_t(v_i)$ to denote the number of buffered packets in device v_i at the end of time slot t, and use $\mathcal{L}_\mathcal{S}$ to represent the length of the convergecast schedule $\mathcal{S}$ (in time slots). The **time-optimal convergecast scheduling** problem, i.e. the minimization of the convergecast schedule length, can then be formulated as follows:

$$\begin{aligned}
&\text{minimize} \quad \mathcal{L}_\mathcal{S} \\
&\text{subject to} \quad p_{\mathcal{L}_\mathcal{S}}(v_0) = N && \text{(1a)} \\
&\qquad \sum_{v_j \in C_{v_i}} s_t(v_j, v_i) + s_t(v_i, f_{v_i}) \leq 1 \qquad \forall v_i \in V, \forall t \in [1, \mathcal{L}_\mathcal{S}] && \text{(1b)} \\
&p_t(v_i) = p_{t-1}(v_i) + \sum_{v_j \in C_{v_i}} s_t(v_j, v_i) - s_t(v_i, f_{v_i}) \qquad \forall v_i \in V, \forall t \in [1, \mathcal{L}_\mathcal{S}] && \text{(1c)} \\
&\qquad s_t(v_i, v_j) \in \{0, 1\} \qquad \forall (v_i, v_j) \in E', \qquad \forall t \in [1, \mathcal{L}_\mathcal{S}] && \text{(1d)}
\end{aligned}$$

Constraint (1a) guarantees that the GW finally receives all packets. Constraint (1b) restricts devices not to transmit and receive at the same time. Constraint (1c) is the conservation of data packets. Constraint (1d) imposes constraint on link status.

As stated in Section 2, spectrum is a scarce resource which should be carefully managed. The second problem to be addressed is the design of a jointly **time- and channel-optimal convergecast scheduling** scheme with the objective to minimize both the number of time slots and the number of channels required to

[1]In WirelessHART, shared slots can be allocated to multi-transmitters to handle retransmission. We do not use shared slots as we assume reliable link-layer communication.

complete convergecast. This problem is formulated as follows

$$\begin{aligned} \text{minimize} \quad & \mathcal{C}_{\mathcal{S}} = \max_{t \in [1, \mathcal{L}_{\mathcal{S}}]} \sum_{(v_i, v_j) \in E'} s_t(v_i, v_j) \\ \text{subject to} \quad & \mathcal{L}_{\mathcal{S}} \text{ solves Problem (1)}, \end{aligned} \tag{2}$$

where $\mathcal{C}_{\mathcal{S}}$ is the number of channels (the maximum number of simultaneous transmissions) used in the schedule $\mathcal{S}$.

As memory is a scarce resource for embedded devices, we consider both the cases where (a) each device can only buffer a single packet at a time slot and (b) all devices can buffer multiple packets.

In some scenarios, the number of channels available for convergecast might be even less than the minimum number of channels obtained by solving Problem (2). Hence another interesting problem is how to minimize the convergecast time with a restricted number of channels, which we refer to as the **channel-constrained time-optimal convergecast scheduling** problem and address it in Section 6.

The scheduling problems above are formulated as integer linear programs and could, at least formally, be solved using optimization tools like MOSEK. However, our experience in [Soldati et al. 2009] indicates that the running time to solve them using ILP tools grows very rapidly and becomes impractical already for rather small problem instances. In this paper, we demonstrate that, for a certain group of routing topologies, the above problems can be solved in polynomial time.

4. LINE ROUTING TOPOLOGY

In this section, we focus on solving the convergecast scheduling problems for networks with a line routing topology. The solution for line topology is instrumental in our developments for more general topologies, but is also interesting in its own right since line is the preferred topology in certain applications such as pipeline monitoring and unmanned offshore gas production [ENGINEERLIVE 2009; HARTCOMM 2009b]. Without loss of generality, the GW is placed at the right end of the line, and the N field devices $(v_1 ... v_N)$ are placed from right to left, as shown in Figure 2.

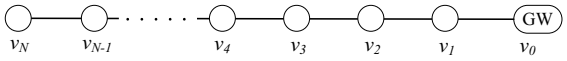


Fig. 2. A network with line routing topology

4.1 Lower bound on convergecast schedule length $\mathcal{L}_{\mathcal{S}}$

THEOREM 1. *The lower bound on the convergecast time $\mathcal{L}_{\mathcal{S}}$ in a network with N field devices organized into a line routing topology is $2N - 1$ time slots.*

PROOF. As shown in Figure 2, link (v_2, v_1) and link (v_1, v_0) can not be scheduled simultaneously due to the half-duplex limitation. To complete convergecast, v_1 needs $N - 1$ time slots to receive packets generated by the other devices, another $N - 1$ time slots to forward these to the gateway, and yet another time slot to forward its own packet. Thus, the lower bound on $\mathcal{L}_{\mathcal{S}}$ is $2(N-1)+1 = 2N-1$. □

4.2 Scenario I: single-packet buffering capability

If each device can buffer at most one packet at any time slot, a device is eligible to receive a packet if and only if it does not hold a packet. The following lemma gives the sufficient and necessary conditions for time-optimal convergecast in a line network with single-packet buffering constraint.

LEMMA 1. *Under single-packet buffering constraint, convergecast in a line network can be completed in* $2N-1$ *time slots iff there is one packet scheduled to be transmitted from* v_1 *to the GW at every odd time slot* $t = 2k-1$*, where* $k \in [1, N]$.

Let $Tx(v_i)$ be the number of packets that device v_i has transmitted since the start of the convergecast operation. Based on Lemma 1, the time-optimal link schedule for convergecast in a line network can be generated using the policy combining the following two rules:

L1. Device v_1 is scheduled to transmit at odd time slots $t = 2k-1$, $k \in [1, N]$.

L2. If device v_{i-1} is scheduled for transmission at time slot $t-1$, then device v_i is scheduled at time t unless v_i has already forwarded all packets it should forward (i.e., unless $Tx(v_i) = N - i + 1$).

The schedule is stored in a two-dimensional dynamic array Sch, where $Sch[t][ch]$ records the device scheduled for transmission at time slot t with channel offset ch. The detailed algorithm to generate time-optimal convergecast schedule in networks with line routing topology is given in Algorithm 1. Figure 3 shows an example of the time-optimal schedule computed by Algorithm 1 in a 5-node line network.

Algorithm 1: Convergecast_Line_Single-packet_Buffering

```
   Input: T = (V, E')
   Output: Sch
1  begin
2      Tx(v_i) ← 0, ∀v_i ∈ V;
3      for t ← 1 to 2N − 1 do
4          ch ← 0;
           /* Schedule device v_1 (Policy L1);                    */
5          if t mod 2 = 1 then
6              Sch[t][ch] ← v_1; ch ← ch + 1;
7              Tx(v_1) ← Tx(v_1) + 1;
           /* Schedule other devices (Policy L2);                 */
8          for each device v_i scheduled in slot t − 1 do
9              if (i + 1 ≤ N) ∧ (Tx(v_{i+1}) < N − (i + 1) + 1) then
10                 Sch[t][ch] ← v_{i+1}; ch ← ch + 1;
11                 Tx(v_{i+1}) ← Tx(v_{i+1}) + 1;
```

THEOREM 2. *For line networks with single-packet buffering capability, the lower bound on the number of channels to complete convergecast in* $2N-1$ *slots is* $\lceil N/2 \rceil$.[2]

[2]Even though frequency reuse may allow to achieve the time bound by using only two channels if each device only interfere with its adjacent neighbors in the line. However, in a real deployment, the interference graph may be very different from the logical routing topology, thus making it hard to fulfill.

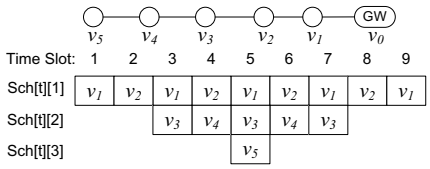


Fig. 3. Optimal schedule for a 5-node line network with single-packet buffering constraint.

COROLLARY 1. *The schedule generated by Algorithm 1 is the only one that can complete convergecast in a line network within* $2N-1$ *time slots under the single-packet buffering constraint.*

By Corollary 1 and Theorem 2, Algorithm 1 yields a time- and channel-optimal schedule. By Theorem 2, the complexity of the for-loop (lines 8-11) in Algorithm 1 is $O(\frac{N}{2})$. Thus, the time complexity of Algorithm 1 is $O((2N-1)\cdot\frac{N}{2}) = O(N^2)$.

4.3 Scenario II: multi-packet buffering capability

If devices can buffer multiple packets, a transmitter does not need to wait until the receiver has emptied its buffer before transmitting the new packet. This yields an opportunity to reduce the number of channels required to complete convergecast in the minimum time. An intuition for this opportunity can be seen in the example in Figure 3. The optimal schedule with single-buffer constraint uses $\lceil\frac{N}{2}\rceil = 3$ channels only once (in time slot 5), and schedules single transmission at slot 1 and slot 2. By allowing v_4 to buffer 2 packets, v_5 may be scheduled for transmission in either time slot 1 or 2, thus reducing the number of channels from 3 to 2.

In the following, we first solve Problem (2) by establishing the lower bound on the number of channels required to complete convergecast in a line network within $2N-1$ time slots for devices with unlimited buffering capability[3]. Then, we present the algorithm to generate the optimal schedule for convergecast in a line network in terms of minimizing both the number of time slots and the number of channels.

THEOREM 3. *Given any schedule* $\mathcal{S}$ *which can complete convergecast in* $2N-1$ *time slots in a line network with* N *field devices with unlimited buffering capability, the lower bound on the number of channels used in* $\mathcal{S}$ *is* $\lceil N - \sqrt{N(N-1)/2}\rceil$.

Theorem 3 gives the lower bound on the number of channels required to complete convergecast in $2N-1$ time slots, but does not prove that the lower bound is always achievable. We next design an algorithm and prove that it always generate time- and channel-optimal schedule for convergecast in networks with line routing topology, thereby demonstrating the tightness of this lower bound.

The basic idea is to allocate as many transmissions as possible at each time slot to capitalize on the available channels. At each time slot t, the algorithm computes the schedule in two steps: **forward scheduling** and **backward scheduling**.

In the **forward scheduling** step, the algorithm searches the eligible devices that can be scheduled for transmission starting from v_1 to v_n. If device v_1 has a packet

[3]In Section 9.2, we analyze the memory efficiency of our scheme and demonstrate that optimal schedule can be generated with very small buffering capability.

in its buffer (i.e., $p_{t-1}(v_1) > 0$), v_1 is scheduled for transmission at this time slot. For device v_i $(i > 1)$, if v_i has a packet in its buffer (i.e., $p_{t-1}(v_i) > 0$) and device v_{i-1} does not have a packet in its buffer, device v_i is scheduled for transmission at this time slot. Therefore, if a device has not finished forwarding all the packets it should transmit and does not have a packet to transmit at the beginning of time slot t, the device receives one packet at time slot t.

After the forward scheduling, the **backward scheduling** step is started if the number of devices scheduled for transmission in the forward scheduling step is less than the maximum transmissions that can be scheduled in this time slot(i.e., $PT_{\max}(t)$ defined in the proof of Theorem 3). Let *end_node* be the farthest device from the GW among all devices still holding packets. In this phase, the algorithm searches devices eligible for transmission in the direction from *end_node* to v_1.

Let $\mathcal{C}_{\mathcal{S}}^{\star} = \lceil N - \sqrt{N(N-1)/2} \rceil$. By Equation (13) and Theorem 1,

$$PT_{\max}(t) = \begin{cases} \mathcal{C}_{\mathcal{S}}^{\star}, & t \in [1, 2N - 2(\mathcal{C}_{\mathcal{S}}^{\star} - 1)]; \\ \lceil \frac{2N-t}{2} \rceil, & t \in (2N - 2(\mathcal{C}_{\mathcal{S}}^{\star} - 1), 2N - 1]. \end{cases} \tag{3}$$

The device v_i that satisfies the following conditions is scheduled for transmission:

—The number of devices scheduled to transmit at time slot t is less than $PT_{\max}(t)$.

—Device v_i is not scheduled in this time slot and device v_i has a packet to transmit. If $i < N$, device v_{i+1} is not scheduled for transmission in this time slot. If $i > 1$, device v_{i-1} is not scheduled for transmission in this time slot.

The detailed algorithm to generate time- and channel-optimal convergecast schedule in networks with line routing topology is given in Algorithm 2. Since both forward and backward scheduling have time complexity $O(N)$, the time complexity of Algorithm 2 is $O(N^2)$. Figure 4 gives the time- and channel-optimal convergecast schedule in a line network with 5 nodes.

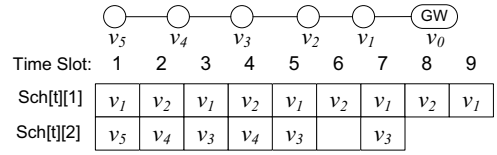


Fig. 4. Optimal schedule for a 5-node line network with multi-packet buffering capability.

Corollary 2. *For a line network with N field devices and unlimited buffering capabilities, the schedule generated by Algorithm 2 can always complete convergecast in $2N-1$ time slots using $\lceil N - \sqrt{N(N-1)/2} \rceil$ channels.*

The forward and backward scheduling can not be easily combined together by scanning the nodes only once due to the following reason: the forward scheduling only needs to schedule necessary transmissions to guarantee that the GW receives one packet in every odd time slot, and leaves as many channels as possible for the second step, whereas the purpose of the backward scheduling is to move the packets at the end of the line as close as possible to the GW, which needs to scan the nodes from the end to the GW. Please refer to the proof of Corollary 2 for insights.

Algorithm 2: ConvergeCast_Line_Multi-packet_Buffering

Input: $T = (V, E')$; $\mathcal{C}_S^\star$
Output: Sch

begin
  $end\_node \leftarrow N$; $p_0(v_i) \leftarrow 1$, $\forall v_i \in V - \{v_0\}$; $p_0(v_0) \leftarrow 0$;
  for $t \leftarrow 1$ **to** $2N - 1$ **do**
    $ch \leftarrow 0$;
    Compute $PT_{\max}(t)$ based on Equation (3);
    /* Forward Scheduling */
    for $j \leftarrow 1$ **to** end_node **do**
      if $j = 1$ **then**
        if $p_{t-1}(v_1) > 0$ **then**
          $Sch[t][ch] \leftarrow v_1$;$ch \leftarrow ch + 1$;
          $p_t(v_1) \leftarrow p_{t-1}(v_1) - 1$; $p_t(v_0) \leftarrow p_{t-1}(v_0) + 1$;
      else
        if $p_{t-1}(v_j) > 0$ && $p_{t-1}(v_{j-1}) = 0$ && $v_j \notin Sch[t]$ && $ch < PT_{\max}(t)$ **then**
          $Sch[t][ch] \leftarrow v_j$; $ch \leftarrow ch + 1$;
          $p_t(v_j) \leftarrow p_{t-1}(v_j) - 1$; $p_t(v_{j-1}) \leftarrow p_{t-1}(v_{j-1}) + 1$;
          if $p_t(v_j) = 0$ && $j = end\_node$ **then**
            $end\_node \leftarrow j - 1$;
    /* Backward Scheduling */
    $j \leftarrow end\_node$;
    while $ch < PT_{\max}(t)$ && $j > 0$ **do**
      if $p_{t-1}(v_j) > 0$ && $(v_{j-1}, v_j, v_{j+1}) \notin Sch[t]$ **then**
        $Sch[t][ch] \leftarrow v_j$; $ch \leftarrow ch + 1$;
        $p_t(v_j) \leftarrow p_{t-1}(v_j) - 1$; $p_t(v_{j-1}) \leftarrow p_{t-1}(v_{j-1}) + 1$;
      $j \leftarrow j - 1$;

5. TREE ROUTING TOPOLOGY

In this section, we investigate the convergecast scheduling problems for networks with tree routing topology. Let D denote the depth of the routing tree. As shown in Figure 5, $T_s(v_i)$ represents the whole subtree rooted at device v_i and n_i is the number of field devices in $T_s(v_i)$. The GW has m children denoted by v_1, v_2... v_m, respectively. Without loss of generality, it is assumed that $n_1 \geq n_2 \geq ... \geq n_m$. For convenience, we logically represent the network into levels so that all devices with the same depth are located at the same level (the GW is at level 0).

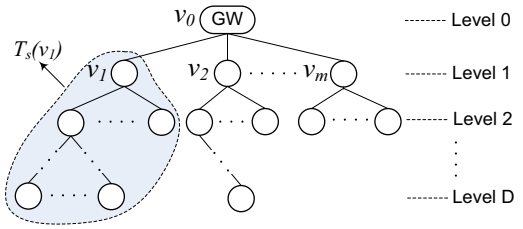


Fig. 5. A network with general tree routing topology

5.1 Lower bound on schedule length ($\mathcal{L}_S$)

THEOREM 4. *The lower bound on the number of time slots required to complete convergecast in a network with tree routing topology and N field devices is $\max\{2n_1 - 1, N\}$, where n_1 is the number field devices in the largest subtree.*

PROOF. For any device v_i, all packets in the subtree $T_s(v_i)$ must go through v_i to the GW. Hence, v_i needs at least $2n_i - 1$ time slots to forward all the packets in subtree $T_s(v_i)$. Since the GW can receive only one packet per time slot, at least N time slots are needed to complete convergecast in a network with N field devices. Thus the lower bound on the convergecast schedule length is $\max\{2n_1 - 1, N\}$. □

Theorem 4 shows that the structure of the routing tree plays a fundamental role in minimizing the convergecast time, and quantifies how unbalanced the tree can be while still admitting time-optimal convergecast. If no subtree has more than $\lfloor (N+1)/2 \rfloor$ nodes, convergecast can be completed in N time slots; otherwise the largest subtree will dominate the achievable convergecast latency. Finding a minimum spanning tree subject to cardinality constraints on the number of nodes in any subtree is called the capacitated minimum spanning tree problem. Although the problem is known to be NP-hard [Papadimitriou 1978], many effective heuristic and approximation algorithms exist [Jothi and Raghavachari 2005].

5.2 Time-optimal convergecast scheduling

In general routing trees, devices can have very different number of children, and hence very different traffic load to forward. It thus makes sense to give devices in larger subtrees higher priority for transmission. The following rules constitute our scheduling policy for time-optimal convergecast in tree networks:

T1. At each time slot t, device $v_i \in C_{v_0}$ is scheduled for transmission if subtree $T_s(v_i)$ has the maximum number of packets left and device v_i is not scheduled for transmission at time slot $t-1$.

T2. At each time slot t, device $v_i \notin C_{v_0}$ is scheduled for transmission if the following three conditions are fulfilled: (1) v_i has not transmitted all the packets it should transmit; (2) device f_{v_i} is scheduled to transmit at time $t-1$; (3) $T_s(v_i)$ has the largest number of packets left among all subtrees rooted at a child of f_{v_i}.

Let $\varphi_t(v_i)$ be the set of candidates that can be scheduled to transmit a packet to device v_i at time slot t. $\varphi_t(v_i) = \{v_j | v_j \in C_{v_i} \wedge Tx(v_j) < n_j \wedge v_j \notin Sch[t-1]\}$, and device v_j that satisfies the following condition is scheduled for transmission:

$$v_j = \arg \max_{v_k \in \varphi_t(v_i)} (n_k - Tx(v_k)).$$

If there are multiple devices with the same maximum number of packets left for transmission, the one with the smallest index is given the highest priority. The detailed algorithm for time-optimal convergecast scheduling is given in Algorithm 3. Figure 6 shows the time-optimal schedule for a sample-tree network. The following corollary proves that the schedule generated by Algorithm 3 can complete convergecast in a tree network using $\max\{2n_1 - 1, N\}$ time slots with single-packet buffering capability.

COROLLARY 3. *Given a network with arbitrary tree routing topology, Algorithm 3 yields a transmission schedule that completes convergecast in* $\max\{2n_1-1, N\}$ *time slots, requiring only single-packet buffering capability.*

Algorithm 3: Convergecast_Tree

Input: $T = (V, E')$
Output: Sch
1 **begin**
2 $\varphi_1(v_i) \leftarrow C_{v_i}, \forall v_i \in V$;
3 $Tx(v_i) \leftarrow 0, \forall v_i \in V$;
4 **for** $t \leftarrow 1$ **to** $\max\{2n_1 - 1, N\}$ **do**
5 $ch \leftarrow 0$;
`/* Schedule the children of the GW (Policy T1); */`
6 **if** $\varphi_t(v_0) \neq \emptyset$ **then**
7 $Sch[t][ch] = \arg\max_{v_k \in \varphi_t(v_0)}(n_k - Tx(v_k))$;
8 $Tx(Sch[t][ch]) \leftarrow Tx(Sch[t][ch]) + 1$; $ch \leftarrow ch + 1$;
9 $\varphi_{t+1}(v_0) \leftarrow \{v_i | v_i \in C_{v_0} \&\& Tx(v_i) < n_i\} - Sch[t][0]$;
`/* Schedule for the other devices (Policy T2); */`
10 **for** *each device* v_i *scheduled in slot* $t-1$ **do**
11 **if** $(C_{v_i} \neq \emptyset) \wedge (Tx(v_i) < n_i)$ **then**
12 $Sch[t][ch] \leftarrow \arg\max_{v_k \in \varphi_t(v_i)}(n_k - Tx(v_k))$;
13 $ch \leftarrow ch + 1$;
14 $Tx(Sch[t][ch]) \leftarrow Tx(Sch[t][ch]) + 1$;
15 $\varphi_{t+1}(v_i) = \{v_j | v_j \in C_{v_i} \wedge Tx(v_j) < n_j\} - Sch[t][ch-1]$;

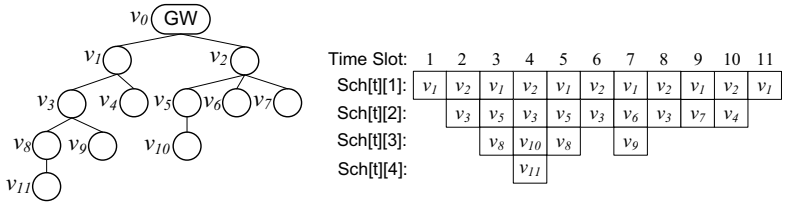


Fig. 6. The time-optimal convergecast schedule for a sample-tree network.

THEOREM 5. *For any network with tree routing topology, the schedule generated by Algorithm 3 completes convergecast using at most* D *channels, where* D *is the depth of the tree.*

It is important to notice that this bound does not depend on the number of nodes in the tree (which can be arbitrarily large), but only on its depth. By Theorem 5, the maximum number of devices that can be scheduled for transmission in one time slot is D. Thus the time complexity of the embedded for-loop (Line 10 - Line 15) is $O(D)$, and the time complexity of Algorithm 3 is $O(D \cdot \max\{2n_1 - 1, N\})$.

5.3 A special tree: balanced complete m-ary tree

To confirm our intuition that the topology of the routing tree is critical, we will demonstrate that Algorithm 3 generates jointly time- and channel-optimal schedules for balanced complete m-ary trees. A balanced complete m-ary tree has m largest subtrees with N/m nodes in each subtree. All leaf nodes are at the same depth and each node (except the leaves) has exactly m children. In the special case of $m = 1$, the m-ary tree collapses into a line topology which admits time- and channel-optimal solutions as proved in Section 4. For $m \geq 2$, the largest subtree has $2N/m - 1 < N$ nodes, and by Theorem 4 the convergecast time bound in a balanced complete m-ary tree is $\max\{2N/m - 1, N\} = N$ time slots.

THEOREM 6. *Given a balanced complete m-ary tree with depth D, the minimum number of channels required to complete convergecast in N time slots is D.*

By Theorem 5, Algorithm 3 generates a time-optimal convergecast schedule using at most as many channels as the depth of the tree. This result, combined with Theorem 6, demonstrates that Algorithm 3 yields time- and channel-optimal convergecast schedules in networks with balanced complete m-ary tree routing topology.

5.4 Analysis for time- and channel-optimal convergecast scheduling

In Algorithm 3, a device is scheduled for transmission as long as it has a packet to transmit and its parent has empty buffer. This greedy policy is not efficient in terms of channel utilization. For instance, the schedule in Figure 6 uses 4 channels (i.e., 4 devices are scheduled at time slot 4) to complete convergecast in 11 time slots. However, it is possible to fulfill the time bound with only 3 channels by re-arranging the scheduling order to better utilize the time slots (e.g., slots 6, 8 and 9) in which only two devices are scheduled. We next establish lower bounds on the number of channels required for time-optimal convergecast in networks with general tree routing topology under different packet buffering capabilities.

5.4.1 *Lower bound on $\mathcal{C}_\mathcal{S}$ with single-packet buffering capability*

THEOREM 7. *For a network with tree routing topology, the number channels required to complete convergecast in $\mathcal{L}^\star_\mathcal{S} = \max\{2n_1 - 1, N\}$ time slots under single-packet buffering constraint satisfies:*

$$\mathcal{C}_\mathcal{S} \geq \left\lceil \frac{(\mathcal{L}^\star_\mathcal{S} + 1) - \sqrt{(\mathcal{L}^\star_\mathcal{S} + 1)^2 - 4\sum_{d=1}^{D} d \cdot n(d)}}{2} \right\rceil, \tag{4}$$

where D is the depth of the tree and $n(d)$ is the number of nodes with depth of d.

5.4.2 *Lower bound on $\mathcal{C}_\mathcal{S}$ with unlimited buffering capability.*

When devices can buffer multiple packets, a device can be scheduled for transmission as long as it has a packet to transmit and its parent has the buffering capability to receive it. In this way, more devices can be scheduled at the first several time slots, thereby reducing the number of channels required for convergecast.

THEOREM 8. *With multi-packet buffering capability at each field device, the number of channels $\mathcal{C}_\mathcal{S}$ required to complete convergecast in $\mathcal{L}^\star_\mathcal{S} = \max\{2n_1 - 1, N\}$ time*

slots in a tree network satisfies:

$$\mathcal{C}_{\mathcal{S}} \geq \left\lceil \frac{(2\mathcal{L}^{\star}_{\mathcal{S}}+1) - \sqrt{(2\mathcal{L}^{\star}_{\mathcal{S}}+1)^2 - 4\sum_{d=1}^{D} d \cdot n(d)}}{2} \right\rceil, \tag{5}$$

where D is the depth of the tree and $n(d)$ is the number of nodes with depth of d.

Remark: It is worth noting that the lower bounds on the number of channels for convergecast may not be always achievable. For instance, consider the example in Figure 7 with a tree structure consisting of 5 lines with 6,2,1,1,1 devices, respectively. By Theorem 7, at least $\mathcal{C}_{\mathcal{S}}$=3 channels are required to complete convergecast in $\max\{2n_1 - 1, N\} = 11$ time slots. The schedule in Figure 7 assigns different colors to devices in different lines. By Corollary 1, the schedule for the longest line in Figure 7 is the only schedule that can complete convergecast in 11 time slots with single-packet buffering constraint. Still, device v_5 is not scheduled for transmission, and the only available position in time slot 9 can not be assigned to v_5 since v_1 is already scheduled to transmit to the GW. Thus, it is not feasible to complete convergecast in 11 time slots with only 3 channels. This example further confirms that the tightness of the channel bounds also depends on the structure of the routing topology.

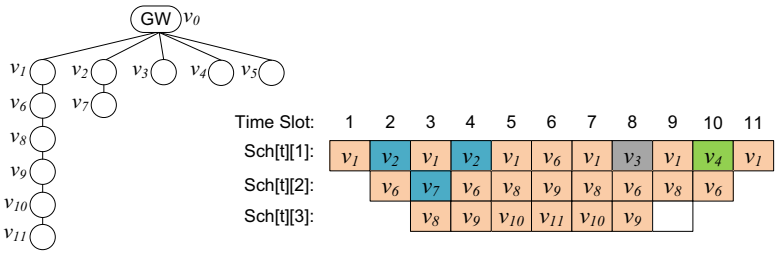


Fig. 7. An example to illustrate that the lower bound on the number of channels is not always feasible. Devices in the same line are labeled with the same color.

5.5 Heuristic algorithm for time- and channel-optimal convergecast scheduling

We now consider the jointly time- and channel-optimal convergecast scheduling problem for arbitrary tree routing topologies. Since we could not find a provably optimal strategy, we propose a heuristic solution inspired by the optimal policies for line topology, and evaluate its performance through extensive simulations.

Unlike the line routing topology, tree topologies may be very irregular with large difference in the depth and size of individual subtrees. If the channel lower bound $\mathcal{C}_{\mathcal{S}}$ computed with Theorem 7 or Theorem 8 is equal to the depth D of the routing tree, then Algorithm 3 returns the time- and channel-optimal schedule according to Theorem 5. However, if $\mathcal{C}_{\mathcal{S}} < D$, the transmissions must be carefully scheduled to achieve time optimality: for example, nodes with more packets to forward should be given higher priority. To prioritize the transmissions, we introduce the definition of *latest release time* as follows: for a packet generated by node v_i, its latest release time at any node v_j in its routing path, denoted by $r_{i,j}$, is the latest time slot in

which node v_j must be scheduled to transmit this packet in order to achieve the lower bound on convergecast time. The basic idea of the heuristic scheduling is to prioritize the nodes holding packets with the smaller latest release time. In the following we first give the approach to compute $r_{i,j}$, and then present the scheduling algorithm.

5.5.1 *Computation of the latest release time.*

For any node v_i, the latest time to release the packet generated by itself, i.e., $r_{i,i}$, can be computed in two steps as illustrated in Fig. 8: the first step initializes $r_{i,i}$ for packets generated by leaf nodes, and the second step deals with non-leaf nodes.

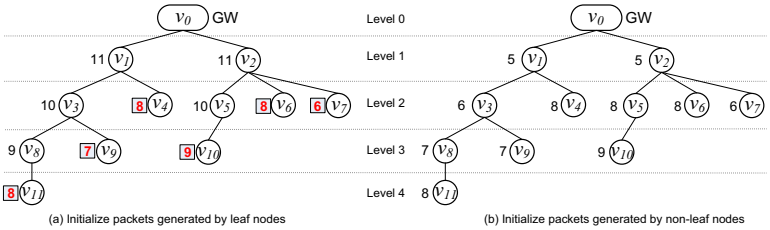


Fig. 8. Computation of latest release time for packets in the tree given in Figure 6. In *a*), starting from the GW, both v_1 and v_2 get $r' = \mathcal{L}^{\star}_{\mathcal{S}} = 11$. Scanning from v_1, the child with maximum descendants is v_3 which gets $r'(v_3) = 10$, while node v_4 gets $r_{4,4} = r'(v_4) = r'(v_3) - 2 = 8$. The procedure continues moving from v_3 to the end of the subtree and is repeated for the subtree rooted at v_2. Case *b*) shows the computation for the non-leave nodes v_8, v_3, v_5, v_1 and v_2.

Let $r'(v_i)$ denote the latest release time for the last packet forwarded by node v_i, and $|C_{v_i}|$ be the number of its children. For any device v_i at Level 1, $r'(v_i) = \mathcal{L}^{\star}_{\mathcal{S}} = \max\{2n_1 - 1, N\}$. For any device $v_i \in V - \{v_0\}$, its children $v_j \in C_{v_i}$ are sorted in non-increasing order of the size of the subtrees they root, and stored in $C_{v_i}[1],\ldots,$ $C_{v_i}[|C_{v_i}|]$, respectively. Assume that the last packet forwarded by v_i comes from the child with the largest number of descendants. Then $r'(C_{v_i}[1]) = r'(v_i) - 1$. The remaining children get $r'(C_{v_i}[i]) = r'(C_{v_i}[i-1]) - 2$. This gives priority to transmit the received packet before receiving a packet from another child, thus lowering the buffering requirement at each node. If v_i is a leaf node, $r_{i,i} = r'(v_i)$. This step is repeated until the packets generated by all leaf nodes have been processed, as shown in Figure 8(a). Starting from depth D-1, the second step scans non-leaf node v_i bottom-up by setting $r_{i,i} = \min_{v_j \in C_{v_i}} r_{j,j} - 1$, resulting in higher priority for owned packet compared to packets from the children. Figure 8(b) illustrates this step. It might occur that nodes at the same level (e.g., v_8 and v_9 in our example) get the same latest release time. This "conflict" is resolved in the scheduling step by prioritizing the node with the largest number of packets left to transmit.

When a packet generated by v_i is sent to its parent v_j at time $r_{i,i}$, the latest release time to forward this packet from v_j is $r_{i,j} = r_{i,i} + 1$. Thus, whenever a node v_j forwards the packet generated by v_i to its parent v_k, $r_{i,k}$ is computed as follows:

$$r_{i,k} = r_{i,j} + 1. \tag{6}$$

5.5.2 *Scheduling algorithm.*

Inspired by the time- and channel-optimal convergecast scheme in line networks, we propose a heuristic algorithm for tree topology that computes the convergecast schedule in two steps: **connectivity keeping** and **priority-based scheduling**. The first step schedules transmissions to guarantee that the GW can receive packets as continuously as possible, while the second step pushes packets forward from the leaves towards the gateway to make full use of the available channels.

Similar to the forward scheduling step for line topology, the purpose of connectivity keeping step is to guarantee that at each level devices are scheduled so that packets can reach the GW timely. This is achieved by avoiding *schedule holes*:

DEFINITION 1. *At any time slot t in schedule $\mathcal{S}$, level $d \in [1, D]$ is called a* ***schedule hole*** *if the following conditions hold:*

(*1*) *There is no packet in level d;*

(*2*) *In each level $j \in [1, d)$, there is only one packet to be transmitted in slot $t+1$;*

(*3*) *There is at least one packet in a level k with $k > d$.*

Once a schedule hole occurs, it can not be removed and results in a time slot without transmission to the gateway, see Figure 9(b). According to Theorem 4, the lower bound on convergecast schedule length is $\max\{2n_1 - 1, N\}$. Thus at most $\max\{2n_1 - 1 - N, 0\}$ schedule holes can be allowed for time optimality. Based on this observation, at any time slot t, connectivity keeping is triggered if the number of schedule holes in the schedule is no smaller than $2n_1 - 1 - N$. The connectivity keeping step is described by the example given in Figure 9(c). The algorithm seeks schedule holes level by level, starting from level 1, and schedules devices as follows:

—At level 1, the packet with the minimum latest release time is scheduled.

—At level $i > 1$, if there is no packet that can be scheduled in slot $t+1$, the device v_i in level i+1 that satisfies the following conditions is scheduled for transmission: (1) v_i can be scheduled in slot t. (2) the packet at device v_i has the minimum latest release time among all packets that are eligible for transmission in slot t.

—If level i has more than one packet that can be scheduled for transmission at slot $t+1$, the connectivity keeping step terminates.

—For each level, when multiple devices hold packets with the same minimum latest release time, the device with the maximum packet left to transmit is prioritized.

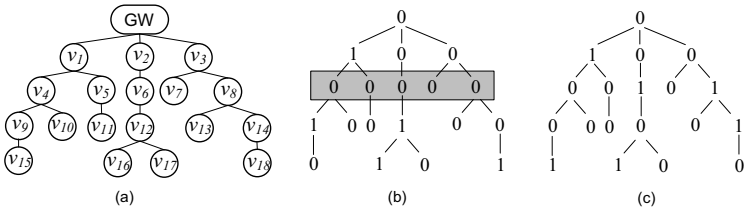


Fig. 9. (a) is a general tree network. In both (b) and (c), the digit 1 identifies a packet in the corresponding device. In (b), level 2 is a schedule hole. In (c), device v_1 and either v_6 or v_8 must be scheduled to avoid schedule hole. If v_6 is scheduled, either v_{15} or v_{16} must be scheduled.

Similar to the backward scheduling in line networks, the basic idea of the priority-based scheduling step is to maximize the number of transmissions after the connectivity keeping step by prioritizing devices holding packets with smaller latest release time. Let Q^t be a priority queue storing the devices that can be scheduled for transmission at time slot t. The devices in Q^t are sorted in non-decreasing order of latest release time for their packets. The priority-based scheduling works as follows: if the number of devices that have been scheduled for transmission in time slot t is less than $PT_{\max}(t)$ (defined in Equation (17)), the device v_k that holds the packet with the minimum latest release time is scheduled for transmission. This procedure is repeated until either $PT_{\max}(t)$ devices have been scheduled for transmission or Q^t is empty. The detailed algorithm is given in Algorithm 4. Figure 10 gives the schedule generated by Algorithm 4 for the tree network in Figure 6. This schedule completes convergecast in 11 time slots with only 3 channels.

Algorithm 4: ConvergeCast_Tree_Single_Buffer

```
   Input: T = (V, E'), C*_S
   Output: Sch
1  begin
2      Packet_GW ← 0; t ← 0; Num_holes ← 0;
3      Initialize t_d(v_i), ∀v_i ∈ V − {v_0};
4      while Packet_GW < N do
5          t ← t + 1; ch ← 0;
6          Generate Q^t;
7          Compute PT_max(t) based on Equation (17);
           /* connectivity keeping                                   */
8          for i ← 1 to D do
9              if Num_holes < 2n_1 − 1 − N then
10                 Num_holes ← Num_holes+1;
11                 break;
12             if level i is a schedule hole then
13                 schedule device v_i to avoid schedule hole;
14                 t_d(v_i) ← t_d(v_i) + 1; Sch[t][ch] ← v_i;
15                 ch ← ch + 1; Q^t ← Q^t − {v_i};
16             else
17                 break;
18             if v_i is in level 1 then
19                 Packet_GW ← Packet_GW + 1;
           /* Priority-based scheduling                              */
20         while ch < PT_max(t) and !Q^t.empty do
21             v_i ← Q^t.GetFirst;
22             t_d(v_i) ← t_d(v_i) + 1; Sch[t][ch] ← v_i;
23             ch ← ch + 1; Q^t ← Q^t − {v_i}; v_i ← Q^t.GetNext;
```

Time Slot:	1	2	3	4	5	6	7	8	9	10	11
Sch[t][1]:	v_1	v_2	v_1	v_2	v_1	v_2	v_1	v_2	v_1	v_2	v_1
Sch[t][2]:		v_3	v_5	v_3	v_6	v_3	v_5	v_3	v_7	v_4	
Sch[t][3]:			v_8	v_{11}	v_8	v_{10}	v_9				

Fig. 10. Time- and channel-optimal schedule generated by Algorithm 4 for the network in Figure 6.

6. CHANNEL-CONSTRAINED TIME-OPTIMAL CONVERGECAST SCHEDULING

Channels are scarce resource in WirelessHART, and it might occur that less than $\mathcal{C}_{\mathcal{S}}^{\star}$(the lower bound) channels are available for convergecast operation. Therefore, understanding the tradeoffs between the number of available channels and the convergecast time is instrumental. This trade-off is readily illustrated in Figure 11: Region I is clearly infeasible, i.e., at least one channel is needed, and the network cannot be evacuated in less than $\mathcal{L}_{\mathcal{S}}^{\star} = \max\{2n_1 - 1, N\}$ time slots; Region II is feasible but unattractive since the upper bound on the number of channels required for time-optimal convergecast, denoted by $\mathcal{C}_{\mathcal{S}}^{t}$, is D (i.e., $\mathcal{C}_{\mathcal{S}}^{t} = D$), as claimed in Theorem 5. In what follows, we characterize the optimal trade-off surface between convergecast time and the number of available channels shown in Region III, and provide scheduling policies capable to minimize the convergecast time for a given number of channels $\mathcal{C}_{\mathcal{S}} < \mathcal{C}_{\mathcal{S}}^{t}$. We have the following result:

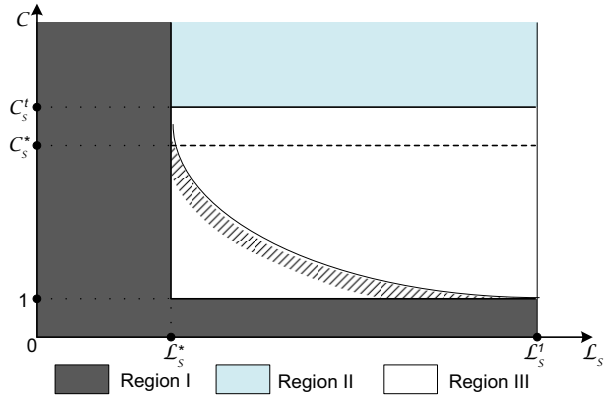


Fig. 11. Relation between the number of channels and the convergecast time.

Corollary 4. *Given $\mathcal{C}_{\mathcal{S}} < \mathcal{C}_{\mathcal{S}}^{t}$ channels, the number of time slots required to complete convergecast satisfies*

(*1*) *Line routing topology with single-packet buffering capability:*

$$\mathcal{L}_{\mathcal{S}} \geq \left\lceil \frac{N(N+1)}{2\mathcal{C}_{\mathcal{S}}} + 2\mathcal{C}_{\mathcal{S}} - 2 \right\rceil. \tag{7}$$

(*2*) *Line routing topology with multi-packet buffering capability:*

$$\mathcal{L}_{\mathcal{S}} \geq \left\lceil \frac{N(N+1)}{2\mathcal{C}_{\mathcal{S}}} + \mathcal{C}_{\mathcal{S}} - 1 \right\rceil. \tag{8}$$

(*3*) *Tree routing topology with single-packet buffering capability:*

$$\mathcal{L}_{\mathcal{S}} \geq \left\lceil \frac{\sum_{d=1}^{D} d \cdot n(d)}{\mathcal{C}_{\mathcal{S}}} + \mathcal{C}_{\mathcal{S}} - 1 \right\rceil. \tag{9}$$

(*4*) *Tree routing topology with muliti-packet buffering capability:*

$$\mathcal{L}_{\mathcal{S}} \geq \left\lceil \frac{\sum_{d=1}^{D} d \cdot n(d)}{\mathcal{C}_{\mathcal{S}}} + \frac{\mathcal{C}_{\mathcal{S}}}{2} - \frac{1}{2} \right\rceil. \tag{10}$$

The convergecast algorithms designed in the previous sections can be easily modified to solve the channel-constrained time-optimal convergecast problem by enforcing a fixed number of channels $\mathcal{C}_{\mathcal{S}}$. For instance, Algorithm 2 uses $\mathcal{C}^{\star}_{\mathcal{S}} = \lceil N - \sqrt{N(N-1)/2} \rceil$ channels to bound the maximum number of transmissions scheduled per time slot. By setting the number of channels to $\mathcal{C}_{\mathcal{S}} < \mathcal{C}^{t}_{\mathcal{S}}$, Algorithm 2 can be used to compute channel-constrained schedules for networks with line routing topology. Similarly, Algorithm 4 can be employed to solve this problem for networks with tree routing topologies. In Section 9.3, we explore Region III by comparing the schedule length obtained by our algorithms with the lower bounds established in Corollary 4.

7. SUB-SCHEDULE EXTRACTION AND CHANNEL HOPPING

In WirelessHART networks, the network manager computes the convergecast schedule, extract the sub-schedule for each device and send the sub-schedule to the device. In the sub-schedule, each transmission is associated with a channel offset which represents the logical channel to be used. In wirelessHART, the logical channels can be mapped to the physical channels as follows:

$$Activechannel = (\text{channeloffset} + \text{ASN}) \ \% \ \text{Number of Active Channels}, \tag{11}$$

where ASN denotes the Absolute Slot Number which is the count of all slots that have occurred since the network is formed.

At every time slot, each device can work in three states: *Transmit* (T), *Receive* (R) and *Sleep* (S). Each field device stores the sub-schedule and channel hopping sequence in a 2-dimensional array $S\_sch[1, \mathcal{L}_S][1, 2]$, where $S\_sch[t][1]$ records the device state associated to slot t and $S\_sch[t][2]$ records the channel offset used by the field device at slot t. At any slot t, the state and channel offset for each device v_i can be generated as follows:

S1. If $Sch[t][ch] = v_i$, v_i works in *Transmit* state using channel with offset ch.

S2. If $Sch[t][ch] \in C_{v_i}$, v_i works in *Receive* state using channel with offset ch; otherwise, v_i remains in *Sleep* state.

The algorithm to generate the sub-schedule and channel hopping sequence for each filed device v_i is given in Algorithm 5. The time complexity of Algorithm 5 is $O(N\mathcal{C}_{\mathcal{S}})$ where $\mathcal{C}_{\mathcal{S}}$ is the number channels used. Figure 12 gives the sub-schedule and channel hopping sequence for device v_3 in the example given in Figure 6.

Time Slot:	1	2	3	4	5	6	7	8	9	10	11
S_sch[t][1]:	S	T	R	T	R	T	R	T	S	S	S
S_sch[t][2]:		1	2	1	2	1	2	1			

Fig. 12. Sub-schedule and channel hopping for device v_3 in the example given in Figure 6.

Algorithm 5: Sub-schedule and Channel Offset Generation(v_i)

```
   Input: Sch[1, L_S][1, C_S]
   Output: S_sch[1, L_S][1, 2]
1  begin
2      for t ← 1 to L_S do
3          for ch ← 0 to C_S − 1 do
               /* Acting as Transmitter                          */
4              if Sch[t][ch] = v_i then
5                  S_sch[t][1] ← T;
6                  S_sch[t][2] ← ch;
7              /* Acting as Receiver                             */
8              else if Sch[t][ch] ∈ C_{v_i} then
9                  S_sch[t][1] ← R;
10                 S_sch[t][2] ← ch;
11             /* Sleep                                          */
12             else S_sch[t][1] ← S;
```

8. DISCUSSION

8.1 Advantages of channel hopping without frequency reuse

Per-transaction channel hopping without channel reuse can greatly simplify the scheduling problems and enables to design scheduling policies with low-complexity for control applications. First, the absence of multi-access interference allows to design simple scheduling policies without constructing interference graphs due to the difficulty in detecting interference and the time-varying interference pattern. Moreover, channels are dynamically assigned during scheduling, avoiding preliminary optimal channel assignment which is NP-hard in general. Second, we proved that the number of channels $C_{\mathcal{S}}$ required for time-optimal convergecast in a network with tree routing topology is bounded as $C^{\star}_{\mathcal{S}} \leq C_{\mathcal{S}} \leq D$, where the upper bound D coincides with the depth of the tree[4], while the lower bound $C^{\star}_{\mathcal{S}}$ can depend on the buffering capability and the structure of routing tree. Even when frequency reuse is allowed, the upper bound on $C_{\mathcal{S}}$ was proven to be $\triangle(G') + 1$, with $\triangle(G')$ being the maximum degree of the interference G' constructed from the original topology [Özlem Durmaz Incel et al. 2008]. Since control applications may require quite dense deployments in a limited area, $\triangle(G')$ will typically be much larger than D. In the worst case, when the network is fully connected, $\triangle(G')$ is equal to the total number of field devices N, leaving no benefit for frequency reuse.

Even though per-transaction channel hopping without channel-reuse is preferred in WirelessHART networks, we argue that our schemes can be easily extended to allow channel reuse, given the knowledge of interference graph. Based on the interference graph, the channel used by node v_i for transmission in time slot t can be assigned in the following way: Let ψ_i denote the set of feasible channels that can be allocated to node v_i for transmission in time slot t, and ψ_i is initialized as the set of all active channels in the network. For each interfering node of v_i, if it is scheduled for transmission in time slot t, the channel allocated for this interfering node is

[4]Although this poses constraints on the depth of routing tree, in control applications it is recommended to deploy networks with small depth D (typically 4-5 hops) due to the stringent delay requirement [HARTCOMM 2009a].

removed from ψ_i. If ψ_i is empty after all interfering nodes have been scanned, node v_i should not be scheduled in this time slot; otherwise node v_i randomly chooses one channel from ψ_i for transmission in time slot t.

8.2 Convergecast with pure relay nodes

In WirelessHART networks, some devices may join as pure relays, i.e., nodes that do not generate traffic. Unlike the pure convergecast problem, the bound on convergecast time in networks with pure relays not only depends on the number of nodes in the network and the structure of the routing topology, but also depends on the packet distribution and the precise ordering of relays-vs-sensor nodes. For example, in a line network with N field devices, of which only one device holds a packet, convergecast can take any time from one time slot (when the packet is generated by the node closest to the gateway) to N time slots (when it is generated at the node furthest away). Therefore, it is hard to get tight closed-form bounds on convergecast time in networks with pure relay nodes.

Let $g(v_i)$ be the number of packets generated by node v_i. If $g(v_i) = 0$, node v_i is a pure relay node. Let $h(v_i)$ be the hop-count from node v_i to the GW. The following corollary gives a lower bound as well as an upper bound on convergecast time in general tree networks with pure relay nodes.

COROLLARY 5. *The minimum convergecast schedule length, denoted by $\mathcal{L}^{\star}_{\mathcal{S}}$, satisfies:*

$$\mathcal{L}^{\star}_{\mathcal{S}} \geq \max\left\{ \max_{v_i \in V} \left(2\sum_{v_j \in T_s(v_i)} g(v_j) - g(v_i) + h(v_i) - 1\right), \sum_{v_i \in V} g(v_i)\right\};$$
$$\mathcal{L}^{\star}_{\mathcal{S}} \leq \max_{i \in [1,m]} \left(\sum_{v_j \in T_s(v_i)} (h(v_j) - 1) \cdot g(v_j)\right) + \sum_{v_i \in V} g(v_i).$$

The schemes designed for the classic convergecast problem can be easily modified to generate efficient convergecast schedules for networks with pure relay nodes. For instance, the policies presented in Section 5.2 can be modified as follows:

1. At each time slot t, device $v_i \in C_{v_0}$ is scheduled for transmission if device v_i has a packet to transmit and $T_s(v_i)$ has the maximum number of packets left.

2. At each time slot t, device $v_i \notin C_{v_0}$ is scheduled for transmission if the following three conditions are fulfilled: (1) v_i has a packet to transmit; (2) device f_{v_i} is not scheduled for transmission in this time slot; (3) $T_s(v_i)$ has the largest number of packets left among all subtrees rooted at a child of f_{v_i}.

It can be seen that the above policies always give transmission priorities to nodes with more packets to transmit and schedule as many transmissions as possible in each time slot with the expectation to evacuate the packets in the network as fast as possible. The solution for channel-constrained minimum convergecast time scheduling can also be extended in a similar way.

8.3 Supporting multiple scan rates

In some applications, the scan rates of sensors in the same logical WirelessHART network might be different. The WirelessHART standard recommends that measurement scan rates are configured as integer multiples of the fastest scan rate, i.e., the supported scan rates are defined as 2^n where n is positive or negative integer

values, e.g., scan rate selections of 500ms, 1s, 2s and 4s [HARTCOMM 2007]. We propose to address such situations by first generating the convergecast schedule for each scan rate, and then merging them together starting with the fastest to the slowest rate as the standard suggests. This policy is illustrated by the example in Fig. 13.

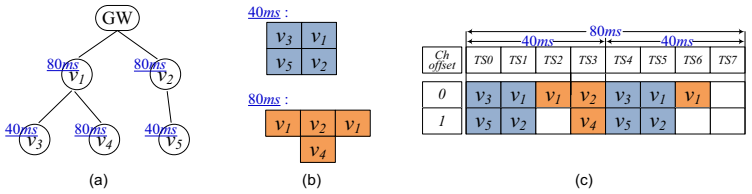


Fig. 13. (a) the network containing devices with different scan rates (b) individual schedules for different scan rates (c) the superframe after merging.

Although the above policy gives one solution that supports multiple scan rates, it is most likely not optimal. However, the WirelessHART standard suggests to leave at least 50% of the slots in the schedule empty for retries and listens. This means that a well-provisioned network will be able to tolerate some suboptimality in the merging phase. The development of more efficient merging policies that also account for the allocation of free slots is a topic for future research, and left outside of the scope of this paper.

8.4 Related work

Previous work on static and distributed link scheduling in radio networks includes [Hajek and Sasaki 1988; Ramanathan and Lloyd 1992; Sekhar and Sivarajan 2000; Chen et al. 2005; Cogill and Hindi 2007; Wan et al. 2009; Wan et al. 2009]. However, most of these works consider saturated data sources and focus on average link rate performance, thereby not applicable to deadline-constrained transmission scheduling in WirelessHART.

Somewhat related is the literature on TDMA-based gossiping (see [Gasieniec and Potapov 2002; Manne and Xin 2006; Huang et al. 2008; Huang et al. 2010] and references therein). However, several distinct and significant features make the gossiping algorithms not immediately suitable for convergecast in WirelessHART. First, gossiping algorithms typically use protocol-based models in which channels are spatially reused by nodes that are sufficiently separated (usually 3-hop). Although it might be possible to extend some existing gossiping schemes for convergecast, much simpler convergecast schemes can be designed for WirelessHART due to its special features. Second, the matching-like solutions used in gossiping problems aim to activate as many non-colliding transmissions as possible, which requires a large number of channels and memory for packet buffering, thereby becoming disadvantageous even for small networks, see e.g., [Soldati et al. 2009].

Much work has been done on designing efficient TDMA-based convergecast protocols for wireless sensor/ad-hoc networks. Choi et al. [Choi et al. 2005] proved that the decision version of time-optimal convergecast scheduling problem for single-channel wireless sensor networks is NP-complete in a weak sense. Tseng and

Pan [Tseng and Pan 2006] studied the minimum delay beacon scheduling problem for quick convergecast in ZigBee/IEEE 802.15.4 single-channel tree-based wireless sensor networks and proved it to be NP-complete. However, we demonstrate that, with multi-channel communication without frequency reuse, time-optimal convergecast scheduling can be solved in polynomial time given enough channels. A closely related work by Gandham et al. [Gandham et al. 2008] focuses on designing distributed scheduling policies to minimize convergecast time in single-channel WSNs. We employ the same methodology (i.e., first study line network and then for tree network) in [Gandham et al. 2008] to study the convergecast scheduling problem in WirelessHART. We present solutions for jointly time- and channel-optimal convergecast scheduling, which can not be obtained by extending Gandham's work.

Interference and multi-hop fading strongly degrade the performance of existing single-channel based schemes. A natural approach to avoid interference and increase throughput is to use multiple channels. A Tree-based Multi-Channel Protocol (TMCP) for data collection was proposed in [Wu et al. 2008]. This scheme allocates different channels to vertex-disjoint subtrees rooted at the gateway. Durmaz Incel and Krishnamachari in [Özlem Durmaz Incel et al. 2008] proved that the receiver-based channel assignment problem in TDMA-based convergecast scheduling with multi-channel communication and frequency reuse is NP-complete. Only when the interference can be completely removed, their results recall our minimum convergecast time. Still, the jointly time- and channel-optimality of the scheduling problem, the channel-constrained time-optimality, as well as the memory efficiency are not considered in [Özlem Durmaz Incel et al. 2008]. The time synchronized mesh protocol [Pister and Doherty 2008], which defines the lower layers of the WirelessHART standard, has been commercialized and successfully deployed on several industrial sites by DUST networks. Impressive reliability and throughput in actual industrial scenarios have been reported by DUST Networks, but there is no public results on the efficiency of TSMP under delay constrained traffic, which is the main theme of this work.

This paper extends our previous experience on real-time data delivery and time- and channel-optimal convergecast scheduling for line routing topology, i.e., [Zhang et al. 2009; Soldati et al. 2009], to networks with general tree routing topologies. The proposed schemes are compliant with the special features of WirelessHART.

9. PERFORMANCE EVALUATION

In this section, we evaluate the performance of our convergecast schemes through both simulations and experiments on real hardware. Since the algorithms for line routing topology and the algorithm for time-optimal convergecast in general tree routing topology have already been proven to be optimal, we focus our evaluations on the heuristic algorithm for jointly time- and channel-optimal convergecast scheduling on arbitrary tree routing topologies. We also demonstrate the memory efficiency of our schemes and investigate the performance of Algorithm 2 and Algorithm 4 applied to the channel-constrained time-optimal convergecast problem. Finally, we evaluate our schemes through simulations in COOJA simulator [Österlind et al. 2006] and validate the simulation results on the Tmote Sky platform [MOTEIV 2004] running the Contiki operating system [Dunkels 2009].

9.1 Performance of the heuristic convergecast scheduling algorithm

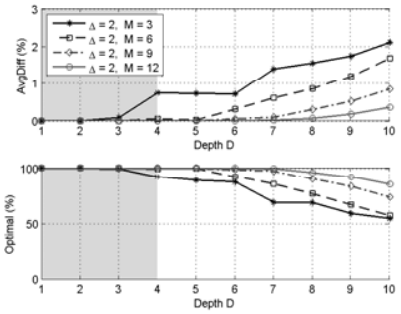
To understand the performance of Algorithm 4, we apply it to the jointly time- and channel-optimal convergecast scheduling problem on a large set of randomly generated tree topologies. We then compare the length of the generated schedules with the theoretical lower bound (although the bound is not always achievable) and study the difference for various topology parameters.

To generate trees of varying depth with large variation in subtree size, we use the following algorithm: First, M children are added to the GW; Then a random number of children chosen with uniform distribution in $[0, \Delta]$ is added to any device at depth $d \in [1, D)$[5]. In this set of simulations, D is varied from 1 to 10 and Δ is varied from 1 to 3. For each setting of $\{M,\ D,\ \Delta\}$, we generate 3000 trees. Note that large values of D and Δ yield extremely large networks, which are very rare in control applications due to the stringent requirements on communication delay, thus of little relevance to the scope of this paper. For example, the setting $\{M,\ D,\ \Delta\} = \{3,\ 10,\ 3\}$ generates trees with up to 88572 nodes (the balanced complete 3-ary tree, cf. Equation (16)).

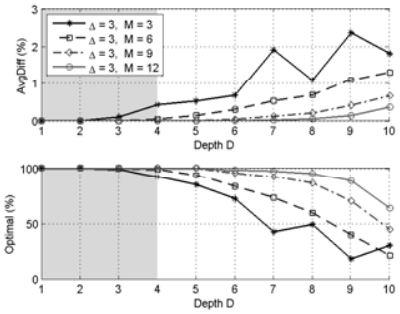
The topmost plots in Figures 14(a)-14(b) show the average deviation (in percent) of actual convergecast schedule length from the lower bound, i.e., AvgDiff $=(\mathcal{L}_S - \mathcal{L}_S^\star)/\mathcal{L}_S^\star$, where $\mathcal{L}_S$ and $\mathcal{L}_S^\star$ are the convergecast schedule length achieved by Algorithm 4 and the lower bound on convergecast schedule length, respectively. For both $\Delta = 2$ and $\Delta = 3$, AvgDiff increases slightly with the increase of depth D of the tree, but it always remains below 2.5%, demonstrating the near-optimality of schedules generated by Algorithm 4. The lower plots in Figures 14(a)-14(b) show the percentage of optimal schedules over the total number of runs. The results show that the schedules are close to optimal, especially for moderate values of D and large values of M. This phenomena is directly connected to the structure of the routing tree: we observe that both large values of D and small values of M may result in highly unbalanced tree topologies. For instance, with $M = 3$, one of the branches may dominate the size of the routing tree, making the direct descendants of the gateway bottlenecks for convergecast. Moreover, the more unbalanced the routing tree, the higher the probability that the lower bound on the number of channels might be not achievable (compare the example given in Figure 7). This condition is alleviated with the increase of the number of children of the GW. When $M = 12$, AvgDiff is less than 0.37% even for $D = 10$, and the percentage of optimal schedules exceeds 97% for $D \leq 7$. The reason for this is that when M increases, the GW has more opportunities to receive packets alternately from different branches, and the connectivity keeping step has more possibilities to avoid schedule holes.

Figure 14(c) plots the maximum deviation of convergecast schedule length from the lower bound (in time slots) for $\Delta = 2$ under different values of M and D. Figure 14(d) shows the frequencies that the discrepancies occur. Although the maximum difference goes up to 15 time slots when $D = 10$, Figure 14(a) shows that this corresponds to less than 2.2% of the lower bound. Furthermore, the large mismatches occur very rarely. For instance, the percentage of schedules demanding

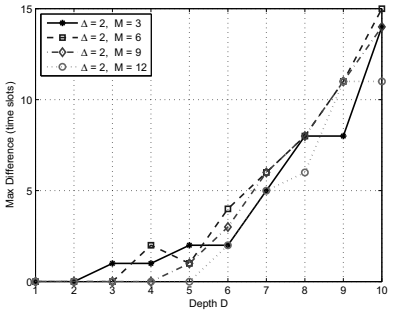
[5]For large number of runs, this approach may enumerate all possible tree topologies with the same setting $\{M,\ D,\ \Delta\}$.



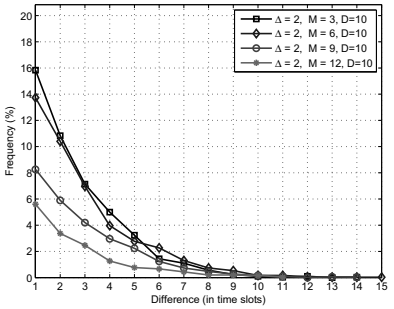
(a) Average deviation from lower bound ($\triangle = 2$)



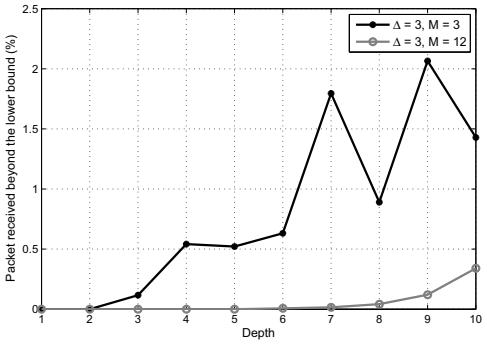
(b) Average deviation from lower bound ($\triangle = 3$)



(c) Max deviation from lower bound ($\triangle = 2$)



(d) Frequency of deviation ($\triangle = 2$)



(e) Percentage of packets received beyond the lower bound

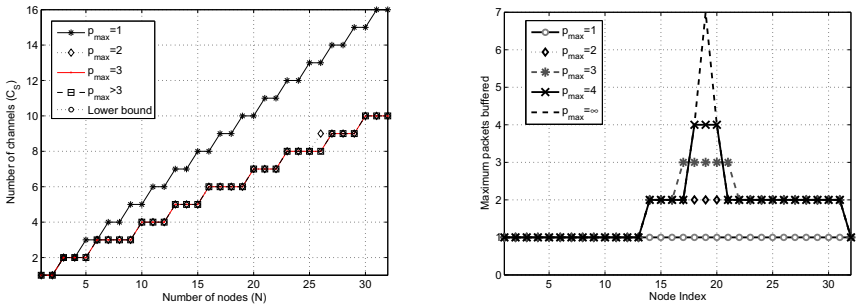
Fig. 14. Performance of Algorithm 4 under different $\{M,\ D,\ \Delta\}$.

more than 9 time slots than the lower bound is below 1.7% for all combinations of $\{M,\ \Delta,\ D\}$ studied, and a mismatch of 15 time slots happens in less than 0.04%, of the runs. Figure 14(e) plots the average percentage of packets received beyond the lower bound on time. This figure is directly connected to Figure 14(b): both large values of D and small values of M may result in highly unbalanced tree topologies,

leading to larger AvgDiff and more packets received beyond the time lower bound. However, when $M = 12$ and $\Delta = 3$, it is less than 0.35% even for $D = 10$, and the average percentage is below 2.1% for all combinations of $\{M,\ \Delta,\ D\}$ studied. All of these demonstrate the near-optimality of Algorithm 4.

9.2 Memory Efficiency

Buffer space is scarce in wireless sensor nodes, and memory-efficiency of scheduling policies is an important performance indicator. Algorithm 1, Algorithm 3 and Algorithm 4 are clearly optimal in terms of memory utilization since each field device is required to buffer at most one packet per time slot. However, the algorithms for jointly time- and channel-optimal convergecast do not impose any restriction on buffer space and could possibly be wasteful. In the following, we analyze the memory efficiency of these algorithms, and study a slight variation where we impose a constraint $p_t(v_i) \leq p_{\max}$ on the number of packets that each node is allowed to buffer. To account for limited buffer space, we make the following slight modification in our algorithms: at each slot, an eligible device can be scheduled for transmission only if the buffer queue at the receiver is not full, *i.e.* $p_t(v_i) < p_{\max}$. We take the schedules generated by Algorithm 2 as examples for illustration[6].



(a) $\mathcal{C}_{\mathcal{S}}$ *v.s.* N and θ (b) Maximum packets buffered

Fig. 15. Memory utilization of the schedules generated by Algorithm 4.

Figure 15(a) shows how the buffer space restriction impacts the minimum number of channels required to solve Problem (1) on a line topology with varying number of nodes, and it also illustrates how the buffer constraint allows us to limit the maximum buffer space to three packets without sacrificing channel (and time) optimality. For all cases except $N = 26$, our algorithm manages to find a time- and channel-optimal schedule which only requires buffer space for two packets. Figure 15(b) shows the buffer space requirement on different nodes in a line with 32 nodes under different buffering restriction. Without buffering constraint, we can see that the algorithm would require node 18 to have sufficient buffer space to hold

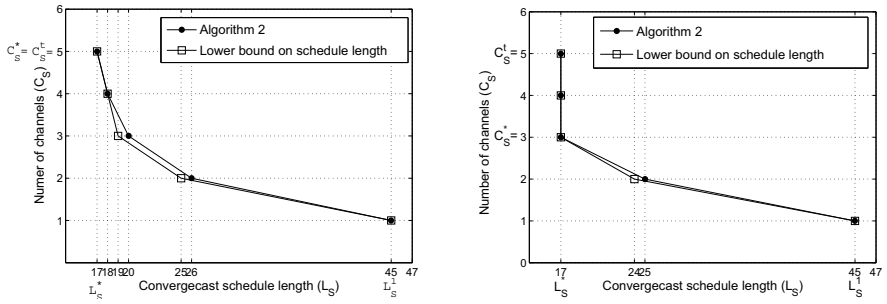
[6]For tree topologies, the buffer utilization may depend strongly on the network structure. Still, the analysis for the line topology provides insights into the general sensitivity of buffer space on channel efficiency.

7 packets. The reason for this is that the backward scheduling step in Algorithm 2 pushes packets generated at the tail of the line to the middle in the first several time slots, in an attempt to maximize the number of scheduled devices.

9.3 Performance of channel-constrained time-optimal convergecast scheduling

In Section 6, we introduced the channel-constrained time-optimal convergecast problem and claimed that Algorithm 2 and Algorithm 4 can be applied to generate good solutions in line and tree routing topologies, respectively. In what follows, we validate our claims and compare the convergecast time achieved by these algorithms against the time bounds established in Corollary 4.

Figure 16 exhibits the performance of Algorithm 2 for channel-constrained convergecast scheduling in a line network with 9 devices. According to Theorem 2, for single-packet buffering capability, Algorithm 2 yields a jointly time- and channel-optimal convergecast schedule, here confirmed by $C_{\mathcal{S}}^{t} = C_{\mathcal{S}}^{\star} = 5$ in Figure 16(a). As



(a) single-packet buffering. (b) multi-packet buffering.

Fig. 16. Performance of Algorithm 2 in a line network with 9 field devices.

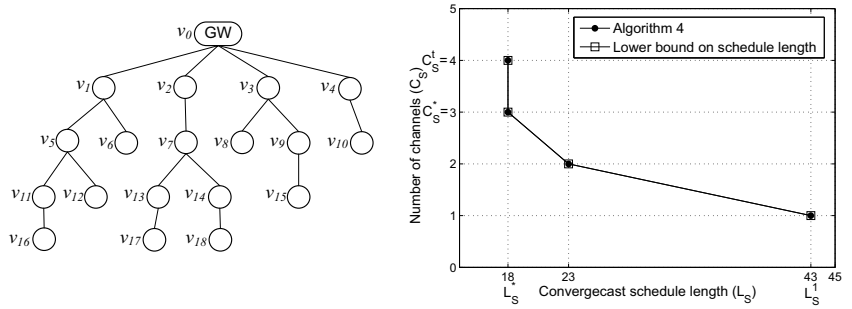
the number of channels decreases, the convergecast time increases. Decreasing the number of available channels from 5 to 1, Algorithm 2 achieves the lower bound established in Corollary 4 in three cases, whereas it requires only one extra time slot when $\mathcal{C}_{\mathcal{S}}$ equals 3 and 2, respectively. However, a manual inspection reveals that the schedules for $\mathcal{C}_{\mathcal{S}} = 3$ and 2 are also time-optimal since the lower bounds from Corollary 4 are not achievable in these two cases. Figure 17 illustrates the case for $\mathcal{C}_{\mathcal{S}} = 3$ with the time bound of $\mathcal{L}_{\mathcal{S}} = 19$ time slots. At most 45 transmissions can be scheduled in 19 time slots with 3 channels, and exactly 45 transmissions are required to complete convergecast in a line with 9 field devices. Thus the dark region in Figure 17 should be completely filled in. However, only 2 devices can be scheduled at time slot 15 since there are only two packets left in the network. This can not be avoided with any other scheduling policies. Thus, the lower bound in Corollary 4 is not achievable for $\mathcal{C}_{\mathcal{S}} = 3$ and one extra time slot is needed to complete convergecast. The same occurs for $\mathcal{C}_{\mathcal{S}} = 2$.

The same analysis can be performed for the multi-packet buffering case in Figure 16(b). By Theorem 3, the channel bound for time-optimal convergecast is $\mathcal{C}_{\mathcal{S}}^{\star} = 3$. In this case, only when $\mathcal{C}_{\mathcal{S}} = 2$ the schedule generated by Algorithm 2 uses

Time Slot:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Sch[t][1]:	v_1	v_2	v_1	v_2	v_1	v_2	v_1	v_2	v_1	v_2	v_1	v_3	v_2	v_1	v_3	v_2	v_1	v_3	v_2	v_1
Sch[t][2]:			v_3	v_4	v_3	v_4	v_3	v_4	v_3	v_5	v_4	v_6	v_5	v_4	v_6	v_5	v_4			
Sch[t][3]:					v_5	v_6	v_5	v_7	v_6	v_8	v_7	v_9	v_8	v_7						

Fig. 17. The schedule generated by Algorithm 2 when $\mathcal{C}_{\mathcal{S}} = 3$.

one more time slot than the lower bound in Corollary 4. Comparing Figures 16(a) and 16(b), we notice that multi-packet buffering capability allows shorter convergecast time in the channel-constrained problem. For instance, with $\mathcal{C}_{\mathcal{S}} = 3$, convergecast requires 17 time slots in multi-packet buffering case compared to 19 (feasible length) time slots in single-packet buffering case.



(a) (b)

Fig. 18. Performance of Algorithm 4 in a example-tree network.

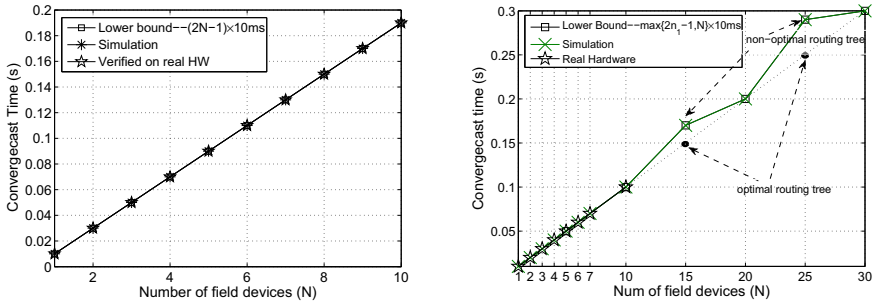
We next employ Algorithm 4 to solve the channel-constraint convergecast scheduling in tree topology using the sample network in Figure 18(a). As demonstrated in Figure 18(b), the schedule generated by Algorithm 4 under single-packet buffering constraint achieves the lower bound on convergecast time in all cases. The above simulation results confirm the efficiency of our scheduling policies.

9.4 Experimental results

Finally, we report results from a real implementation of our scheduling policies in the Contiki operating system using the Rime protocol stack [Dunkels et al. 2007] on the Tmote Sky platform. We validate our implementations in extensive simulations using the COOJA simulator and real-world experiments.

Our implementation relies on Contiki's built-in time synchronization service with hardware timers set to 16 kHz. The service is accurate enough to maintain synchronized time slots of 10 ms length, as required in our experiments. We use a simplified 802.15.4 MAC packet format where the MAC Protocol Data Unit (MPDU) only consists of application data payload (64 bytes in our experiments) and an additional 2-byte Frame Check Sequence (FCS) with CRC information. For each real experiment, we verified that the convergecast schedule works by observing 10 consecutive

convergecast rounds without lost radio packets (as stressed earlier, reliability mechanisms are outside the scope of this paper). The experimental results agree fully with simulation results under ideal channel conditions.



(a) Line topology (b) Three topology

Fig. 19. Convergecast time *vs.* different routing topologies.

Figure 19(a) exhibits the convergecast latency in networks with line routing topology by comparing the simulation/experiment results with the theoretical lower bound. In both simulations and experiments, our scheme achieves the theoretical lower bound on convergecast latency. It can be seen that collecting all packets in a line with 10 devices takes less than 0.2s. Figure 19(b) shows the simulation and experimental results for convergecast in networks with general tree routing topology. The theoretical bound is also met in both simulations and experiments. Figure 19(b) also shows how the structure of the routing topology affects the achievable convergecast time, cf. Theorem 4: when one sub-tree dominates, more than N time slots are needed to complete convergecast. Figure 20 gives the routing tree used in our simulations for $N = 15$, where the dominant subtree rooted at v_1 has $n_1 = 9$ field devices. Instead of 15 time slots, $2n_1 - 1 = 2 \times 9 - 1 = 17$ time slots are needed to complete convergecast with this routing tree. By replacing the link $(v_{10},\ v_5)$ with $(v_{10},\ v_3)$, the largest subtree would remain with $n_1 = 6$ devices and convergecast could be completed in 15 time slots.

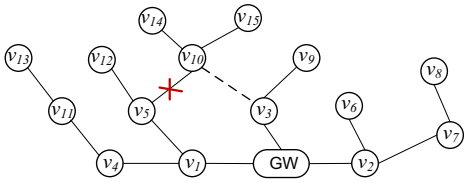


Fig. 20. The routing tree topology used when $N = 15$.

10. CONCLUSION

Convergecast is an important communication primitive for data collection in WirelessHART networks. This paper studied the problem of optimal convergecast scheduling, and we proposed jointly time- and channel-optimal scheduling policies for convergecast in WirelessHART networks with both line and balanced complete m-ary tree routing topologies. For general tree routing topology, we presented time-optimal scheduling policies which can complete convergecast in $\max\{2n_1 - 1, N\}$ time slots, established lower bounds on the number of channels for time-optimal convergecast, and proposed a heuristic algorithm to generate near-optimal convergecast schedule in order to minimize both the number of time slots and the number of channels. Moreover, we demonstrated that our algorithms can also be employed to solve the channel-constrained time-optimal convergecast problem when the number of available channels is less than the lower bound required for time-optimal convergecast. Simulations and experiments on real hardware show that our scheme can provide very fast and efficient convergecast in WirelessHART networks. We are currently studying optimal convergecast scheduling over unreliable links, in an attempt to understand the fundamental limits of delay and reliability, and develop efficient scheduling policies that make full use of time-, frequency- and spatial diversity in WirelessHART networks.

Acknowledgment

We wish to thank Fredrik Österlind at Swedish Institute of Computer Science (SICS) for helping implementing parts of the convergecast schemes for simulation in COOJA simulator and experiments on Tmote sky platform.

Appendix

Proof of Lemma 1.

$' \rightarrow'$: To complete convergecast, v_1 must be scheduled to transmit in N time slots. Due to the single-packet buffering constraint, v_1 can not transmit in two consecutive time slots. Thus, to complete convergecast in $2N - 1$ time slots, v_1 must transmit a packet to the GW at any odd time slot $t = 2k - 1$, with $k \in [1, N]$.

$' \leftarrow'$: If v_1 transmits one packet to the GW at every odd slot $t = 2k - 1$ where $k \in [1, N]$, obviously the total number of time slots for convergecast is $2N - 1$. □

Proof of Corollary 1.

Let $\mathcal{S}$ be the schedule returned by Algorithm 1. Given any schedule $\mathcal{S}^{'} \neq \mathcal{S}$, there must be a time slot t with a device v_i scheduled in $\mathcal{S}$ but not in $\mathcal{S}^{'}$:

—If $i = 1$, v_1 does not transmit in all odd time slots. Hence by Lemma 1, schedule $\mathcal{S}^{'}$ can not be time-optimal.

—If $i = 2$, device v_2 does not feed v_1 with new packets at all even slots. Hence, there will be an odd time slot in which v_1 does not have a packet to transmit. So schedule $\mathcal{S}^{'}$ can not be time-optimal either.

—If $i \geq 3$, device v_i does not feed v_{i-1} with a new packet to transmit at slot $t + 1$, and device v_{i-1} will not feed v_{i-2} with a new packet to transmit at slot $t + 2$, and so on. Suppose that the packet transmitted by device v_i is forwarded to the GW by v_1 at an odd time slot $t^{'}$ in schedule $\mathcal{S}$. Since the transmission for device

v_i is delayed in $\mathcal{S}^{'}$ and the continuity is broken, v_1 must do not have a packet to transmit at odd time slot $t^{'}$ in $\mathcal{S}^{'}$. Thus schedule $\mathcal{S}^{'}$ can not be time-optimal .

□

Proof of Theorem 2.
By Corollary 1, the schedule generated by Algorithm 1 is the only optimal schedule that can complete convergecast in $2N-1$ time slots with single-packet buffering constraint. The maximum number of parallel transmissions scheduled at a time slot in this optimal schedule is $\frac{N+1}{2}$ if N is odd and $\frac{N}{2}$ if N is even. Since all parallel transmissions must use different channels, the lower bound on the number of channels required to complete convergecast in $2N-1$ time slots is $\lceil \frac{N}{2} \rceil$. □

Proof of Theorem 3.
Let $\mathcal{C}_{\mathcal{S}}$ be the number of channels used in schedule $\mathcal{S}$, and $PT_{\max}(t)$ denote the maximum number of Parallel Transmissions that can be scheduled in time slot $t \in [1, \mathcal{L}_{\mathcal{S}}]$. As can be seen from Figure 4, to guarantee that convergecast can be completed in $\mathcal{L}_{\mathcal{S}}$ time slots, only one device can be scheduled when $t = \mathcal{L}_{\mathcal{S}}$ or $t = \mathcal{L}_{\mathcal{S}}-1$, and at most two devices can be scheduled when $t = \mathcal{L}_{\mathcal{S}}-2$ or $t = \mathcal{L}_{\mathcal{S}}-3$, and so on. Thus at most $\lceil \frac{\mathcal{L}_{\mathcal{S}}-t+1}{2} \rceil$ devices can be scheduled for transmission in time slot t. When $t \in [1, \mathcal{L}_{\mathcal{S}} - 2(\mathcal{C}_{\mathcal{S}}-1)]$, $\lceil \frac{\mathcal{L}_{\mathcal{S}}-t+1}{2} \rceil \geq \mathcal{C}_{\mathcal{S}}$. Since the maximum number of channels can be used is $\mathcal{C}_{\mathcal{S}}$, the maximum number of devices can be scheduled in time slot $t \in [1, \mathcal{L}_{\mathcal{S}} - 2(\mathcal{C}_{\mathcal{S}}-1)]$ is $\mathcal{C}_{\mathcal{S}}$. Hence,

$$PT_{\max}(t) = \begin{cases} \mathcal{C}_{\mathcal{S}}, & \text{if } t \in [1, \mathcal{L}_{\mathcal{S}} - 2(\mathcal{C}_{\mathcal{S}}-1)]; \\ \lceil \frac{\mathcal{L}_{\mathcal{S}}-t+1}{2} \rceil, & \text{if } t \in (\mathcal{L}_{\mathcal{S}} - 2(\mathcal{C}_{\mathcal{S}}-1), \mathcal{L}_{\mathcal{S}}]. \end{cases} \tag{13}$$

By Equation (13), the maximum number of transmissions that can be scheduled in $\mathcal{L}_{\mathcal{S}}$ time slots using $\mathcal{C}_{\mathcal{S}}$ channels is thus

$$\sum_{t=1}^{\mathcal{L}_{\mathcal{S}}} PT_{\max}(t) = \sum_{t=1}^{\mathcal{L}_{\mathcal{S}}-2(\mathcal{C}_{\mathcal{S}}-1)} \mathcal{C}_{\mathcal{S}} + \sum_{t=\mathcal{L}_{\mathcal{S}}-2(\mathcal{C}_{\mathcal{S}}-1)+1}^{\mathcal{L}_{\mathcal{S}}} \left\lceil \frac{\mathcal{L}_{\mathcal{S}}-t+1}{2} \right\rceil = -\mathcal{C}_{\mathcal{S}}^2 + \mathcal{C}_{\mathcal{S}}(\mathcal{L}_{\mathcal{S}}+1)$$

where we have used the fact that $\sum_{t=\mathcal{L}_{\mathcal{S}}-2(\mathcal{C}_{\mathcal{S}}-1)+1}^{\mathcal{L}_{\mathcal{S}}} \lceil \frac{\mathcal{L}_{\mathcal{S}}-t+1}{2} \rceil = (\mathcal{C}_{\mathcal{S}}-1) + (\mathcal{C}_{\mathcal{S}} - 1), ..., +2+2+1+1 = 2 \cdot \frac{\mathcal{C}_{\mathcal{S}}(\mathcal{C}_{\mathcal{S}}-1)}{2} = \mathcal{C}_{\mathcal{S}}(\mathcal{C}_{\mathcal{S}}-1)$. Moreover, at least $N(N+1)/2$ transmissions must be scheduled to complete convergecast, so

$$-\mathcal{C}_{\mathcal{S}}^2 + (\mathcal{L}_{\mathcal{S}}+1) \cdot \mathcal{C}_{\mathcal{S}} \geq \frac{N(N+1)}{2}. \tag{14}$$

Replacing $\mathcal{L}_{\mathcal{S}}$ with $2N-1$, $\mathcal{C}_{\mathcal{S}} \geq \lceil N - \sqrt{N(N-1)/2} \rceil$. □

Proof of Corollary 2.
By Equation (3), $PT_{\max}(t) \leq \mathcal{C}_{\mathcal{S}}^{\star} = \lceil N - \sqrt{N(N-1)/2} \rceil$ where $t \in [1, 2N-1]$. Since the number of devices scheduled by Algorithm 2 at any time slot t is no more than $PT_{\max}(t)$, the number of channels used does not exceed $\mathcal{C}_{\mathcal{S}}^{\star}$. Since $\mathcal{C}_{\mathcal{S}}^{\star}$ is the lower bound on the number of channels for time-optimal convergecast, the schedule generated by Algorithm 2 uses exactly $\mathcal{C}_{\mathcal{S}}^{\star}$ channels.

Algorithm 2 schedules device v_1 for transmission whenever it has a packet in its buffer. By similar arguments as in Lemma 1, if convergecast can not be completed in $2N-1$ time slots, there must be at least one odd slot in which device v_1 is not

scheduled. Let t' denote the first odd slot with no transmission from v_1 to the GW. Since v_1 is not scheduled at even slot $t'-1$ either, there must be two adjacent nodes with no packet to transmit before t'. We refer to this condition as to a *schedule hole*. To prove time-optimality of the schedule generated by Algorithm 2, it is only needed to show that there are no schedule holes.

We logically split the line network into two parts: device v_i is in Region A if $i \leq 2\mathcal{C}_{\mathcal{S}}^{\star}$; otherwise v_i belongs to Region B. In forward scheduling step, a device can be scheduled for transmission iff it has a packet and the buffer at its parent is empty. Thus, in the first several time slots the forwarding scheduling step activates only a few devices in Region A, and the remaining channels are used to move the packets from Region B to Region A (see Figure 21). Since at most $\mathcal{C}_{\mathcal{S}}^{\star}$ devices in Region A can be scheduled for transmission at any slot due to the half-duplex constraint, the forward scheduling can always guarantee that there are no two adjacent devices without packet to transmit in Region A if the packets in Region B can be fed into Region A timely. Since backward scheduling is performed after forward scheduling, a schedule hole can not be induced in backward scheduling step.

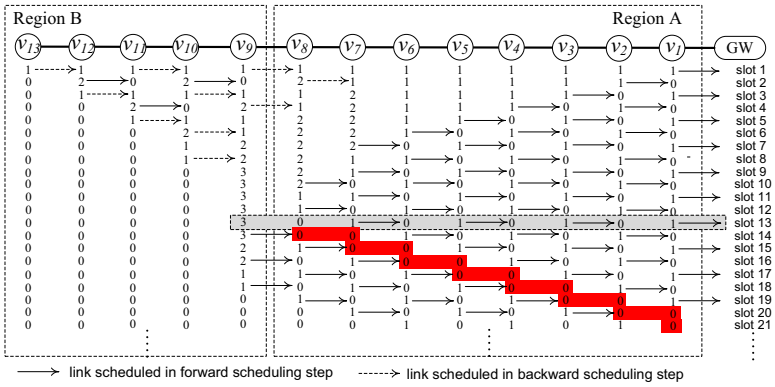


Fig. 21. Illustration of schedule hole.

Thus a schedule hole can be induced only at the boundary between two regions if the remaining packets in Region B can not be transmitted to Region A due to the lack of channels, as illustrated in the example in Figure 21 (To demonstrate the generation of a schedule hole, the number of devices in Region A is set to $2(\mathcal{C}_{\mathcal{S}}^{\star}-1)$ instead of $2\mathcal{C}_{\mathcal{S}}^{\star}$). Therefore, if a schedule hole is induced in Region A, the pattern marked with light gray, in which there are exactly $\mathcal{C}_{\mathcal{S}}^{\star}$ devices in Region A holding one packet each and placed at every second node starting from v_1, must occur. Once generated, the two adjacent zeroes, highlighted with red color in Figure 21, move forward one hop at each slot until arriving at the GW, and can not be removed once generated, resulting in no transmission for device v_1 at an odd slot. To complete the proof, we show by contradiction that Algorithm 2 never yields this pattern.

Suppose the pattern occurs in Algorithm 2 at time slot t. Let P_B be the number of packets in the devices located in Region B at the beginning of time slot t. Since the number packets that the GW receives before time slot t is $\frac{t-1}{2}$ (t must be odd

if this pattern occurs), $P_B = N - \frac{t-1}{2} - \mathcal{C}^\star_\mathcal{S}$. In the best case all the P_B packets are in the buffers of device $v_{2\mathcal{C}^\star_\mathcal{S}+1}$, and the minimum number of transmissions needed to deliver the P_B packets to the GW is $P_B(2\mathcal{C}^\star_\mathcal{S}+1)$. For the $\mathcal{C}^\star_\mathcal{S}$ packets in Region A, each device v_j where $j = 1, 3, 5, ..., 2\mathcal{C}^\star_\mathcal{S} - 1$ holds a packet, and j transmissions are needed to transmit the packet at device v_j to the GW. Thefore, the number of transmissions needed to deliver the $\mathcal{C}^\star_\mathcal{S}$ packets to the GW is $1+3+5+, ..., +2\mathcal{C}^\star_\mathcal{S}-1 = C^{*2}$. Since there is at least one packet still located in Region B, the number of transmissions scheduled at any time slot $t' < t$ must be $\mathcal{C}^\star_\mathcal{S}$. Hence, the minimum number of transmissions required to complete convergecast is

$$\begin{aligned} P_A(2\mathcal{C}^\star_\mathcal{S}+1) + C^{*2} + \mathcal{C}^\star_\mathcal{S}(t-1) &= (N - \tfrac{t-1}{2} - \mathcal{C}^\star_\mathcal{S})(2\mathcal{C}^\star_\mathcal{S}+1) + C^{*2} + \mathcal{C}^\star_\mathcal{S}(t-1); \\ &= -C^{*2} + 2N\mathcal{C}^\star_\mathcal{S} + (N - \tfrac{t-1}{2} - \mathcal{C}^\star_\mathcal{S}). \end{aligned}$$

As there is at least one packet in the devices located in Region B, $N - \frac{t-1}{2} - \mathcal{C}^\star_\mathcal{S} > 0$. According to the proof of Theorem 3, the maximum number of transmissions that can be scheduled in $2N-1$ time slots using $\mathcal{C}^\star_\mathcal{S}$ channels is $-C^{*2} + 2N\mathcal{C}^\star_\mathcal{S}$. If this pattern happens, the lower bound on the number of channels established in Theorem 3 is not correct. Therefore, this pattern never happens. □

Proof of Corollary 3.

According to policy T2, a device $v_i \notin C_{v_0}$ is scheduled for transmission at time slot t only if its parent f_{v_i} has been scheduled at time slot $t-1$. Thus each field device needs to buffer at most one packet at any time slot.

Suppose that device v_i is scheduled for transmission at time slot t. If v_i has not finished forwarding all the packets it should forward, the earliest time slot where it can be re-scheduled is $t+2$. An time $t+1$, a child of device v_i is scheduled to transmit, which guarantees that v_i has a packet in its buffer at the end of time slot $t+1$. Thus, for any device v_i scheduled at time t by Algorithm 3, v_i must have a packet to transmit at the beginning of time slot t. To prove that Algorithm 3 always yields a time-optimal convergecast schedule, we need to prove that the transmissions for the children of the GW can be scheduled in $\max\{2n_1 - 1, N\}$ time slots.

Case: $2n_1 - 1 > N$

The lower bound for convergecast scheduling is $2n_1 - 1$ time slots. After time slot $t = 2k$ $(1 \le k \le N-1)$, the minimum number of packets left in the largest subtree is $n_1 - t/2$, and the maximum number of packets left in one of the other subtrees is $\sum_{i=2}^{m} n_i - t/2$. Since $2n_1 - 1 > \sum_{i=1}^{m} n_i$ (i.e., $n_1 > \sum_{i=2}^{m} n_i + 1$), $n_1 - t/2 > \sum_{i=2}^{m} n_i - t/2$, thus there must be a packet scheduled to be transmitted to the GW from the largest subtree at time $t = 2k+1$. Hence, convergecast in the largest subtree can be done in $2n_1 - 1$ time slots. Since $n_1 - 1 > \sum_{i=2}^{m} n_i$, the transmissions from other subtrees to the GW can be scheduled in the remaining $n_1 - 1$ time slots. Therefore, the schedule generated by Algorithm 3 completes convegecast in $2n_1 - 1$ time slots.

Case: $2n_1 - 1 \le N$

The lower bound on the number of time slots for convergecast is N. Hence, there must be a transmission from one child of the GW to the GW at any time slot t $(1 \le t \le N)$. Note that the first scheduling policy can always guarantee this. □

Proof of Theorem 5.

At the first time slot, only device v_1 is scheduled for transmission. At time slot 2,

device v_2 and a child of v_1 are scheduled. The two devices are located at different levels. Assume that the devices scheduled in time slot t are located in different levels. Based on policy T2, a device can be scheduled for transmission at time slot $t+1$ only if its parent has been scheduled at time slot t. Thus the devices scheduled in time slot $t+1$ must be located in different levels. By induction, the devices scheduled at any time slot are located in different levels. Thus the maximum number of devices scheduled in a time slot is D for any tree network with depth D. Since the devices scheduled at the same time slot must use different channels, the schedule generated by Algorithm 3 uses at most D channels. □

Proof of Theorem 6.
To deliver a packet from any device v_i at depth d to the GW, d transmissions must be scheduled. In a balanced complete m-ary tree, there are m^d nodes at depth d. Hence the total number of transmissions required to complete convergecast is

$$\sum_{d=1}^{D} d \cdot m^d = \frac{m - m^{D+1}[1+D(1-m)]}{(1-m)^2}, \tag{15}$$

and the number of nodes in the tree is

$$N = \sum_{d=1}^{D} m^d = \frac{m^{D+1}-1}{m-1} - 1 = m\frac{m^D-1}{m-1}. \tag{16}$$

By Equations (15) and (16), the average number of transmissions scheduled per time slot is $\frac{1}{N}\sum_{d=1}^{D} d \cdot m^d$. Thus, at least $\lceil \frac{1}{N}\sum_{d=1}^{D} d \cdot m^d \rceil$ channels are needed to fulfill convergecast in N time slots. To complete the proof, we show that $D-1 < \frac{1}{N}\sum_{d=1}^{D} d \cdot m^d \leq D$. For any $m > 1$ and $D > 1$, $\frac{1}{m-1} \leq 1$ and $\frac{m^D}{m^D-1} > 1$. Hence

$$\frac{1}{N}\sum_{d=1}^{D} d \cdot m^d = D\frac{m^D}{m^D-1} - \frac{1}{m-1} \geq D\frac{m^D}{m^D-1} - 1 > D-1.$$

Consider the difference between $\frac{\sum_{d=1}^{D} d \cdot m^d}{N}$ and D, $\frac{1}{N}\sum_{d=1}^{D} d \cdot m^d - D = \frac{(mD-m^D)+(1-D)}{(m^D-1)(m-1)}$. For any $m > 1$ and $D > 1$, $mD - m^D \leq 0$, $1 - D \leq 0$ and $(m^D-1)(m-1) > 0$. Thus $\frac{\sum_{d=1}^{D} d \cdot m^d}{N} - D \leq 0$, i.e., $\frac{\sum_{d=1}^{D} d \cdot m^d}{N} \leq D$. □

Proof of Theorem 7.
As shown in Figure 6, due to single-packet buffering constraint, the maximum number of transmissions scheduled at time slot t, denoted by $PT_{max}(t)$, is t. To complete convergecast in $\mathcal{L}_{\mathcal{S}}^{\star}$ time slots, at most one device can be scheduled in time slot $\mathcal{L}_{\mathcal{S}}^{\star}$, and at most two devices can be scheduled at time slot $\mathcal{L}_{\mathcal{S}}^{\star}-1$ and so on. Thus $PT_{\max}(t)$ should be no larger than $\mathcal{L}_{\mathcal{S}}^{\star}-t+1$ which is equal to $\mathcal{C}_{\mathcal{S}}$ when $t = \mathcal{L}_{\mathcal{S}}^{\star} - \mathcal{C}_{\mathcal{S}} + 1$. When $t \in [\mathcal{C}_{\mathcal{S}}+1, \mathcal{L}_{\mathcal{S}}^{\star} - \mathcal{C}_{\mathcal{S}}]$, the number of devices eligible to be scheduled might be larger than $\mathcal{C}_{\mathcal{S}}$. However, the maximum number of channels can be used is $\mathcal{C}_{\mathcal{S}}$. Thus $PT_{\max}(t) = \mathcal{C}_{\mathcal{S}}$ when $t \in [\mathcal{C}_{\mathcal{S}}+1, \mathcal{L}_{\mathcal{S}}^{\star} - \mathcal{C}_{\mathcal{S}}]$. Hence

$$PT_{\max}(t) = \begin{cases} t, & \text{if } t \in [1, \mathcal{C}_{\mathcal{S}}]; \\ \mathcal{C}_{\mathcal{S}}, & \text{if } t \in [\mathcal{C}_{\mathcal{S}}+1, \mathcal{L}_{\mathcal{S}}^{\star} - \mathcal{C}_{\mathcal{S}}]; \\ \mathcal{L}_{\mathcal{S}}^{\star} - t + 1, & \text{if } t \in [\mathcal{L}_{\mathcal{S}}^{\star} - \mathcal{C}_{\mathcal{S}} + 1, \mathcal{L}_{\mathcal{S}}^{\star}]. \end{cases} \tag{17}$$

By Equation (17), the maximum number of transmissions that can be scheduled in $\mathcal{L}_{\mathcal{S}}^{\star}$ time slots using $\mathcal{C}_{\mathcal{S}}$ channels is thus

$$\sum_{t=1}^{\mathcal{L}_{\mathcal{S}}^{\star}} PT_{\max}(t) = \sum_{t=1}^{\mathcal{C}_{\mathcal{S}}} t + \sum_{t=\mathcal{C}_{\mathcal{S}}+1}^{\mathcal{L}_{\mathcal{S}}^{\star}-\mathcal{C}_{\mathcal{S}}} \mathcal{C}_{\mathcal{S}} + \sum_{t=\mathcal{L}_{\mathcal{S}}^{\star}-\mathcal{C}_{\mathcal{S}}+1}^{\mathcal{L}_{\mathcal{S}}^{\star}} (\mathcal{L}_{\mathcal{S}}^{\star} - t + 1) = -\mathcal{C}_{\mathcal{S}}^2 + (\mathcal{L}_{\mathcal{S}}^{\star} + 1)\mathcal{C}_{\mathcal{S}},$$

where $\sum_{t=1}^{\mathcal{C}_{\mathcal{S}}} t = \sum_{t=\mathcal{L}_{\mathcal{S}}^{\star}-\mathcal{C}_{\mathcal{S}}+1}^{\mathcal{L}_{\mathcal{S}}^{\star}} (\mathcal{L}_{\mathcal{S}}^{\star} - t + 1) = \frac{(\mathcal{C}_{\mathcal{S}}+1)\mathcal{C}_{\mathcal{S}}}{2}$. Moreover, at least $\sum_{d=1}^{D} d \cdot n(d)$ transmissions are required to complete convergecast, thus

$$-\mathcal{C}_{\mathcal{S}}^2 + (\mathcal{L}_{\mathcal{S}}^{\star} + 1)\mathcal{C}_{\mathcal{S}} \geq \sum_{d=1}^{D} d \cdot n(d). \tag{18}$$

Rearranging terms, we find that $\mathcal{C}_{\mathcal{S}} \geq \left\lceil \frac{(\mathcal{L}_{\mathcal{S}}^{\star}+1) - \sqrt{(\mathcal{L}_{\mathcal{S}}^{\star}+1)^2 - 4\sum_{d=1}^{D} d \cdot n(d)}}{2} \right\rceil$. □

Proof of Theorem 8.

Different from the case of single-packet buffering, up to $\mathcal{C}_{\mathcal{S}}$ devices can be scheduled at time slot $t \in [1, \mathcal{C}_{\mathcal{S}}]$ because of the multi-packet buffering capability. Thus

$$PT_{\max}(t) = \begin{cases} \mathcal{C}_{\mathcal{S}}, & \text{if } t \in [1, \mathcal{L}_{\mathcal{S}} - \mathcal{C}_{\mathcal{S}}]; \\ \mathcal{L}_{\mathcal{S}} - t + 1, & \text{if } t \in (\mathcal{L}_{\mathcal{S}} - \mathcal{C}_{\mathcal{S}}, \mathcal{L}_{\mathcal{S}}]. \end{cases} \tag{19}$$

The maximum number of transmissions that can be scheduled within $\mathcal{L}_{\mathcal{S}}$ time slots with $\mathcal{C}_{\mathcal{S}}$ channels is

$$\sum_{t=1}^{\mathcal{L}_{\mathcal{S}}} PT_{\max}(t) = (\mathcal{L}_{\mathcal{S}} - \mathcal{C}_{\mathcal{S}})\mathcal{C}_{\mathcal{S}} + \sum_{t=\mathcal{L}_{\mathcal{S}}-\mathcal{C}_{\mathcal{S}}+1}^{\mathcal{L}_{\mathcal{S}}} (\mathcal{L}_{\mathcal{S}} - t + 1) = \frac{-\mathcal{C}_{\mathcal{S}}^2}{2} + (\mathcal{L}_{\mathcal{S}} + \tfrac{1}{2})\mathcal{C}_{\mathcal{S}}.$$

Similar to the proof of Theorem 7,

$$\frac{-\mathcal{C}_{\mathcal{S}}^2}{2} + (\mathcal{L}_{\mathcal{S}} + \tfrac{1}{2})\mathcal{C}_{\mathcal{S}} \geq \sum_{d=1}^{D} d \cdot n(d). \tag{20}$$

By arranging terms, we get $\mathcal{C}_{\mathcal{S}} \geq \left\lceil \frac{(2\mathcal{L}_{\mathcal{S}}+1) - \sqrt{(2\mathcal{L}_{\mathcal{S}}+1)^2 - 4\sum_{d=1}^{D} d \cdot n(d)}}{2} \right\rceil$. □

Proof of Corollary 4.

Case (1): The proof is the same as the proof for Theorem 3 except the definition of $PT_{\max}(t)$. As can be seen from Figure 3, due to the single-packet buffering constraint, at most $\lceil \frac{t}{2} \rceil$ can be scheduled at time slot t. To guarantee that convergecast can be completed in $\mathcal{L}_{\mathcal{S}}$ time slots, at most $\lceil \frac{\mathcal{L}_{\mathcal{S}}-t+1}{2} \rceil$ devices can be scheduled for transmission in time slot t. Since the maximum number of channels can be used is $\mathcal{C}_{\mathcal{S}}$,

$$PT_{\max}(t) = \begin{cases} \lceil \frac{t}{2} \rceil, & \text{if } t \in [1, 2(\mathcal{C}_{\mathcal{S}} - 1)]; \\ \mathcal{C}_{\mathcal{S}}, & \text{if } t \in (2(\mathcal{C}_{\mathcal{S}} - 1), \mathcal{L}_{\mathcal{S}} - 2(\mathcal{C}_{\mathcal{S}} - 1)]; \\ \lceil \frac{\mathcal{L}_{\mathcal{S}}-t+1}{2} \rceil, & \text{if } t \in (\mathcal{L}_{\mathcal{S}} - 2(\mathcal{C}_{\mathcal{S}} - 1), \mathcal{L}_{\mathcal{S}}]. \end{cases} \tag{21}$$

The maximum number of transmissions that can be scheduled in $\mathcal{L}_{\mathcal{S}}$ time slots using $\mathcal{C}_{\mathcal{S}}$ channels is

$$\begin{aligned} \sum_{t=1}^{2(\mathcal{C}_{\mathcal{S}}-1)} PT_{\max}(t) &= \sum_{t=1}^{\mathcal{L}_{\mathcal{S}}-2(\mathcal{C}_{\mathcal{S}}-1)} \lceil \frac{t}{2} \rceil + [\mathcal{L}_{\mathcal{S}} - 4(\mathcal{C}_{\mathcal{S}} - 1)]\mathcal{C}_{\mathcal{S}} + \sum_{t=\mathcal{L}_{\mathcal{S}}-2(\mathcal{C}_{\mathcal{S}}-1)+1}^{\mathcal{L}_{\mathcal{S}}} \left\lceil \frac{\mathcal{L}_{\mathcal{S}} - t + 1}{2} \right\rceil \\ &= -2\mathcal{C}_{\mathcal{S}}^2 + (\mathcal{L}_{\mathcal{S}} + 2) \cdot \mathcal{C}_{\mathcal{S}}. \end{aligned}$$

Similar to the proof of Theorem 3,

$$-2\mathcal{C}_{\mathcal{S}}^2 + (\mathcal{L}_{\mathcal{S}} + 2) \cdot \mathcal{C}_{\mathcal{S}} \geq \frac{N(N+1)}{2}. \tag{22}$$

Thus $\mathcal{L}_{\mathcal{S}} \geq \left\lceil \frac{N(N+1)}{2\mathcal{C}_{\mathcal{S}}} + 2\mathcal{C}_{\mathcal{S}} - 2 \right\rceil$.

Cases (2),(3) and (4): The lower bounds given in cases *b*), *c*) and *d*) can be easily obtained based on Equation (14), Equation (18) and Equation (20). □

Proof of Corollary 5.

For any node v_i, it needs at least $\sum_{v_j \in T_s(v_i)/v_i} g(v_j)$ time slots to receive the packets generated by other nodes in subtree $T_s(v_i)$, and needs at least $\sum_{v_j \in T_s(v_i)} g(v_j)$ time slots to transmit the received packets together with the packets generated by itself to its parent. Thus the earliest time slot in which node v_i can finish transmitting all packets is $t = \sum_{v_j \in T_s(v_i)/v_i} g(v_j) + \sum_{v_j \in T_s(v_i)} g(v_j) = 2\sum_{v_j \in T_s(v_i)} g(v_j) - g(v_i)$. At the end of time slot t, there must be at least one packet at the parent of v_i, which needs another $h(v_i) - 1$ time slots to reach the GW. Therefore, the minimum number of time slots required to deliver all packets in subtree $T_s(v_i)$ is $2\sum_{v_j \in T_s(v_i)} g(v_j) - g(v_i) + h(v_i) - 1$. Since $\mathcal{L}_{\mathcal{S}}^{\star}$ should be no smaller than the number of packets in the network, $\mathcal{L}_{\mathcal{S}}^{\star} \geq \max\left\{ \max_{v_i \in V} \left(2\sum_{v_j \in T_s(v_i)} g(v_j) - g(v_i) + h(v_i) - 1\right), \sum_{v_i \in V} g(v_i) \right\}$.

The packets in any subtree $T_s(v_i)$ can be delivered to node v_i in $\sum_{v_j \in T_s(v_i)} (h(v_j) - 1) \cdot g(v_j)$ time slots by scheduling only one node at any time slot. Thus the packets in the network can be delivered to the children of the GW in $\max_{i \in [1,m]} \left(\sum_{v_j \in T_s(v_i)} (h(v_j) - 1) \cdot g(v_j) \right)$. Hence $\mathcal{L}_{\mathcal{S}}^{\star} \leq \max_{i \in [1,m]} \left(\sum_{v_j \in T_s(v_i)} (h(v_j) - 1) \cdot g(v_j) \right) + \sum_{v_i \in V} g(v_i)$. □

REFERENCES

Chen, L., Low, S. H., and Doyle, J. C. 2005. Joint congestion and media access control design for ad hoc wireless networks. In *Proceeding of the 24th Conference on Computer Communications (INFOCOM)*. 2212–2222.

Choi, H., Wang, J., and Hughes, E. A. 2005. Scheduling on sensor hybrid network. In *Proceedings of the 14th International Conference onComputer Communications and Networks (ICCCN)*. 503– 508.

Cogill, R. and Hindi, H. 2007. Optimal routing and scheduling in flexible manufacturing systems using integer programming. In *Proceeding of IEEE Conference on Decision and Control(CDC)*. 4095–4102.

Dunkels, A. 2009. *The Contiki Operating System*. http://www.sics.se/contiki/.

Dunkels, A., Österlind, F., and He, Z. 2007. An adaptive communication architecture for wireless sensor networks. In *Proceedings of the Fifth ACM Conference on Networked Embedded Sensor Systems(SenSys)*. 335–349.

ENGINEERLIVE. 2009. *Gas pipeline monitoring: moving from analogue to wireless*. http://www.engineerlive.com/Oil-and-Gas-Engineer/Communications/Gas_pipeline_monitoring:_moving_from_analogue_to_wireless/21951/.

Gandham, S., Zhang, Y., and Huang, Q. 2008. Distributed time-optimal scheduling for convergecast in wireless sensor networks. *Computer Networks 52*, 610–629.

Gasieniec, L. and Potapov, I. 2002. Gossiping with unit messages in known radio networks. In *Proceedings of the IFIP 17th World Computer Congress*. 193–205.

Gutierrez, J. 2008. *WirelessHART: The Industrial Wireless Standard*. http://www.petro-online.com/articles/safety/15/jose-gutierrez/wirelesshart-the-industrialwireless-standard/359/.

Hajek, B. and Sasaki, G. 1988. Link scheduling in polynomial time. *IEEE Transactions on Information Theory 34*, 910–917.

HARTCOMM. 2007. *WirelessHART specifications*. http://www.hartcomm2.org.

HARTCOMM. 2009a. *Control with WirelessHART*. http://www.hartcomm.org/protocol/training/training_resources_wihart.html.

HARTCOMM. 2009b. *Unmanned Offshore Gas Production with HART Networks*. http://www.hartcomm.org/protocol/realworld/realworld_companies_offshore.html.

Huang, S. C.-H., Du, H., and Park, E.-K. 2008. Minimum-latency gossiping in multi-hop wireless networks. In *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing(MobiHoc)*. 323–330.

Huang, S. C.-H., Wan, P.-J., Du, H., and Park, E.-K. 2010. Minimum latency gossiping in radio networks. *IEEE Transactions on Parallel and Distributed Systems 21*, 790–800.

Jothi, R. and Raghavachari, B. 2005. Approximation algorithms for the capacitated minimum spanning tree problem and its variants in network design. *ACM Transactions on Algorithms 1*, 265 – 282.

Kim, A. N., Hekland, F., Petersen, S., and Doyle, P. 2008. When HART goes wireless: understanding and implementing the WirelessHART standard. In *IEEE international conference on emerging technologies in factory automation*. Hamburg, Germany.

Manne, F. and Xin, Q. 2006. Optimal gossiping with unit size messages in known topology radio networks. In *Procceding of the Third Workshop on Combinatorial and Algorithmic Aspects of Networking(CAAN)*. 125–134.

MOTEIV. 2004. *Tmotesky Datasheet*. http://www.sentilla.com/pdf/eol/tmote-sky-datasheet.pdf.

Österlind, F., Dunkels, A., Eriksson, J., Finne, N., and Voigt, T. 2006. Cross-level sensor network simulation with COOJA. In *In Proceedings of the First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*. 641–648.

Özlem Durmaz Incel, Ghosh, A., Krishnamachari, B., and Chintalapudi, K. K. 2008. Multi-channel scheduling for fast convergecast in wireless sensor networks. Tech. rep., Department of Computer Science, University of Twente.

Özlem Durmaz Incel and Krishnamachari, B. 2008. Enhancing the data collection rate of tree-based aggregation in wireless sensor networks. In *Proceedings of the 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*. 569–577.

Papadimitriou, C. H. 1978. The complexity of the capacitated tree problem. *Networks 8*, 217 – 230.

Pesonen, J., Zhang, H., Soldati, P., and Johansson, M. 2009. Methodology and tools for controller-networking codesign in WirelessHART. In *IEEE International Conference on Emerging Techonologies and Factory Automation (ETFA)*.

Pister, K. S. J. and Doherty, L. 2008. TSMP: Time Synchronized Mesh Protocol. In *Proceedings of the IASTED Internatinal Symposium on Distributed Sensor Networks(DSN)*. 391–398.

Ramanathan, S. and Lloyd, E. L. 1992. On the complexity of link scheduling in multi-hop radio networks. In *Proc of the 26th Conference on Information Science and Systems*.

Sekhar, M. M. and Sivarajan, K. N. 2000. Routing and scheduling in packet radio networks. In *IEEE International Conference on Personal Wireless Communications(PWC)*. 335–339.

Soldati, P., Zhang, H., and Johansson, M. 2009. Deadline-constrained transmission scheduling and data evacuation in WirelessHART networks. In *Proceedings of The European Control Conference (ECC)*.

Song, W.-Z., Yuan, F., LaHusen, R., and Shirazi, B. 2007. Time-optimum packet scheduling for many-to-one routing in wireless sensor networks. *International Journal of Parallel, Emergent and Distributed Systems*, 355–370.

Tseng, Y.-C. and Pan, M.-S. 2006. Quick convergecast in zigbee/ieee 802.15.4 tree-based wireless sensor networks. In *Proceedings of the 4th ACM international workshop on Mobility management and wireless access(MOBIWAC)*. 60 – 66.

Upadhyayula, S., Annamalai, V., and Gupta, S. K. S. 2003. A low-latency and energy-efficient algorithm for convergecast in wireless sensor networks. In *Proceedings of IEEE Global Telecommunications Conference(GLOBECOM)*. 3525– 3530.

Wan, P.-J., Huang, S. C.-H., Wang, L., Wan, Z., and Jia, X. 2009. Minimum-latency aggregation scheduling in multihop wireless networks. In *MobiHoc '09: Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*. 185–194.

Wan, P.-J., Xu, X., Wang, L., Jia, X., and Park, E. 2009. Minimum-latency beaconing schedule in multihop wireless networks. In *Proceeding of the 28th Conference on Computer Communications (INFOCOM)*. 2340–2346.

Wu, Y., Stankovic, J., He, T., and Lin, S. 2008. Realistic and efficient multi-channel communications in wireless sensor networks. In *Proceeding of the 27th Conference on Computer Communications (INFOCOM)*. 1193–1201.

Zhang, H., Soldati, P., and Johansson, M. 2009. Optimal link scheduling and channel assignment for convergecast in linear WirelessHART networks. In *Proceeding of the 7th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*.