Rapid Convergecast on Commodity Hardware: Performance Limits and Optimal Policies

Haibo Zhang*, Fredrik Österlind†, Pablo Soldati*, Thiemo Voigt† and Mikael Johansson*
*School of Electrical Engineering, KTH, SE-10044 Stockholm, Sweden
Email: {haibo.zhang | pablo.soldati | mikael.johansson}@ee.kth.se
†Swedish Institute of Computer Science, Box 1263, SE-164 29 Kista, Sweden
Email: {fros|thiemo}@sics.se

***Abstract*—The increased industrial interest in wireless sensor networks demands a shift from optimizing protocols for reporting sporadic events, to solutions for high-rate data collection and dissemination. We study time-optimal convergecast under the communication constraints of commodity sensor network platforms. We propose a novel convergecast model in which packet copying between the microcontroller and the radio transceiver is separated from packet transmission, thereby improving channel utilization and system throughput. Based on this model, we establish the tight lower bound on the number of time slots for convergecast in networks with tree routing topology, and present both centralized and distributed algorithms for computing time-optimal convergecast schedules. Our scheme is memory-efficient as each node buffers at most one packet at any time. We evaluate our scheme in simulation and on real hardware, and show that our scheme can achieve a throughput of 203 kbit/s (86.4% of the theoretical upper bound): up to 86.24 % improvement compared with traditional TDMA-based convergecast. With an optimal routing tree and the maximum MAC layer payload, convergecast in a network with 20 sensor nodes can be completed in only** 100**ms.**

I. Introduction

The emergence of low-cost low-power radios, along with the recent surge in research on wireless communication and networked control, has enabled the deployment of wireless sensor networks (WSNs) for industrial process monitoring and control [1]. A typical control installation can have several hundreds of sensors and actuators, with control loops demanding sampling rates ranging from sub-second to minutes. During a sample interval, several operations must be completed: sensor measurements must be taken and communicated to the controller, controller code should be executed to compute the desired control commands, and the commands need to be disseminated to the actuators. Thus, for a control loop running on one second sampling interval, the time left for each communication (sensor to controller or controller to actuator) is less than 500 milliseconds [2]. This stringent latency requirement, coupled with the special characteristics of WSNs such as limited energy, bandwidth, computing and storage capability, poses great challenges to the design of real-time communication protocols for wireless control applications.

To achieve the strict latency requirement for control applications, contention-free medium access protocols such as time-division multiple access (TDMA) have many advantages over contention-based protocols, e.g., the ability to deliver packets with deterministic delay bounds by eliminating collisions and retransmissions. Moreover, TDMA-based protocols are energy-efficient since the amount of idle listening and overhearing can be reduced by turning the radio off in non-scheduled time slots. *Wireless*HART [3], the recently approved standard for control applications, uses a TDMA data link layer to control access to the network.

Distributing the communication across multiple channels for parallel transmissions is another attractive approach to reduce communication delay. State-of-the-art low-power radios support a number of orthogonal channels, e.g., the IEEE 802.15.4 Chipcon CC2420 packet-based radio supports 16 non-overlapping channels in the 2.4 GHz ISM band [4]. Moreover, many popular sensor platforms, including MicaZ [5], Tmote Sky [6], and Telos [7], have recently gravitated towards a single data-link protocol 802.15.4, and even a single radio chip ChipCon CC2420. However, state-of-the-art bulk data transport protocols report a multi-hop data throughput of approximately 10 kbit/s over a 250 kbit/s 802.15.4 radio [8]. In [9] we identified that it is the packet copying operation between the radio transceiver and the microcontroller that limits the achievable throughput (similar measurement results were also reported in [10]). By moving packet copying off the critical path of packet forwarding, our conditional immediate transmission primitive gave close to a 10-fold increase in network throughput. This result indicates that the impact of packet copying must be taken into account in the design of real-time communication schemes for control applications.

Periodic data collection from sensors to the controller is one of the basic operations in control applications. The networking primitive for collecting data from multiple sources to a single sink is called *convergecast*. We have previously shown that control command dissemination can be performed by simply running convergecast ”in reverse” [11]. Thus efficient policies for convergecast scheduling are instrumental for industrial control applications. Despite the vast literature on TDMA scheduling in WSNs [12]–[16], most existing convergecast schemes are based on single-channel communication, which are not able to provide timely communication at high data rate due to channel interference and limited bandwidth. Studies that consider multi-channel TDMA convergecast focus on frequency-reuse to decrease the convergecast delay, but do not handle packet copying in a separate time slot [17] [18]. Chintalapudi and Venkatraman proposed a transmission pipelining scheme for single-hop networks that takes into account the hardware overhead in the packet transmission process [10].

In this paper, we revisit the time-optimal convergecast problem, aiming at answering the following question: *How fast can data be collected from sensors to the sink on commodity hardware?* In contrast to previous work, we investigate this problem by exploring the advantages of TDMA-based multi-channel communication, separated packet copying and time-slotted channel hopping. We propose a novel *Wireless*HART-compliant convergecast model in which channel hopping is performed on a per-timeslot basis without frequency-reuse and packet copying is separated from transmission and reception. Based on this model, we first prove that, for a tree network with N sensor nodes and one sink, the lower bound on the convergecast time is $\max\{3n_1 - \Delta, N\}$ time slots, where n_1 is the maximum number of nodes in any subtree and $\Delta \in \{1, 2\}$. Then we present both centralized and distributed algorithms for computing time-optimal convergecast schedules. Finally, we evaluate our scheme through extensive simulations and validate the simulation results on Tmote Sky platform. Both simulations and experiments on real hardware show that our scheme can achieve a throughput of 203 kbit/s, 86.4% of the theoretical upper bound. With maximum MAC layer payload of 125 bytes and optimal routing tree with depth no larger than the number of available channels, convergecast in a network with 20 sensor nodes can be completed in only 100 ms.

The paper is organized as follows. We motivate our work in Section II. Section III gives the system model and problem statement. Section IV and Section V present the centralized and distributed scheduling algorithms, respectively. The solution for sub-schedule extraction and channel hopping is given in Section VI. Section VIII summarizes the related work and discusses possible extensions to deal with unreliable links. Section IX concludes the paper.

II. Motivation

A. The Practical 802.15.4 Bottleneck: Packet Copying

The *microcontroller* and *radio transceiver* are two main components of a wireless sensor node. Communication between the microcontroller and the radio is done via a Serial Peripherals Interface (SPI) bus, see the timing diagram in Fig. 1 (a). After receiving a packet, the microcontroller triggers an interrupt to notify a process to fetch the incoming data from the radio's receive buffer over the SPI bus. Before transmitting a packet, the microcontroller first copies the packet into the radio's transmit buffer over the SPI bus, and then sends a command to start transmission[1].

Based on the Tmote Sky platform with Contiki operating system [19] running at 3.9 MHz, we measure that the time for one-way packet copying (i.e., microcontroller to radio transmit buffer or radio receive buffer to microcontroller) with

[1]The CC2420 packet radio has two separate on-chip buffers: a receive buffer and a transmit buffer. Received packets cannot be copied to the transmit buffer directly without going through the microcontroller.

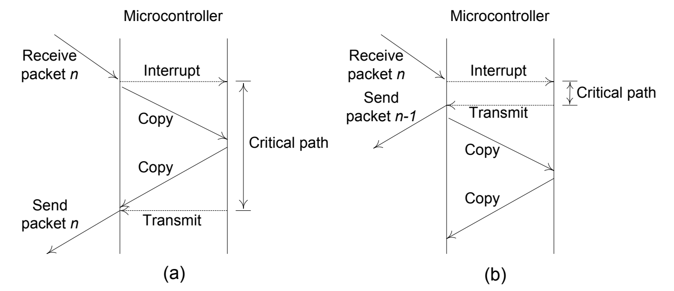


Fig. 1. Packet copying is on the critical path in (a). Packet copying is removed from the critical path in (b). [9]

125 bytes MAC layer data payload is around 1.45ms and the time for transmitting/receiving 125 bytes is around 4.55ms, which indicates that the two-way packet copying takes nearly the same time as packet transmitting/receiving. As illustrated by Fig. 1 (b), by arranging the order of packet copying and transmission, packet copying can be removed from the critical path of packet forwarding, i.e., packet $n-1$ is pre-copied into the transmit buffer, and immediately forwarded when packet n arrives, thereby significantly reducing forwarding delay.

B. Advantages of Channel Hopping without Frequency Reuse

Channel-hopping exploits frequency diversity by sending subsequent packets on different channels. Since the packet loss is usually bursty in nature, a transmission failure on a particular channel is likely to result in a failed subsequent transmission whereas a different channel would behave independently. Thus channel hopping can effectively reduce external interference and multi-path fading effects. To fully explore the advantages of channel hopping, each node should have the ability to switch channels with short delay. Recent results show that most IEEE 802.15.4-compliant radio chips can switch channels in less than 192μs [6].

Although frequency reuse is often advocated to enhance the channel utilization, per-timeslot channel hopping without frequency reuse still has several advantages in practice: first, it provides seamless integration of channel assignment with link scheduling. In schemes based on frequency reuse, channels are initially and statically assigned to links/nodes based on interference graphs, and link scheduling is performed in a separate stage. Since interference could potentially be removed by scheduling interfering links on different slots, the traditional approaches might underperform dynamic scheduling and channel hopping. Second, it significantly reduces the complexity of transmission scheduling. Since no (in network) multi-access interference is generated without channel reuse, transmissions can be scheduled without constructing an interference graph, whereas most channel assignment problems in traditional scheduling approaches are based on interference graphs and have been proven to be NP-hard [18] [20]. Moreover, most control applications require quite dense deployments in a limited area such as a small factory. In the worst case when each node falls in the interference range of all the others, there is no benefit for frequency reuse.

III. System Model and Problem Statement

We model a WSN network as a directed graph $G = (V, E)$ where the vertices in $V = \{v_0, v_1, ..., v_N\}$ denote the network devices (sink and sensors) and the edges in E represent communication links. There is only one sink, denoted by v_0 and N sensors v_1,...,v_N. Both the sensors and the sink remain static after deployment. Time is synchronized and divided into slots of equal size, and the length of a time slot allows transmitting/receiving exactly one packet. Each device is equipped with one half-duplex radio transceiver, which means that a device cannot transmit and receive at the same time. Channel hopping is performed on a per time-slot basis without frequency reuse. We assume that the links in E can sustain reliable transmission. Possible extensions to deal with unreliable links are discussed in Section VIII-B.

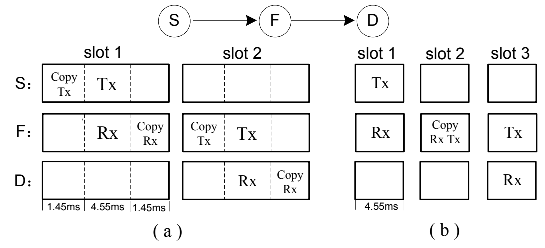


Fig. 2. The source (S) transmits a packet to the destination (D) via the forwarder (F). Traditionally, packet copying is performed within the time slot for transmission/reception (a). By separating packet copying from transmission and performing it in a separate time slot (b), the length of time slots is reduced.

Existing TDMA-based convergecast schemes do not take into account the impact of packet copying, which is commonly performed within the time slots allocated for packet transmission or reception, as shown in Fig. 2(a). In this mechanism, one channel is reserved during the entire time slot for either transmission or reception, resulting in low channel utilization. In our scheme, packet copying is separated from packet transmission/reception, as shown in Fig. 2 (b). Based on our measurements, the time for two-way packet copying (i.e. 2×1.45ms) is less than the time for transmission/reception (i.e. 4.55ms). Thus the two-way packet copying can be completed within one time slot in our scheme. After receiving a packet, a separate time slot is allocated for two-way packet copying (i.e., Copy Rx Tx). At the next available transmission time slot, the radio can immediately transmit the packet. There are two advantages for this mechanism. First, the channel can be released immediately after transmitting/receiving a packet and can be allocated to another node for transmission/reception in next time slot, thereby improving channel utilization. Second, although the number of time slots needed increases, the length of time slots is significantly reduced, enabling convergecast with lower latency and higher throughput.

Based on the above model, we revisit the convergecast operation in which each sensor generates one packet destined to the sink, and the convergecast packets are routed along a spanning tree, denoted by $T = (V, E')$ where $E' \subset E$. For any routing tree shown in Fig. 3, let $ST(v_i)$ be the largest subtree rooted at node v_i, and n_i be the number of nodes in $ST(v_i)$. The sink 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 every node v_i, $f(v_i)$ denotes its parent and $C(v_i)$ represents the set of its children. We use $\mathcal{L}_{\mathcal{S}}$ to represent the length of the convergecast schedule $\mathcal{S}$ (in time slots). The objective is to compute optimal link schedule so that the convergecast time can be minimized.

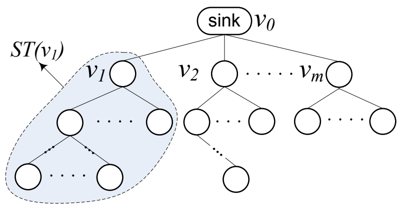


Fig. 3. Tree routing topology with m branches.

IV. Optimal Scheduling

In this section, we first establish the lower bound on the number of time slots required for convergecast, and then present a centralized algorithm to compute the time-optimal convergecast schedule for networks with tree routing topology.

A. Lower Bound on Convergecast Schedule Length($\mathcal{L}_{\mathcal{S}}$)

Theorem 1: For a routing tree with N sensor nodes, the lower bound on the number of time slots required to complete convergecast is $\max\{3n_1 - \Delta, N\}$ where $\Delta = 1$ if $n_1 = n_2$, and $\Delta = 2$ otherwise.

Proof: For any subtree $ST(v_i)$ rooted at node v_i, all packets generated by the nodes in $ST(v_i)$ must go through node v_i. For the packet generated by node v_i, it can be preloaded into the transmit buffer, and transmitted at the first available time slot. To forward any other packet, node v_i needs one slot to receive the packet, one slot for the two-way packet copying, and one slot for transmitting the packet. Thus node v_i needs at least $3(n_i - 1) + 1 = 3n_i - 2$ time slots to forward the n_i packets generated by nodes in $ST(v_i)$.

Suppose that subtrees $ST(v_1), ST(v_2), ..., ST(v_k) (k \leq m)$ have the same number of nodes (i.e., $n_1 = n_2 = ... = n_k > n_{k+1}, ..., \geq n_m$). Let t_i be the earliest time slot in which node v_i is scheduled for transmission. Let v_j be the node where $t_j = \max_{i\in[1,k]} t_i$. Clearly, $t_j \geq k$ because the sink can be scheduled to receive from only one of its children at any time slot. Since at least $3n_j - 2$ time slots are needed to complete convergecast in subtree $ST(v_j)$, the earliest time slot in which node v_j finishes forwarding all packets in $ST(v_j)$ is no less than $3n_1 - 2 + (k - 1)$. Therefore, at least $\mathcal{L}_{\mathcal{S}} = \max\{3n_1 - 2 + (k - 1), N\}$ time slots are needed for convergecast in a tree network. If $k = 1$, $\mathcal{L}_{\mathcal{S}} = \max\{3n_1 - 2, N\}$. If $k = 2$, $\mathcal{L}_{\mathcal{S}} = \max\{3n_1 - 1, N\}$. If $k > 2$, $\mathcal{L}_{\mathcal{S}} = 3n_1 - 2 + (k - 1) \leq kn_1 \leq N$. Thus the theorem holds. ■

Theorem 1 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 $(N-\Delta)/3$ nodes, convergecast can be completed in N time slots; otherwise the largest subtree will dominate the achievable convergecast latency, and the worst case is a line routing topology where the convergecast length is $3N-2$. 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 it is known to be NP-hard [21], many effective heuristic and approximation algorithms exist [22].

B. Time-optimal Convergecast Scheduling

In a tree network, each node might have more than one child. To minimize the number of time slots for convergecast, each node should be optimally scheduled to receive packets from its children. Let $Tx(v_i)$ be the number of packets that device v_i has transmitted since the start of the convergecast operation. The convergecast 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 time-optimal convergecast schedule can be computed using the policy combining the following two rules:

- **T1**: At each time slot t, node $v_i \in C(v_0)$ is scheduled for transmission if $ST(v_i)$ has the maximum number of packets left and node v_i is not scheduled for transmission at time slots $t-1$ and $t-2$.
- **T2**: At each time slot t, node $v_i \notin C(v_0)$ is scheduled for transmission if the following three conditions are satisfied: (1) node v_i has not finished transmitting all the packets it should transmit (i.e., $Tx(v_i) < n_i$); (2) node $f(v_i)$ is scheduled for transmission at time slot $t-1$; (3) $ST(v_i)$ has the maximum number of packets left among all subtrees $ST(v_j)$ rooted at the children of node $f(v_i)$.

Let $\varphi_t(v_i)$ be the set of candidates that can be scheduled to transmit a packet to node v_i at time slot t. $\varphi_t(v_i) = \{v_j | v_j \in C(v_i) \wedge Tx(v_j) < n_j \wedge v_j \notin Sch[t-1] \wedge v_j \notin Sch[t-2]\}$, and node $v_j \in C(v_i)$ that satisfies the following condition is scheduled for transmission.

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

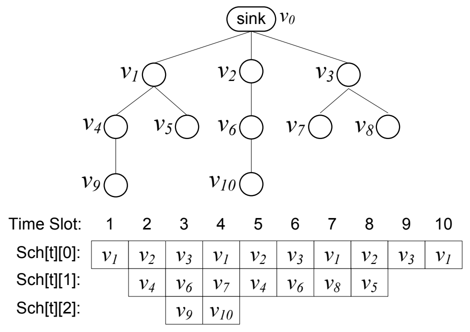
In case that there is a tie (i.e., multiple nodes satisfy the conditions given in the above policies), the node with the lowest ID is given the highest priority. The detailed algorithm for generating the time-optimal convergecast schedule is given in Algorithm 1. Fig. 4 gives an example of the schedule generated by Algorithm 1.

Corollary 1: The convergecast schedule generated by Algorithm 1 requires each sensor to buffer at most one packet at any time slot.

Proof: According to policy **T2**, any node v_i with hop-count larger than one is scheduled for transmission only if its parent has been scheduled in the previous time slot. Note that each sensor generates only one packet in each convergecast round. If node v_i is scheduled for transmission, there must be no packet at its parent. Thus the corollary holds. ■

Algorithm 1: Centralized_Convergecast_Scheduling

```
begin
    $\mathcal{L}_{\mathcal{S}} \leftarrow \max\{3n_1 - \Delta, N\}$;
    $\varphi_1(v_i) \leftarrow C(v_i)\ (0 \le i \le N)$;
    $Tx(v_i) \leftarrow 0$;
    for $t \leftarrow 1$ to $\mathcal{L}_{\mathcal{S}}$ do
        $ch \leftarrow 0$;
        /* Schedule children of the
           sink--Policy T1;                    */
        if $\varphi_t(v_0) \neq \phi$ then
            $Sch[t][ch] \leftarrow \arg\max_{v_k \in \varphi_t(v_0)}(n_k - Tx(v_k))$;
            $ch \leftarrow ch + 1$;
            $Tx(Sch[t][ch]) \leftarrow Tx(Sch[t][ch]) + 1$;
        /* Schedule other nodes--Policy
           T2;                                 */
        $k \leftarrow 1$;
        while $Sch[t-1][k] > 0$ do
            $v_i \leftarrow \arg\max_{v_k \in \varphi_t(Sch[t-1][k])}(n_k - Tx(v_k))$;
            if $v_i > 0$ then
                $Sch[t][ch] \leftarrow v_i$; $ch \leftarrow ch + 1$;
                $Tx(v_i) \leftarrow Tx(v_i) + 1$;
            $k \leftarrow k + 1$;
        for $i \leftarrow 0$ to $N$ do
            Generate $\varphi_{t+1}(v_i)$;
```



Time Slot:	1	2	3	4	5	6	7	8	9	10
Sch[t][0]:	v_1	v_2	v_3	v_1	v_2	v_3	v_1	v_2	v_3	v_1
Sch[t][1]:		v_4	v_6	v_7	v_4	v_6	v_8	v_5		
Sch[t][2]:			v_9	v_{10}						

Fig. 4. Optimal convergecast schedule for a sample tree network.

Theorem 2: Given any tree network, the schedule generated by Algorithm 1 can complete convergecast in $\max\{3n_1 - \Delta, N\}$ time slots where $\Delta = 1$ if $n_1 = n_2$; otherwise $\Delta = 2$.

Corollary 2: For any tree network, the convergecast schedule generated by Algorithm 1 uses at most d channels where d is the depth of the tree.

Due to space limitations, the proofs of Theorem 2 and Corollary 2 are omitted. The complete proofs can be found in a technical report [23]. Let $|C|$ be the maximum number of children a node has. The time complexity for generating $\varphi_t(v_i)$ (line 19-20) is $O(|C|N)$. Based on Corollary 2, the time complexity for the while-loop in Algorithm 1 is $O(|C|d) < O(|C|N)$.

Thus the time complexity of Algorithm 1 is $O(|C|N\mathcal{L}_{\mathcal{S}})$.

From Corollary 2, it can be seen that Algorithm 1 has constraint on the depth of the routing tree, i.e., the depth of the routing tree should not be larger than the number of available channels. Even though some channels might be blocked due to the bad quality caused by adjacent channel interference or co-existed 802.11 device, the number of available channels might be enough for convergecast in most control applications. This is because that it is recommended to deploy networks with small depth (typically 4-5 hops [2]) due to the problems such as long delay for packet delivery, the difficulty with maintaining accurate time-synchronization for TDMA, etc. It is also worth noting that our scheme does not pose any restriction on the number of nodes in the network since there is no limitation on the number of children a node can have. For example, for a balanced complete 3-ary tree routing tree in which all leaf nodes are at the same depth and each node (except the leaves) has exactly 3 children, the number of sensor nodes, excluding the sink, goes up to 1092 when the depth of the tree is 6.

V. Distributed Scheduling

While centralized management is natural in some applications and employed in standards such as *Wireless*HART, it is sometimes useful to perform the scheduling in a distributed manner. In this section, we demonstrate that it is possible to develop a distributed algorithm for computing the time-optimal link schedule for convergecast in tree routing topologies.

During tree construction stage, each node v_i can obtain the following information: n_i, the number of nodes in sub-tree $ST(v_i)$; $C(v_i)$, the set of children of node v_i; and n_j for $v_j \in C(v_i)$, the number of nodes in sub-tree $ST(v_j)$ rooted at a child of node v_i. Let $Dsch_{v_i}[1, \mathcal{L}_{\mathcal{S}}][1, 2]$ be the transmission schedule for the children of node v_i, where $Dsch_{v_i}[t][1]$ records the child scheduled to transmit at time slot t and $Dsch_{v_i}[t][2]$ is the channel offset allocated to the child scheduled for transmission at time slot t.

From Policy **T1** in Section IV, it can be seen that the transmission schedule for the children of the sink is computed based only on information about (a) the set of children of the sink, and (b) the number of nodes in each sub-tree rooted at a child of the sink. Thus the sink can compute the transmission schedule for its children, and the channel offset for each transmission is initialized as follows: If $Dsch_{v_i}[t][1]$ is not empty, $Dsch_{v_i}[t][2]$ is set to 0; otherwise $Dsch_{v_i}[t][2]$ is set to -1. After generating $Dsch_{v_0}$, the sink multicasts $Dsch_{v_0}$ to its children. When a node v_i receives the $Dsch_{f(v_i)}$ from its parent $f(v_i)$, node v_i generates $Dsch_{v_i}$ based on the $Dsch_{f(v_i)}$ using Policy **T2** in Section IV, and the channel offset is determined as follows: $Dsch_{v_i}[t][2] = Dsch_{f(v_i)}[t][2] + 1$ if $Dsch_{v_i}[t][1]$ is not empty; otherwise $Dsch_{v_i}[t][2] = Dsch_{f(v_i)}[t][2]$. Then node v_i multcasts $Dsch_{v_i}$ to its children. The distributed algorithm performed at each node to generate time-optimal convergecast schedule is given in Algorithm 2.

Algorithm 2: Distributed_Convergecast_Scheduling(node v_i)

```
begin
    $\mathcal{L}_{\mathcal{S}} \leftarrow \max\{3n_1 - \Delta, N\}$;
    if $v_i = sink$ then
        Generate $Dsch_{v_i}$          /* Policy T1 */;
        for $t \leftarrow 1$ to $\mathcal{L}_{\mathcal{S}}$ do     /* channel offset */
            if $Dsch_{v_i}[t][1] \neq 0$ then
                $Dsch_{v_i}[t][2] = 0$;
            else $Dsch_{v_i}[t][2] = -1$;
        Multicast $Dsch_{v_i}$ to all nodes in $C(v_i)$;
    if receive Dsch from $f(v_i)$ and $C(v_i) \neq \phi$ then
        Generate $Dsch_{v_i}$          /* Policy T2 */;
        for $t \leftarrow 1$ to $\mathcal{L}_{\mathcal{S}}$ do     /* channel offset */
            if $Dsch_{v_i}[t][1] \neq 0$ then
                $Dsch_{v_i}[t][2] = Dsch_{f(v_i)}[t][2] + 1$;
            else $Dsch_{v_i}[t][2] = Dsch_{f(v_i)}[t][2]$;
        Multicast $Dsch_{v_i}$ to all nodes in $C(v_i)$;
```

Theorem 3: If there is no packet loss and $Dsch_{v_i}$ can be encapsulated in one packet, Algorithm 2 can generate the time-optimal convergecast schedule for a tree network by multicasting only $N - n_l + 1$ packets where n_l is the number of leaf nodes in a tree network.

Proof: In algorithm 2, each non-leaf node in the tree needs to generate the transmission schedule for its children and multicast the schedule to its children. If there is no packet loss and $Dsch_{v_i}$ can be encapsulated into one packet, each non-leaf node only needs to multicast one packet. Thus the total number of packets multicasted is $N - n_l + 1$ where n_l is the number of leaf nodes in the tree network. ■

VI. Sub-schedule Extraction and Channel Hopping

The schedule generated by our algorithms is recorded in compact structures (Sch in Algorithm 1 and $Dsch$ in Algorithm 2) specifying which node is scheduled for transmission on each channel at each time slot. Once obtained Sch or $Dsch$, each node should extract its sub-schedule and generate the corresponding channel offset sequence.

Each transmission is associated with a channel offset which represents the logical channel to be used. At every time slot, each node can work in the following four states: *Transmit* (T), *Receive* (R), *Copy* (C) and *Sleep* (S). The sub-schedule and channel hopping sequence for each node is recorded in a 2-dimensional array $S\_sch[1, \mathcal{L}_{\mathcal{S}}][1, 2]$, where $S\_sch[t][1]$ records the state that the node should work in at time slot t and $S\_sch[t][2]$ records the channel offset used at time slot t.

A. Centralized Scenario

In this scenario, the convergecast schedule Sch is first computed at the sink, and then distributed to all sensors in the network. Let v_k be the node recorded in $Sch[t][ch]$. The sub-schedule and channel offset for node v_i can be generated using the following policies:

- At time slot t, node v_k works in *Transmit* state, and both v_k and $f(v_k)$ are assigned with channel offset of ch.
- At time slot $t+1$, node v_k works in *Receive* state if $Tx(v_k) < n_k$; otherwise, node v_k enters into *Sleep* state.
- At time slot $t+2$, node v_k works in *Copy* state if the state at time slot $t+1$ is *Receive*; otherwise, node v_k enters into *Sleep* state.

The associated state transition diagram is illustrated in Fig. 5. The algorithm for generating the sub-schedule and channel

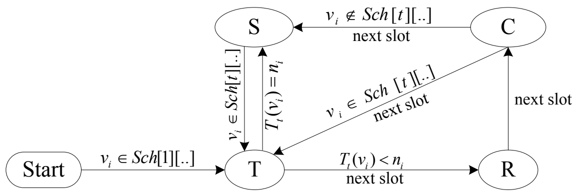


Fig. 5. State transition at node v_i for sub-schedule extraction.

hopping for node v_i is given in Algorithm 3. Fig. 6 presents the resulting subschedule and channel hopping sequence for node v_1 based on the schedule in Fig. 4.

Algorithm 3: Sub-schedule Generation(v_i)

```
begin
    for $t \leftarrow 1$ to $\mathcal{L}_S$ do
        for $ch \leftarrow 0$ to $Ch$ do
            /* Transmit                                  */
            if $Sch[t][ch] = v_i$ then
                $S\_sch[t][1] \leftarrow T$;
                $S\_sch[t][2] \leftarrow ch$;
            /* Receive                                   */
            else if $Sch[t][ch] \in C(v_i)$ then
                $S\_sch[t][1] \leftarrow R$;
                $S\_sch[t][2] \leftarrow ch$;
            /* Copy                                      */
            else if $S\_Sch[t-1][1] = R$ then
                $S\_sch[t][1] \leftarrow C$;
            /* Sleep                                     */
            else $S\_sch[t][1] \leftarrow S$;
```

Time Slot:	1	2	3	4	5	6	7	8	9	10
S_sch[t][1]:	T	R	T	R	T	R	S	T	S	S
S_sch[t][2]:	0	1	0	1	0	1		0		

Fig. 6. The sub-schedule and channel assignment for node v_1 based on the schedule given in Fig. 4.

B. Distributed Scenario

In Algorithm 2, each node v_i computes the transmission schedule of its children $Dsch_{v_i}$, and the channel offset for each transmission has already been assigned. Based on $Dsch_{v_i}$, node v_i can determine the time slots in which v_i should be in *Receive* state and the channel offset allocated for packet reception. If node v_i works in *Receive* state in time slot t, it works in *Copy* state in time slot $t+1$. After node v_i receives the transmission schedule $Dsch_{f(v_i)}$ from its parent $f(v_i)$, node v_i can determine the time slots in which v_i should be in *Transmit* state and the channel offset allocated for packet transmission. For the time slots in which the work state has not been assigned, node v_i works in *Sleep* state.

C. Channel Hopping

To support channel hopping, all devices maintain the same channel table, called ChannelMap, that tracks the active channels. For a given slot and channel offset, the physical channel used is determined as follows:

$$Activechannel = (channeloffset + ASN)\%NumChannels$$

where ASN denotes the Absolute Slot Number which is the count of all slots that have occurred since the network is formed. The remainder of this operation (i.e., $Activechannel$) is used as the index into ChannelMap to obtain the physical channel. For a given absolute slot, the devices scheduled in this slot are assigned with different channel offset, and thus operate on different physical channels. Since ASN is only increased and never resetted, the same device operates on different physical channels in different time slots.

VII. Performance Evaluation

In this section, we evaluate the performance of our convergecast schemes through both simulations in the COOJA simulator [24] and experiments on the Tmote Sky platform. All schemes are implemented on the Contiki operating system using the Rime protocol stack.

A. Experiment Setup

Time synchronization plays an important role in TDMA-based schemes. Recent work has shown high synchronization accuracy (around $1\mu s$) with low power consumption [25]. For simplicity, we use the following techniques to provide high-resolution time synchronization in our experiments: first, we use Contiki's built-in time synchronization service (i.e., pair-wise time synchronization), with hardware timers configured to 16 kHz. Second, we place all sensors in the communication range of the sink. At the beginning of each convergecast, the sink broadcasts beacons to synchronize all sensors. We use a simplified 802.15.4 MAC packet format in which the MAC Protocol Data Unit (MPDU) only consists of application data payload in addition to the 2-byte Frame Check Sequence (FCS) with CRC information. As the maximum physical layer payload size of 802.15.4 is 127 bytes, the maximum MAC layer data payload is 125 bytes. For each simulation/experiment, we verified that the tested slot length works by observing 10 consecutive convergecast rounds without lost radio packets. We use the following two metrics to evaluate the performance of our schemes:

- **Latency**: The latency for convergecast is defined as the time starting from the moment the first node transmits a packet until the sink has received all packets.

- **Throughput**: The throughput of the network is defined as the number of bits successfully received by the sink per second.

To highlight the performance, we compare the latency and throughput achieved by our scheme with the corresponding theoretical bounds which are calculated as follows: The 802.15.4 physical layer frame format consists of a 4-byte preamble, a 1-byte Start of Frame delimiter (SFD), a 1-byte frame length field, and a 2-byte CRC field. Given MAC layer data payload of size s_{mldp} bytes, the theoretical upper bound on the single-hop throughput is $T(s_{mldp}) = s_{mldp} \times 250/(4 + 1 + 1 + 2 + s_{mldp})$ kbit/s, and the minimum time slot length is $l_{slot} = 8 \times s_{mldp}/T(s_{mldp})$ ms. The lower bound on convergecast latency is $\mathcal{L}_{\mathcal{S}} \times l_{slot}$. In all experiments for a fixed MAC layer data payload, the time slot length is decreased until the radio packets start colliding and the sink no longer receives all data.

B. Latency

To evaluate the convergecast latency of our system, we maximize the packet payload: 125 bytes. Fig. 7 shows the minimum latency from simulation, verified results on real hardware as well as the theoretical lower bound on latency for convergecast in tree networks. As discussed in Section IV-A, the worst routing topology is a line with convergecast schedule length lower bound of $3N - 2$, and the best routing topology is a spanning tree which satisfies that $3n_1 - \Delta \leq N$. As can be seen from Fig. 7, in simulations with 20 sensor nodes and optimal spanning tree, the convergecast latency can be reduced to 0.1s, $1/3$ of the latency in the corresponding line network. This means that, in one second, the sink can collect data from up to 200 nodes using our scheme. Fig. 7 also shows that our scheme can work on real hardware using the same minimum time slot length obtained in simulations.

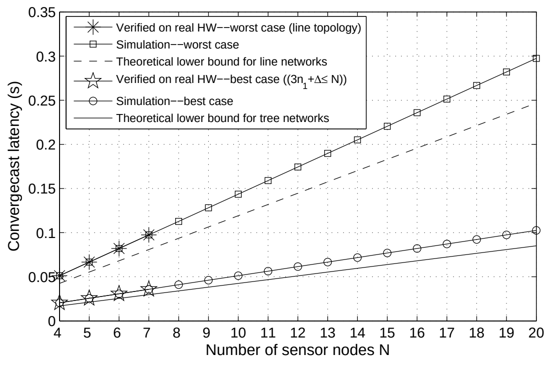


Fig. 7. Convergecast latency vs. number of nodes in a tree network.

From Fig. 7, it can be seen that there is a small gap between the simulation/experiment results and the theoretical lower bound. This can be explained as follows: first, the CC2420 radio platform has a 192 microseconds turnround time for changing state and channel [4]. This time cannot be moved off from the critical path. Second, there is some communication stack overhead in the copy slots. Although it might be possible to slightly further reduce convergecast latency by reducing interrupt latency, there is no too much improvement left.

C. Throughput

Fig. 8 plots the maximum throughput achieved in simulations on a network with 20 sensors and 1 sink by comparing with the theoretical upper bound. With maximum MAC layer data payload of 125 bytes, the throughput achieved in simulations is 203 kbit/s, around 86.4% of the upper bound (234.96 kbit/s). As the overhead such as turnround time remains fixed, this overhead dominates a large portion when MAC layer data payload is small. With the decrease of the MAC layer data payload, the effect of such overhead on throughput becomes larger. With MAC layer data payload of 11 bytes, the throughput achieved in simulations is 70.4 kbit/s, which is 48.64% of the upper bound (144.74 kbit/s).

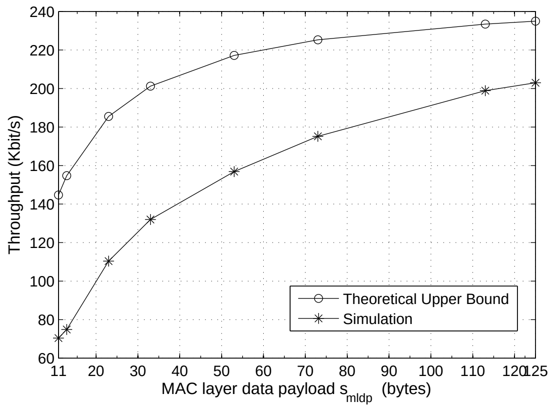


Fig. 8. Throughput vs. MAC layer packet size.

To show the advantage of our scheme, Fig. 9 compares with traditional convergecast in terms of throughput improvement. In Fig. 9, "traditional" denotes the multi-channel convergecast schemes such as the scheme proposed in [17], in which packet copying is performed within the time slots for transmission/reception, while our scheme is denoted by MCC_SC

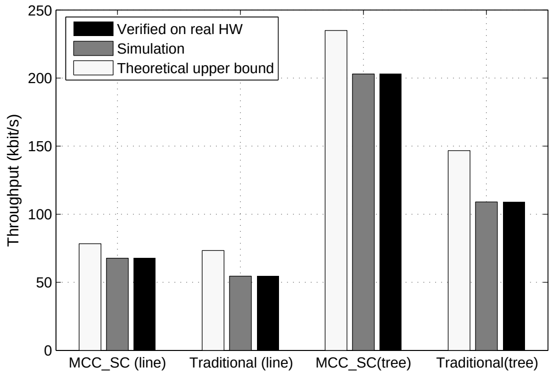


Fig. 9. Throughput improvement.

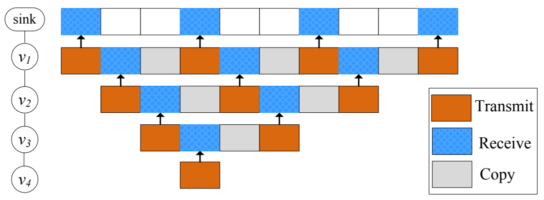


Fig. 10. Example for convergecast schedule in a line with separated packet copying.

(Multi-Channel Convergecast with Separated Copying). In our experimental setup, the time for transmitting 125 byte data is around 4.55ms, and the time for copying 125 bytes in two ways, i.e., from the receive buffer at the radio transceiver to the microcontroller and from the microcontroller to the transmit buffer at the radio transceiver, is around $2 \times 1.45 = 2.9$ ms, equivalent to $125 \times 2.9/4.55 = 40$ bytes. Therefore, the theoretical upper bound on throughput for traditional convergecast is $125 \times 250/(125 + 8 + 80) = 146.71$ kbit/s. As shown in Fig. 9, for networks with line routing topology, the theoretical upper bound on throughput for MCC_SC is only $1/3$ of the maximum throughput. This is because that the sink can be scheduled to receive only once in every 3 adjacent time slots, as demonstrated by the example in Fig. 10. For traditional convergecast schemes, the theoretical upper bound on throughput is roughly the same as that for MCC_SC because packet copying time slots can not be reused by other nodes. However, the achievable throughput for MCC_SC has been improved by 24.16 $\%$ compared with traditional schemes, which can be explained based on Fig. 2. In traditional schemes, there is 1.45 ms reserved for packet copying at both the beginning and the end of each time slot. However, only one part can be used, i.e., in transmit slot, only the 1.45 ms at the beginning is used for Copy Tx, and only the 1.45 ms at the end is used for Copy Rx in receive slot. In MCC_SC, the unused 1.45 ms in each slot has been removed, thereby improving throughput.

For networks with optimal tree routing topologies (i.e., $3n_1 - \Delta \leq N$), the throughput achieved by MCC_SC is 203 kbit/s, improved by 86.24 $\%$ compared with traditional convergecast. This is because that the copy time slots can be reused by other nodes. For example, if the network given in Fig. 10 has 3 lines, the copy time slots in the schedule for one line can be reused by nodes in other lines for transmission, and the sink is kept busy receiving packet at every time slot, thus significantly improving system throughput.

VIII. Discussion

A. Related Work

TDMA can provide collision-free and energy-efficient real-time data delivery. A number of TDMA-based convergecast protocols have been developed for single-radio single-channel architectures. Choi *et al*. proved that the decision version of time-optimal convergecast scheduling problem for single-channel wireless sensor networks is NP-complete in a weak sense [13]. Similarly, Gandham *et al*. proved that the lower bound for time-optimal TDMA convergecast schedule length in tree networks is $\max\{3n_1 - 3, N\}$ time slots, and proposed a distributed algorithm that can complete convergecast in $\max\{3n_1 - 1, N\}$ time slots [15]. Although the bound we established in this study look similar, they are in fact very different. By removing copying from the critical path, our schemes support much shorter time slot length, hence yielding much faster convergecast. Moreover, Gandham's scheme is based on single-channel communication, and focuses on spacial reuse to reduce convergecast latency.

Interference and multi-path fading are two major factors that affect the performance of existing single-channel based schemes. A natural approach to avoid interference and increase throughput is to use multiple channels. Wu *et al*. proposed a Tree-based Multi-Channel Protocol (TMCP) for data collection applications [18]. TMCP divides the network into multiple subtrees and allocates different channels to each subtree. Durmaz Incel and Krishnamachari proposed a simple receiver-based frequency and time scheduling approach for data aggregated convergecast [17]. However, all the above schemes are focused on making clever utilization of frequency-reuse to decrease convergecast delay, whereas the formulated channel assignment problems have been proven to be NP-complete. In [26], we proposed a self-organizing, multi-channel protocol for convergecast based on constructing collision-free trees. This scheme, however, is not time-optimal. We also proposed a general framework for real-time data delivery problems in *Wireless*HART networks and presented the optimal solution for convergecast in WirelssHART networks with binary-tree topology [27], and time- and channel-optimal convergecast solutions for *Wireless*HART networks with line topology [28]. However, all the above TDMA-based convergecast schemes do not take into account the impact of packet copying. The time synchronized mesh protocol (TSMP) [29], has reported impressive reliability in actual industrial scenarios, but there are no public results on the efficiency of TSMP under delay constrained traffic.

Packet copying between the radio transceiver and the microcontroller has been identified as the bottleneck for low latency packet delivery in networks with 802.15.4-based radios in [9]. Similar measurement results were also presented in [10], and a pipelining transmission scheme was proposed. However, the scheme only works for single-hop networks. Unlike the aforementioned related works, we propose novel low-complexity algorithms that exploit the advantages of both time-slotted channel hopping and separated packet copying, and validate our algorithms by extensive simulations and experiments on real sensor nodes.

B. Reliability Issues

Links in WSNs are notorious for poor and unstable quality, and packets are likely to get lost even without collisions. Thus retransmission mechanism might still be necessary for applications with high requirement on reliability. To retransmit a packet, the transmitter needs to know that the previously transmitted packet has been lost. An intuitive solution is to use an ACK mechanism. Similar to the *Wireless*HART standard, the receiver acknowledges each successfully received data packet, and the transmission of an ACK message is scheduled in the same time slot for packet receiving.

The basic idea for retransmission scheduling is to reserve some slots immediately after the first transmission for possible retransmissions. When a transmitter sends a data packet, the data packet still remains in the radio's transmit buffer. If the receiver successfully receives the data packet and the transmitter successfully receives the associated acknowledgement, the transmitter stays in sleep states in the following reserved slots for retransmission; otherwise the transmitter retransmits the data packet buffered in the radio transmit buffer unless the reserved slots for retransmission have been used up. There are other schemes to deal with lost packets such as repeating the convergecast frame several times to enhance reliability [11] [10]. However, we only present initial ideas in this work. Developing robust and reliable convergecast scheduling scheme and evaluating different convergecast schemes are out of the scope of this paper and are planned future work.

IX. CONCLUSION

Convergecast is an important communication primitive for data collection in WSNs. To achieve fast convergecast, we propose a novel model that combines TDMA-based multi-channel communication, per-timeslot channel hopping and separated packet copying, and present the time-optimal solution for convergecast in network with general tree routing topologies. Our schemes are memory-efficient, as each node is required to buffer at most one packet at any time slot. Both simulations and real experiments show that our scheme can achieve a maximum throughput of 203 kbit/s, 86.4% of the theoretical upper bound. With optimal routing structure and maximum MAC layer payload, convergecast in a network with 20 sensor nodes can be completed in only 100ms.

X. ACKNOWLEDGEMENTS

This work was sponsored by VINNOVA, the Swedish Agency for Innovation Systems, and the Swedish Foundation for Strategic Research.

REFERENCES

[1] *WirelessHART applications*, http://www.hartcomm2.org/hart_protocol/wireless_hart/wireless_hart_main.html.

[2] HARTCOMM, *Control with WirelessHART*, http://www.hartcomm.org/protocol/training/training_resources_wihart.html, 2009.

[3] *WirelessHART Data Sheet*, http://www.hartcomm2.org/hart_protocol/wireless_hart/wireless_hart_main.html, 2007.

[4] *Chipcon AS, CC2420 Datasheet(rev.1.3)*, 2005.

[5] *Crossbow Technology*, www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAZ_Datasheet.pdf, 2006.

[6] *Moteiv*, http://www.sentilla.com/pdf/eol/tmote-sky-datasheet.pdf, 2004.

[7] *Crossbow Technology*, http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf, 2004.

[8] S. Kim, R. Fonseca, P. Dutta, A. Tavakoli, D. Culler, P. Levis, S. Shenker, and I. Stoica, "Flush: A reliable bulk transport protocol for multihop wireless networks," in *Proceedings of the Fifth ACM Conference on Networked Embedded Sensor Systems(SenSys)*, 2007, pp. 351 – 365.

[9] F. Österlind and A. Dunkels, "Approaching the maximum 802.15.4 multi-hop throughput," in *Proceedings of the Fifth ACM Workshop on Embedded Networked Sensors*, 2008.

[10] K. K. Chintalapudi and L. Venkatraman, "On the design of MAC protocols for low-latency hard real-time distance control applications over 802.15.4 hardware," in *Proceedings of International Conference on Information Processing in Sensor Networks(IPSN)*, pp. 356–367.

[11] J. Pesonen, H. Zhang, P. Soldati, and M. Johansson, *Methodology and tools for controller-networking codesign inWirelessHART*, Proceedings of the 14th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2009.

[12] I. Rhee, A. Warrier, J. Min, and L. Xu, "DRAND: Distributed randomized tdma scheduling for wireless adhoc networks," in *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, 2006, pp. 190 – 201.

[13] H. Choi, J. Wang, and E. A. Hughes, "Scheduling on sensor hybrid network," in *Proceedings of the 14th International Conference on Computer Communications and Networks (ICCCN)*, 2005, pp. 503– 508.

[14] M. Adler, R. K. Sitaraman, A. L. Rosenberg, and W. Unger, "Scheduling time-constrained communication in linear networks," in *Proceeding of ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, 1998, pp. 269 – 278.

[15] S. Gandham, Y. Zhang, and Q. Huang, "Distributed time-optimal scheduling for convergecast in wireless sensor networks," *Computer Netowrks*, vol. 52, pp. 610–629, 2008.

[16] P. Djukic and S. Valaee, "Distributed link scheduling for tdma mesh networks," in *Proceedings of IEEE International Conference on Communications (ICC)*, 2007, pp. 3823–3828.

[17] O. Durmaz Incel and B. Krishnamachari, "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)*, 2008, pp. 569–577.

[18] Y. Wu, J. Stankovic, T. He, and S. Lin, "Realistic and efficient multichannel communications in wireless sensor networks," in *Proceeding of the 27th Conference on Computer Communications (INFOCOM)*, 2008, pp. 1193–1201.

[19] *The Contiki Operating System*, http://www.sics.se/contiki/.

[20] O. Durmaz Incel, A. Ghosh, B. Krishnamachari, and K. K. Chintalapudi, "Multi-channel scheduling for fast convergecast in wireless sensor networks," Department of Computer Science, University of Twente, Tech. Rep., 2008.

[21] C. H. Papadimitriou, "The complexity of the capacitated tree problem," *Networks*, vol. 8, pp. 217 – 230, 1978.

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

[23] H. Zhang, F. Österlind, P. Soldati, T. Voigt, and M. Johansson, "Time-optimal convergecast with separated packet copying: Scheduling policies and performance," School of Electrical Engineering, Royal Institute of Technology (KTH), Tech. Rep., 2009.

[24] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "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)*, 2006.

[25] T. Schmid, P. Dutta, and M. B. Srivastava, "High-resolution, low-power time synchronization and oxymoron no more," in *Proceedings of the 9th International Conference on Information Processing in Sensor Networks (IPSN)*, 2010, pp. 151 – 161.

[26] T. Voigt and F. Österlind, "CoReDac: Collision-free command-response data collection," in *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2008.

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

[28] H. Zhang, P. Soldati, and M. Johansson, "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)*, 2009.

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