

# Distributed Minimal Time Convergecast Scheduling in Wireless Sensor Networks

Shashidhar Gandham  
Department of Computer Science  
University of Texas at Dallas  
Email: gshashi@utdallas.edu

Ying Zhang and Qingfeng Huang  
Computer Science Laboratory  
Palo Alto Research Center (PARC) Inc.  
Email: {yzhang, qhuang}@parc.com

**Abstract**—We consider applications of sensor networks wherein data packets generated by every node have to reach the base station. This results in a many-to-one communication paradigm referred to as convergecast. We are interested in determining a TDMA schedule that *minimizes the total time* required to complete the convergecast. We consider a simple version of the problem wherein every node generates exactly one packet. We propose a *distributed convergecast scheduling algorithm* that requires at most  $3N$  timeslots, where  $N$  represents the number of nodes in the network. Through extensive simulations, we demonstrate that actual number of timeslots needed is around  $1.5N$ . In addition to time efficiency, we prove that our convergecast scheduling algorithm requires the nodes to buffer no more than two packets at any instance. We propose a sleep schedule that conserves more than 50% of the energy. We present simulation results for a real application scenario to show that our convergecast scheduling algorithm performs significantly better than existing convergecast algorithms.

**Keywords:** Wireless Sensor Networks, Convergecast, Scheduling, TDMA.

## I. INTRODUCTION

Convergecast is a typical many-to-one communication pattern in sensor network applications. In convergecast many or all nodes in the network send data to a base station during a relatively short time period. For instance, convergecast occurs in applications requiring periodic global snapshots such as monitoring the residual energy of nodes. Event-driven data collection as in the sniper localization [17] also results in convergecast. In these applications all the packets generated in the network have to reach a base station either for record or for computationally intensive analysis.

A guarantee on packet delivery and a bound on convergecast latency are highly desirable in mission critical applications, e.g., surveillance and security. Guarantee on packet delivery ensures availability of accurate information about the sensing field. An optimal bound on convergecast latency leads to timely detection of the events. Furthermore, a known time bound on convergecast latency can help the base station to schedule convergecast requests and related computations.

It is well-known that collisions present a major challenge in convergecast when contention-based MAC protocols like CSMA are employed. Collisions result in loss of packets and recovery methods such as retransmissions increase the latency. Moreover, retransmissions might lead to further collisions in high data rate scenarios [4]. In addition, retransmissions drain the scarce energy reserves of sensor nodes. Radially coordinated transmission [13] was proposed to decrease the probability of collisions. However, as a result of using CSMA

MAC layer, the convergecast latency incurred by radial coordination is far from the optimal.

Contention-free MAC protocols like TDMA can be used to eliminate collisions and obtain a bound on the time required to complete convergecast. Hence, we are interested in determining a TDMA schedule such that the entire convergecast can be completed in minimal number of timeslots. We refer to such a TDMA schedule as *minimal time convergecast schedule*. The important contributions of our work include:

(1) Distributed convergecast scheduling algorithm that requires at most  $3N$  timeslots in any network. Simulation results show that the number of timeslots required is about  $1.5N$ .

(2) A bound of two on the number of packets that are required to be buffered at a node during convergecast. This result is significant considering the fact that sensor nodes have a limited amount of memory.

(3) A sleep schedule for nodes that conserves at least 50% of energy. This result is important considering the limited amount of energy available at sensor nodes.

### A. Related Work

In TDMA, the time domain is sliced into *timeslots*. Multiple, spatially separated, non-interfering transmissions can be scheduled in each timeslot. Typically, TDMA scheduling algorithms assign timeslots to either the nodes (broadcast scheduling [9], [14], [15]) or to the edges (link scheduling [5], [6], [14]). These algorithms attempt to minimize the number of timeslots required for every node to communicate once with all their neighbors. Convergecast can be accomplished by employing existing TDMA scheduling algorithms. However, the latency incurred might be quite high.

Choi et. al. [7], show that convergecast scheduling problem is NP-hard. Kesselman and Kowalski consider the problem of convergecast in ad hoc geometric networks [1]. They assume that a transmitting node is capable of detecting collisions within its transmission range. In addition, the duration of a timeslot was assumed to be long enough to allow multiple packet transmissions in one timeslot. Typically wireless nodes cannot detect collisions while transmitting. We consider timeslot durations that allow transmission of *exactly one* packet.

Florens and McEliece have considered the problem of scheduling for packet distribution in sensor networks [2]. An algorithm to determine the minimal length schedule to send packets from the base station to the sensor nodes was proposed. The packet distribution scheduling problem can be considered as an inverse of the convergecast scheduling problem. The algorithm proposed in [2] can be employed for

convergecast. However, the proposed algorithm is centralized where the schedule is computed at the base station. Our algorithm is *distributed* where each node computes its own schedule after the initialization phase.

### B. System Model

We consider sensor networks wherein the nodes and the associated base station are static. All the nodes are equipped with a single omnidirectional transceiver. Hence, the nodes (including the base station) cannot transmit and receive at the same time. All the communications are carried over a single frequency band. The bandwidth of every wireless link in the network is assumed to be the same. We assume that the network connectivity is fixed over time. No specific assumptions, like unit-disk radio range model, are made about the propagation characteristics of the wireless medium. However, we consider only symmetric links for scheduling.

We assume that the maximum length of a packet is fixed. The duration of a timeslot allows transmission of exactly one packet. Similar to most of the TDMA MAC protocols, we assume that the drift in the clock of a node is bounded all the time. We consider applications wherein the base station has to receive every data packet sent by the nodes without aggregation of the data.

## II. CONVERGECAST IN TREE NETWORKS

We first consider a *linear* network of sensor nodes and propose an optimal convergecast scheduling algorithm. Next, we consider a network that can be reduced into multiple linear networks with base station as the point of intersection. Such networks are referred to as *multi-line* networks. We propose a distributed scheduling algorithm for convergecast in multi-line networks. We show that *tree* networks can be reduced into equivalent multi-line networks. Thus, providing a convergecast schedule for tree networks.

### A. Linear Networks

Consider a linear network as shown in Figure 1(a). We define the following states that a node can be in each timeslot during the convergecast:

- *R*: The node may receive from a neighboring node.
- *T*: The node can transmit.
- *I*: The node neither transmits nor receives.

Assume that every node obtains its hop-count from the base station before the convergecast begins. Each node in the network is assigned an initial state based on its hop-count from the base station. A node with hop-count  $h$  is assigned a (i) state *T* if  $h \bmod 3$  is 1, (ii) state *I* if  $h \bmod 3$  is 2 and (iii) state *R* if  $h \bmod 3$  is 0. Here, the *mod* operator returns the remainder after division of the operands. A node moves from one state to another in the subsequent timeslots according to the state transition diagram shown in Figure 2.

For example, consider the node  $i$  and its schedule as shown in Figure 1(b). The initial state of the node  $i$  is *T*; i.e.  $i$  transmits in the first timeslot ( $f_1(i) = s$ ). In the next timeslot  $i$  moves to the state *I*; i.e.  $i$  remains idle in the second timeslot. In the third timeslot the node  $i$  moves into state *R*; i.e.  $i$  receives a packet from  $h$  in the third timeslot. A node comes back to its start state after every three timeslots.

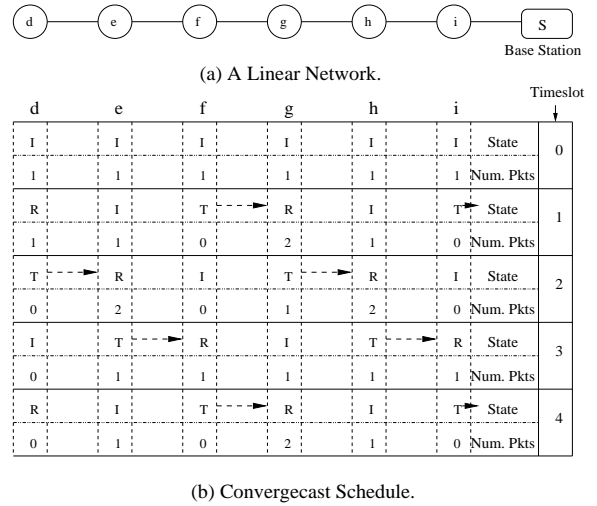


Fig. 1. Convergecast Scheduling in Linear Networks.

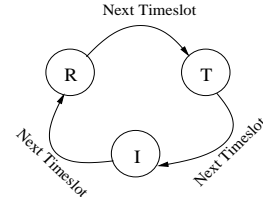


Fig. 2. State Transition for Convergecast Scheduling.

Nodes continue to move from one state to another until the completion of convergecast.

Figure 1(b) shows the state of nodes and the number of packets in the buffer of each node at the end of timeslots 1 to 4. Consider the state of the nodes in the first timeslot. Notice that for every node in state *R* there is only one node in its neighborhood which is in state *T*. As a result, all the packet transmissions will be successful.

Next, we show that at least  $3N - 3$  timeslots are required to complete convergecast in a linear network with  $N$  nodes. Then, we use mathematical induction to show that the above-described algorithm requires  $3N - 2$  timeslots. We suggest a simple modification to obtain an optimal schedule in  $3N - 3$  timeslots when  $N > 1$ .

**Theorem 2.1:** The lower bound on the number of timeslots required to complete convergecast is  $3N - 3$ , where  $N$  is the number of nodes in the network.

**Proof:** Consider the node  $g$  in Figure 1(a) which is three hops away from the base station. Among the links  $(g, h)$ ,  $(h, i)$  and  $(i, s)$  transmissions can be scheduled simultaneously along only one link due to interference. Hence, every packet that flows through the node  $g$  will require at least 3 timeslots to reach the base station. Note that in convergecast  $g$  has to forward  $N - 2$  packets. Also, packets that originate at nodes  $h$  and  $i$  can reach the base station in 2 and 1 timeslot(s) respectively. Hence, at least  $3(N - 2) + 2 + 1 (= 3N - 3)$  timeslots are required to complete convergecast. ■

A similar analysis was used in [10] to obtain lower bounds for gossiping in a linear wireless network.

**Theorem 2.2:** The convergecast scheduling algorithm re-

quires  $3N - 2$  timeslots to complete convergecast in linear networks, where  $N$  is the number of nodes in the network.

**Proof:** Note that when  $N$  is equal to 1 the convergecast scheduling algorithm requires one timeslot. Hence, the above statement is valid when the network has one node. Let the statement be true when  $N$  is equal to  $K$ ; if the network has  $K$  nodes then  $3K - 2$  timeslots are required.

Now, let us consider a linear network that has  $K + 1$  nodes. Let the last node in the network be  $u$ . As per our scheduling algorithm,  $u$  will transmit once in the first three timeslots. As  $u$  is the last node in the network, it will not have any packets to transmit after the third timeslot. Except for node  $u$ , every other node transmits a packet and receives a packet in the first three timeslots. As a result, every node except  $u$  has one packet in its buffer at the end of the third timeslot. Considering only those node which need to transmit after first three timeslots, we have a network with  $K$  nodes. We know that for the remaining network convergecast can be completed in  $3K - 2$  timeslots. Hence, for a linear network with  $K + 1$  nodes convergecast is completed in  $3(K + 1) - 2$  timeslots.

By induction, we can conclude that the above statement is correct for all values of  $N$ . ■

To obtain a convergecast schedule using  $3N - 3$  timeslot, we use the above described algorithm in the first  $3(N - 2)$  timeslots. After  $3(N - 2)$  timeslots, only the nodes that are at one hop-away and two hops-away from the base station have one packet each. A three-timeslot schedule is started to forward the last two packets to the base station: (i) In the first timeslot the neighbor of the base station transmits its packet. (ii) Second and third timeslots are used obtain the packet of node that is two-hops away from the base station.

Typically sensor nodes have a limited amount of memory. For example, the MICA2 series motes from Crossbow [8] have 4 K bytes of RAM. Hence, the following result pertaining to our convergecast algorithm is significant.

*Theorem 2.3:* The convergecast scheduling algorithm requires nodes to buffer at most two packets in any timeslot.

**Proof:** Note that every node starts with one packet generated by it. In our scheduling algorithm, a node transmits once between two successive packet receptions. Hence, at most two packets are buffered at a node in any timeslot during the convergecast. ■

Our convergecast scheduling algorithm for other networks, described in next few sections, is based on the scheduling algorithm for linear networks described in this section. Hence, the above result about the maximum buffer size required at nodes is valid for networks of any topology.

### B. Multi-line Networks

In linear networks, the base station receives exactly one packet in every three timeslots. In multi-line networks (see Figure 3) more than one packet can be received by the base station in every three timeslots. The basic idea behind our convergecast scheduling algorithm for multi-line networks is to schedule transmissions *in parallel* along multiple branches. As the base station is equipped with a single transceiver, at most one packet can be received in a timeslot. Hence, in each timeslot we need to decide the branch along which a packet can be forwarded to the base station. In this section, we present a distributed algorithm to schedule packet transmissions from

branches of multi-line networks. The algorithm presented in this section is implemented at *each node* along with the state transitions described in Section II-A.

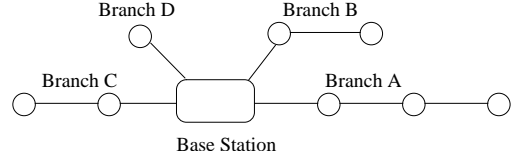


Fig. 3. A Multi-line Network.

During the initialization, each node obtains its hop-count to the base station and determines its initial state as described in Section II-A. The base station assigns a unique ID to every branch. The information about the number of nodes in each branch is collected and disseminated to all the nodes by the base station. After the initialization phase, each node is aware of (1) its initial state ( $T$ ,  $I$ , or  $R$ ), (2) its branch ID and (3) the number of nodes in all the branches of the network. Note that no node, including the base station, needs to be aware of the entire network topology.

If a packet is scheduled to be received by the base station from a branch, say  $\mathcal{I}$ , in timeslot  $t$  then the nodes in the branch are said to be *active* in timeslots  $t$ ,  $t + 1$  and  $t + 2$ ; i.e., all the nodes in branch  $\mathcal{I}$  change their states as per the state transition diagram (see Figure 2) in three successive timeslots starting at  $t$ . As the first node along each branch is initially in state  $T$ , the base station receives a packet from branch  $\mathcal{I}$  in timeslot  $t$ . Note that the first node in branch  $\mathcal{I}$  will have zero packets until the end of slot  $t + 2$ . As a result, no more packets can be received from branch  $\mathcal{I}$  in slots  $t + 1$  and  $t + 2$ . Hence,  $\mathcal{I}$  is *eligible* to forward the next packet only after timeslot  $t + 2$ .

In a given timeslot, an *eligible* branch with highest number of packets left is given priority to forward a packet to the base station. In the event of a tie the branch with lowest ID is given preference. As every node knows the number of packets in each branch they can independently decide which branch forwards a packet in a timeslot and act accordingly.

Branch	A	B	C	D
Pkts Left	3	2	2	1
Last Slot	0	0	0	0

(a) Before Timeslot 1

Branch	A	B	C	D
Pkts Left	2	2	2	1
Last Slot	1	0	0	0

(b) End of Timeslot 1

Branch	A	B	C	D
Pkts Left	2	1	2	1
Last Slot	1	2	0	0

(c) End of Timeslot 2

Branch	A	B	C	D
Pkts Left	2	1	1	1
Last Slot	1	2	3	0

(d) End of Timeslot 3

Branch	A	B	C	D
Pkts Left	1	1	1	1
Last Slot	4	2	3	0

(e) End of Timeslot 4

Branch	A	B	C	D
Pkts Left	1	0	1	1
Last Slot	4	5	3	0

(f) End of Timeslot 5

Branch	A	B	C	D
Pkts Left	1	0	0	1
Last Slot	4	5	6	0

(g) End of Timeslot 6

Branch	A	B	C	D
Pkts Left	0	0	0	1
Last Slot	7	5	6	0

(h) End of Timeslot 7

Branch	A	B	C	D
Pkts Left	0	0	0	0
Last Slot	7	5	6	8

(i) End of Timeslot 8

Fig. 4. Convergecast Schedule for Multi-line Networks.

For example, consider the multi-line network shown in Figure 3. The network consists of branches  $A$ ,  $B$ ,  $C$  and  $D$  ( $A < B < C < D$ ) with 3, 2, 2 and 1 nodes respectively. Each node in the network maintains a schedule table as shown

<p>Notation:</p> <p><math>Q</math>: transmission queue;</p> <p><math>\mathcal{I} \in \{1, \dots, k\}</math>: branch ID of this node;</p> <p><math>\mathcal{S} \in \{T, I, R\}</math>: current state;</p> <p><math>\mathcal{N} : [n_1, n_2, \dots, n_k]</math>: number of packets in each branch;</p> <p><math>t</math>: current timeslot;</p> <p><math>\mathcal{T} : [t_1, t_2, \dots, t_k]</math>: last <i>active</i> timeslot</p> <p><b>initialization:</b></p> <p><math>\mathcal{S}</math> set according to hop-count;</p> <p><math>\mathcal{N}</math> set with initial numbers given by the base station;</p> <p><math>t = 1</math>;</p> <p><math>t_i = 0 \forall i \in \{1, 2, \dots, k\}</math>;</p> <p><b>0. at timeslot <math>t</math>:</b></p> <p>1. Let <math>\mathcal{L} = \{i   t_i &lt; t\}</math>; set of eligible branches to transmit at <math>t</math>;</p> <p>2. Let <math>j = \arg \max\{n_i   i \in \mathcal{L}\}</math> be the first branch in <math>\mathcal{L}</math> that has the maximum number of packets left;</p> <p>3. <math>n_j \leftarrow \max(n_j - 1, 0)</math>; branch <math>j</math> will transmit at timeslot <math>t</math></p> <p>4. <math>t_j \leftarrow t_j + 2</math>; branch <math>j</math> will not be eligible for next two time slots</p> <p>5. <b>if</b> <math>t \leq t_{\mathcal{I}}</math>; this branch is active</p> <p>6.     <b>switch</b> <math>\mathcal{S}</math></p> <p>7.         case T: transmit one packet from <math>Q</math></p> <p>8.         case I: idle</p> <p>9.         case R: push the received packet into <math>Q</math></p> <p>10.     <b>end switch</b></p> <p>11.     <math>\mathcal{S} \leftarrow \text{next}(\mathcal{S})</math>; according to Figure 2</p> <p>12. <b>end if</b></p> <p>13. <math>t \leftarrow t + 1</math>;</p> <p>14. <b>if</b> <math>\max(\mathcal{N}) = 0</math> and <math>t &gt; \max(\mathcal{T})</math></p> <p>15.     <b>initialization</b></p> <p>16. <b>end if</b></p>
---

TABLE I  
SCHEDULING ALGORITHM FOR MULTI-LINE NETWORK

in Figure 4(a). The `Pkts Left` field is used to track the number of packets remaining in each branch. The `Last Slot` field shows the last timeslot in which a branch has forwarded a packet to the base station (two less than the last *active* timeslots). Branch *A*, with the maximum number of packets, forwards a packet in the first timeslot. Every node in the network can conclude the same and update their schedule tables as shown in Figure 4(b). All the nodes that belong to branch *A* will be active in timeslots 1, 2 and 3. In the second timeslot branches *B*, *C* and *D* are eligible to forward. As branches *B* and *C* have more packets than *D* and  $B < C$ , branch *B* forwards a packet in the second timeslot. The contents of the schedule table maintained at *each node* by the end of second timeslot is shown in Figure 4(c). In the current example, eight timeslots are required to complete the convergecast. Figures 4(d) - (i) show the contents of the schedule table at the end of timeslot 3 to 8. The pseudo code that has to be implemented on each node for the convergecast scheduling is shown in Table I. Next, we obtain the lower bound on the number of timeslots required for convergecast in multi-line networks. We show that our algorithm uses at most two timeslots more than the lower bound.

**Theorem 2.4:** If  $N$  represents the number of nodes in the network and  $n_k$  represents the maximum number of nodes in a branch, then the lower bound on the number of timeslots required for convergecast scheduling in multi-line networks is given by  $\max(3n_k - 3, N)$ .

**Proof:** As the base station requires  $N$  timeslots to receive all the packets that originate in the network,  $N$  is a lower bound for convergecast in any networks. Consider the branch

with  $n_k$  nodes. From Theorem 2.1, we know that at least  $3n_k - 3$  timeslots are required for convergecast in this branch. Thus,  $\max(3n_k - 3, N)$  is the lower bound on the number of timeslots required for convergecast in multi-line networks.

**Theorem 2.5:** If  $N$  represents the number of nodes in the network and  $n_k$  represents the maximum number of nodes in a branch, then the number of timeslots required by our convergecast-scheduling algorithm for multi-line networks is given by  $\max(3n_k - 1, N)$ .

**Proof:** Let  $n_i$  represent the number of nodes in branch  $i$  and  $n_k \geq n_{k-1} \dots \geq n_1$ . When there are at most two branches in the network, it is easy to show that the schedule determined by our algorithm requires at most  $3n_k - 1$  timeslots. From hereon, we consider networks which have at least three branches.

Let  $n_k > \lceil \frac{1}{2}(\sum_{i=1}^{k-1} n_i) \rceil$ . In this scenario it can be easily verified that  $\max(3n_k - 1, N) = 3n_k - 1$ . From Theorem 2.2, we know that at least  $3n_k - 2$  timeslots are required for convergecast in branch  $k$ . Of these  $3n_k - 2$  timeslots, branch  $k$  cannot forward any packet to the base station in  $2n_k - 2$  timeslots. The base station can receive packets from other branches in these timeslots. Note that the total number of packets in the remaining branches of the network is less than or equal to  $2n_k - 1$ . Also, our algorithm schedules branch  $k$  once in every three timeslots. Hence, the convergecast schedule determined by our algorithm requires at most  $3n_k - 2 + 1$  timeslots.

Let  $n_k \leq \lceil \frac{1}{2}(\sum_{i=1}^{k-1} n_i) \rceil$ . We can show that  $\max(3n_k - 1, N) = N$ . Here, we prove by induction that our convergecast-scheduling algorithm for multi-line networks requires exactly  $N$  timeslots. Consider a network in which the longest branch has one node (i.e.,  $n_k = 1$ ). Other branches in the network can have at most one node. In this scenario, our algorithm will schedule each branch in one timeslot. Hence, in  $N$  timeslots the convergecast will be completed.

Assume that our algorithm uses  $N$  timeslots for convergecast when the longest branch in the network has  $M$  nodes. Now, consider a network where the longest branch has  $M + 1$  nodes (i.e.,  $n_k = M + 1$ ). Our algorithm will schedule branches  $k, k - 1$  and  $k - 2$  in the first, second and third timeslots respectively. After the third timeslot branch  $k$  will have  $M$  packets left. If  $k$  is still the branch with highest number of packets left then according to our assumption convergecast in the remaining network can be completed in  $N - 3$  timeslots. Thus, the entire convergecast can be completed in  $N$  timeslots.

However, if more than three branches in the network have  $M + 1$  nodes then even after the third timeslot the longest branch in the network will have  $M + 1$  nodes. Assume that the network has  $l$  branches ( $l \leq k$  and  $l > 3$ ) with  $M + 1$  nodes. We know that our algorithm will always schedule the longest branch first. Hence, by the end of  $l^{\text{th}}$  timeslot we will have a network where the longest branch has  $M$  nodes (we consider only nodes which have a packet to transmit) and  $N - l$  packets are left in the network. According to our assumption, the convergecast in the remaining network can be completed in  $N - l$  timeslots. As a result, the convergecast for the entire network can be completed in  $N$  timeslots. Hence, by induction, we conclude that our convergecast scheduling algorithm requires  $N$  timeslots when  $n_k \leq \lceil \frac{1}{2}(\sum_{i=1}^{k-1} n_i) \rceil$ . ■

### C. Tree Networks

In the rest of the paper we use the term *one-hop-subtree* to refer to any subtree that is rooted at a one-hop neighbor of the base station. Our convergecast scheduling algorithm for tree networks is based on the observation that a tree network can be reduced to a multi-line network with each line represented as a combination of linear branches of nodes. For example, the tree shown in Figure 5(a) can be represented as a two-line network: (i)  $a-b-c-d$  and  $a-b-e-f$  rooted at  $b$ , and (ii)  $a-g-h$  and  $a-g-i-j$  rooted at  $g$ .

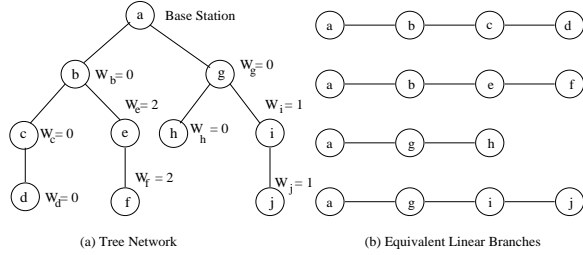


Fig. 5. Reduction of a Tree Network into Linear Branches.

Consider the example of the one-hop-subtree rooted at  $b$ . Initially, we schedule the transmissions along  $a-b-c-d$ . Once the packets generated by nodes  $c$  and  $d$  are received by  $b$ , we switch to scheduling transmissions along  $a-b-e-f$ . Note that by the time node  $b$  receives the packet generated by  $d$  two packets will be forwarded from the branch  $a-b-c-d$  to the base station. In order to make a seamless schedule switch, nodes  $e$  and  $f$  should know a priori how long they need to wait before forwarding packets. Alternatively, nodes  $e$  and  $f$  should know how many packets from their one-hop-subtree should be forwarded to the root before they can become *active*. Recall from Section II-B, that a node is said to be *active* when its changing states in three successive timeslots.

In general, let  $v$  be any node in the tree. Let  $W_v$  be the number packets that have to be forwarded by the corresponding subtree before node  $v$  becomes *active*. Let  $v$  have  $k$  child nodes  $v_1, v_2, \dots, v_k$ . Let  $N_1, N_2, \dots, N_k$  respectively be the number of nodes in each subtree rooted at  $v$ . Then  $W_{v_1} = W_v$ ; i.e.,  $v_1$  (the first child node) becomes active along with  $v$ .  $W_{v_2} = W_v + N_1$ ; i.e., node  $v_2$  becomes active after all the packets in subtree rooted at  $v_1$  have reached the node  $v$ . Similarly,  $\forall i \leq k$   $W_{v_i} = \sum_{j=1}^{i-1} N_j + W_v$ . To schedule in a tree network, nodes coordinate to determine their respective  $W_v$  values in the initialization phase. For example, an in-order tree traversal [12] initiated by the base station can be used. Figure 5 shows the value of  $W_v$  at each node.

Note that all the one-hop-subtrees (e.g. subtrees rooted at nodes  $b$  and  $g$  in Figure 5) can be scheduled in parallel as in a multi-line network (see Section II-B). Each one-hop-subtree will correspond to a branch in the multi-line network. An *eligible one-hop-subtree with highest number of packets left will be scheduled in each timeslot*; a one-hop-subtree is said to be *eligible* in timeslot  $t$  if it was not scheduled in timeslots  $t-2$  and  $t-1$ .

The pseudo code for scheduling in the tree network is the same as the one for multi-line networks, except that the active condition for a node (line 5 in Table I) has to be modified to:

$t < t_{\mathcal{I}}$  and  $\Delta n_{\mathcal{I}} \geq W_v$ . Where,  $\Delta n_{\mathcal{I}} = n_{\mathcal{I}}^0 - n_{\mathcal{I}}$  is the number of packets forwarded by one-hop-subtree  $\mathcal{I}$  to the base station before timeslot  $t$ .

We conclude that *the number of timeslots required by our convergecast scheduling algorithm for tree networks is at most  $\max(3n_k - 1, N)$  (Theorem 2.5)*. Here,  $n_k$  represents the number of nodes in the largest one-hop-subtree. For example, the network shown in Figure 5 can be viewed as a two-line network, and 14 ( $n_k = 5$ ) timeslots are required to complete convergecast. For tree networks, in addition to the information required for multi-line networks, each node  $v$  has to know (1)  $W_v$ : the linear order in its one-hop-subtree, and (2)  $N_v$ : the number of nodes rooted at  $v$ . Again, no node has information about connectivity of the whole network.

### D. Sleep Schedule for Energy Conservation

It is well known that idle listening consumes a significant amount of energy in sensor networks [18]. Nodes employing TDMA MAC can switch off their transceivers to conserve energy spent in idle listening. We propose to switch off the transceiver of a node when *either the node is inactive or the node has forwarded all the packets*. Note that given a node with  $W_v$ , it is inactive if (1) the branch of the one-hop-subtree is not active or (2) the number of packets left from this branch is less than  $W_v$ .

To illustrate the effectiveness of the sleep schedule, consider a linear network with  $N$  nodes. Assume that the energy spent by a node in transmit state  $T$ , receive state  $R$  and idle state  $I$  be equal (say,  $e$  Joules). Also, assume that energy spent in sleep state is negligible. We know that at most  $3N$  timeslots are required to finish the convergecast. If no sleep schedule is employed by the nodes then the total energy consumption in the network will be  $3N^2e$  Joules. If nodes employ sleep schedule then the  $N-i^{th}$  node, counting from the base station, will have its transceiver switched on for at most  $3i$  timeslots. As a result, the total energy consumption in the network is  $\frac{3N(N+1)e}{2}$  Joules. Hence, we conclude that *the sleep schedule results in about 50% energy conservation in linear networks*.

In multi-line networks employing sleep schedule results in relatively more energy savings. Consider a multi-line network of  $N$  nodes and  $k$  branches. Assume that the number of nodes in each branch is equal. Note that convergecast in this network can be accomplished in  $N$  timeslots (Theorem 2.5). If no sleep schedule is employed then the total energy consumption in the network will be  $N^2e$  Joules. Using the result for energy consumption in linear networks with sleep schedule, we conclude that energy spent in each branch during convergecast is  $\frac{3\frac{N}{k}(\frac{N}{k}+1)e}{2}$  Joules. As a result, the total energy spent in multi-line networks using sleep schedule is  $k * \frac{3\frac{N}{k}(\frac{N}{k}+1)e}{2}$  Joules. Hence, we conclude that *sleep schedule reduces energy consumption in multi-line networks by about a factor of  $\frac{1.5}{k}$* . Note that the more the number of branches the more saving in energy consumption by using the sleep schedule.

As we reduce the tree networks to multi-line networks where each branch corresponds to a one-hop-subtree the above result holds for tree networks too. As shown in the next section, we build a tree for convergecast scheduling in general networks. Hence, the same result can be extended for general networks.

### III. CONVERGECAST IN GENERAL NETWORKS

For networks of general topology, we construct a spanning tree and apply the convergecast scheduling algorithm described in Section II-C. However, such a schedule might not be feasible. Note that the scheduling algorithm for a tree network considers interference due to the edges that belong to the tree. In a general network, we need to consider the interference due to edges that are not part of the spanning tree. Given a spanning tree, we classify the edges of a network as *spanning tree edges* and *non-spanning tree edges*. As the names suggest, spanning tree edges (resp. non-spanning tree edges) are part of (resp. not part of) the constructed spanning tree.

Consider a linear branch of a spanning tree,  $s \leftarrow v_1 \leftarrow v_2 \dots \leftarrow v_n$ . Here, the edges  $(v_i, v_{i+1}) \forall i \in \{1, 2, \dots, n-1\}$  are the spanning tree edges. Consider any non-spanning tree edge  $(v_i, v_j)$ , where  $j \neq i-1 \wedge j \neq i+1$ . Note that if nodes  $v_i$  and  $v_j$  are assigned the same initial states then the edge  $(v_i, v_j)$  does not cause any collisions. However, if nodes  $v_i$  and  $v_j$  are assigned different initial states then the edge  $(v_i, v_j)$  will result in collisions. The reason is as follows: When one of the nodes in  $v_i$  and  $v_j$  has an initial state  $R$  and the other has an initial state  $T$ , it is easy to see that a collision occurs. On the other hand, say  $v_i$  has  $R$  as its initial state and  $v_j$  has  $I$  as its initial state. In such scenario a collision will not occur in the first timeslot. However, in the second timeslot  $v_i$  will move into state  $T$  (see Figure 2) and  $v_j$  will move into state  $R$ . As a result a collision will occur in the second timeslot. We refer to the edge  $(v_i, v_j)$  as a **conflicting edge**.

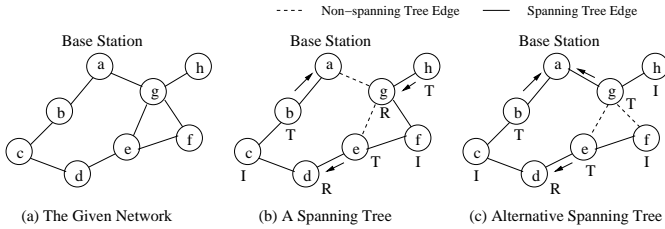


Fig. 6. Spanning Trees.

For example, consider the network shown in Figure 6(a). A spanning tree and the initial state assigned to each node by our convergecast scheduling algorithm can be seen in Figure 6(b). Notice that the edge  $(e, g)$  is a **conflicting edge**. Node  $g$  cannot receive from node  $h$  in the same timeslot that node  $e$  transmits. As a result the schedule determined by our convergecast algorithm is infeasible. Alternatively, consider a spanning tree shown in Figure 6(c). Here, nodes  $e$  and  $g$  are not part of the same linear branch and schedule determined by our algorithm will be feasible. In general, the following theorem holds.

**Theorem 3.1:** *No conflicting edges exist when the spanning tree constructed is a Breadth First Search (BFS) tree.*

**Proof:** (By contradiction) Let constructing a BFS tree result in a conflicting edge  $(v_i, v_j)$ . By definition the end points of a conflicting edge belong to a linear branch of the tree. Without loss of generality let the node  $v_i$  be the node closer (in hop-count) to the root. As the edge  $(v_i, v_j)$  exists in the network,  $(v_i, v_j)$  should be part of the BFS tree [3]. However, by the definition of a conflicting edge,  $(v_i, v_j)$  has to be a non-spanning tree edge. A contradiction. ■

As a BFS tree guarantees that conflicting edges do not exist, we propose to construct a BFS tree for convergecast-scheduling in general networks. Note that a BFS tree is also a shortest path tree if all the edges are assigned unit weight. Any existing distributed shortest path tree or BFS tree construction algorithm can be used in the initialization phase. After the BFS tree is constructed, each one-hop-subtree can be scheduled one after another. Such a scheduling mechanism requires at most  $3N - 2$  timeslots (Theorem 2.2). Multiple one-hop-subtrees can be scheduled in parallel to obtain a convergecast schedule using relatively less number of timeslots. However, interference between nodes belonging to different one-hop-subtrees has to be considered.

Two one-hop-subtrees, say  $A$  and  $B$ , are said to be *interfering* if there exist a non-spanning tree edge  $(a, b)$  such that  $a$  is in  $A$  and  $b$  is in  $B$ ;  $(A, B)$  is referred to as a *conflicting-subtree-pair* and the edge  $(a, b)$  is referred to as an *interfering-edge*. If multiple one-hop-subtrees are scheduled simultaneously, interfering-edges might result in collisions. Hence, in a timeslot  $t$  we schedule an **eligible one-hop-subtree** which has the maximum number of packets left and does not interfere with one-hop-subtrees scheduled in timeslots  $t - 2$  and  $t - 1$ .

During the initialization, nodes announce their one-hop-subtree ID to all their neighbors. As a result, nodes can learn about existence of conflicting-subtree-pairs and report to the base station. For example, if node  $a$  in one-hop-subtree  $A$  discovers that its neighbor  $b$  belongs to a different one-hop-subtree  $B$  then  $a$  reports the conflicting subtree pair  $(A, B)$  to the base station. After the base station receives all the conflicting subtree pairs, a *conflict map*  $M$  is disseminated to each node in the network;  $M(A, B) = 1$  if  $(A, B)$  is a conflict-subtree-pair and  $M(A, B) = 0$  otherwise. Using the conflict map  $M$ , each node independently schedules its transmissions. Note that for general networks, in addition to the information for tree networks, each node has to know the conflict map  $M$  at the initialization phase. However, again, no node knows the global connectivity of the network.

The line 1 of the pseudo code shown in Table I is to be modified as: Let  $\mathcal{J} = \{i | t_i \geq t\}$  be the set of active one-hop-subtrees at timeslot  $t$  and  $\mathcal{L} = \{i | t_i < t \wedge \forall j \in \mathcal{J}, M(i, j) = 0\}$ . We will refer to this filtering (mechanism of not scheduling conflicting-subtree-pairs in parallel) as **collision resolution**. As this filtering mechanism is conservative, the resulting schedule might not be optimal. However, we show in simulations that for a network of  $N$  nodes about  $1.5N$  timeslots are actually needed using our convergecast algorithm. This is about half of  $3N$ , the upper bound on the number of timeslots required by our algorithm.

### IV. PERFORMANCE EVALUATIONS

For performance evaluation we conducted several simulation experiments using Rmase [19]. Rmase is a wireless sensor network simulator built on Prowler [16].

**Radio and MAC Model:** The default deterministic radio model in Prowler is used in our simulation. All the communication links in the network are symmetric and deterministic. Packets are lost only when there is a collision. The data rate of wireless links is set to be 40Kbps. The maximum length of a packet is fixed as 960 bits. All algorithms except

our convergecast scheduling algorithm use the default CSMA MAC implemented in Prowler. Our convergecast scheduling algorithm uses CSMA during initialization. Nodes switch to TDMA after the initialization. The duration of each timeslot is set to 1/40 seconds, which is slightly greater than the time required to transmit one packet.

*Performance Metrics:* We use the following metrics of evaluation: (1) **Latency** is a measure of the time taken by a packet to reach the base station from its source node. We report the average latency for convergecast given by  $\sum_i d_i/n$ . Here,  $n$  represents the number of packets received by the base station and  $d_i$  represents the latency of the  $i^{th}$  packet. (2) **Throughput** of a network measures the number of data packets received at the base station per second. (3) **Delivery ratio** measures the fraction of packets successfully received by the base station.

*Experimental Set Up:* We considered a square sensor field of  $4 \times 4$  square-unit area. Initially, nodes were placed on points of a uniform grid. The coordinates of each node were shifted by a uniform random distribution with range  $[-0.5, 0.5]$ . A node which was closest to the center of the sensor field was designated as the base station. Networks with 25, 36, 49, 64, 81 and 100 nodes were considered. Note that increasing the number of nodes deployed in the same area increases the network density. We implemented two variants of our convergecast scheduling algorithm. These variants are distinguished based on whether nodes employ collision resolution (see Section III) between different subtrees or not. Note that when collision resolution is not employed some packets might be lost due to collisions. We implemented the following convergecast algorithms to compare with the performance of our convergecast algorithm:

**Directed Flooding:** Directed flooding [11], [17] uses *hop-count* to flood the packets towards the base station. Duplicate packet transmissions are used to increase the reliability.

**Radial Coordination:** In radially coordinated convergecast [13] each node waits for an additional time based on the *hop-count* before transmitting.

### A. Simulation Results

Each simulation experiment was repeated for ten times. Average values of metrics under consideration are reported. We first report the number of timeslots required by our scheduling algorithm. Then, we present the results comparing our convergecast scheduling algorithm with directed flooding and radial coordination for convergecast.

1) *Number of Timeslots:* The number of timeslots used to complete convergecast by our scheduling algorithm can be seen in Figure 7. It can be noticed that the number of timeslots required grows linearly with the number of nodes. The variants without collision resolution require 50% less timeslots comparatively. However, not all packets are guaranteed to arrive at the base station without employing conflict resolution. An interesting observation is that *the number of timeslots required by both the variants of our algorithm is less than  $1.5N$* , which is significantly less than the upper bound of  $3N$ .

2) *Performance Comparisons:* Figure 8 compares the performance of all the four convergecast mechanisms under consideration. In these simulations every node generated exactly one data packet at the beginning. It can be noticed that both the

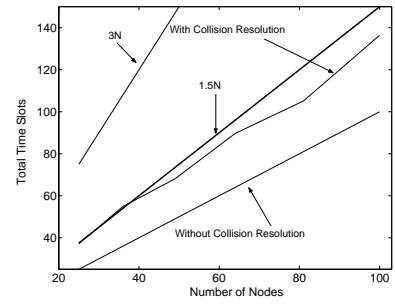


Fig. 7. Number of Timeslots Used for Convergecast

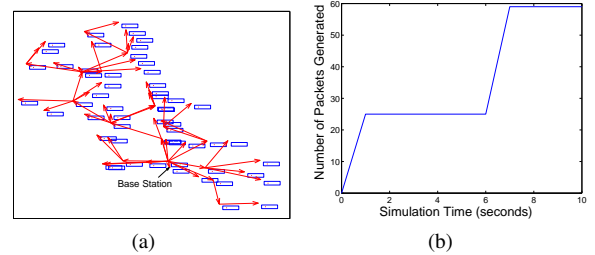


Fig. 10. (a) Node Distribution in Shooter Localization Experiment (b) Event traces for the first 10 seconds.

scheduled convergecast variants outperform directed flooding and radial coordination on all the three metrics.

When collision resolution was employed, almost 100% of the packets were received by the base station (see Figure 8(a)). An important observation to be noticed is that *even without collision resolution the delivery ratio is significantly higher (about 85%)*. We offer the following explanation: The nodes that interfere and belong to two different one-hop-subtrees might not be active simultaneously (see Section III). In addition, not all nodes of interfering one-hop-subtrees interfere with each other. As a result, not all simultaneous transmissions along the interfering subtrees result in collisions.

From Figure 8(b), it can be observed that the scheduled variant without collision resolution has the highest throughput. Hence, based on the application requirements, one can trade-off delivery ratio for higher throughput by not using the collision resolution mechanism. The convergecast scheduling algorithm results in low latencies in comparison with directed flooding and radial coordination (see Figure 8(c)).

We have repeated all the above experiments by varying the number of nodes while ensuring that the network density remained same. Similar trends were observed.

### B. Shooter Localization

In order to evaluate the convergecast algorithms in a more realistic setting, we considered the shooter localization [17] application. For shooter localization, the acoustic measurements of the sensor nodes that are above a certain threshold are collected at the base station. These measurements are used by the base station to determine location from where the shot was fired. The network topology (Figure 10(a)) and the event trace data (Figure 10(b)) are obtained from the actual experiments on the hardware platform. We conducted simulation experiments using this data. From Figure 9 we can conclude that the delivery ratio of both the scheduled

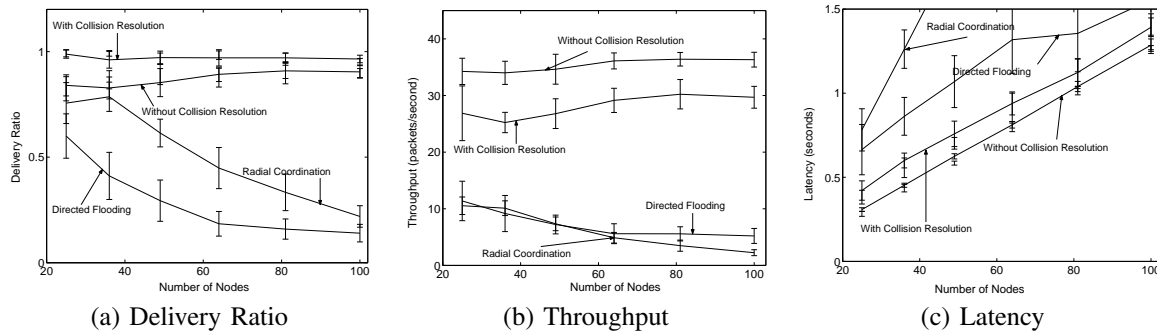


Fig. 8. Performance Comparisons: Networks with Varying Number of Nodes

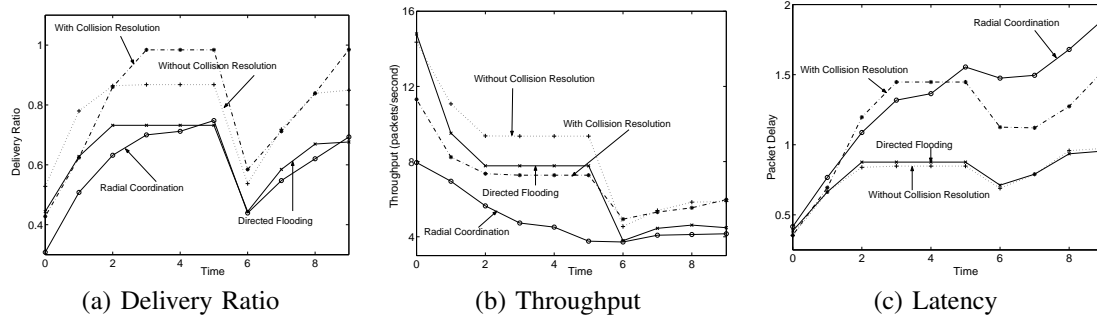


Fig. 9. Performance Comparisons: Shooter Localization

convergecast variants is better than the directed flooding and radial coordination. The convergecast variant without collision resolution has the highest throughput and the lowest latency. We conclude that our scheduling algorithm performs better even in realistic scenarios.

## V. CONCLUSIONS AND FUTURE WORK

We proposed a minimal time distributed convergecast scheduling algorithm for sensor networks. Our convergecast scheduling algorithm requires at most  $3N$  timeslots, where  $N$  is the number of nodes in the network. Our simulation results show that the actual number of required timeslots is significantly less than  $3N$ . In specific, we observed the number of timeslots required is about  $1.5N$ . We also proved that nodes employing our convergecast scheduling algorithm need not buffer more than two packets at any instance. Also, more than 50% of the energy can be saved by using the proposed sleep schedule. As sensor nodes have limited memory and energy these results are of importance.

In the current work, we proposed to construct a breadth first search tree for convergecast scheduling. However, construction of a lifetime optimal tree remains an open issue which we are interested in addressing. Performance evaluation of the proposed scheme in presence of intermittent connectivity and channel errors is in progress. Implicit acknowledgement schemes and extensions to handle asymmetric links are interesting directions of future research.

## REFERENCES

- [1] Kesselman A. and Kowalski D. Fast distributed algorithm for convergecast in ad hoc geometric radio networks. In *The 2<sup>nd</sup> Annual Conference on Wireless On demand Network Systems and Services*, 2005.
- [2] Florens C. and McEliece R. Packet distribution algorithms for sensor networks. In *IEEE INFOCOM*, 2003.
- [3] Rivest R. L. Cormen T. H., Leiserson C. E. and Stein C. *Introduction to Algorithms*. MIT Press and McGraw-Hill.
- [4] Bianchi G. Performance analysis of the ieee 802.11 distributed coordination function. *IEEE JSAC*, 18(3), March 2000.
- [5] Gandham S., Dawande M. and Prakash R. Link Scheduling in Sensor Networks: Distributed Edge Coloring Revisited. *Proceedings of IEEE INFOCOM*, March 2005.
- [6] Hajek B. and Sasaki G. Link Scheduling in Polynomial Time. *IEEE Transactions on Information Theory*, 34:910–917, Sept. 1988.
- [7] Ju Wang Hongsik Choi and Esther A. Hughes. Scheduling on Sensor Hybrid Network. In *IEEE ICCCN*, 2005.
- [8] Crossbow Technologies Inc. [www.xbow.com](http://www.xbow.com).
- [9] Krumke S. O., Marathe M. V. and Ravi S.S. Models and Approximation Algorithms for Channel Assignment in Radio Networks. *Wireless Networks*, 7:575–584, 2001.
- [10] Gasieniec L. and Potapov I. Gossiping with unit messages in known radio networks. In *IFIP TCS*, 2002.
- [11] M. Maroti. Directed flood-routing framework. In *5th International Middleware Conference*, 2004.
- [12] Goodrich M.T. and Tamassia R. *Data Structures and Algorithms in Java*. John Wiley and Sons, Inc., second edition, 2001.
- [13] Huang Q. and Zhang Y. Radial coordination for convergecast in wireless sensor networks. In *First IEEE Workshop on Embedded Networked Sensors*, 2004.
- [14] Ramanathan S. A Unified Framework and Algorithm for Channel Assignment in Wireless Networks. *Wireless Networks*, 5:81–94, 1999.
- [15] Ramaswami R. and Parhi K.K. Distributed Scheduling of Broadcasts in a Radio Network. *IEEE INFOCOM*, 1989.
- [16] G. Simon. Probabilistic wireless network simulator. <http://www.isis.vanderbilt.edu/projects/nest/proowler/>.
- [17] Balogh Gy. Kusy B. Ndas A. Pap G. Sallai J. Simon Gy., Ldeczi A. and Frampton K. Sensor network-based countersniper system. In *Sensys*, 2004.
- [18] Ye W., Heidemann J. and Estrin D. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. *INFOCOM*, 2002.
- [19] Y. Zhang, M. Fromherz, and L. Kuhn. Rmase: Routing modeling application simulation environment. <http://www.parc.com/era/nest/Rmase>.