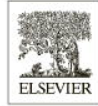


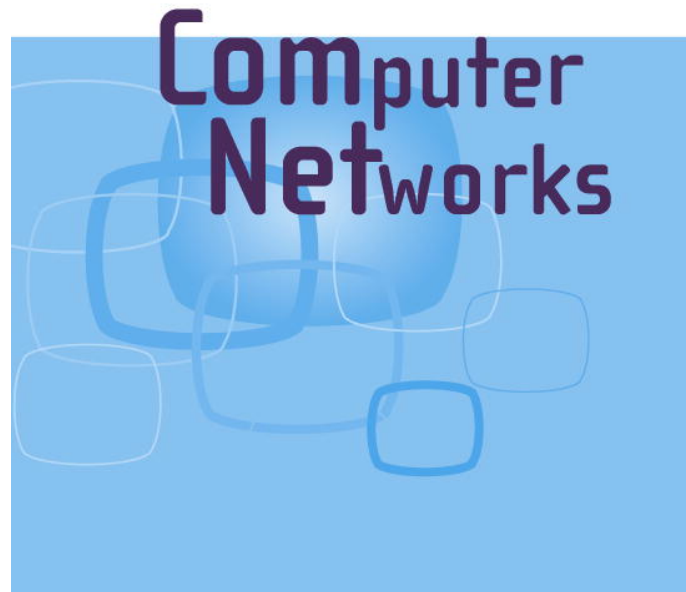
Provided for non-commercial research and education use.
Not for reproduction, distribution or commercial use.



Volume 52 Issue 3

22 February 2008

ISSN 1389-1286



This article was published in an Elsevier journal. The attached copy is furnished to the author for non-commercial research and education use, including for instruction at the author's institution, sharing with colleagues and providing to institution administration.

Other uses, including reproduction and distribution, or selling or licensing copies, or posting to personal, institutional or third party websites are prohibited.

In most cases authors are permitted to post their version of the article (e.g. in Word or Tex form) to their personal website or institutional repository. Authors requiring further information regarding Elsevier's archiving and manuscript policies are encouraged to visit:

<http://www.elsevier.com/copyright>



Distributed time-optimal scheduling for convergecast in wireless sensor networks [☆]

Shashidhar Gandham ^{a,*}, Ying Zhang ^b, Qingfeng Huang ^b

^a *xG Technology Inc., Wireless MAC Protocol Group, 240 South Pineapple Avenue, Suite, Sarasota, FL 34236, United States*

^b *Palo Alto Research Center Inc., Formerly Xerox PARC, United States*

Received 21 February 2007; received in revised form 16 October 2007; accepted 24 October 2007

Available online 7 November 2007

Responsible Editor: Dr. I.F. Akyildiz

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. Initially, we consider a simple version of the problem wherein every node generates exactly one packet. We provide a *distributed* scheduling algorithm for tree networks that requires at most $\max(3n_k - 1, N)$ timeslots for convergecast, where n_k represents the maximum number of nodes in any subtree and N represents the number of nodes in the network. We propose a distributed convergecast scheduling algorithm for general networks that requires at most $3N$ timeslots. 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 propose simple modifications to apply our algorithm when (i) the convergecast is initiated by the base station, (ii) nodes generate multiple packets and (iii) the wireless channel propagation characteristics are not ideal. Simulation results for a real application scenario show that our convergecast scheduling algorithm performs significantly better than existing convergecast algorithms.

© 2007 Elsevier B.V. All rights reserved.

Keywords: Wireless sensor networks; Convergecast; Scheduling; TDMA

1. 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 [8]. Event-driven data collection as in the sniper localization [23] also results in convergecast. In these applications all the packets generated in the network have to reach a base station either

[☆] Preliminary version of this paper appeared in IEEE ICDCS 2006.

* Corresponding author. Tel.: +001 972 679 4907.
E-mail addresses: shashig@xgtechnology.com (S. Gandham),
yzhang@parc.com (Y. Zhang), qhuang@parc.com (Q. Huang).

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 more effectively.

It is well-known that collisions present a major challenge in convergecast when contention-based MAC protocols like CSMA [1] 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 [6]. In addition, retransmissions drain the scarce energy reserves of sensor nodes. Radially coordinated transmission [18] 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:

- Distributed convergecast scheduling algorithm for tree networks that requires at most $\max(3n_k - 1, N)$ timeslots. Here, N represents the number of nodes in the network and n_k represents the maximum number of nodes in any subtree.
- Distributed convergecast scheduling algorithm that requires at most $3N$ timeslots in any network. Simulation results show that the number of timeslots required is about $1.5N$.
- 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.
- A sleep schedule for nodes to conserve energy. The sleep schedule reduces the energy consumption by at least 50% in comparison to scenario

wherein nodes are on all the time. This result is important considering the limited amount of energy available at sensor nodes.

The above results are based on the assumption that every node generates exactly one packet. We propose simple modifications to apply our algorithm when (i) the convergecast is initiated by the base station, (ii) nodes generate multiple packets and (iii) the wireless channel propagation characteristics are not ideal.

1.1. 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 [14,19,21]) or to the edges (link scheduling [7,9,19,20,24]). 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. For example, the link scheduling algorithm proposed in [7] requires at least $2(\delta + 1)$ timeslots for networks with a maximum degree of δ . Employing such a schedule requires at least $2(\delta + 1)$ timeslots to forward a packet at each hop. As the longest path in a network with N nodes can have at most $N - 1$ hops, a maximum of $2(N - 1)(\delta + 1)$ timeslots are required for convergecast. We show that the maximum number of timeslots required for convergecast is linear in number of nodes and is *independent* of the maximum degree.

Florens and McEliece have considered the problem of scheduling for packet distribution in sensor networks [4]. 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 [4] can be employed for convergecast. However, the proposed algorithm is centralized where the schedule is computed at the base station. Choi and Hughes [10], also propose a centralized convergecast scheduling algorithm. Our algorithm is *distributed* where each node computes its own schedule after the initialization phase.

Kesselman and Kowalski consider the problem of convergecast in ad hoc geometric networks [2].

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.

The remaining part of this paper is organized as follows. System model considered for this work and the problem formulation are described in Section 2. Convergecast scheduling algorithms for tree networks is presented in Section 3. Our scheduling algorithm for general networks is described in Section 4. Convergecast in other operational scenarios is discussed in Section 5. Section 6 shows simulation results and compares the performance of our convergecast scheduling algorithm with existing algorithms. Section 7 concludes the paper.

2. System model and problem formulation

2.1. Assumptions

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. Initially, we assume that transmission range of a node is equal to its interference range, i.e., during initial analysis we assume protocol interference model. We relax this assumption in Section 5.3.

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 [12,13] of the data.

2.2. Problem formulation

Let $G(V, E)$ represent the sensor network where (i) $V - \{s\}$ represents the set of sensor nodes, (ii) s represents the base station and (iii) $E \subset V \times V$ represents the set of wireless links. Let $p_0(v)$ be the number of packets that originate at a node v , and $p_j(v)$ be the number of packets at node v at the end of timeslot j . Let $f_j(u, v) \in \{0, 1\}$ represent the action of nodes in timeslot j ; $f_j(u, v) = 1$ if node u transmits a packet to v in timeslot j and $f_j(u, v) = 0$ otherwise. Note that $u \neq v$. Let N_u represent the one-hop neighbors of node u and l be the number of timeslots required to complete convergecast. The convergecast scheduling problem can be formulated as an Integer Linear Program (ILP) shown below:

Minimize l

Subject to

$$p_l(s) = \sum_{u \in V} p_0(u) \tag{1}$$

$$\sum_{w \in N(v)} \sum_{v \in N(u)} f_j(v, w) \leq 1, \quad \forall u \in V, \forall j \in \{1, \dots, l\} \tag{2}$$

$$\sum_{v \in N(u)} f_j(u, v) \leq 1, \quad \forall u \in V, \forall j \in \{1, \dots, l\} \tag{3}$$

$$\sum_{v \in N(u)} f_j(v, u) + f_j(u, v) \leq 1, \quad \forall u \in V, \forall j \in \{1, \dots, l\} \tag{4}$$

$$p_j(u) + \sum_{j \in N(u)} f_j(u, v) = p_{j-1}(u), \quad \forall u \in V, \forall j \in \{1, \dots, l\} \tag{5}$$

$$p_j(u) - \sum_{j \in N(u)} f_j(v, u) = p_{j-1}(u), \quad \forall u \in V, \forall j \in \{1, \dots, l\} \tag{6}$$

$$f_j(u, v) \in \{0, 1\}, \quad \forall u \in V \ \& \ \forall v \in V \tag{7}$$

The objective is to minimize l , the total number of timeslots used for convergecast. Constraint (1) ensures that all the packets are collected at the base station by the end of convergecast. Constraint set (2) states that at most one neighbor of a node can transmit in a timeslot. Thus, addressing the hidden terminal problem. Similarly, constraint set (3) states that a node transmits to at most one neighbor in a timeslot. Restricting a node from transmitting and receiving in the same timeslot is accomplished by constraint set (4). Constraint sets (5) and (6) are a direct result of conservation of messages.

The above ILP can be solved using tools like CPLEX. Typically, solutions to ILP's have exponential running time. Moreover, such a solution would be centralized in nature and will not be scalable. The convergecast scheduling problem is known to be NP-hard. In [10] authors show that the problem can be reduced graph partitioning problem, a well-known NP-hard problem.

In Sections 3 and 4 we present our distributed heuristic for convergecast scheduling. To simplify the discussion in remaining sections, we assume that every node generates exactly one packet. Convergecast scheduling when nodes generate multiple packets is addressed in Section 5.2.

3. 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.

3.1. Linear networks

Consider a linear network as shown in Fig. 1a. 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 Fig. 2.

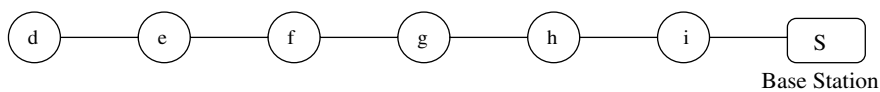
For example, consider the node i and its schedule as shown in Fig. 1b. The initial state of the node i is T, i.e., i transmits in the first timeslot ($f_1(i) = s$). In the next timeslot i moves to the state I, i.e., i remains idle in the second timeslot. In the third timeslot the node i moves into state 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. Nodes continue to move from one state to another until the completion of convergecast.

Fig. 1b shows the state of nodes and the number of packets in the buffer of each node at the end of timeslots 1–4. 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 scheduled in accordance to our algorithm 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 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 Fig. 1a 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 three 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. \square



(a) A Linear Network.

d	e	f	g	h	i	State	Timeslot
I	I	I	I	I	I		0
1	1	1	1	1	1	Num. Pkts	
R	I	T → R	I	I	T →	State	1
1	1	0	2	1	0	Num. Pkts	
T → R	I	I	T → R	I	I	State	2
0	2	0	1	2	0	Num. Pkts	
I	T → R	I	I	T → R	I	State	3
0	1	1	1	1	1	Num. Pkts	
R	I	T → R	I	I	T →	State	4
0	1	0	2	1	0	Num. Pkts	

(b) Convergecast Schedule.

Fig. 1. Convergecast scheduling in linear networks.

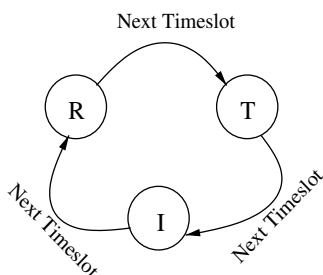


Fig. 2. State transition for convergecast scheduling.

A similar analysis was used in [15] to obtain lower bounds for gossiping in a linear wireless network.

Theorem 2. *The convergecast scheduling algorithm requires $3N - 2$ 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 , i.e., if the network has K nodes then the convergecast can be completed in $3K - 2$ timeslots.

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 to 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 [11] have 4K bytes of RAM. Hence, the following result pertaining to our convergecast algorithm is significant.

Theorem 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. \square

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.

3.2. Multi-line networks

In linear networks, the base station receives exactly one packet in every three timeslots. In multi-line networks (see Fig. 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 *parallelly* 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 imple-

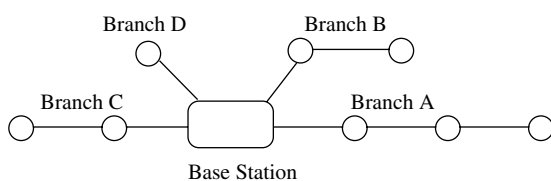


Fig. 3. A multi-line network.

mented at *each node* along with the state transitions described in Section 3.1.

During the initialization, each node obtains its hop-count to the base station and determines its initial state as described in Section 3.1. 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. The following simple algorithm can be used to collect this information: (1) the last node in each branch informs its predecessor or parent node that there is only one node succeeding it. (2) Every other node in the network waits until he gets this information from its successor or child node. Increments the value by one and forwards it to its parent. An alternative approach to obtain the number of nodes in each branch is to make the last node in each branch report its hop-count to 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 Fig. 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 a packet to the base station 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.

For example, consider the multi-line network shown in Fig. 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 in Fig. 4a. The *Pkts Left* field is used to track the number of packets remaining in each branch. The *Last*

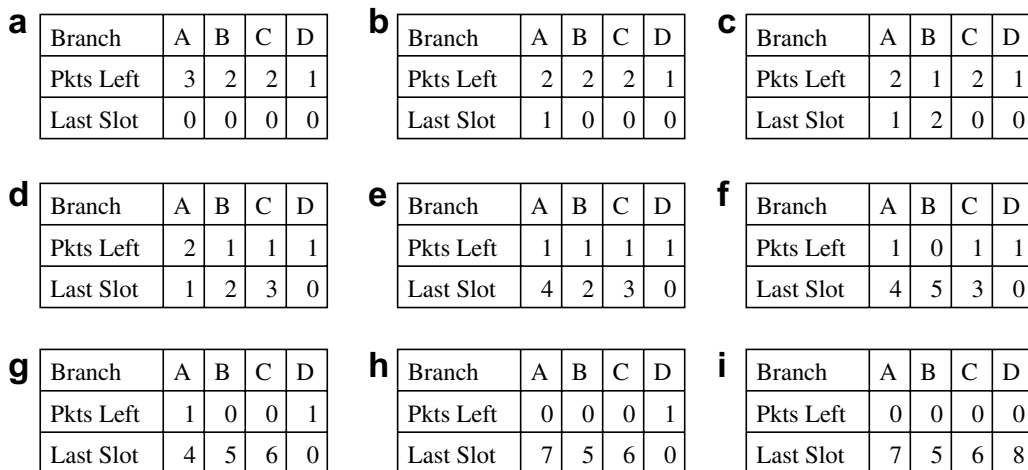


Fig. 4. Convergecast schedule for multi-line networks.

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 Fig. 4b. 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 Fig. 4c. In the current example, eight timeslots are required to complete the convergecast. Fig. 4d–(i) show the contents of the schedule table at the end of timeslot 3–8. The pseudocode that has to be implemented on each node for the convergecast scheduling is shown in Table 1.

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 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

Table 1
Scheduling algorithm for multi-line network

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

Theorem 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. \square

Theorem 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 $n_k \geq n_{k-1} \geq n_{k-2} \cdots \geq n_1$. 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, 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. 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 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 $n_k \leq \lceil \frac{1}{2}(\sum_{i=1}^{k-1} n_i) \rceil$. \square

3.3. Tree networks

In 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 Fig. 5a 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.

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 3.2, that a nodes 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

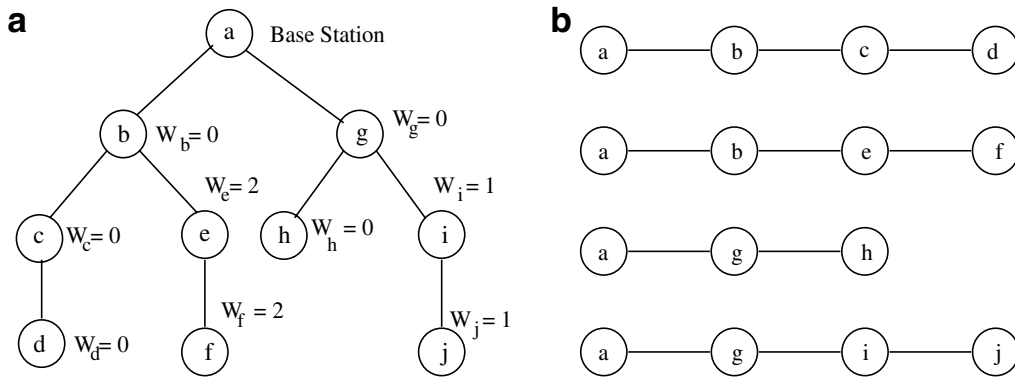


Fig. 5. Reduction of a tree network into linear branches.

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 [17] initiated by the base station can be used. Fig. 5 shows the value of W_v at each node.

Note that all the one-hop-subtree (e.g. subtrees rooted at nodes b and g in Fig. 5) can be scheduled parallelly as in a multi-line network (see Section 3.2). Each one-top-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 pseudocode for scheduling in tree network is the same as the one for multi-line networks, except that the active condition for a node (line 5 in Table 1) has to be modified to: $t < t_{\mathcal{F}}$ and $\Delta n_{\mathcal{F}} \geq W_v$. Where, $\Delta n_{\mathcal{F}} = n_{\mathcal{F}}^0 - n_{\mathcal{F}}$ is the number of packets forwarded by one-hop-subtree \mathcal{F} 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 5). Here, n_k represents the number of nodes in the largest one-hop-subtree. For example, the network shown in Fig. 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.

3.4. Sleep schedule for energy conservation

It is well known that idle listening consumes a significant amount of energy in sensor networks [25]. 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 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 $\frac{3N(\frac{N}{k}+1)e}{2}$ Joules. As a result, the total energy spent in multi-line networks using sleep schedule is $k * \frac{3N(\frac{N}{k}+1)e}{2}$ 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.

4. Convergecast in general networks

For networks of general topology, we construct a spanning tree and apply the convergecast scheduling algorithm described in Section 3.3. 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, $s \leftarrow v_1 \leftarrow v_2 \cdots \leftarrow v_n$. Here, the edges $(v_i, v_{i+1}) \forall i \in \{1, 2, \dots, n - 1\}$ are the spanning tree edges. Con-

sider 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 Fig. 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*.

For example, consider the network shown in Fig. 6a. A spanning tree and the initial state assigned to each node by our convergecast scheduling algorithm can be seen in Fig. 6b. 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 Fig. 6c. 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 6. *No conflicting edges exist when the spanning tree constructed is a Breadth First Search (B.F.S) tree.*

Proof. (By contradiction) Let constructing a B.F.S 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

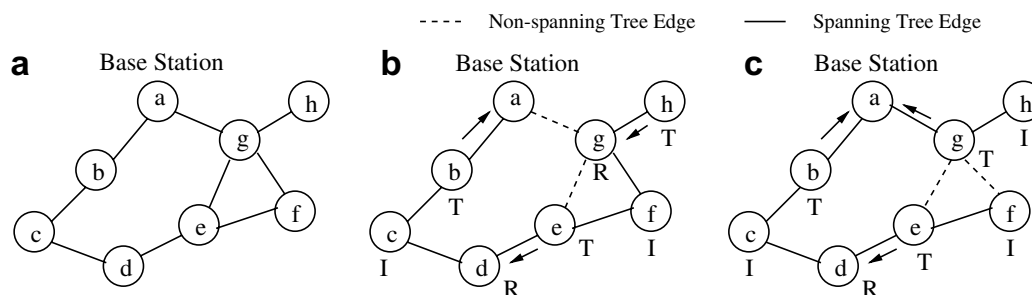


Fig. 6. Spanning trees.

the network, (v_i, v_j) should be part of the B.F.S tree [5]. However, by the definition of a conflicting edge, (v_i, v_j) has to be a non-spanning tree edge. A contradiction. \square

As a B.F.S tree guarantees that conflicting edges do not exist, we propose to construct a B.F.S tree for convergecast scheduling in general networks. Note that a B.F.S tree is also a shortest path tree if all the edges are assigned unit weight. Any existing distributed shortest path tree or B.F.S tree construction algorithm can be used in the initialization phase. After the B.F.S 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). Multiple one-hop-subtrees can be scheduled parallelly 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 pseudocode shown in Table 1 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 parallelly) 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.

5. Convergecast in other scenarios

We started our discussion assuming a simple scenario wherein every node generated exactly one data packet and the convergecast was initiated by the sensor nodes. In addition, we considered an ideal wireless channel propagation characteristics. In this section we show that our convergecast scheduling algorithm is applicable even when these assumptions do not hold.

5.1. Base station initiated convergecast

In certain applications, the base station might start the convergecast process by sending a *convergecast initiation message* to all its one-hop neighbors. Assume that every node in the network is actively listening to the wireless medium by using the *promiscuous* mode. The one-hop neighbors of the base station can schedule their transmissions to the base station on receiving a convergecast initiation message. On listening to the transmissions of one-hop neighbors, the two-hops neighbors of the base station can know that the convergecast has began and they need to start scheduling their transmissions. Similarly, any node in the network can start scheduling its transmission when it notices that its parent in the B.F.S spanning tree has started transmitting packets.

For the base station initiated convergecast, we propose the following modifications to our scheduling algorithm: (i) the direction of transitions in the state transition diagram shown in Fig. 2 is reversed. For example, a node in state R moves into state I in the next timeslot instead of moving into state T. Similarly, a node in state I (resp. state T) moves into state T (resp. state R) in the next timeslot. (ii) All the nodes are assigned the same initial state I. However, a node starts changing its state to T only after it overhears the first transmission from its parent.

Consider first the linear network shown in Fig. 1a. The transmission schedule for base station initiated convergecast in this network can be seen in Fig. 7. Initially every node is in state I and actively listens for transmissions from its parent, a node one-hop closer to the base station. On noticing that its parent has transmitted a packet, a node will move into state T and transmits its packet in the next timeslot. Hereupon, the node changes its state after every timeslot according to the modified state transition diagram. From Fig. 7, we can notice that it takes five timeslots for the last node *d* to notice that convergecast has began. In the sixth timeslot, *d* transmits.

In general, the last node in a linear network transmits only in the *N*th timeslot. After the *N*th timeslot, the effective length (considering only nodes which have to transmit) of the linear network reduces by one unit in every two timeslot. Hence, the base station initiated convergecast can be completed in linear networks using at most $3N - 2$ timeslots (after the base station transmits initialization message). An inductive argument, similar to the one used in Theorem 2, can be used to prove this statement. Notice that every node first transmits the packet that it has generated before receiving a packet from a neighboring node. Hence, in this case, at most one packet is buffered by a node at any instance of time.

In multi-line networks a node which is *h* hops away from the base station will come to know about the convergecast after $\lceil \frac{h}{3} \rceil$ packets are forwarded to

the base station from the corresponding branch. In tree networks a node, say v_i , learns about the convergecast after W_v (see Section 3.3) packets are forwarded by its one-hop-subtree. Where, v is the parent of node v_i and v waits until W_v packets are forwarded by its one-hop-subtree before becoming *active*. For scheduling in general networks we propose that node v_i updates its schedule table to reflect the all transmissions until W_v packets have left its one-hop-subtree. From the instance a node transmits for the first time, the scheduling algorithm is very similar to the one described for general networks in Section 4. As a result we conclude that at most $3N$ timeslots are required for base station initiated convergecast in general networks.

5.2. Nodes with multiple packets

In some applications nodes might generate more than one packet during each convergecast. For example, the message generated by some nodes might be longer than the maximum packet size the MAC data frame can carry. The maximum size of a packet is defined by the available data rate and duration of the timeslot. If the captured event (say an image in case of video sensors as in surveillance system for intruder detection) cannot be reported in a single packet then the event is reported using multiple packets. The source node knows exactly how many packets would be required to report that event. Such multi-packet possibilities also exist when each sensor node is equipped with more than

							Timeslot	
d	e	f	g	h	i			
I		I		I		I	Status	0
I		I		I		T	Status	1
I		I		I		T	-----▶	R
I		I		T		R		I
I		I		T		R		I
I		T		R		I		T
I		T		R		T	-----▶	R
T		R		I		T	-----▶	R
R		I		T		R		I
R		I		T		R		I
I		T		R		I		T
I		T		R		T	-----▶	R
T		R		I		T	-----▶	R
T		R		I		T	-----▶	R
T		R		I		T	-----▶	R

Fig. 7. Base station initiated convergecast in linear networks.

one sensing device and not all sensor reading can be packed into a single packet.

We present a simple modification to our convergecast scheduling algorithm to handle these scenarios. Here, we show that a linear network where nodes generate multiple packets can be reduced into an equivalent tree network where every node generates exactly one packet.

Consider the linear network shown in Fig. 8a. Here, the node f generates three packets. All the other nodes generate exactly one packet. We add logical nodes f_0 and f_1 , each with one data packet, as neighbors of the node g . We set the number of packets generated by the node f to one. The resultant network can be seen in 8b. Note that for convergecast scheduling both the networks are equivalent. In the original network, shown in Fig. 8a, the node g has to receive and transmit six packets. In the reduced network (Fig. 8b) too the node g has to receive and transmit six packets.

We propose that nodes having multiple packets act as multiple logical nodes; one logical node per packet. As a result, the linear network reduces to a logical tree network. In the resulting logical tree network every node has one packet to be forwarded to the base station. The convergecast scheduling algorithm for tree networks proposed in Section 3.3 can be employed on the logical tree network. Note that the logical tree network has P nodes, where P is the number of packets in the original network. Hence, we can conclude that convergecast can be completed in at most $3P$ timeslots (see Sec-

tion 3.3). Our convergecast scheduling algorithm for general networks is based on scheduling in tree networks (Section 4). Hence, the current reduction can be used in general networks where nodes generate multiple packets. Note that the proposed method can be applied only in scenarios where every node is aware of number packets it will generate before the initialization phase.

5.3. Non-ideal radio characteristics

An inherent assumption we made about the wireless channel was that two nodes can interfere with each other only if they can communicate, i.e., the transmission range of a node is equal to its interference range. However, it is known that in certain scenarios the interference range of a node can be greater than its transmission range. For a detailed explanation of the phenomenon we refer the reader to [27].

To illustrate, consider the linear network with the initial states of the nodes as shown in Fig. 9. Though nodes i and g cannot communicate, transmissions of node i will result in interference at node g . In order to avoid such interference we introduce one more idle state I between state R and state I of our state transition diagram shown in Fig. 2. The resulting initial states of the nodes can be seen in Fig. 9. Here, at most $4N$ timeslots are required to complete convergecast.

Let D be the maximum number of communication links across which interference can span. In

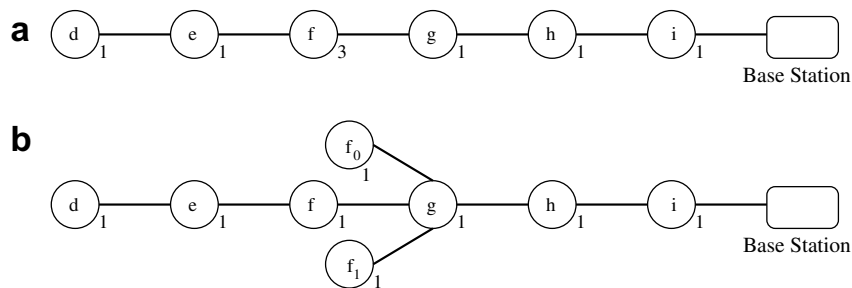


Fig. 8. Multi-packet network.

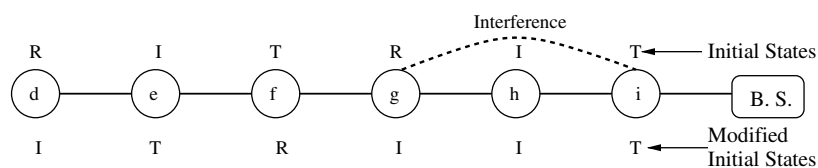


Fig. 9. Non-ideal radio propagation characteristics.

the current example nodes i and g are separated by two-hops. Hence, the value of D will be two. By an inductive argument, similar to the one used in [Theorem 2](#), we can show that *at most $(D + 2)N$ timeslots are required for convergecast in a linear network*. In multi-line networks, the value of D might vary in different branches. As a result, we propose the following modification to our scheduling algorithm for multi-line networks: *a branch with maximum value of $(D + 2)P'$ will be scheduled in a timeslot*. Here, P' is the number of packets left in a branch. A similar modification can be used for scheduling in tree networks. In general networks, a one-hop-subtree that has maximum value of $(D + 2)P'$ and that does not interfere with currently active one-hop-subtrees will be scheduled in a timeslot. *We conclude that at most $(D + 2)N$ timeslots are required for convergecast in any network*.

In this discussion we assume that the value of D is known. An upper bound on D can be obtained off-line based on the radio propagation characteristics. In future, we are interested in developing a distributed algorithm to determine D on-line.

6. Performance evaluations

To evaluate the performance of our convergecast scheduling algorithm we conducted several simulation experiments using Rmase [26]. Rmase is a wireless sensor network simulator built on Prowler [22].

6.1. Radio and MAC model

Prowler provides a radio propagation model and the CSMA MAC protocol for Mica nodes [3]. The radio model measures the received signal strength at each node. All the simultaneous transmissions in the network are taken into account for the measurements. A packet can be received successfully only when the received signal strength is greater than a pre-determined threshold. For example, a node j can receive a packet from node i if the received strength $P_{\text{rec}}(i, j) > \Delta$; where $\Delta > 0$ is the threshold. A collision occurs at a node when two overlapping transmissions are received with a signal strength greater than Δ . Unless stated otherwise, 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. We also report simulations where the radio model is based

on the SINR. In SINR-based radio model links could be asymmetric and non-deterministic.

The data rate of wireless links is set to be 40 Kbps. 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 s, which is slightly greater than the time required to transmit one packet of size 960 bits.

6.2. Performance metrics

We use the following metrics to evaluate the performance of our convergecast scheduling algorithm:

Latency: 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. Note that latency accounts for the number of hops traversed, the length of transmission queues, the random delays at the MAC layer, and additional delays added to minimize collisions.

Throughput: Throughput of a network measures the number of data packets received at the base station per second. Note that the concept of throughput is identical to good-put since only desired data packets are counted.

Delivery ratio: The delivery ratio measures the fraction of packets successfully received by the base station. It is defined as a ratio between total number of data packets received by the base station and the total number of data packets generated.

6.3. Experimental set up

We considered a square sensor field of 4×4 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. [Fig. 10](#) shows connectivity of networks with 25 and 49 nodes. We implemented two variants of our convergecast

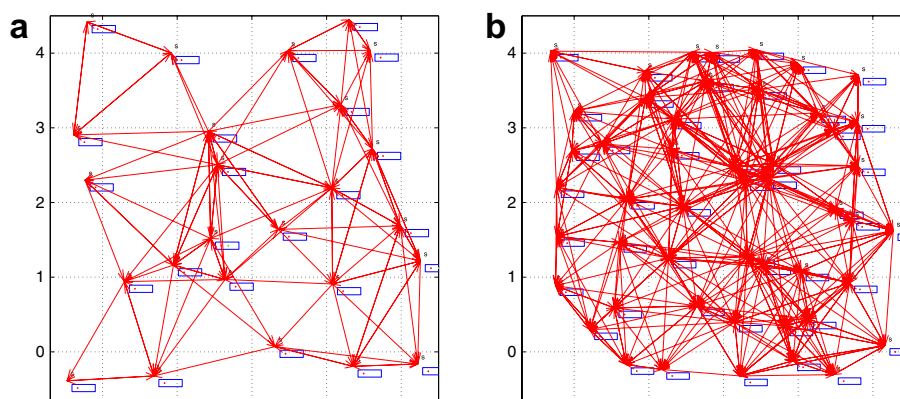


Fig. 10. Connectivity of networks with different densities. (a) Network with 25 nodes; (b) network with 49 nodes.

scheduling algorithm. These variants are distinguished based on whether nodes employ collision resolution (see Section 4) 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 scheduling algorithm:

- *Directed flooding*: Directed flooding [16,23] uses *hop-count* to flood the packets towards the base station. Duplicate packet transmissions are used to increase the reliability of packet delivery. Each packet might be transmitted up to three times.
- *Radial coordination*: In radially coordinated convergecast [18] each node waits for an additional time based on the *hop-count* before transmitting.

6.4. Simulation results

Each simulation experiment was repeated for 10 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.

6.4.1. Number of timeslots

The number of timeslots used to complete convergecast by our scheduling algorithm can be seen in Fig. 11a. 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 $3N$. As shown in Fig. 11b, the same observation is valid with the base station initiated convergecast.

6.4.2. Performance comparisons

Fig. 12 compares the performance of all the four convergecast mechanisms under consideration. In

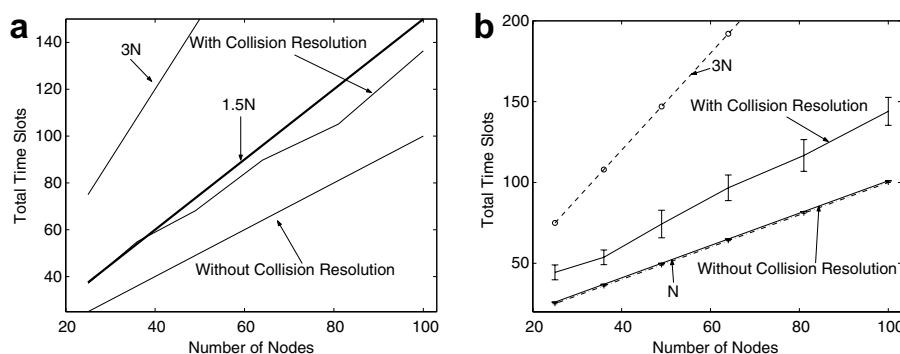


Fig. 11. Number of timeslots used by scheduled convergecast variants. (a) Scheduled convergecast; (b) base station initiated convergecast.

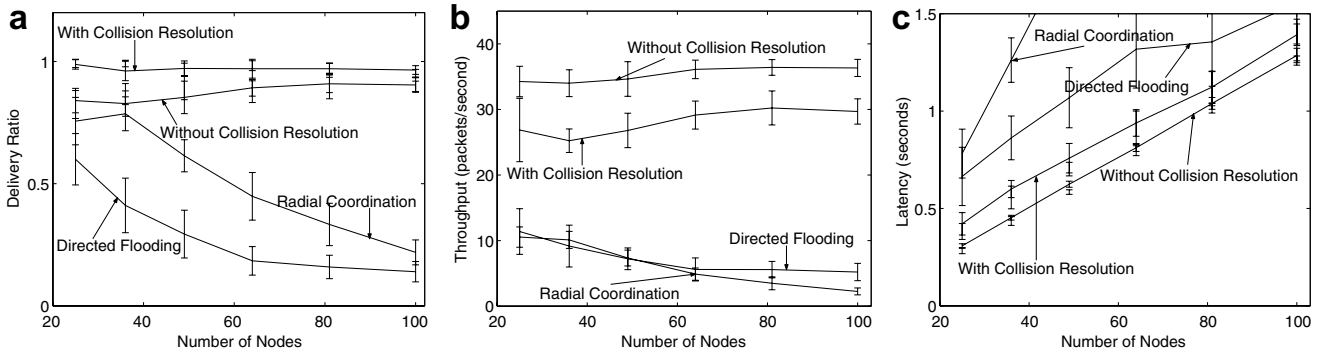


Fig. 12. Performance comparisons: networks with varying number of nodes. (a) Delivery ratio; (b) throughput; (c) latency.

these simulations every node generated exactly one data packet at the beginning. It can be noticed that both the 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 Fig. 12a). 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 4). 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 Fig. 12b, 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 Fig. 12c).

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.

6.4.3. SINR-based simulations

We repeat the above experiments using SINR-based radio model provided in Prowler. A detailed description of the radio model can be found in [22]. As can be noticed in 13a, the delivery ratio of all the four methods drops significantly. A corresponding drop in throughput and increase in latency can be noticed in Fig. 13a and b, respectively. Note that despite the drop in performance both the convergecast variants outperform directed flooding and radial coordination. Presence of asymmetric links and ignored interference instances explain degradation in the performance.

A simple link layer solution as explained here can be used to ignore asymmetric links for determining convergecast schedule. Every node, on deployment, learns about its neighbors and transmits its neighborhood list to all its neighbors. A node would consider a link during scheduling only if its node ID is in the neighborhood list of the corresponding neigh-

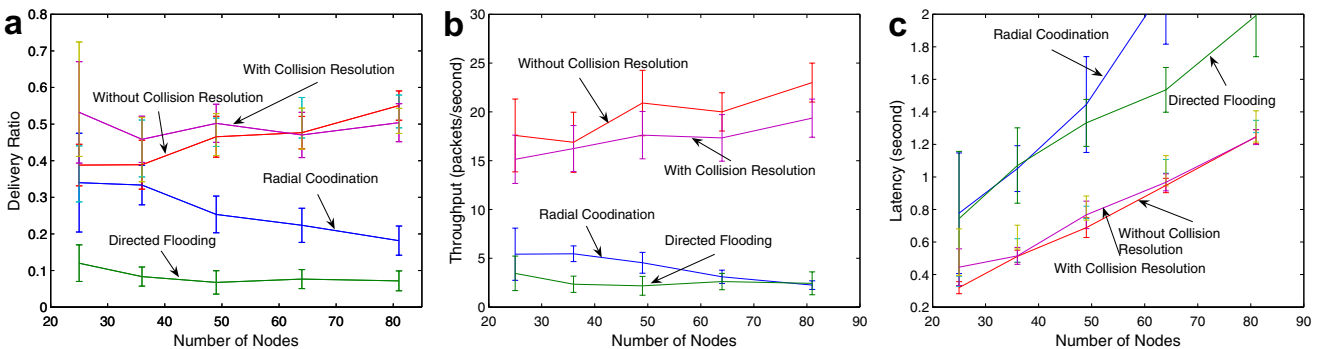


Fig. 13. Performance comparisons based on SINR simulations: networks with varying number of nodes. (a) Delivery ratio; (b) throughput; (c) latency.

bor. The solution proposed in Section 5.3 should improve the delivery ratio at the expense of increasing the latency and decreasing the throughput.

6.4.4. Varying percentage of source nodes

Our convergecast scheduling algorithm can be employed even when a fraction of nodes in the network generate packets. For performance comparison in these scenarios, we simulated a sensor network with 49 nodes. The percentage of nodes generating packets was varied from 20 to 100 in increments of 20.

It can be noticed from Fig. 14a that our convergecast scheduling algorithm with collision resolution has consistent high delivery ratios. Without collision resolution, the delivery ratio decreases with increasing percentage of source nodes. For directed flooding and radial coordination the decrease in delivery ratio with the increase of percentage of source nodes is drastic. For our scheduled convergecast variants the throughput increases in proportion to the percentage of source nodes (see Fig. 14b). For directed flooding and radial coordination, throughput remains almost constant (around 7 packets/s); suggesting that these are not scalable. The latency

incurred by our scheduled convergecast variants remains almost constant with increasing percentage of source nodes (see Fig. 14c).

6.5. Shooter localization

In order to evaluate the convergecast algorithms in a more realistic setting, we considered the shooter localization [23] 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 (Fig. 15a) and the event trace data (Fig. 15b) are obtained from the actual experiments on the hardware platform. We conducted simulation experiments using this data. The results are shown in Fig. 16.

From Fig. 16 we can conclude that the delivery ratio of both the scheduled 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 convergecast

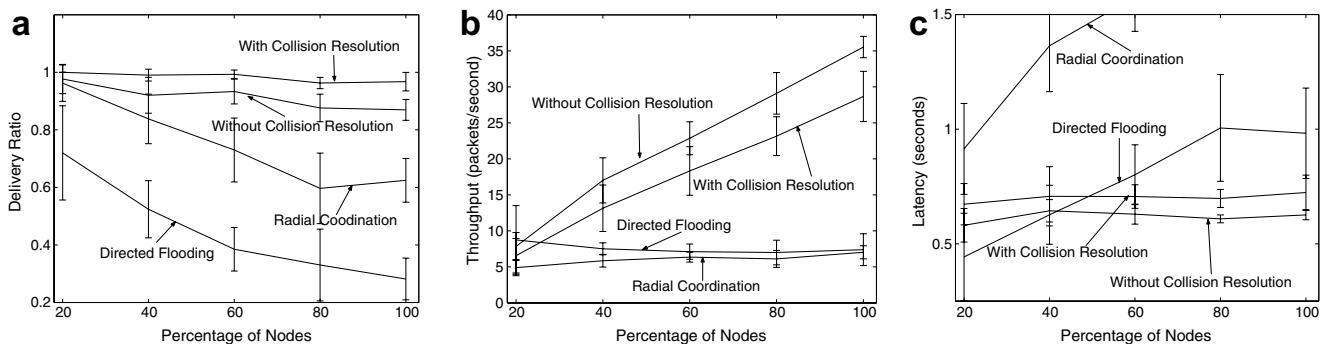


Fig. 14. Performance comparisons: convergecast with varying percentage of source nodes. (a) Delivery ratio; (b) throughput; (c) latency.

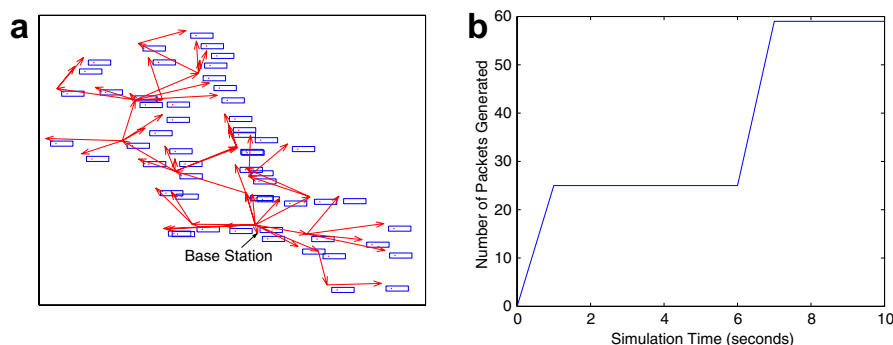


Fig. 15. (a) Node distribution in shooter localization experiment. (b) Event traces for the first 10 s.

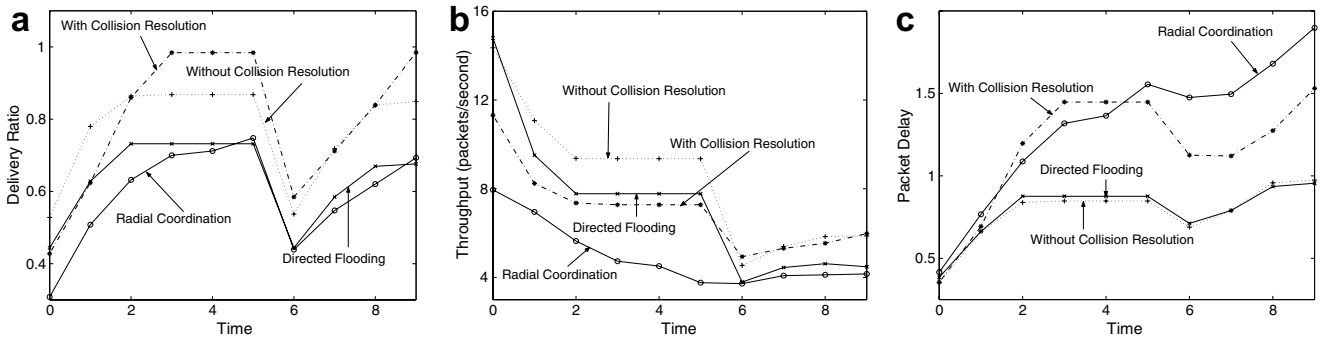


Fig. 16. Performance comparisons: shooter localization. (a) Delivery ratio; (b) throughput; (c) latency.

scheduling algorithm performs better even in realistic scenarios.

6.6. Energy consumption

The scheduled convergecast also saves energy consumption if the sleep schedule is used. We measured the energy consumption of all the convergecast schemes. For scheduled convergecast, nodes employ sleep schedule explained in Section 3.4. Energy spent by a nodes is determined by $P_t * T_t + P_r * T_r + P_i * T_i + P_s * T_s$. Where T_t, T_r, T_i, T_s represent the duration (in s) of transmission, reception, idle and sleep, respectively. Similarly, P_t, P_r, P_i, P_s represent the transmission power, power required for receiving, power required in idle mode and power required in sleep mode, respectively. Power is determined by $P_x = I_x * V$. Where $I_t = 7.1 \text{ mA}$, $I_r = 7 \text{ mA}$, $I_i = 7 \text{ mA}$ and $I_s = 0.000002 \text{ mA}$. And V is constant $3V$. These values are based on energy model for Mica2 motes [28]. Fig. 17 shows that both variants of scheduled convergecast using sleep schedule consume significantly less energy than directed flooding and radial coordination.

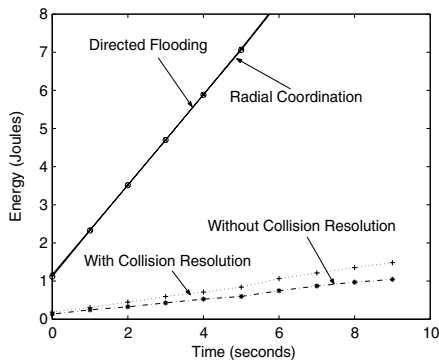


Fig. 17. Energy consumption.

7. Conclusions and future work

We proposed a minimal time distributed convergecast scheduling algorithm for sensor networks. Initially, we considered a simple system where every sensor node had one packet to be forwarded to the base station. For linear networks we proved that the optimal convergecast schedule consists of $3N - 3$ timeslots, where N is the number of nodes in the network. We proposed a scheduling algorithm that requires $3N - 3$ timeslots for convergecast in linear networks. For multi-line and tree networks we proved that our convergecast scheduling algorithm requires at most $\max(3n_k - 1, N)$ timeslots, where n_k represents the highest number of nodes in any branch or subtree. We have shown that the lower bound for convergecast in multi-line and tree networks is $\max(3n_k - 3, N)$. For tree networks, our distributed algorithm requires $\max(3n_k - 1, N)$ timeslots, where N is the total number of nodes and n_k is the size of the largest subtree.

Our convergecast scheduling algorithm proposed for general networks requires at most $3N$ timeslots. We conducted extensive simulation results to 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.

When nodes generate multiple packets we proposed a simple modification to our convergecast scheduling algorithm. The resulting schedule is guaranteed to consist at most $3P$ timeslots, where P is the number of packets in the network. We proposed extensions to our scheduling algorithm in

order to handle scenarios where (i) the convergecast is initiated by the base station and (ii) the interference range of a node is greater than its transmission range. We conducted extensive simulation results to demonstrate effectiveness of our scheduling algorithm over the existing convergecast algorithms.

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.

Acknowledgements

This work is funded in part by the Defense Advanced Research Project Agency (DARPA) contract F33615-01-C-1904. We thank the Institute for Software Integrated Systems (ISIS), Vanderbilt University for providing us the data related to the shooter localization application.

References

- [1] A.S. Tanenbaum, Computer Networks, third ed., Prentice-Hall Inc., 1996.
- [2] A. Kesselman, D. Kowalski, Fast distributed algorithm for convergecast in ad hoc geometric radio networks, in: The Second Annual Conference on Wireless On demand Network Systems and Services, 2005.
- [3] U.C. Berkeley, Berkeley wireless embedded systems. <<http://webs.cs.berkeley.edu/>>, 2003.
- [4] C. Florens, R. McEliece, Packet distribution algorithms for sensor networks, in: IEEE INFOCOM, 2003.
- [5] R.L. Rivest, T.H. Cormen, C.E. Leiserson, C. Stein, Introduction to Algorithms, MIT Press and McGraw-Hill.
- [6] G. Bianchi, Performance analysis of the IEEE 802.11 distributed coordination function, IEEE JSAC 18 (3) (2000).
- [7] S. Gandham, M. Dawande, R. Prakash, Link scheduling in sensor networks: distributed edge coloring revisited, in: Proceedings of IEEE INFOCOM, March 2005.
- [8] S. Gandham, M. Dawande, R. Prakash, S. Venkatesan, Energy efficient schemes for wireless sensor networks with multiple mobile base stations, IEEE Globecom (2003).
- [9] B. Hajek, G. Sasaki, Link scheduling in polynomial time, IEEE Transactions on Information Theory 34 (1988) 910–917.
- [10] Ju Wang Hongsik Choi, Esther A. Hughes, Scheduling on Sensor Hybrid Network, in: IEEE ICCCN, 2005.
- [11] Crossbow Technologies Inc. <<http://www.xbow.com/>>.
- [12] R. Govindan, C. Intanagonwiwat, D. Estrin, J. Heidemann, Impact of network density on data aggregation in wireless sensor networks, in: ICDCS, 2002.
- [13] D. Estrin, B. Krishnamachari, S. Wicker, The impact of data aggregation in wireless sensor networks, in: International Workshop of Distributed Event Based Systems, 2002.
- [14] S.O. Krumke, M.V. Marathe, S.S. Ravi, Models and approximation algorithms for channel assignment in radio networks, Wireless Networks 7 (2001) 575–584.
- [15] L. Gasieniec, I. Potapov, Gossiping with unit messages in known radio networks, in: IFIP TCS, 2002.
- [16] M. Maroti, Directed flood-routing framework, in: Proceedings of the Fifth International Middleware Conference, 2004.
- [17] M.T. Goodrich, R. Tamassia, Data Structures and Algorithms in Java, second ed., John Wiley and Sons, Inc., 2001.
- [18] Q. Huang, Y. Zhang, Radial coordination for convergecast in wireless sensor networks, in: First IEEE Workshop on Embedded Networked Sensors, 2004.
- [19] S. Ramanathan, A unified framework and algorithm for channel assignment in wireless networks, Wireless Networks 5 (1999) 81–94.
- [20] S. Ramanathan, E.L. Lloyd, Scheduling algorithms for multi-hop radio networks, IEEE/ACM Transactions on Networking 1 (1993) 166–177.
- [21] R. Ramaswami, K.K. Parhi, Distributed scheduling of broadcasts in a radio network, IEEE INFOCOM (1989).
- [22] G. Simon. Probabilistic wireless network simulator. <<http://www.isis.vanderbilt.edu/projects/nest/prowler/>>.
- [23] Gy. Balogh, B. Kusy, A. Ndas, G. Pap, J. Sallai, Gy. Simon, A. LTdeczi, K. Frampton, Sensor network-based counter-sniper system, in: Sensys, 2004.
- [24] H. Tamura, K. Watanabe, M. Sengoku, S. Shinoda, On a new edge coloring related to multi-hop wireless networks, APCCAS'02, 2002 Asia-Pacific Conference on Circuits and Systems, 2002.
- [25] W. Ye, J. Heidemann, D. Estrin, An energy-efficient MAC protocol for wireless sensor networks, INFOCOM (2002).
- [26] Y. Zhang, M. Fromherz, L. Kuhn. Rmase: Routing modeling application simulation environment. <<http://www.parc.com/era/nest/Rmase/>>.
- [27] G. Zhou, T. He, J.A. Stankovic, T.F. Abdelzaher, RID: Radio interference detection in wireless sensor networks, Proceedings of IEEE Infocom (2005).
- [28] Marco Zuniga, Bhaskar Krishnamachari. Analyzing the transitional region in low power wireless links, in: First IEEE International Conference on Sensor and Ad hoc Communications and Networks (SECON), October 2004.



Shashidhar Rao Gandham received the Bachelor of Technology degree from National Institute of Technology – Warangal (formerly Regional Engineering College – Warangal), India, in May 1999. He received the Master of Science degree in Computer Science from University of Texas at Dallas in December 2002. During summer of 2005 he was an intern at Palo Alto Research Center Inc. (formerly Xerox PARC). In May 2006,

he received his Doctorate degree from University of Texas at Dallas for his research on link scheduling, routing and base station positioning in wireless sensor networks. Currently he is working as Network Protocol Research Engineer at xG Technology, Inc., where he leads a team that is developing the

protocols for next generation all IP-based cellular networks. His research interests are in wireless sensor networks, mobile ad hoc networks, wireless LANs, mathematical programming, combinatorial optimization and distributed algorithms.



Ying Zhang (<http://www.parc.com/yzhang>) is a member of research staff at Palo Alto Research Center (PARC). She obtained her Ph.D. in Computer Science from University of British Columbia in 1994. Her current research interests are sensor/actuator networks and embedded control architectures. She has led software architecture development for modular reconfigurable robots at PARC and has been working on ad hoc routing and

localization technologies in sensor networks. She is the information director and an associate editor of the ACM Transaction on Sensor Networks, and has served for many technical program committees and NSF review panels on sensor networks. She is a member of IEEE and ACM.



Qingfeng Huang received his D.Sc. degree in Computer Science from Washington University in St. Louis in 2003. He also holds A.M. and M.S. degrees in Physics, from Washington University and Fudan University, respectively. He has been a research scientist at the Palo Alto Research Center (Xerox PARC) since 2003. He has published more than 40 academic papers and filed more than 25 patents in areas such

as in areas including software engineering, context-aware and mobile computing, sensor networks, intelligent transportation systems, robotics, neuroscience, and quantum physics. His current research interests include algorithms and middleware for transportation networks and sensor actuator networks, systems for knowledge sharing and harvesting in networks, and mechanisms that facilitate innovation. He is a member of the IEEE and IEEE Computer Society.