CHAPTER 1

SS-TDMA: A SELF-STABILIZING MAC FOR SENSOR NETWORKS

Sandeep S. Kulkarni
Mahesh (Umamaheswaran) Arumugam

Software Engineering and Network Systems (SENS) Lab
Computer Science and Engineering, Michigan State University
Web: `http://www.cse.msu.edu/~{sandeep,arumugam}`
Email: `{sandeep,arumugam}@cse.msu.edu`

1.1 ABSTRACT

We focus on the problem of designing a TDMA service for a grid-based sensor network. Such networks are readily found in many applications in the area of monitoring, hazard detection, and so on. We consider three communication patterns, *broadcast*, *convergecast* and *local gossip*, that occur frequently in these systems. We develop TDMA service that can be customized based on the application requirements and also provide guidance about using this service when the communication pattern is unknown or varies with time. With these customizations, whenever a sensor receives a message, it can forward it to its successors with a small delay. We show that this TDMA service is collision free whereas existing CSMA based approaches suffer significant collisions. We also show how this service can be extended to deal with other deployments in a 2-D field, failure of sensors, and sensors that are sleeping as part of a power management scheme. Further, we show that this service can be used in a mobile sensor network that provides localization service.

1.2 INTRODUCTION

Sensor networks are becoming popular nowadays due to their applications in both academic and industrial environments. Currently, sensor networks are highly useful in applications such as data gathering, active/passive monitoring and tracking of undesirable objects [1], and unattended hazard detection [2, 3, 4]. Further, the sensors can be rapidly deployed in the field due to their small size and low cost. However, these sensors are often resource constrained. Specifically, the sensors are constrained by limited power, limited communication distance, and limited sensing capability. Therefore, they need to collaborate with each other to perform a particular task.

Sensor networks can be classified based on the nature of deployment of sensors. One approach for such deployment is random where sensors are distributed with some expected density distribution. Another approach for such deployment is systematic geometric distribution. The results in this chapter are targeted towards such systematic geometric distribution where sensors are deployed in a (rectangular, hexagonal or triangular) grid. One such application that uses such systematic distribution and requires collaborative effort is the DARPA NEST technology demonstration project *A Line in the sand* (LITeS) [1]. In this demonstration, sensors are arranged in a thick line (grid). When an intruder (for example, a person, a soldier, a car, or a heavy vehicle such as a tank) comes in the vicinity of this line or crosses this line, the sensors detect it. Now, to classify the intruder, the sensors that observed the intruder communicate with each other. These sensors have two options for classification: internal or external. In an internal classification, the sensors that detect the intruder communicate with each other. And, in an external classification, the sensors send their observed values to a base station that exfiltrates the data outside the network.

Message collision and communication patterns. One of the important challenges in applications such as LITeS is message collision. Specifically, if a sensor receives two messages simultaneously then they collide and both messages become incomprehensible. Also, it is difficult for a sensor to know whether a given message reached all its neighbors. This is due to the fact that a message sent by a sensor may collide at one neighbor and be received correctly at another neighbor.

Another important challenge is to deal with the existence of different communication patterns. We consider three commonly occurring communication patterns: *broadcast*, *convergecast*, and *local gossip*. In broadcast, a message is sent to all the sensors in the network. Broadcast is useful when a base station wants to transmit some information (e.g., program capsules for reprogramming the sensors [5, 6], diffusion message to revalidate the time slots allotted to sensors, etc) to all the sensors in the network. We also consider two other communication patterns, convergecast and local gossip. These communication patterns are based on our experience with LITeS demonstration [1]. Based on the internal classification technique mentioned earlier, we consider the communication pattern, local gossip, where a sensor sends a message to its neighboring sensors within some distance. And, based on the external classification technique mentioned earlier, we consider the communication pattern, convergecast, where a group of sensors send a message to a particular sensor such as the base station.

We present our TDMA service[1] for sensor networks. Our TDMA service ensures collision-freedom and fair bandwidth allocation among different sensors. Further, our TDMA service lets one customize the assignment of time slots to different sensors by considering the common communication patterns that occur in the application.

[1] A preliminary version of this work appears in [7].

Improper initializations and state corruption. In a large sensor network, it is possible that some of the sensors are improperly initialized. Further, if the slots assigned to the sensors are corrupted or the clock skew is higher than expected in some interval, then collisions occur during communication. Hence, starting from such arbitrary states, the TDMA service should be able to recover to states from where the service satisfies its specification. In other words, it is necessary that the TDMA service be stabilizing fault-tolerant [8, 9], i.e., starting from an arbitrary state, the system should recover to states from where subsequent communication is collision-free.

Results. We concentrate on designing collision-free communication algorithms for sensor networks. We compare the performance of our TDMA algorithm with collision-avoidance protocols applicable in sensor networks. The main results of this chapter are as follows:

- We present self-stabilizing TDMA (*SS-TDMA*) service that can be customized for different communication patterns, namely, broadcast, convergecast, and local gossip. Starting from an arbitrary state, SS-TDMA recovers to states from where collision-free communication among sensors is restored. Furthermore, we present simulation results to validate that SS-TDMA is collision-free. We compare SS-TDMA with collision-avoidance protocols like CSMA and show that they suffer significant number of collisions.

- SS-TDMA can be used in several deployment scenarios. We first consider deployment in a rectangular grid. Then, we extend it to hexagonal/triangular grid. Finally, we show how it can extended to any geometric distribution provided localization service [10, 11] is available.

- We show how slots are assigned to sensors so that the delay in broadcast, convergecast, and local gossip is reduced. Also, under reasonable assumptions, we prove that SS-TDMA is delay-optimal for broadcast. Further, we show how SS-TDMA is used in the context of power management.

- We show that SS-TDMA is fault-tolerant. More specifically, we show that it is possible to reclaim the slots assigned to failed sensors and reassign them to active sensors. Furthermore, we show that SS-TDMA can tolerate errors in the location, i.e., even if the sensors are moved slightly from their ideal location, the percentage of collisions is within application requirements. Additionally, we show that mobility is supported in SS-TDMA if localization service is available.

- We outline the middleware architecture of SS-TDMA. Towards this end, we present how our algorithms are implemented as a middleware service. We identify the application programming interfaces (APIs) of our service.

The rest of the chapter is organized as follows. In Section 1.3, we describe the sensor network model and state the assumptions made in this chapter. Then, in Section 1.4, we present SS-TDMA algorithms for sensor networks. Specifically, we present the SS-TDMA algorithms for different communication patterns. Further, in Section 1.5, we show that SS-TDMA is self-stabilizing, delay-optimal (under certain conditions), and energy-efficient. In Section 1.6, we present the simulation and implementation results of SS-TDMA. In Section 1.7, we provide extensions to SS-TDMA. Finally, in Section 1.8, we discuss related work and in Section 1.9, we make concluding remarks.

1.3 MODEL AND ASSUMPTIONS

In this section, we present the sensor network model and state our assumptions. The assumptions are in three categories: existence of base station (or exfiltration point), deployment topology of the network, and sensor radio capabilities.

Base station. We assume that there exists a base station that initiates a diffusing computation to assign initial slots to all sensors. Typically, the base station is more powerful compared to other sensors in the network. And, it has long range wireless network capability (e.g., IEEE 802.11). If there are multiple base stations in the network, first, they elect a leader among themselves. Note that this process is independent of the sensor network since the base stations have powerful radio and also, use a different wireless network protocol. The elected base station is now responsible for initiating the diffusing computation to assign initial TDMA slots to all sensors. Furthermore, the base station could be co-located with another sensor in the grid, or it could be assigned a separate grid position. For simplicity, we assume that the base station is located at the left-top position (at location $\langle 0, 0 \rangle$). However, it can be placed in any grid position. The extension for this case is straightforward and, hence, is omitted.

Topology. Initially, we assume that sensors are arranged in a grid where each sensor knows its location in the network (geometric position). Each message sent by a sensor includes this geometric position. Thus, a sensor can determine the position, direction and distance (with respect to itself) of the sensors that send messages to it. Initially, for simplicity, we assume that the sensor network has a perfect grid topology and no sensors have failed or are in sleeping state. By making these assumptions, we can design algorithms for perfect grid-based sensor networks. Then, we extend the algorithms to deal with the case where sensors (other than the base station) have failed. (Note that the assumption about non-failure of base station is acceptable; base station is responsible for exfiltrating data from the network. Hence, if it fails, another base station must be available to utilize the sensor network.)

In this chapter , initially, we assume a rectangular grid topology. Later, we extend the algorithms for hexagonal and triangular grids. We will also show that our algorithm works for the case where sensors are randomly deployed in a geometric distribution. Finally, we also show that our algorithm works even if the sensor nodes are mobile. The last two extensions require localization service [10, 11] so that the sensors can identify their location in the field.

Communication and interference ranges. We assume that each sensor has a communication range and an interference range. Communication range is the distance up to which a sensor can communicate with certainty/high probability. Interference range is the distance up to which a sensor can communicate, although the probability of such a communication may be low. This assumption is based on the ability of sensors to communicate with each other in a geometric topology, or the existence of some sensors that have larger communication range due to higher power levels or elevation in the sensor plane. Hence, to introduce uniformity in the communication capabilities of the sensors, we consider interference range. In such a scenario, one possible way to estimate the interference range is to take the maximum value of the communication ranges of all sensors. (Note that the communication range that is used in this case should be the minimum of the communication ranges of all sensors.)

Now, based on these assumptions, given two sensors, j and k, if k is in the interference range of j but k is not in the communication range of j then k receives messages sent by j with a low probability. However, if k receives another message while j is sending a message, it is possible that collision between these two messages can prevent k from receiving either

of those messages. Based on the definition of the interference range, it follows that it is at least equal to the communication range. Also, we assume that the sensors are aware of their communication range and interference range. Initially, we assume communication range = 1, i.e., a sensor can only talk to its neighbors in the grid. Then, we extend the algorithms to deal with other communication ranges.

Remark. We only consider collisions where at least one of the messages was expected to be received at the destination. In other words, when j and k send a message that collide at l then that collision is counted only if l is in the communication range of either j or k. If l is outside the communication range of both j and k, l was not expected to receive either of the two messages. Hence, such a collision is not counted.

1.4 TDMA SERVICE FOR SENSOR NETWORKS

In this section, we present time-division multiple access (TDMA) algorithms for sensor networks. Time-division multiplexing is the problem of assigning time slots to each sensor. Two sensors j and k can transmit in the same time slot if j does not interfere with the communication of k and k does not interfere with the communication of j. In this context, we define the notion of *collision-group*. The *collision-group* of sensor j includes the sensors that are in the communication range of j and the sensors that interfere with the sensors in the communication range of j. Hence, if two sensors j and k are alloted the same time slot then j should not be present in the collision-group of k and k should not be present in the collision-group of j. Thus, the problem of time-division multiple access is defined in Figure 1.1.

Time Division Multiple Access
Assign time slots to each sensor such that,
If two sensors j and k transmit at the same time then
($j \notin$ collision-group of sensor k).

Figure 1.1. Problem statement of TDMA

Now, we present the algorithm for allotting time slots to the sensors. In Section 1.4.1, we present the TDMA algorithm for broadcast. Then, in Section 1.4.2, we present the TDMA algorithm for convergecast and in Section 1.4.3, we present the algorithm for local gossip.

1.4.1 TDMA Service for Broadcast

In this section, we present our TDMA algorithm for broadcast. Consider a grid network where a sensor can communicate with sensors that are distance 1 away and interfere with sensors that are distance $y, y \geq 1$ away (cf. Figure 1.2, where $y = 2$). To present the TDMA algorithm, we first present an algorithm for initial slot assignment. Then, we present an algorithm for subsequent slot assignments.

Initial slot assignment. Let us assume that the base station starts a diffusing computation to assign initial slots at time slot 0. From Figure 1.2, we observe that sensors $\langle 1, 0\rangle$ and $\langle 0, 1\rangle$ will receive the diffusion message. Now, both sensors $\langle 1, 0\rangle$ and $\langle 0, 1\rangle$ should not forward the diffusion message at the same time, as their message would collide at sensor $\langle 1, 1\rangle$. Hence, sensor $\langle 1, 0\rangle$ is allowed to forward the message at slot 1. Sensor $\langle 2, 0\rangle$ is allowed to forward the message at slot 2. We note that sensors $\langle 2, 0\rangle$ and $\langle 0, 1\rangle$ should not

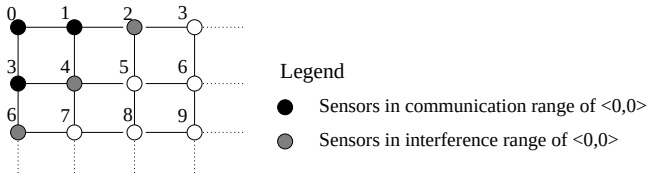


Figure 1.2. Initial slot assignment for broadcast where communication range$=1$, interference range$=2$. The number associated with each sensor denotes the slot at which it should forward the diffusion message.

transmit at the same time as their messages will collide at $\langle 1,1\rangle$ and $\langle 2,1\rangle$. Hence, sensor $\langle 0,1\rangle$ is allowed to forward the message at slot $(y+1)$ (i.e, slot 3 in Figure 1.2. In general, if a sensor receives a diffusion message from its west neighbor, it forwards after 1 slot, and if it receives the message from its north neighbor, it forwards after $(y+1)$ slots. In short, if the base station initiates the diffusion at slot 0, sensor $\langle i,j\rangle$ will forward the message at slot $i+(y+1)j$.

TDMA slots. Once the initial slot (the slot in which a sensor forwards the diffusion message) is determined, a sensor determines its future slots using the TDMA period. Let j and k be two sensors such that j is in the collision-group of k. Let t_j (respectively, t_k) be the initial slot of j (respectively, k). We propose an algorithm where the slots assigned for j are $t_j + c * MCG$ where $c \geq 0$ and MCG captures information about the maximum collision-group in the system.

From the initial slot assignment algorithm, we know that $t_j \neq t_k$. Now, future messages sent by j and k can collide if $t_j + c_1 * MCG = t_k + c_2 * MCG$, where $c_1, c_2 > 0$. In other words, future messages from j and k can collide iff $|t_j - t_k|$ is a multiple of MCG. More specifically, to ensure collision-freedom, it suffices that for any two sensors j and k such that j is in the collision-group of k, MCG does not divide $|t_j - t_k|$. We can achieve this by choosing MCG to be $max(|t_j - t_k| : j$ is in the collision-group of $k) + 1$.

If j is in the collision-group of k then $|t_j - t_k|$ is at most $(y+1)^2$; such a situation occurs if j is at distance of $y+1$ in north/south of k. Hence, the TDMA period is $(y+1)^2 + 1$. The algorithm for assigning TDMA slots is shown in Figure 1.3.

```
const P_b = (y+1)^2 + 1;
// Initial slot assignment for broadcast
When sensor j receives a diffusion message from k
    if (k is west neighbor at distance 1)
      transmit after 1 slot.
    else if (k is north neighbor at distance 1)
      transmit after (y+1) slots.
    else // duplicate message
      ignore

// TDMA algorithm for broadcast
If sensor j transmits a diffusion message at time slot t_j,
    j can transmit at time slots, ∀c : c ≥ 0 : t_j + c * P_b.
```

Figure 1.3. TDMA algorithm for broadcast

The algorithm assigns time slots for each sensor based on the time at which it transmits the diffusion message. Thus, a sensor (say, j) can transmit in slots: t_j, $t_j + ((y+1)^2+1)$, $t_j + 2((y+1)^2+1)$, . . ., etc. Figure 1.4 shows a sample allocation of slots to the sensors.

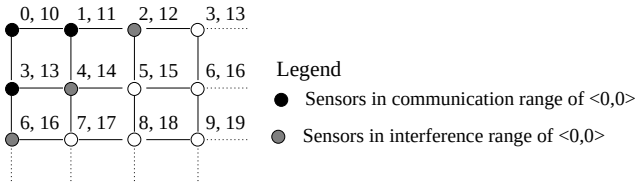


Figure 1.4. TDMA slot assignment for broadcast where communication range $= 1$, interference range $= 2$. The numbers associated with each sensor denote the slots at which it can send a message.

Theorem 1.1 The initial slot assignment for broadcast in the above algorithm is collision-free.

Proof. Let us assume that the source sensor $\langle 0, 0\rangle$ (i.e., base station) starts transmitting at slot 0. By induction, we observe that sensor $\langle i, j\rangle$ will transmit at time slot $t = i+(y+1)j$. Now, we show that collisions will not occur in this algorithm. Consider two sensors $\langle i_1, j_1\rangle$ and $\langle i_2, j_2\rangle$. Sensor $\langle i_1, j_1\rangle$ will transmit at time slot $t_1 = i_1 + (y+1)j_1$ and $\langle i_2, j_2\rangle$ will transmit at time slot $t_2 = i_2 + (y+1)j_2$. Collision is possible only if the following conditions hold:

- $t_1 = t_2$, i.e., $(i_1 - i_2) + (y+1)(j_1 - j_2) = 0$.
- The Manhattan distance between $\langle i_1, j_1\rangle$ and $\langle i_2, j_2\rangle$ is less than or equal to $(y+1)$, i.e., $|i_1 - i_2| + |j_1 - j_2| \leq y + 1$.
- $\langle i_1, j_1\rangle$ and $\langle i_2, j_2\rangle$ are distinct, i.e., $|i_1 - i_2| + |j_1 - j_2| \geq 1$.

From the first condition, we conclude that $(i_1 - i_2)$ is a multiple of $(y+1)$. Combining this with the second condition, we have $|i_1 - i_2| = 0$ or $|j_1 - j_2| = 0$. However, if $|i_1 - i_2| = 0$ (respectively, $|j_1 - j_2| = 0$) then from the first condition $(j_1 - j_2)$ (respectively, $(i_1 - i_2)$) must be zero. If both $(i_1 - i_2)$ and $(j_1 - j_2)$ are zero then the third condition is violated. Thus, collision cannot occur in this algorithm. □

Theorem 1.2 The above algorithm satisfies the problem specification of TDMA.

Proof. Consider two distinct sensors j and k such that j is in the collision-group of k. The time slots assigned to j and k are $t_j + c * P_b$ and $t_k + c * P_b$ respectively, where c is an integer and $P_b = (y+1)^2 + 1$ is the period between successive slots. Suppose a collision occurs when j and k transmit a message at slot $t_j + c_1 * P_b$ and $t_k + c_2 * P_b$ respectively, where $c_1, c_2 > 0$. In other words, $t_j + c_1 * P_b = t_k + c_2 * P_b$. From Theorem 1.1, we know that $t_j \neq t_k$. Therefore, collision will occur iff $|t_j - t_k|$ is a multiple of P_b. However, since j is in the collision-group of k, $|t_j - t_k|$ is at most $(y+1)^2$ (less than P_b). In other words, $|t_j - t_k| \leq (y+1)^2 < P_b$. Hence, if j and k transmit at the same time, then they are not present in the collision-group of each other. This is a contradiction. Thus, collisions cannot occur in this algorithm. □

Remark. In our algorithms, we have used Manhattan distance in the calculation of interference range. If we consider geometric distance instead, our algorithms can be extended

appropriately by using a larger interference range that accommodates all sensors where interference may occur.

1.4.2 TDMA Service for Convergecast

Suppose, sensor $\langle 1, 1\rangle$ sends a message to the base station at slot 14 (cf. Figure 1.4). Sensors $\langle 0, 1\rangle$ and $\langle 1, 0\rangle$ receive the message. However, these sensors have just missed their slots (11 and 13 respectively) and hence, need to wait until the next slot before forwarding the message to the base station. Therefore, the algorithm in Section 1.4.1 introduces a significant delay for convergecast, where a group of sensors send data (for example, information about the activities of an intruder in the field [1]) to the base station. Hence, in this section, we customize the algorithm in Section 1.4.1 for convergecast.

To reduce the delay for convergecast, we change the slot assignment as follows: If j receives a message from its left neighbor then it chooses to transmit the diffusion in $(-1)^{th}$ slot (in circular sense). In other words, j transmits in the $(P-1)^{th}$ slot, where P (=$(y+1)^2+1$) is the interval between slots assigned to a sensor and y is the interference range of the sensors. If j receives a message from its top neighbor then it transmits in the $(-(y+1))^{th}$ slot. (For example, see Figure 1.5 for slot assignment for the case where $y = 2$.) After the first slot is determined, the sensors can then transmit once in every P slots.

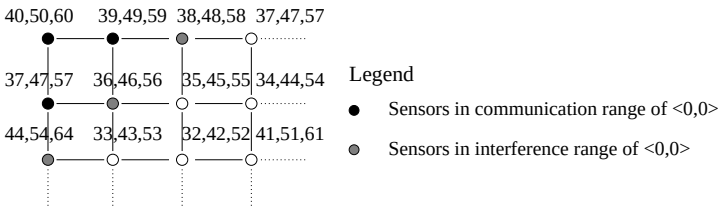


Figure 1.5. TDMA slot assignment for convergecast where communication range $= 1$, interference range $= 2$. The numbers associated with each sensor denote the slots at which it can send a message. Some initial slots are not shown.

As we can see from Figure 1.5, with the above slot assignment, delay for convergecast is reduced. Specifically, when a sensor transmits a message that is to be relayed by sensors closer to the base station (left-top sensor), such a relay introduces only a small (respectively, no) delay. Thus, the TDMA algorithm customized for convergecast is shown in Figure 1.6.

Theorem 1.3 The above algorithm satisfies the problem specification of TDMA.
Proof. The proof is similar to Theorem 1.2. □

1.4.3 TDMA Service for Local Gossip

For local gossip, the communication is in all directions. Hence, we need an approach that combines the slot assignment for broadcast and convergecast. We proceed as follows: We increase the value of the period (P) to $2((y+1)^2+1)$, twice the previous value. With this increased value, each sensor gets two slots (even and odd) in this period. Let the slots assigned to the base station be 0 and $P-1$. To simplify the presentation, let us assume that the base station starts a diffusion in its even or the 0^{th} slot. When j receives the diffusion

const $P_c = (y+1)^2 + 1$;
// Initial slot assignment for convergecast
When sensor j receives a diffusion message from k
    if (k is west neighbor at distance 1)
      transmit in the $P_c + (-1)^{th}$ slot.
    else if (k is north neighbor at distance 1)
      transmit in the $P_c + (-(y+1))^{th}$ slot.
    else // duplicate message
      ignore

// TDMA algorithm for convergecast
If sensor j transmits a diffusion message at time slot t_j,
    j can transmit at time slots, $\forall c : c \geq 0 : t_j + c * P_c$.

Figure 1.6. TDMA algorithm for convergecast

from its left neighbor, it chooses the slot that is 2 higher than that used by the left neighbor. Likewise, when j receives the diffusion from its top neighbor, it chooses the slot that is $2(y+1)$ higher than that used by the top neighbor. (For example, see Figure 1.7 for slot assignment for the case where $y = 2$.) Note that the diffusion messages are forwarded in the even slots. In our solution for gossip, whenever sensor k transmits in the even slot, say t_k, it can also transmit in $((P-1)-t_k)\ mod\ P$, the odd slot. Thus, the TDMA algorithm customized for local gossip is shown in Figure 1.8.

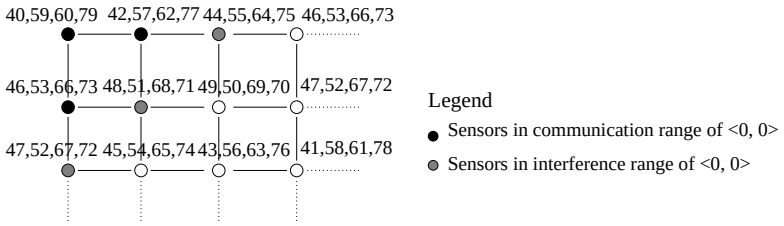


Figure 1.7. TDMA slot assignment for gossip where communication range $= 1$, interference range $= 2$. The numbers associated with each sensor denote the slots at which it can send a message. Some initial slots are not shown.

Theorem 1.4 The above algorithm satisfies the problem specification of TDMA.
Proof. The proof is similar to Theorems 1.2 and 1.3. Note that in the gossip algorithm, even slots behave like broadcast algorithm and odd slots behave like convergecast algorithm.

□

Based on Figure 1.7, in the case where TDMA is customized for local gossip, the interval between two successive slots of a sensor can be twice as much as in the case where TDMA is customized for broadcast/convergecast. Thus, if a sensor needs to transmit a message then the worst case delay is larger when the TDMA service is customized for local gossip. In spite of this deficiency, the TDMA service provides substantial benefits for broadcast and convergecast even if it is customized for local gossip. To see this, observe that once the base station sends the broadcast message in its even slot, any sensor receiving it can forward it

```
const P_g = 2((y + 1)^2 + 1);
// Initial slot assignment for local gossip
When sensor j receives a diffusion message from k
    if (k is west neighbor at distance 1)
      transmit after 2 slots.
    else if (k is north neighbor at distance 1)
      transmit after 2(y + 1) slots.
    else // duplicate message
      ignore

// TDMA algorithm for local gossip
If sensor j transmits a diffusion message at time slot t_j,
    j can transmit at time slots,
      ∀c : c ≥ 0 : t_j + c * P_g,
          (((P_g − 1) − t_j) mod P) + c * P_g.
```

Figure 1.8. TDMA algorithm for local gossip

with a small delay (cf. Figure 1.7). Likewise, if a sensor transmits a convergecast message in the odd slot, any sensor receiving it can forward it with a small delay. In fact, as seen from Figure 1.7, in the TDMA service customized for local gossip, if any sensor wants to transmit a message in any given direction (east, west, north, south, southeast, southwest, northeast, or northwest) then any sensor that receives that message can forward it with a small delay.

Based on the above discussion, if the most common communication pattern is known to be broadcast or convergecast, we can customize the TDMA service accordingly. Even if the communication pattern is unknown or varies with time, customizing the TDMA service for local gossip provides a significant benefit for other communication patterns.

1.5 SS-TDMA: PROPERTIES

In this section, we present some of the properties of our algorithms. First, in Section 1.5.1, we show how stabilization can be added to the algorithms in Section 1.4. Then, in Section 1.5.2, we show that under certain conditions, the delay in delivering a broadcast message using the algorithm in Section 1.4 is optimal. In Section 1.5.3, we show that the algorithms proposed in Section 1.4 can be used in the context of power management.

1.5.1 Stabilization and Reliability

We now add stabilization to the TDMA algorithms discussed in Section 1.4, i.e., if the network is initialized with arbitrary clock values (including the case where there is a phase offset among clocks), we ensure that it recovers to states from where collision-free communication is achieved. The TDMA algorithm in Section 1.4 relies on the initial slot assignment algorithm. We modify that algorithm to obtain self-stabilizing TDMA (SS-TDMA). Specifically, in SS-TDMA, the base station periodically sends a diffusion message in a slot it believes to be its TDMA slot (according to the algorithm in Section 1.4). Whenever a sensor receives the diffusion message, it recalculates its TDMA slot based on the appropriate algorithm in Section 1.4. Then, it forwards the diffusion message it that slot.

If the clock values are corrupted then some sensors may not receive the diffusion message. To deal with this case, in SS-TDMA, whenever a sensor does not get the diffusion message for certain consecutive number of times, the sensor shuts down, i.e., it will not transmit any message until it receives a diffusion message from a sensor closer to the base station. The network will eventually reach a state where the diffusion message can be received by all sensors. From then on, the sensors can use the TDMA algorithm in Section 1.4 to transmit messages. Based on the above description, we observe that if there are no faults in the network and the links are reliable then no sensor will ever shut down. Moreover, if faults perturb clock values, then eventually they will be restored so that subsequent communication is collision-free. Thus, we have

Theorem 1.5 Starting from arbitrary initial states, SS-TDMA recovers to states from where collision-free communication among sensors is restored. □

Remark. We do not specify the parameters such as the period between successive diffusing computations, the number of diffusions a sensor waits before shutting down, etc. This choice depends on how frequently we want to perform validation of slots to account for clock drifts, acceptable overhead when no clock drifts occur, and acceptable time for recovery. Since we use time-synchronization service [12] along with SS-TDMA, the clock drift among sensors is very small. Further, the overhead incurred by time-synchronization service is very low (one beacon every 15 seconds). Hence, the period between successive diffusing computations to revalidate the time slots could be higher. Thus, the frequency of diffusing computations is expected to be very low. However, we do not consider the issue of optimizing these values based on the requirements of the application. This issue is orthogonal to the service proposed in this chapter.

Dealing with unreliable links. So far, we assumed that if a sensor sends a message then in the absence of collisions, it would be correctly received. However, in a sensor network, message could be lost due to other environmental factors. In SS-TDMA, such failures are already tolerated. However, in such cases, some sensors may shut down incorrectly. The probability of such shut downs can be reduced. Towards this end, let p be the probability that a sensor receives a message from its neighbor. Also, let n be the number of diffusion periods a sensor waits before shutting down. Now, consider a sensor j that receives a diffusion message after l intermediate transmissions. The probability that this sensor does not receive the diffusion message is $1 - p^l$ and the probability that this sensor shuts down in the absence of faults is $(1-p^l)^n$. Note that this is an overestimate since a sensor receives the diffusion message from more than one sensor. If we consider $p = 0.90$, $l = 10$ and $n = 10$, the probability that a sensor that is 10 hops away from the base station will incorrectly shut down is 0.0137. Thus, we have

Theorem 1.6 If p is the probability of a successful communication over a link, n is the number of diffusion periods a sensor waits before shutting down, then the probability that a sensor l hops away from the base station shuts down incorrectly is at most $(1 - p^l)^n$. □

Corollary 1.7 If there are no faults in the network and the links are reliable then no sensor will ever shut down.

Proof. If the links are reliable, then $p = 1$. Hence, from Theorem 1.6, this theorem follows. □

1.5.2 Delay Optimality

In this section, we prove that, under certain assumptions, the broadcast algorithm is optimal and our algorithm reduces the delay in delivering a message to its intended receivers. Towards this end, we prove Theorems 1.8 and 1.9, next.

Theorem 1.8 A broadcast where (1) communication range and interference range of the sensors is 1, (2) every sensor must transmit at least once, (3) if $i_1 \leq i_2$ and $j_1 \leq j_2$ then the slot used by sensor $\langle i_1, j_1 \rangle$ to transmit the broadcast message should be less that or equal to the slot used by sensor $\langle i_2, j_2 \rangle$, and (4) no collisions should occur, will take at least $3(n-1)+1$ slots in a $n * n$ grid where the initiator of broadcast is at one corner.

Proof. We prove this by induction. For the base case, consider a $2 * 2$ grid as shown in Figure 1.9.

Figure 1.9. Broadcast in a $2 * 2$ grid

Suppose sensor $\langle 0, 0 \rangle$ starts the broadcast at slot 0. Sensors $\langle 1, 0 \rangle$ and $\langle 0, 1 \rangle$ receive the message. Now, both these sensors cannot transmit in the next slot, since there will be a collision at $\langle 1, 1 \rangle$. Hence, without loss of generality, let sensor $\langle 1, 0 \rangle$ transmit the broadcast at slot 1. Now, either sensors $\langle 0, 1 \rangle$ or $\langle 1, 1 \rangle$ can transmit next. However, sensor $\langle 0, 1 \rangle$ should transmit the message before sensor $\langle 1, 1 \rangle$. Hence, sensor $\langle 0, 1 \rangle$ transmits at slot 2 and sensor $\langle 1, 1 \rangle$ transmit at slot 3. Hence, the broadcast takes 4 slots. Thus, the theorem holds for $2 * 2$ grid.

For inductive case, let us assume that the theorem holds for $n * n$ grid, i.e., the broadcast takes $3(n-1)+1$ slots. Now, we prove that in a $(n+1) * (n+1)$ grid, the broadcast takes $3n + 1$ slots.

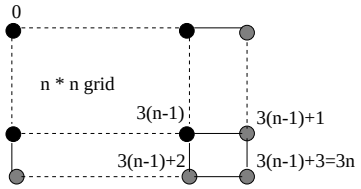


Figure 1.10. Broadcast in a $(n+1) * (n+1)$ grid

From the induction hypothesis, we know that the last (bottom-right) sensor (i.e., $\langle n-1, n-1 \rangle$) in the $n * n$ grid transmits the broadcast message at slot $3(n-1)$. Hence, sensor $\langle n, n-1 \rangle$ to the right in $(n+1) * (n+1)$ grid, transmits the message at slot $3(n-1)+1$ (cf. Figure 1.10). Similar to the argument in $2 * 2$ grid, sensor $\langle n-1, n \rangle$ should transmit the message before sensor $\langle n, n \rangle$. Therefore, sensor $\langle n-1, n \rangle$ transmits at slot $3(n-1)+2$ and

sensor $\langle n, n\rangle$ transmits at slot $3(n-1)+3=3n$. Hence, the broadcast takes at least $3n+1$ slots. Thus, the theorem holds for $(n+1)*(n+1)$ grid. □

Theorem 1.9 For communication range and interference range of 1, SS-TDMA requires $3(n-1)+1$ slots for broadcast in a $n*n$ grid. □

Thus, under the assumptions of Theorem 1.8, SS-TDMA customized for broadcast pattern is delay-optimal. Further, in the broadcast algorithm presented in Section 1.4.1, whenever a sensor sends a message, the sensors farther away from the base station receive the message just before their allotted time slots. Hence, they can transmit the message with a small delay. Similarly, for other communication patterns and interference ranges, a sensor can forward the message to others with a small delay.

1.5.3 Energy Efficiency

In this section, we discuss the energy-efficiency of SS-TDMA. Energy-efficient algorithms are important in sensor networks due to the inherent power constraints of the sensors.

Energy-efficiency can be achieved in SS-TDMA as follows. A sensor remains in active mode only in its alloted time slots (if it needs to send any data) and in the alloted time slots of the sensors within its communication range. In the remaining slots, the sensor can save energy by turning off its radio and remaining in idle mode. Suppose the communication range of a sensor is 1, then a sensor will have at most 4 neighbors. Let P be the period between successive time slots alloted to a sensor. A sensor will have to be in active mode in its alloted time slot and in the alloted time slot of its 4 neighbors, during every period. In other words, a sensor will have to be in active mode in 5 slots each period. For interference range, $y=2$, period $P=(y+1)^2+1=10$, a sensor will have to be active in 5 slots in every 10 slots, i.e., 50% of the time. In general, if the communication range is 1 then a sensor needs to be awake for at most 5 slots in every $(y+1)^2+1$ slots.

We note that more optimizations are possible in the above scheme. In SS-TDMA, whenever a sensor has some message to send, it will send the message at the start of its alloted time slot. Hence, neighboring sensors can decide whether they should continue listening in that time slot. If a sensor does not receive any message in the first part of the time slot, it can turn its radio off. Further, depending on the communication pattern, a sensor can choose to listen only to a subset of its neighbors. For example, in broadcast, a sensor always gets a message from the sensors that are closer to the base station. Hence, sensors can choose to listen only to the slots assigned to their neighbors that are closer to the base station. Thus, energy-efficiency can be achieved in SS-TDMA.

1.6 SS-TDMA: SIMULATION AND IMPLEMENTATION RESULTS

We have implemented SS-TDMA for different communication patterns on MICA motes [3, 2]. Further, we have simulated our algorithms in *prowler* [13], which allows one to simulate arbitrarily large number of sensors (especially MICA motes). In Section 1.6.1, we present the middleware architecture of SS-TDMA for MICA motes. In Section 1.6.2, we present the simulation model and in Section 1.6.3, we present the simulation results.

1.6.1 SS-TDMA: Middleware Architecture

SS-TDMA service includes APIs for initialization, send and receive. We discuss each of these APIs and their internal details, next.

Initialization. As discussed in Section 1.4, one of the parameters to the service is the interference range used by sensors. We assume that the interference range of all sensors is identical. For initialization, SS-TDMA assumes that once the sensor network is deployed, there is a delay before the application begins. This delay is used to perform a diffusing computation and to assign initial slots. Additionally, the diffusion is performed periodically to (re)validate the slots and to deal with clock drift among sensors. Thus, one of the parameters to the service is the period after which diffusion is used to (re)validate the slots that sensors need to use for TDMA.

Yet another parameter for SS-TDMA is the time slot (in physical time) that should be assigned for sending a message. We choose the slot time so that it is larger than the time required to send a message of maximum length (including preamble, CRC, etc.).

SS-TDMA also takes the parameter that identifies the communication pattern for which the service should be customized. The application can use this parameter to customize the communication that occurs most frequently. As discussed in Section 1.4, if the commonly occurring communication pattern is not known then customizing SS-TDMA for local gossip is beneficial.

Send. Although SS-TDMA ensures that when two sensors, say j and k, transmit simultaneously, neighbors of j (respectively, k) receive messages from j (respectively, k) without collision, we still use CSMA. Thus, if j is about to transmit in its TDMA slot and it observes that the channel is busy then j backs off until the next TDMA slot. Although in our simulations and in experiments with small number of motes such a back off never occurred, it is expected that it may occur in a larger experimental setup. We expect that using CSMA in addition to TDMA will reduce the collisions that may occur due to unsynchronized clocks, larger interference range than that used by SS-TDMA or interference range that varies with time or other environment factors.

The send is non-blocking. Hence, if SS-TDMA receives more than two messages and the sum of their lengths is less than the maximum message length, we combine those messages and send them in the next time slot.

Receive. There are no special tasks performed when SS-TDMA receives a message. All received messages are forwarded to upper layer. Additionally, if the received message includes multiple embedded messages then the receive action separates them.

1.6.2 Simulation Model

In this section, we discuss the simulation model of the experiments. We use a probabilistic wireless network simulator, *prowler* [13], that is a simulation environment for embedded systems especially for MICA motes [3, 2]. The simulator has a modular design. Each layer of the system architecture is designed as a separate module.

Using prowler, one can prototype different sensor network applications, communication models, propagation models and topology. For our TDMA simulations, we use the radio/communication model that is based on the algorithms in Section 1.4. Using this model, we implement the notion of communication range and interference range. To compare our algorithms with the existing implementation, we use the default radio models provided by prowler. These models include CSMA and a primitive model that uses no access protocol. Finally, we use the rectangular grid as the underlying topology since it reflects the topology

used in LITeS [1]. Specifically, in LITeS, the sensors are arranged in a rectangular grid and the base station is placed at one corner in this grid.

Now, we discuss the simulations we performed in the context of these communication patterns based on the experiences with LITeS. Then, we discuss the simulations we performed to study the effect of location errors. SS-TDMA groups up to 4 messages in the queue into a single message before transmitting.

Broadcast. The base station (sensor at left-top corner) initiates a broadcast. It sends the broadcast message to its neighbors in the communication range. Whenever a sensor receives the broadcast message for the first time, it relays it (for sensors farther from the base station). We conduct the broadcast simulations for different network sizes. In these simulations, we consider the following metrics: maximum delay incurred in receiving the broadcast message, number of sensors that receive the broadcast message, and number of collisions. Since CSMA (respectively, no MAC layer) does not guarantee reception by all sensors, we also consider the delay when a certain percentage of sensors receive the broadcast message. Regarding collisions, we compute the ratio of the number of collisions to the number of messages.

Convergecast. For convergecast, a set of sensors send a message to the base station (approximately) at the same time. In our experiments, we keep the network size fixed at 10x10. We choose a subgrid of varying size; sensors in this subgrid transmit the data to the base station. We assume that the subgrid that sends the data to the base station is in the opposite corner from the base station, i.e., the subgrid is farthest from the base station. For these simulations, we compute maximum delay incurred for receiving messages at the base station, the percentage of sensors whose messages are received by the base station and the number of collisions. Once again, as in broadcast simulations, we compute the delay for the case where a certain percentage of messages are received by the base station.

Local gossip. In local gossip, a subgrid of sensors send the data. The goal is to transmit the data from these sensors to the sensors in the subgrid and the neighbors of the sensors in the subgrid. Thus, local gossip is applicable in *locally* determining the set of sensors that observed a particular event. In our experiments, we keep the network size fixed at 10x10. We choose different sizes of subgrids. For these simulations, we compute the average delay incurred for receiving messages at the sensors that are expected to receive the local gossip and number of collisions.

Location errors. An important concern for a communication protocol is the errors in sensor location. Errors are introduced in sensor location due to misplacement of sensors, or external factors like wind, vehicle movement, etc. Communication protocols that depend on sensor location should be able to tolerate this kind of error.

To model location errors, we randomly perturbed the sensors. In our simulation, the error in sensor location is determined using a normal distribution, $N(\mu, \sigma)$, where μ is the mean error distance and σ is the standard deviation of the error. The direction of perturbation was randomly selected from 0 degrees to 360 degrees. To ensure that the grid remains connected in spite of perturbations, during these simulations we increased the communication range. We note that this is a reasonable assumption in that if we need to tolerate location errors then the ideal distance between two neighboring sensors should be smaller the communication range.

1.6.3 Simulation Results

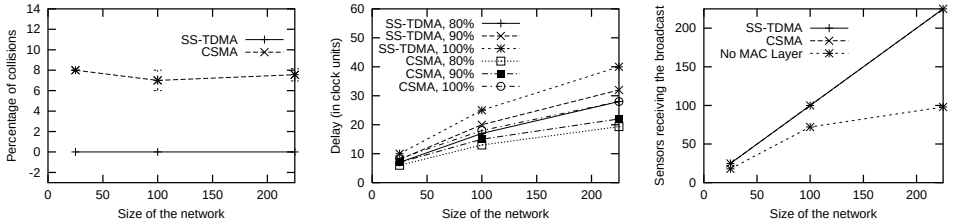
In this section, we present our simulation results that compare SS-TDMA with the case where CSMA is used and with the case where no MAC layer is used. The results presented

in this section are mean of 3 experiments. Also, we have shown the variance in the graphs, whenever it is more than 5% of the mean.

Broadcast. In Figure 1.11, we present our simulation results for broadcast for the case where communication range and interference range is 1. Figure 1.11 (a) identifies the number of collisions that occur in different algorithms. As expected, SS-TDMA is collision free for all network sizes. By contrast, in CSMA, about 10% of messages suffer from collisions.

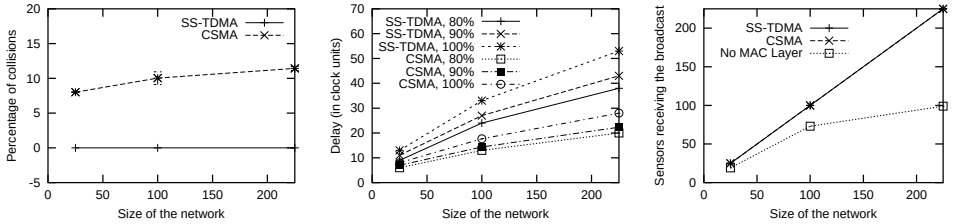
Figure 1.11 (b) identifies the maximum delay incurred in receiving broadcast messages. Since all sensors may not receive the broadcast message when CSMA is used, we consider the delay when a certain percentage, 80-100%, of sensors receive the broadcast message. As we can see, the delay in SS-TDMA is only slightly higher.

Figure 1.11 (c) identifies the number of sensors that receive the broadcast message. We find that with SS-TDMA/CSMA, all sensors receive the message. However, without the MAC layer, the number of sensors that receive the message is less than 50%.



(a) (b) (c)

Figure 1.11. Results for broadcast with communication range = 1, interference range = 1. With SS-TDMA/CSMA, all sensors receive the broadcast and hence, the graphs for them are identical



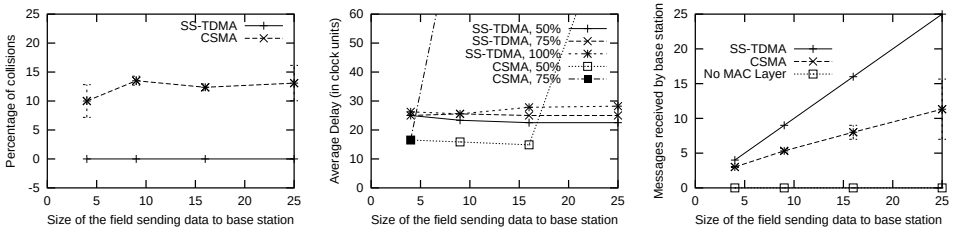
(a) (b) (c)

Figure 1.12. Results for broadcast with communication range = 1, interference range = 2. With SS-TDMA/CSMA, all sensors receive the broadcast and hence, the graphs for them are identical.

In Figure 1.12, we present the simulation results for broadcast for the case where communication range is 1 and interference range is 2. As we can see, these results are similar to those in Figure 1.11.

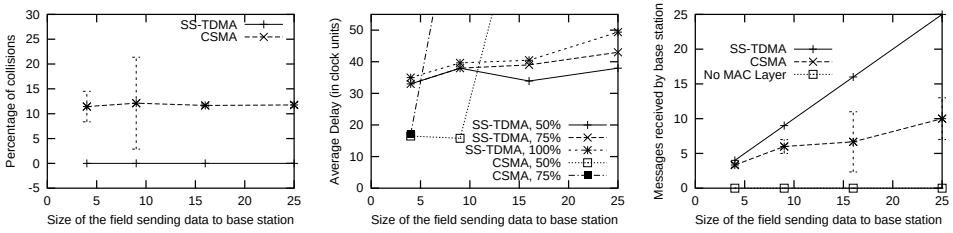
Convergecast. In Figure 1.13, we present our simulation results for convergecast for the case where communication range and interference range is 1. Figure 1.13 (a) identifies the number of collisions that occur in different algorithms. As we can see from Figure 1.13

(a), although SS-TDMA is collision free, there are a significant number of collisions with CSMA. Regarding delay, as we can see from Figures 1.13 (b) and (c), the delay incurred by SS-TDMA is reasonable and that the base station receives all the messages sent by the sensors. By contrast, with CSMA, approximately 50% of the messages are received when the number of sensors sending the data to the base station increases. The delay incurred for 75% of the messages to reach the base station is infinity when the the field size is greater than 4. This is represented in the graph (cf. Figure 1.13 (b)) by an arrow that goes vertically outside the graph. And, without the MAC layer, no data reaches the base station (cf. Figure 1.13 (c)).



(a) (b) (c)

Figure 1.13. Results for convergecast with communication range = 1, interference range = 1



(a) (b) (c)

Figure 1.14. Results for convergecast with communication range = 1, interference range = 2

In Figure 1.14, we present the simulation results for convergecast for the case where communication range is 1 and interference range is 2. As we can see, these results are similar to those in Figure 1.13.

Local gossip. In Figures 1.15 (a) and 1.15 (b), we present our simulation results for local gossip for the case where communication range is 1 and interference range is 1. Figure 1.15 (a) identifies the number of collisions as the size of the group performing local gossip increases. As we can see, CSMA based solution suffers significant collisions whereas SS-TDMA is collision free. Also, as seen from Figure 1.15 (b), the delay in SS-TDMA is somewhat more than that in CSMA. However, unlike SS-TDMA where all sensors receive the necessary messages, in CSMA, the sensors receive approximately 50% of messages.

Once again, the results are similar for the case where interference range is increased to 2 (cf. Figures 1.15 (c) and 1.15 (d)).

Effect of location errors. In our location error experiments, even if the sensors are perturbed from their ideal position, as long as the perturbation is small and the communication range is increased so that the network remains connected, the results are close to those

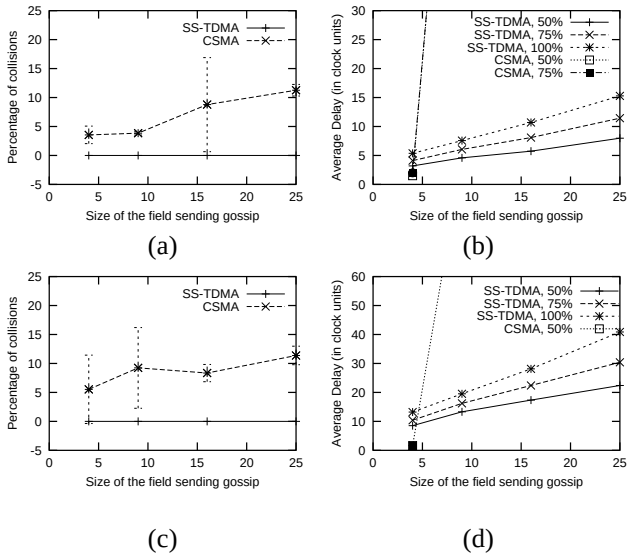


Figure 1.15. Results for local gossip; (a) and (b) with communication range = 1, interference range = 1, (c) and (d) with communication range = 1, interference range = 2

presented earlier. We introduce location errors in the sensors as follows. Let $\langle a, b\rangle$ be the ideal location of a sensor. Let e_d be the distance a sensor is perturbed from its ideal location, and θ_d be the angle of perturbation. The error distance e_d is determined using the normal distribution $N(\mu, \sigma)$, where μ is the mean error distance and σ is the standard deviation of e_d. Thus, the error in location on 95% of the sensors is in the range $(-\mu-2\sigma, \mu+2\sigma)$. Hence, to determine the topology, we increase the *physical communication range* by $\mu+2\sigma$. However, the communication and interference range used by the algorithm is 1. For small perturbations (i.e., $\mu \leq 0.2$), increasing the physical communication range is sufficient to ensure that the network is connected. However, for larger perturbations (i.e, $\mu > 0.2$), if the communication and interference range used by the algorithm is 1, number of collisions increase significantly. Hence, we need to increase the interference range that the algorithm uses, say, to 2. Additionally, if the predicted μ is less than the actual mean error, the algorithm can increase its interference range when it observes significantly higher number of collisions using the approach to change the collision-group size (cf. Section 1.7.4).

In our simulations, μ takes the following values: $0.0 - 0.4$ and σ takes the following values: $0.0 - 0.2$. And, θ_d is determined using the uniform distribution $U(0, 2\pi)$. Thus, the actual location of the sensor is $\langle a+e_d\cos(\theta_d), b+e_d\sin(\theta_d)\rangle$.

Broadcast. In Figure 1.16, we present the simulations results for broadcast with location errors. Figure 1.16 (a) identifies the percentage of collisions during broadcast. As we can see, when μ increases, the number of collisions increases. However, the collisions are within 2%. Figure 1.16 (b) identifies the maximum delay involved in delivering the broadcast message to all the sensors. We can note that the delay is within 5% when compared to the case where no location errors are introduced. Finally, Figure 1.16 (c) identifies the number of sensors receiving the broadcast message. As we can observe, all the sensors receive the message except for the case where $\mu = 0.2$ and $\sigma = 0.1$. Even in this case, more

than 98% of the sensors receive the broadcast message. Table 1.1 shows the percentage of collision for the case where $\mu = 0.4$ and interference range=2. As we can observe, the percentage of collisions is small with increased interference range.

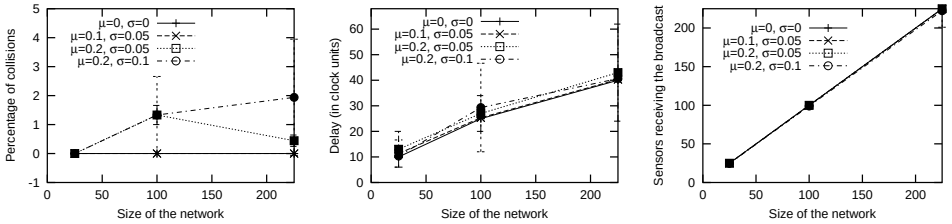


Figure 1.16. Results for broadcast with location errors

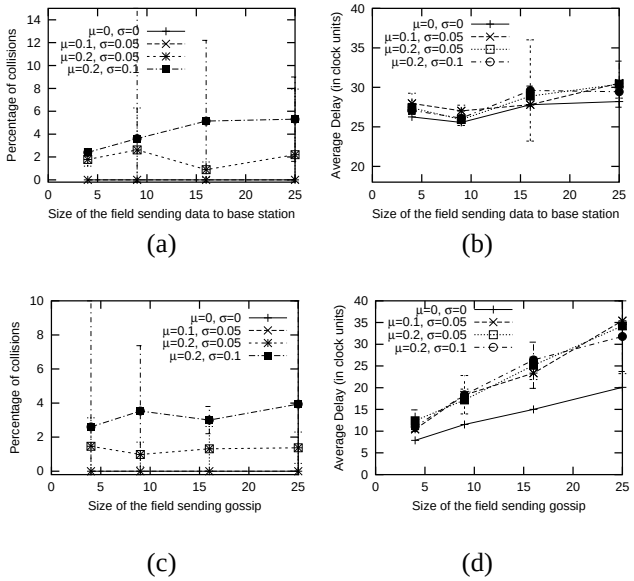


Figure 1.17. Results for convergecast and local gossip with location errors

Convergecast. In Figures 1.17 (a) and 1.17 (b), we present the simulation results for convergecast with location errors. Figure 1.17 (a) identifies the number of collisions during the message communication. We note that, as the error in sensor location increases, collisions increase. Further, as observed earlier, the collisions are within acceptable limits, i.e., within 6%. Figure 1.17 (b) identifies the average delay involved in delivering the convergecast messages to the base station; the average delay is within 5% when compared to the case where were no location errors are introduced. Further, similar to the case where no location errors are present, the base station receives all the convergecast messages. Moreover, if the

mean error distance increases, we can keep the percentage of collisions small by increasing the interference range (cf. Table 1.1).

Local gossip. In Figures 1.17 (c) and 1.17 (d), we present the simulation results for local gossip with location errors. Similar to the observations made earlier in this section, from Figure 1.17 (c), we observe that the number of collisions during message communication is small (i.e., within 4%). Further, the delay involved in delivering the local gossip messages is within 15% when compared to the case where no location errors are introduced. Finally, all the local gossip messages are delivered to the group that expects such messages. Moreover, if the mean error distance increases, we can keep the percentage of collisions small by increasing the interference range (cf. Table 1.1). From these simulations, we conclude that the location errors do not significantly affect the performance of our TDMA service.

Table 1.1. Percentage of collisions for $\mu = 0.4$, $\sigma = 0.2$, and interference range=2

	Broadcast			Convergecast		Local Gossip	
Network Size	Mean	Variance	Field Size	Mean	Variance	Mean	Variance
			4	8.46	2.26	5.15	0.21
25	0	0	9	7.64	1.97	5.35	2.35
100	5	4	16	9.55	18.36	7.82	1.67
225	6.31	2.81	25	10.30	3.17	10.07	14.18

Effect of grouping. In the proposed SS-TDMA service for sensor networks, if the service receives two or more messages and the sum of the message lengths is less than the maximum message length of a TDMA message, SS-TDMA combines these messages into a single TDMA message. In this section, we study the effect of grouping. Specifically, we study the effect of varying the number of messages grouped into a single TDMA message for convergecast and local gossip. Note that in the simulations for broadcast only one message is transmitted, and hence, we do not consider the issue of grouping for broadcast. Further, in this section, we present results for the delay in delivering the messages. In our simulations, the base station (respectively, the group expecting the gossip messages) receives all the convergecast (respectively, gossip) messages. Also, the percentage of collisions is zero.

In Figures 1.18 (a) and 1.18 (b), we present the simulation results for convergecast and local gossip where the communication range is 1 and interference range is 1. We consider the following values for grouping constant (GP): 1, 2, and 4. In Figure 1.18 (a), we can observe that the delay increases when the number of messages grouped into single TDMA message decreases. With $GP = 1$, only one message is sent in a TDMA slot. With $GP = 2$, one or two messages are sent in a slot. In this case, SS-TDMA service will group messages in its queue depending on the total message size. We note that the TDMA message queue will also contain messages generated at other sensors. This is due to the fact that a sensor needs to forward a message generated at other sensors in convergecast. Thus, the grouped messages may contain messages from different sensors and, hence, the delay will be different with different group sizes.

In Figure 1.18 (b), we present the results for local gossip. We observe that the results are similar to convergecast. Specifically, as GP decreases, delay in delivering the local gossip messages increases.

Once again, the results for the case where the communication range is 1 and interference range is 2 are similar (cf. Figures 1.18 (c) and 1.18 (d)).

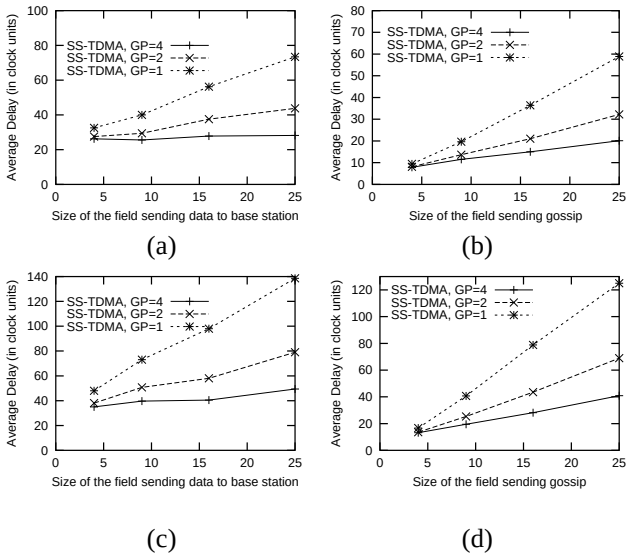


Figure 1.18. Effect of grouping constant; with communication range = 1, interference range = 1, for (a) convergecast and (b) local gossip, and with communication range = 1, interference range = 2, for (c) convergecast and (d) local gossip

1.7 SS-TDMA: EXTENSIONS

So far, we had assumed that sensors are arranged in a rectangular grid as this is the topology used in LITeS [1]. In this section, we discuss some of the extensions to SS-TDMA. Specifically, in Section 1.7.1, we show how SS-TDMA can be extended to support larger communication ranges. In Sections 1.7.2 and 1.7.3, we present SS-TDMA algorithms for other grid-based topologies as well as for arbitrary geometric topologies. Also, we show how this approach can be used to support mobile sensor nodes. And, in Section 1.7.4, we show how SS-TDMA deals with failed sensors. Finally, in Section 1.7.5, we show how SS-TDMA can applied to network programming of sensors [5].

1.7.1 SS-TDMA: Larger Communication Ranges

In this section, we present the TDMA algorithm for communication range$=2$ customized for broadcast pattern. We note that similar extensions are possible for even large communication ranges and other communication patterns (i.e., convergecast and local gossip).

Consider a rectangular grid where a sensor can communicate with its distance 2 neighbors, i.e., communication range = interference range = 2, and $y = \lceil \frac{interference\ range}{communication\ range} \rceil = 1$ (cf. Figure 1.19). The base station is located at the left-top position ($\langle 0, 0 \rangle$) in the network.

Given a sensor location $\langle a, b \rangle$, depending upon whether a, b are even or odd, we can split the network into 4 subgrids, even-even, even-odd, odd-even, and odd-odd. Now, SS-TDMA algorithm from Section 1.4.1 can be used in each of these subgrids. In these subgrids, communication range = interference range = 1. Let the base station (location $\langle 0, 0 \rangle$), located in the even-even subgrid, start the diffusion to assign initial slots for broadcast pattern at

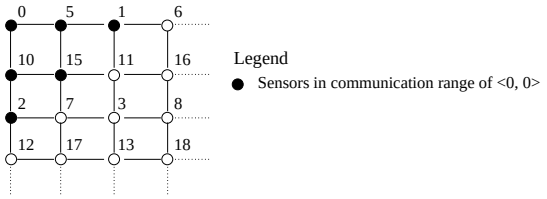


Figure 1.19. TDMA slot assignment in a network with communication range$=2$

slot 0. Sensors $\langle 1,0\rangle$, $\langle 2,0\rangle$, $\langle 0,1\rangle$, $\langle 1,1\rangle$, and $\langle 0,2\rangle$ will receive the message. Figure 1.20 (a) shows when these sensors can forward the diffusion message.

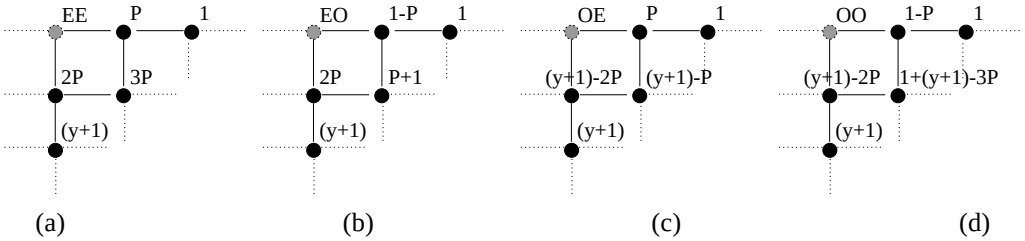


Figure 1.20. Initial slot assignment when the sender is in (a) even-even subgrid, (b) even-odd subgrid, (c) odd-even subgrid, and (d) odd-odd subgrid. The sensors shaded in gray are the senders and the sensors shaded in black are in the communication range of the respective senders.

From Figure 1.20 (a), we observe that sensors $\langle 2,0\rangle$ and $\langle 0,2\rangle$ are allowed to forward the diffusion message after 1 and 2 slot(s) respectively. This is similar to the algorithm in Section 1.4.1 where communication range = interference range = 1. Other sensors transmit in such a way that they do not interfere with the even-even subgrid. Specifically, sensor $\langle 1,0\rangle$ forwards the diffusion after $P=(y+1)^2+1$ slots, i.e., after 5 slots, sensor $\langle 0,1\rangle$ forwards the diffusion message after $2P$ slots, and sensor $\langle 1,1\rangle$ forwards the message after $3P$ slots (cf. Figure 1.19). If the sender is in a even-odd subgrid (respectively, odd-even and odd-odd subgrid), then the initial slots are assigned using the slot assignment specified in Figure 1.20 (b) (respectively, Figures 1.20 (c) and 1.20 (d)).

Once the initial slots are assigned, sensors can determine the future slots using the period between successive slots. The TDMA period for communication range$=2$ in a rectangular grid is $4P$.

Theorem 1.10 The above algorithm satisfies the problem specification of TDMA. □

1.7.2 SS-TDMA: Hexagonal Grids

Consider a hexagonal grid network where a sensor can communicate with its distance 1 neighbors and interfere with its distance 2 neighbors (cf. Figure 1.21). Note that by simple geometry, we can show that the distance between opposite corners of a hexagon is twice as long as an edge of the hexagon. We assume that the base station is located at the left-most corner on the left-top hexagon in the network (cf. Figure 1.21).

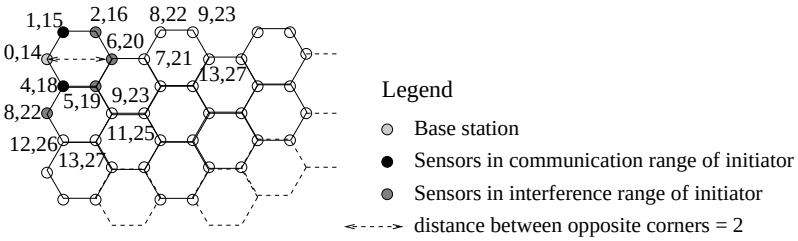


Figure 1.21. TDMA slot assignment in hexagonal-grid network where communication range=1 and interference range=2. The numbers associated with each sensor denote the slots at which it can send a message. Slots for some sensors are not shown.

From Figure 1.21, we observe that whenever the base station transmits, sensors located at the top (say, j) and bottom (say, k) of the base station at geometric distance 1 from the base station can transmit next. However, if both these sensors transmit simultaneously then collision occurs at the base station. Hence, we proceed as follows: whenever j receives the diffusion message from the base station, it retransmits the message after 1 slot. Likewise, whenever k receives the diffusion message from the base station, it retransmits the message after $2y$ slots, where y is the interference range of the sensors. Further, whenever a sensor receives a message from its neighbor on the straight edge (cf. Figure 1.21), it forwards the message after 1 slot.

Once the initial slots are assigned, each sensor can determine future slots based on the time it forwards the diffusion message and the period between successive slots. For a hexagonal grid, the period between successive slots, $P_{hex} = 2y(y+1) + \lfloor \frac{y}{2} \rfloor + 1$ suffices. Thus, the SS-TDMA algorithm for hexagonal grids is shown in Figure 1.22.

```
const P_hex = 2y(y+1) + ⌊y/2⌋ + 1;
// Initial slot assignment in hexagonal grids
when sensor j receives a diffusion message from k
    if (k is at distance 1 in the same level
    (i.e., j − k is a straight edge))
      transmit after 1 slot.
    else if (k is at distance 1 in the lower level)
      transmit after 1 slot.
    else if (k is at distance 1 in the upper level)
      transmit after 2y slots.
    else // duplicate message
      ignore

// TDMA algorithm for broadcast in hexagonal grids
If sensor j transmits a diffusion message at time slot t_j,
    j can transmit at time slots, ∀c : c ≥ 0 : t_j + c * P_hex.
```

Figure 1.22. TDMA algorithm for hexagonal grid

Theorem 1.11 The above algorithm satisfies the problem specification of TDMA. □

We note that the above algorithm is customized for broadcast. We can also customize SS-TDMA on a hexagonal grid for convergecast and local gossip. Towards this end, we need to change the initial slot assignments and the TDMA period P_{hex} similar to the modifications discussed for rectangular grids in Sections 1.4.2 and 1.4.3. Specifically, for convergecast, whenever sensor j receives the diffusion message, it forwards the message in its negative slot (i.e., it forwards the message in $(P_{hex} - t_j)^{th}$ slot, where t_j is the slot in which it is supposed to forward according to the above algorithm). Likewise, for local gossip, similar modifications can be applied.

Remark. We observe that it is possible to convert a triangular grid into a hexagonal grid. Once the hexagonal grid is obtained, SS-TDMA for the hexagonal network can be applied to allot time slots to different sensors. However, in this algorithm, the sensors within the hexagon will not get time slots. We can allow the sensors in the boundary of the hexagon (called *boundary* sensors) to share their time slots with the intermediate sensors. In order to allow boundary sensors to share their TDMA slots with the intermediate sensors, we need to increase the collision-group size. In this case, collision-group of a sensor includes the sensors that are within distance $y + 2$. For more details of this extension, we refer the reader to [14].

1.7.3 Two-Dimensional Clustering

In this section, we discuss how SS-TDMA can be implemented in other geometric 2-D deployments. This approach is based on the virtual grid idea in geographical-adaptive fidelity (GAF) algorithm from [15]. The idea behind this approach is to embed a rectangular grid on the field where the sensors are deployed.

With the help of localization service [10, 11], a sensor can determine its x, y coordinates in the field. Based on these values, the sensor can determine the square in which it is present in the grid embedded on the field. Once the sensor knows its location in the grid, it can determine its communication slots using the algorithm presented in Section 1.4. To ensure that the sensors in neighboring squares in the grid can communicate with each other, the distance between the sensors in the neighboring squares should be less than or equal to the communication range of the sensors. If two sensors fall on the longest diagonal between the neighboring squares then the distance between these two sensors is $\sqrt{5}r$, where r is the length of the square. Hence, we need to ensure that the length of the square, $r \leq \frac{communication\ range}{\sqrt{5}}$ in order to allow these two sensors to communicate. With this embedding, we now obtain a 2-D grid where some sensors have failed; such a scenario will occur if there is no sensor that falls in the given $r * r$ square. Hence, we can use the extension considered in Section 1.7.4 to obtain the TDMA service for such a network.

In this scheme, however, two or more sensors may fall into the same square. Hence, we need other mechanisms that allows these sensors to collaborate among themselves on deciding how to share the communication slots assigned to that square. Since the number of sensors that fall in the same square are expected to be small, a simple protocol can be easily designed to decide how these sensors will share their TDMA slots. However, this issue is outside the scope of this chapter.

Remark. In most applications, if more than one sensor is present in a square, these sensors provide redundant information. Hence, it is preferable that only one sensor is active at any

instant of time. Periodically, the active sensor can delegate its role to other sensors in the square. Remaining sensors can thus turn off their radio and conserve energy.

Supporting mobile sensor nodes. SS-TDMA can be applied to mobile sensor networks provided localization service [10, 11] is available. In presence of mobile sensor nodes, time slots used by a sensor keeps on changing due to its motion. Hence, a sensor needs to know its current position. In other words, SS-TDMA can be used in mobile sensor networks if localization service is available.

1.7.4 Dealing with Failed Sensors

In this section, we focus on providing TDMA service in the presence of failed/sleeping sensors. We assume that the base station does not fail and that the network remains connected.

In SS-TDMA, a sensor normally receives the diffusion message for the first time from a sensor that is closer to the base station. In presence of failed/sleeping sensors, a sensor may receive the diffusion message for the first time from the sensor that is (physically) farther away from the base station. SS-TDMA ignores such message as duplicate. However, in presence of failed/sleeping sensors, a sensor should forward such a diffusion message. This ensures that the diffusion message reaches all the active sensors. The algorithm for the initial slot assignment is in Figure 1.23.

```
when sensor j receives a diffusion message for the first time from sensor k
    update local clock;
    determine the ideal diffusion slot;
    find the TDMA slots using the appropriate algorithm from Section 1.4;
    transmit in ideal diffusion slot or next TDMA slot, whichever is earlier;
```

Figure 1.23. Initial slot assignment while dealing with failed sensors

This modification, however, also assigns slots to failed/sleeping sensors. Hence, if the number of such failed/sleeping sensors is large, then the bandwidth is wasted. To improve the bandwidth utilization, we consider the problem of reducing the collision-group size to deal with the sensors that are sleeping or have failed. Our solution involves three tasks: (1) allowing each sensor to determine its collision-group, (2) computing the maximum collision-group size (MCG) in the network, and (3) communicating the MCG to all sensors

Determining collision-group of each sensor. Regarding the first part, if a sensor, say j, plans to be inactive for a long time, it should inform the sensors in its collision-group before it becomes inactive. This can be achieved as follows: When j wants to become inactive, it informs its neighbors (in a slot assigned by SS-TDMA). These neighbors, in turn, inform their neighbors until the information reaches all sensors in the collision-group of j. Alternatively, if j fails (or becomes inactive without informing its neighbors), its neighbors can detect this fact by observing that no communication was received in the slot allotted to j. This information, in turn, will be communicated to the sensors in the collision-group of j. A sensor, say k, updates its collision-group to max (collision-group of k, $\forall i : |t_i - t_k|$ where sensor i is in the collision-group of k and t_i is the time slot at which sensor i transmits a diffusion message).

Computing the MCG. Regarding the second part, we use the initial slot assignment algorithm for broadcast (cf. Section 1.4.1) where the communication range is 1 and the interference range can be greater than 1. Once again, for simplicity, we assume that left-top sensor ($\langle 0, 0 \rangle$) communicates the size of the collision-group when it initiates the diffusion.

Whenever a sensor, say j, propagates the diffusion, it sets the collision-group to max (collision-group included in a message that was received by j, collision-group of j). It follows that the sensor in the right-bottom corner will be able to obtain the MCG in the network. This MCG can then be communicated to the left-top sensor using the current collision-free SS-TDMA algorithm. If the right-bottom sensor has failed, this responsibility can be delegated to other sensors.

Communicating the MCG. Finally, regarding the third part, once the left-top sensor learns the new collision-group, it can include this when it initiates the next diffusion. This diffusion will allow the sensors to learn the size of the new collision-group that will then be used by SS-TDMA.

We would like to note that the above description is intended to show that it is possible to change the size of the collision-group to ensure optimal bandwidth utilization. The parameters involved in changing the collision-group are the frequency with which sensors update their collision-group and the frequency with which the sensor(s) at the right-bottom communicate the group change information. Also, it is possible to accelerate the change using the distributed reset [16, 17]. However, this issue is outside the scope of this chapter.

Improving bandwidth utilization further. Bandwidth utilization in SS-TDMA can improved further by the following techniques. First, the time slot interval used in SS-TDMA can be increased. Now, time slot is divided into two phases; listen phase and send phase. Whenever a sensor wants to send data in its assigned time slot, it will start transmitting at the beginning of the listen phase. With this modification, a sensor (say, j) that requires more bandwidth will listen in the slots assigned to other sensors in its collision-group. If j notices that medium is free in the listen phase of a time slot (assigned to a sensor, say, k), j can use the send phase of this slot to send its data. However, collisions may occur with this scheme, since two or more sensors in the collision-group of k can try to access the medium in the send phase simultaneously. Note that the send phase is long enough to transmit a message, and the listen phase is small enough so that other sensors can detect whether the medium is free or not.

Second, some of the slots can be left unassigned. With this modification, whenever a sensor (say, j) requires more bandwidth, it can choose to send the data in the slots not used by any sensor in its collision-group. Again, no guarantees can be made about this communication. Another approach is that sensors can collaborate among themselves to dynamically allocate the unassigned slots. With this modification, unassigned slots can be requested by a sensor that requires more bandwidth. Based on a collaborative decision making algorithm, future slots can be assigned to a sensor. This guarantees that the communication is collision-free.

1.7.5 SS-TDMA: Application to Network Programming

In this section, we discuss how SS-TDMA can be applied to network programming of sensors. As mentioned earlier, sensor network applications often include thousands of sensors that will be deployed in a very large hostile field. Programming/upgrading the software in these sensors is difficult. To deal with this problem, wireless programming [5] is used. However, the solution in [5] is applicable only to single-hop networks. In other words, this solution is useful only if the sensors are one-hop away from the base station. Hence, we need an approach for multi-hop wireless programming of sensors, where intermediate sensors also forward the program. Some of the existing multi-hop network programming solutions include multi-hop over-the-air programming (MOAP) [18], Trickle [19], and multi-hop network reprogramming (MNP) [6]. These approaches are based on CSMA and,

hence, rely on advertisement/request and back-off mechanisms for propagating the code in the network. However, this approach can delay programming some sensors considerably.

SS-TDMA can be effectively used to perform network programming. Specifically, we can extend the single-hop programming solution in [5] using SS-TDMA to reprogram a multi-hop network. Towards this end, we make the following enhancements to [5]. Whenever a sensor receives a program capsule (code segment containing 16 bytes of instructions), it stores the program in its secondary storage or EEPROM in the appropriate address. Additionally, it buffers the capsule in SS-TDMA's internal queue. SS-TDMA will forward the capsule in the sensor's alloted time slot. Hence, in this approach, a sensor need not wait to get all/part of the program before forwarding. Therefore, program capsules are sent in a pipeline. Moreover, all the sensors will be reprogrammed approximately at the same time. Theoretically, to reprogram a $n * n$ network with a program containing x capsules, SS-TDMA takes $(x-1) * P_b + 2(n-1) * P_b$ amount of time, where P_b is the period between successive slots in broadcast. As a result of pipelining, the last $x-1$ capsules can be forwarded within $(x-1) * P_b$, since the base station will send one capsule per period. The first capsule takes $2(n-1) * P_b$ to reach the last or the bottom-right sensor in the grid. For time slot $= 30ms$, interference range $= 2$, $P_b = 10*$time slot$= 300ms$, a 1000 capsule program can be forwarded in a 10x10 network within 305.1 seconds or 5.1 minutes.

Currently, we have implemented network programming using SS-TDMA in MICA motes [3, 2]. Other than the issue of collision considered in this chapter, SS-TDMA deals with other causes of message loss such as environmental factors. SS-TDMA provides an ability to obtain implicit ACKs; whenever a sensor forwards a message, it acts as an implicit ACK for the predecessor. The network programming service utilizes these implicit ACKs to ensure that no packets are lost and to perform retransmissions only when necessary. The details of this service, however, is outside the scope of this chapter.

1.8 RELATED WORK

In this section, we discuss the related work on MAC protocols for sensor networks. To deal with the problem of message collision, approaches like collision-avoidance and collision-freedom based MAC protocols are proposed.

Collision-avoidance protocols. Collision-avoidance protocols like carrier sense multiple access (CSMA) [20, 21] try to avoid collisions by sensing the medium before transmitting a message. If the medium is busy then the protocol retries after a random exponential back-off period. Another example of collision-avoidance protocol is carrier sense multiple access and collision detection (CSMA/CD). CSMA/CD [21] is difficult to use in the context of sensor networks as the collisions are often detected at some receivers whereas other receivers and sender(s) may not detect the collision.

In [22], Ye, Heidemann, and Estrin propose an energy-efficient sensor-MAC (S-MAC) protocol. The authors identify the main sources of energy-waste, namely, collisions, overhearing, idle-listening, and control packet overhead. In order to reduce the energy consumption, S-MAC uses periodic listen and sleep cycle and IEEE 802.11 style RTS/CTS/Data/ACK sequence for communication. S-MAC minimizes the number of data-collisions by using RTS/CTS control signals before transmitting the data. It reduces the energy spent on overhearing by scheduling network-allocation vector (NAV) whenever a sensor overhears RTS/CTS not associated with itself. Until NAV fires, S-MAC allows the overhearing sensor to sleep. Further, S-MAC reduces idle-listening by restricting communication to occur only in the receiver's scheduled listen interval. Finally, it reduces the control overhead by trans-

mitting small packets like RTS, CTS, or ACK. Timeout-MAC (T-MAC) [23] is an extension to S-MAC that allows adaptive duty cycle to handle load variations in the network unlike the fixed duty cycle operation in S-MAC.

S-MAC differs from SS-TDMA in a number of ways. First, SS-TDMA does not use IEEE 802.11 style control for communication. Overhearing and idle-listening are not a problem in SS-TDMA, since a sensor will listen only in the time slots alloted to its neighbors. And, protocol control overhead is very minimum in SS-TDMA. SS-TDMA just requires the base station to send periodic diffusing computations to validate the slots assigned to the sensors. However, the frequency of such diffusions can be low, since time synchronization service takes care of most clock drifts in the sensors. We emphasize that while time synchronization service is necessary for both S-MAC and SS-TDMA, SS-TDMA requires coarse-grained synchronization. S-MAC requires fine-grained synchronization, since whenever a sensor wants to send a message, it needs to know when to send RTS and when to expect the corresponding CTS. Towards this end, the listen interval in S-MAC is split into three parts; first part for SYNC exchanges, which allows a sensor to discover the existence of other sensors and to know their listen/sleep schedules, second part for RTS exchanges, and third part for CTS exchanges. Since SYNC/RTS/CTS packets are small, the interval for such control packet transmissions is small. Hence, a sensor needs to have a highly precise timing information.

Collision-free protocols. Collision-freedom protocols like frequency division multiple access (FDMA), code division multiple access (CDMA), and time division multiple access (TDMA) ensure that collisions do not occur while the sensors communicate. FDMA [21] ensures collision-freedom by allotting different frequencies for the sensors. FDMA is not applicable in the context of sensor networks since the sensors (e.g., MICA motes [3, 2]) are often restricted to transmit only on one frequency. CDMA [24] requires that the codes used to encode the message be orthogonal to each other so that the destination can separate different messages. Thus, CDMA requires expensive operations for encoding/decoding a message. Therefore, CDMA is not preferred for sensor networks that lack the special hardware and that have limited computing power.

TDMA ensures collision-freedom by allotting time slots for communication. TDMA based protocols can be classified as either randomized or deterministic protocols, based on the way time slots are alloted to different sensors or the startup algorithm works. Randomized TDMA protocols include [25, 26, 27]. And, deterministic TDMA protocols include [28, 29].

Randomized startup. In [25], Claesson, Lönn, and Suri propose a randomized startup algorithm for TDMA. Whenever a collision occurs during startup (synchronization phase), exponential back-off is used for determining the time to transmit next. In contrast, SS-TDMA uses a self-stabilizing deterministic algorithms for assigning the initial slots. Further, the complexity of the algorithm in [25] is $O(N)$, where N is the number of system nodes, whereas the complexity of the initial slot assignment algorithm in SS-TDMA is $O(D)$, where D is the diameter of the network. Moreover, an important assumption in [25] is that each node has a unique message length.

In [26, 27], initially, nodes are in random-access mode and TDMA slots are assigned to the nodes during the process of network organization. Specifically, in Low-Energy Adaptive Clustering Hierarchy (LEACH) [27], clusters are formed and each cluster elects a cluster head. All the nodes in the network are assumed to have enough radio power to communicate with the base station. However, only the cluster-heads are allowed to communicate with the base station directly (single-hop). Other nodes should communicate with their cluster-head in order to forward a message to the base station. To achieve this hierarchy, non cluster-

head nodes reduce their radio power in a such a way that they communicate only with their cluster-heads. Inter-cluster interference is avoided by using spread spectrum or CDMA.

LEACH differs from SS-TDMA in that the slot assignment in LEACH is randomized. Further, LEACH requires CDMA to prevent inter-cluster interference. By contrast, SS-TDMA does not need CDMA. Further, SS-TDMA does not assume that all sensors can communicate with the base station, which is reasonable in a large sensor field.

Deterministic startup. In [28], Arisha, Youssef, and Younis propose a clustering scheme to allot time slots to different sensors. Each cluster has a gateway node. The gateway node informs each sensor in its cluster about the time slots in which the sensors can transmit messages and also, the time slots in which the sensors should listen. In this algorithm, slot assignment is performed by the gateway and communicated to different sensors. This approach assumes that a node can act as a sensor and/or a gateway. If a node acts as a sensor, it gets one slot in every period, where period depends on the number of the nodes in the network. And, if a node acts as a gateway, it gets multiple-slots in every period depending on the number of its children. The approach in [28] differs from SS-TDMA in that in SS-TDMA slot assignment is uniform. The period in SS-TDMA depends on the collision-group size (or the interference range) unlike the approach in [28].

1.9 CONCLUSION AND FUTURE WORK

In this chapter, we presented SS-TDMA and showed that it can be customized for broadcast, convergecast, and local gossip. While SS-TDMA is designed for grid-based topologies, we showed how we can extend it to deal with non-grid topologies, mobile sensor nodes, and failed sensors. Thus, SS-TDMA can deal with commonly occurring difficulties, e.g., failed sensors, sleeping sensors, unidirectional links, and unreliable links, in sensor networks.

As discussed in Section 1.4, we recommend that if the application requirements are unknown, then the TDMA service for the local gossip be used. Towards this end, we observe that the period used for local gossip is twice that for the case of broadcast/convergecast. Hence, it is possible that initiator(s) of broadcast/convergecast suffer extra delay when the local gossip solution is used. However, once the initiator sends its message, subsequent relaying occurs quickly. This is due to the fact that the solution for local gossip also ensures that the communication patterns such as broadcast and convergecast incur small delays at intermediate sensors. Thus, the solution for local gossip provides substantial benefit to broadcast and convergecast. In fact, SS-TDMA optimized for local gossip also enables a sensor to send data in any given direction in such a way that the delay incurred by the data at intermediate sensors is small.

SS-TDMA is energy-efficient and it also allows sensors to save more power by turning off the radio completely as long as the remaining sensors remain connected. These sleeping sensors can periodically wake up, wait for one diffusion message from one of its neighbors and return to sleeping state. This will allow the sensors to save power as well as keep the clock synchronized with their neighbors. Since SS-TDMA is stabilizing fault-tolerant, if all sensors are deactivated for a long time causing arbitrary clock drift, SS-TDMA ensures that eventually the diffusion will complete successfully and collision-free communication will be restored.

One of the important issues in SS-TDMA is the *exposed terminal* problem. For example, consider 4 sensors $A, B, C,$ and D arranged in a line. If B wants to send a message to A and C wants to send a message to D, either B or C will have to suppress their communication in SS-TDMA, since D (respectively, A) belongs to the collision-group of B (respectively,

C). One way to overcome this problem is as follows. Unlike vertex-coloring in SS-TDMA (where, slots are assigned to the individual sensors), slots are assigned to individual edges in the network (edge-coloring). Now, B and C can simultaneously send a message to A and D respectively. However, edge-coloring is expensive in the sense that if a sensor wants to send a broadcast message, it has to send up to d messages, where d is the number of its neighbors.

Another important issue in SS-TDMA is to determine the communication and interference range of a sensor. There are several ways to achieve this. Initially, we overestimate the interference range by considering the manufacturer specification about the ability of sensors to communicate with each other. Subsequently, we can use the biconnectivity experiments [30] to determine appropriate communication range and appropriate interference range. Given any two sensors, j and k, these results allow these sensors to determine the probability that j can communicate with k and the probability that k can communicate with j. Using these results, we can update the communication range and the interference range: j is in the communication range of k iff min(probability with which j can communicate with k, probability with which k can communicate with j) exceeds a certain threshold. And, j is in the interference range of k iff max(probability with which j can communicate with k, probability with which k can communicate with j) exceeds another threshold. Using the approach in Section 1.7.4 for dealing with failed sensors, we can communicate the communication range and interference range of all sensors to the base station. Base station can then change the communication range and interference range appropriately.

We have combined SS-TDMA with previous algorithms on time synchronization (e.g., [12]). SS-TDMA and time synchronization complement each other. SS-TDMA is useful in ensuring that messages sent for time synchronization do not collide. And, time synchronization helps to reduce the clock drift and to ensure that the drift does not cause TDMA slots of nearby sensors to overlap.

We have combined SS-TDMA with previous work on *implicit* acknowledgments [1]. We expect that for known communication patterns such as broadcast, convergecast and local gossip, combining SS-TDMA with implicit acknowledgments will be especially useful. In these communication patterns, when sensor j transmits a message to k, k is expected to retransmit it to its successor (unless k is the last sensor to receive that communication). Since message sent by k is broadcast to all its neighbors, j can also hear that message. Thus, the retransmission by k acts as an implicit acknowledgment for j. With SS-TDMA, j can wait until the TDMA slot assigned to k; if k does not transmit in that slot, j can conclude that k did not receive its message. Thus, j can reduce the power spent in waiting for the implicit acknowledgment by listening to the radio only in the TDMA slot for k.

There are several questions raised by this work: First, an interesting question is how to determine the initial sensor that is responsible for initiating the diffusion. In some heterogeneous networks where some sensors are more powerful and more reliable, these powerful/reliable sensors can be chosen as the base station. Alternatively, during deployment of sensors (e.g., by dropping them from a plane), we can keep several potential base stations that communicate with each other directly and use the approach in [31, 32] so that one of them is chosen to be the initiator. Another interesting issue is regarding the convergecast communication pattern. As mentioned in Section 1.4.2, during convergecast, a bottleneck is created near the base station. An extension to SS-TDMA then is to account for this bottleneck by allotting more slots to the sensors near the base station compared to the rest of the network.

Acknowledgments. This work was partially sponsored by NSF CAREER CCR-0092724, DARPA Grant OSURS01-C-1901, ONR Grant N00014-01-1-0744, NSF Equipment Grant EIA-0130724, and a grant from Michigan State University.

REFERENCES

1. A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y-R. Choi, T. Herman, S. S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A line in the sand: A wireless sensor network for target detection, classifi cation, and tracking. *Computer Networks (Elsevier)*, 2004, to appear.
2. J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. Pister. System architecture directions for network sensors. *In Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, November 2000.
3. D. E. Culler, J. Hill, P. Buonadonna, R. Szewczyk, and A. Woo. A network-centric approach to embedded software for tiny devices. In *EMSOFT*, volume 2211 of *Lecture Notes in Computer Science*, pages 97–113. Springer, 2001.
4. A. Mairwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. *In Proceedings of the ACM International Workshop On Wireless Sensor Networks and Applications (WSNA)*, 2002.
5. TinyOS: A component-based OS for the networked sensor regime. `http://www.tinyos.net`. Latest source available at: `http://sourceforge.net/cvs/?group_id=28656`.
6. S. S. Kulkarni and L. Wang. MNP: Multihop network reprogramming service for sensor networks. Technical Report MSU-CSE-04-19, Department of Computer Science, Michigan State University, May 2004.
7. S. S. Kulkarni and M(U). Arumugam. TDMA service for sensor networks. *In Proceedings of the Third International Workshop on Assurance in Distributed Systems and Networks*, March 2004.
8. E. W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Communications of the ACM*, 17(11), 1974.
9. S. Dolev. *Self-Stabilization*. The MIT Press, 2000.
10. G. Agha, W. Kim, Y. Kwon, K. Mechitov, and S. Sundresh. Evaluation of localization services (preliminary report), 2003. DARPA NEST Program. Available at: `http://osl.cs.uiuc.edu/docs/nest-localization-report-2003/`.
11. T. He, C. Huang, B. Blum, J. Stankovic, and T. Abdelzaher. Range-free localization schemes for large scale sensor networks. *In Proceedings of the 9th Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 81–95, 2003.
12. T. Herman. NestArch: Prototype time synchronization service. `http://www.ai.mit.edu/people/sombrero/nestwiki/index/ComponentTimeSync`, January 2003.
13. G. Simon, P. Volgyesi, M. Maroti, and A. Ledeczi. Simulation-based optimization of communication protocols for large-scale wireless sensors networks. *In Proceedings of the IEEE Aerospace Conference*, March 2003.
14. U. Arumugam. Collision-free communication in sensor networks. Master's thesis, Computer Science and Engineering, Michigan State University, September 2003. Available at: `http://www.cse.msu.edu/~arumugam/research/MastersThesis/main.ps`.
15. Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. *In Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 70–84, July 2001.

16. A. Arora and M. Gouda. Distributed reset. *IEEE Transactions on Computers*, 43(9):1026–1038, 1994.

17. S. S. Kulkarni and A. Arora. Multitolerance in distributed reset. *Chicago Journal of Theoretical Computer Science*, 1998.

18. T. Stathopoulos, J. Heidemann, and D. Estrin. A remote code update mechanism for wireless sensor networks. Technical Report CENS-TR-30, University of California, Los Angeles, Center for Embedded Networked Computing, November 2003.

19. P. Levis, N. Patel, S. Shenker, and D. Culler. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor network. *In Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI)*, 2004.

20. A. Woo and D. Culler. A transmission control scheme for media access in sensor networks. *In Proceedings of the Seventh Annual International Conference on Mobile Computing and Networking*, pages 221–235, 2001.

21. R. Rom and M. Sidi. *Multiple Access Protocols: Performance and Analysis*. Springer-Verlag, 1989. Also available at: `http://www.comnet.technion.ac.il/rom/PDF/MAP.pdf`.

22. W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac for wireless sensor networks. *In Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1567–1576, June 2002.

23. T. van Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. *In Proceedings of the First International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 171–180, November 2003.

24. A. J. Viterbi. *CDMA: Principles of Spread Spectrum Communication*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, 1995.

25. V. Claesson, H. Lönn, and N. Suri. Efficient TDMA synchronization for distributed embedded systems. *In Proceedings of the 20th IEEE Symposium on Reliable Distributed Systems (SRDS)*, pages 198–201, October 2001.

26. K. Sohrabi and G. J. Pottie. Performance of a novel self-organization protocol for wireless ad-hoc sensor networks. *In Proceedings of the IEEE Vehicular Technology Conference*, pages 1222–1226, 1999.

27. W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, 1(4):660–670, October 2002.

28. K. Arisha, M. Youssef, and M. Younis. Energy-aware TDMA-based MAC for sensor networks. *In Proceedings of the IEEE Workshop on Integrated Management of Power Aware Communications, Computing and Networking (IMPACCT)*, May 2002.

29. S. S. Kulkarni and U. Arumugam. Collision-free communication in sensor networks. *In Proceedings of the Sixth Symposium on Self-stabilizing Systems (SSS), Springer*, LNCS:2704:17–31, June 2003.

30. Y-R. Choi, M. Gouda, M. C. Kim, and A. Arora. The mote connectivity protocol. *In Proceedings of the 12th International Conference on Computer Communications and Networks*, pages 533–538, October 2003.

31. L. Gasieniec, A. Pelc, and D. Peleg. The wakeup problem in synchronous broadcast systems. *SIAM Journal of Discrete Mathematics*, 14(2):207–222, 2001.

32. B. S. Chlebus, L. Gasieniec, A. Gibbons, A. Pelc, and W. Rytter. Deterministic broadcasting in ad hoc radio networks. *Distributed Computing*, 15(1):27–38, 2002.