Collision-free Communication in Sensor Networks*

Appears in the Symposium on Self-Stabilizing Systems (SSS) 2003, ©Springer-Verlag, LNCS:2704

Sandeep S. Kulkarni and Umamaheswaran Arumugam

Software Engineering and Networks Laboratory
Department of Computer Science and Engineering
Michigan State University
East Lansing MI 48824 USA
Email: {sandeep, arumugam}@cse.msu.edu
Web: `http://www.cse.msu.edu/~{sandeep,arumugam}`

Abstract. In this paper, we provide a stabilizing solution for collision-free diffusion in sensor networks. Such diffusions are often necessary in sensor networks when information from one sensor needs to be communicated to other sensors that satisfy certain geographic properties. Our solution deals with several difficulties, e.g., unidirectional links, unreliable links, long links, failed sensors, and sensors that are sleeping in order to save energy, that occur in sensor networks. It also ensures that there are no collisions during the diffusion and that the time required for the diffusion is $O(D)$ where D is the diameter of the network. Moreover, while the solution can be applied to an arbitrary topology, it is more suitable for a commonly occurring topology, a two-dimensional grid.
We show how our solution for collision-free diffusion can be used for time-division multiplexing (TDM) in sensor networks. TDM ensures that the message communication (other than the messages sent by diffusion) among sensors is also collision-free. While collision-free diffusion and time-division multiplexing are interdependent, we show how both these properties can be achieved simultaneously. Our algorithms are stabilizing fault-tolerant, i.e., collision-free diffusion and time-division multiplexing are restored even if the system reaches an arbitrary state where the sensors are corrupted or improperly initialized.

1 Introduction

In recent years, sensor networks have become popular in the academic and industrial environment due to their application in data gathering, active and passive tracking of unexpected/undesirable objects, environment monitoring and unattended hazard detection. Due to their low cost and small size, it is possible to

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

rapidly deploy them in large numbers. These sensors are resource constrained and can typically communicate with other (neighboring) sensors over a wireless network. However, due to limited power and communication range, they need to collaborate to achieve the required task.

One of the important issues in sensor networks is message collision: Due to the shared wireless medium, if a sensor simultaneously receives two messages then they collide and, hence, both messages become incomprehensible. Such collision is undesirable as it results in wastage of power to transmit the message that resulted in a collision.

Collision among messages is especially problematic in the context of *system-wide computations* where some sensor needs to communicate some information to the entire network (respectively, a subset of the network that satisfies the geographic properties of interest). Such computations arise when a sensor needs to communicate the observed value to the *base station* or when we want to organize these sensors in a suitable topology (e.g., tree). In such diffusion, every sensor that receives a message transmits it to its neighbors (in the given direction). It follows that at any time multiple sensors may be forwarding the sensor values to their respective neighbors. Therefore, the possibility of a collision increases.

Challenges in sensor networks. One of the important issues in sensor networks is the scenario where two sensors can communicate with each other only with a very low probability. Also, sensor networks suffer from unidirectional links where one sensor can communicate with another with a high probability although the probability of the reverse communication is very low. In a situation where sensor j can only communicate occasionally with sensor k, it is expected that the signal strength received by k is so small that k cannot correctly determine all the bits in that message. However, the signal strength that k receives is often strong enough that it can corrupt another message that was sent to k at the same time. The above discussion suggests that we need to consider the situation where two sensors cannot effectively *communicate* with each other although they can effectively *interfere* with each other.

A collision-free diffusion is advantageous in obtaining clock synchronization and time-division multiplexing. More specifically, when a sensor receives the diffusion message, it can uniquely determine its clock by considering the clock value when the diffusion was initiated and the path that the diffusion took to reach that sensor. If clocks are synchronized, we can assign slots to each sensor such that simultaneous message transmissions by two sensors do not collide.

A closer inspection of diffusion and time-division multiplexing shows that these two problems are interdependent. More specifically, if clocks are not synchronized then the diffusion may fail in the following scenario: the clock values of two sensors differ, one of these sensors transmits the diffusion message and the other sensor transmits an unrelated message at the same time. It follows that a collision in this situation will prevent the diffusion message from reaching all the desired destinations. Moreover, if the diffusion does not complete successfully then the clocks may remain unsynchronized forever. It is therefore necessary that diffusion be stabilizing fault-tolerant [1], i.e., starting from an arbitrary state,

the system should recover to states from where subsequent diffusing computation is collision-free. A stabilizing fault-tolerant solution also deals with the case where the sensors are inactive for a long time and subsequently become active, although at slightly different times.

Contributions of the paper. With the above motivation, in this paper, we focus on stabilizing collision-free algorithm for diffusing computations in sensor networks. The main contributions of this paper are as follows.

1. We present an algorithm (with 4 versions) for collision-free diffusion in sensor networks. The first version focuses on an ideal sensor network where a sensor communicates perfectly with the neighbors in its grid and does not interfere with any other sensors in the network. Our second version improves the performance of the first version for the case where a sensor can communicate with other non-neighboring sensors at some distance, say x. The third version extends the first version to deal with the case where a sensor interferes with other sensors at some distance, say y. Finally, the fourth version combines the second and the third versions to deal with the case where a sensor may communicate with other sensors at distance, say x, and interfere with sensors at a larger distance, say y.
2. We show how the algorithm for collision-free diffusion enables us to obtain time-division multiplexing in sensor networks. In a two-dimensional grid, we show that messages of two sensors that transmit simultaneously do not collide with each other.
3. We show how collision-free diffusion can be obtained if some sensors have failed or have been shut off to save power.
4. We show how stabilizing fault-tolerance can be added to our algorithms. Thus, starting from an arbitrary state, each of our algorithms recovers to states from where collision-free diffusion and time-division multiplexing are achieved. Finally, these algorithms can also be tailored to deal with the case where communication between (even the neighboring) sensors is unreliable.

Organization of the paper. The rest of the paper is organized as follows: In Section 2, we discuss the model of the sensor network. Then, in Section 3, we present our algorithm for collision-free diffusion. In Section 4, we extend our algorithm to provide time-division multiplexing and show how both these properties can be achieved even if one begins in a state where the sensor clocks are not synchronized. Subsequently, in Section 5, we discuss extensions of our algorithms. Finally, in Section 6, we discuss the related work and make concluding remarks in Section 7.

2 Model and Assumptions

In this section, we present the system model and identify the assumptions made in this paper. 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. Also, we assume that each sensor is aware of its communication range and an interference range. Note that, interference range is greater than or equal to the communication range.

Additionally, we assume that one clock tick of the sensor corresponds to the propagation time of a message. Though the sensors can have high-precision clocks, for communication, we use only the higher-order bits that correspond to the propagation time of a message.

We assume that there is exactly one initiator that initiates the diffusion. For simplicity, initially, we assume that the sensor at the left-top (at location $\langle 0, 0 \rangle$) is the initiator. This assumption is made since we can view the diffusion as propagating from this sensor to all the sensors in the network in one single (south-east) quadrant. We remove this assumption in Section 3.5 and provide an algorithm where the initiator is not at the left-top position.

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 initiator) have failed.

3 Collision-free Diffusion

In this section, we provide an algorithm for collision-free diffusion for sensors arranged in a two-dimensional grid. As mentioned in the Introduction, whenever a sensor receives a diffusion message for the first time, it retransmits the message to its neighbors. However, if two (nearby) sensors transmit the diffusion message at the same time then the messages collide. The collision becomes even more problematic in sensor networks where it is often not possible to detect whether collision has occurred or not. Also, it is possible that message sent by one sensor collides at one receiver whereas another receiver correctly receives it.

To deal with these problems, we define the problem of collision-free diffusion as follows. The first requirement for collision-free diffusion is that the diffusion message should reach every sensor. The second requirement is that collisions should not occur. More specifically, when sensor j transmits a message, it is necessary that a collision should not occur at any sensor that is expected to receive the message from j, i.e., a sensor in the communication range of j. Thus, if sensor k transmits concurrently then the set of sensors in the communication range of j should be disjoint from the set of sensors in the interference range of k. Thus, the problem statement is defined as follows:

Problem Statement: Collision-free Diffusion
Given a sensor grid; if a sensor initiates diffusion then the following properties should be satisfied:

1. Diffusion message should reach every sensor.
2. If two sensors j and k transmit at the same time,
   (Sensors in communication range of j) $\bigcap$
   (Sensors in interference range of k) $= \emptyset$.

We present four versions of our collision-free diffusion algorithm. For simplicity, in Sections 3.1-3.4, we assume that the sensor at $\langle 0, 0 \rangle$ initiates the diffusion. In Section 3.1, we discuss the algorithm for diffusion in networks where a sensor can communicate only with its distance 1 neighbors. In Section 3.2, we extend this version for diffusion in networks where a sensor can communicate with its distance $x, x \geq 1$, neighbors. In both these algorithms, we assume that the interference range of a sensor is same as its communication range. We weaken this requirement in Sections 3.3, and 3.4. Specifically, in Section 3.3, we extend the first version (cf. Section 3.1) to deal with the case where a sensor can communicate with its distance 1 neighbors and interfere with its distance $y, y \geq 1$, neighbors. And, in Section 3.4, we extend the third version (cf. Section 3.3) to deal with the case where a sensor can communicate with its distance $x, x \geq 1$, neighbors and interfere with its distance $y, y \geq x$, neighbors. In Section 3.5, we provide an algorithm for collision-free diffusion initiated by an arbitrary sensor. Although in Sections 3.1-3.5 we assume that the communication range (respectively, interference range) of all sensors are identical, observations made in Section 3.6 show that the algorithm can be applied even if they are different.

3.1 Version 1: Communicate 1, Interfere 1

Consider a simple grid network where a sensor can communicate with sensors that are distance 1 away (cf. Figure 1)[1].

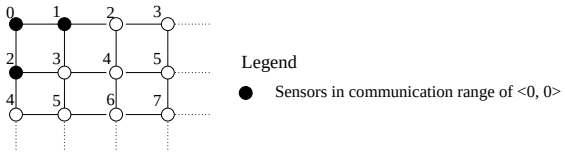

Fig. 1. Sample diffusion in networks where a sensor communicates with its distance 1 neighbors. The number associated with a sensor shows the slot in which it should transmit.

[1] In these examples we have used the manhattan distance between sensors. Our algorithms can be applied even if we consider the geographic distance between sensors. See the first observation in Section 3.6.

From this figure, we observe that sensors $\langle 1,0\rangle$ and $\langle 0,1\rangle$ should not transmit at the same time as their messages will collide at sensor $\langle 1,1\rangle$. The following algorithm provides a collision-free diffusion in networks where sensors can communicate only with their distance 1 neighbors.

```
when sensor j receives a diffusion message from sensor k
      if (k is west neighbor at distance 1)
        transmit after 1 clock tick.
      else if (k is north neighbor at distance 1)
        transmit after 2 clock ticks.
      else // duplicate message received from east/south neighbor
        ignore
```

Theorem 3.1 The above algorithm satisfies the problem specification of collision-free diffusion.

Proof. The proof is similar to that of Theorem 3.3 where $y=1$. □

3.2 Version 2: Communicate x, Interfere x

Consider a grid network where a sensor can communicate with sensors that are distance $x, x \geq 1$ away (cf. Figure 2, where $x = 2$).

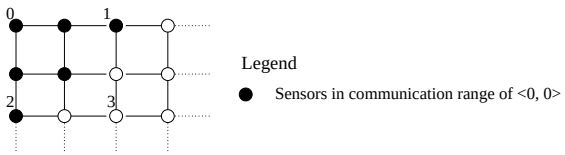

Fig. 2. Sample diffusion in networks where a sensor communicates with its distance 2 neighbors. The number associated with a sensor shows the slot in which it should transmit.

From Figure 2, we observe that sensors $\langle 2,0\rangle$ and $\langle 0,2\rangle$ should not transmit at the same time as these messages will collide at $\langle 2,2\rangle$. Since the sensor $\langle 0,0\rangle$ can communicate at a larger distance and the sensors $\langle 0,2\rangle$, $\langle 2,0\rangle$ propagate the diffusion, sensors $\langle 0,1\rangle$, $\langle 1,0\rangle$ and $\langle 1,1\rangle$ need not transmit. The following algorithm provides a collision-free diffusion in networks where sensors can communicate with their distance $x, x \geq 1$, neighbors.

```
when sensor j receives a diffusion message from sensor k
      if (k is west neighbor at distance x)
        transmit after 1 clock tick.
      else if (k is north neighbor at distance x)
        transmit after 2 clock ticks.
      else //duplicate message from east/south neighbor or too close to source
        ignore
```

Theorem 3.2 The above algorithm satisfies the problem specification of collision-free diffusion. □

3.3 Version 3: Communicate 1, Interfere y

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 3, where $y = 2$).

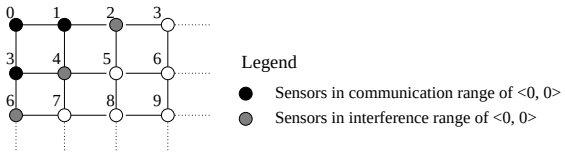

Fig. 3. Sample diffusion in networks where a sensor communicates with its distance 1 neighbors and interferes with its distance 2 neighbors. The number associated with a sensor shows the slot in which it should transmit.

Once again, we observe that sensors $\langle 1, 0\rangle$ and $\langle 0, 1\rangle$ should not transmit at the same time. Also, sensors $\langle 2, 0\rangle$ and $\langle 0, 1\rangle$ should not transmit at the same time as their messages will collide at $\langle 1, 1\rangle$ and $\langle 2, 1\rangle$. The third version of the algorithm is as follows:

```
when sensor j receives a diffusion message from sensor k
      if (k is west neighbor at distance 1)
        transmit after 1 clock tick.
      else if (k is north neighbor at distance 1)
        transmit after y + 1 clock ticks.
      else // duplicate message received from east/south neighbor
        ignore
```

Theorem 3.3 The above algorithm satisfies the problem specification of collision-free diffusion.

Proof. Let us assume that the source sensor $\langle 0, 0\rangle$ starts transmitting at time $t = 0$. By induction, we observe that sensor $\langle i, j\rangle$ will transmit at time $t = i + (y + 1)j$. 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 $t_1 = i_1 + (y+1)j_1$ and $\langle i_2, j_2\rangle$ will transmit at time $t_2 = i_2 + (y+1)j_2$. Collision is possible only if the following conditions hold:

- $t_1 = t_2$, i.e., $(i_1 - i_2) + (y + 1)(j_1 - j_2) = 0$.
- $|i_1 - i_2| + |j_1 - j_2| \leq y + 1$.
- $|i_1 - i_2| + |j_1 - j_2| \geq 1$.

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. □

3.4 Version 4: Communicate x, Interfere y

Consider a grid network where a sensor can communicate with sensors that are distance $x, x \geq 1$ away and interfere with sensors that are distance $y, y \geq x$ away. This network can be viewed as a modified network where the intermediate sensors are removed. Hence, in this modified network, a sensor can communicate with its distance 1 neighbors and interfere with its distance $\lceil \frac{y}{x} \rceil$ neighbors. Now, we apply the version 3 of our algorithm with parameters *communicate* 1, *interfere* $\lceil \frac{y}{x} \rceil$.

3.5 Diffusion by an Arbitrary Sensor

If sensor k (other than, $\langle 0, 0 \rangle$) initiates the diffusion, we split the network into four quadrants with sensor k at the intersection of x and y axes. For each quadrant, we can use the algorithm similar to that in Sections 3.1-3.4; We simply need to ensure that messages in different quadrants do not collide (on x and y axes). For the case where communication range = interference range = 1, we can achieve this as follows: (Extensions for other values of communication and interference range are also similar.)

Sensors in south-east quadrant transmit the diffusion message as before (i.e., a sensor $\langle i, j \rangle$ will transmit the diffusion at $|i| + 2|j|$). Sensors in the north-east and south-west quadrants (including the $-$ve x-axis and $+$ve y-axis but excluding the $+$ve x-axis and $-$ve y-axis) transmit the diffusion similar to the south-east quadrant, but with 2 clock ticks delay. This is to ensure the diffusion messages do not collide at the x and y axes. Specifically, a sensor $\langle i, j \rangle$ in the north-east quadrant or in the south-west quadrant transmits the diffusion at $|i| + 2|j| + 2$. Sensors in the north-west quadrant (excluding the axes) transmits the diffusion similar to the other quadrants except that the delay here is 4 clock ticks. In other words, a sensor $\langle i, j \rangle$ in the north-west quadrant transmits the diffusion at $|i| + 2|j| + 4$. We leave it to the reader to verify that with this modification, diffusion initiated by an arbitrary sensor is collision-free.

3.6 Observations about Our Algorithm

We make the following observations about our algorithm:

1. In Sections 3.1-3.5, we considered the manhattan distance between sensors, i.e., if the interference range is y then we said that sensors $\langle i_1, j_1 \rangle$ and $\langle i_2, j_2 \rangle$ interfered with each other only if $|i_1 - i_2| + |j_1 - j_2| \leq y$. We note that our

algorithm works correctly even if we say that sensors $\langle i_1, j_1 \rangle$ and $\langle i_2, j_2 \rangle$ interfere only if $|i_1 - i_2| \leq y$ and $|j_1 - j_2| \leq y$. It follows that our algorithm works correctly even if we consider the geographic distance between sensors and say that two sensors $\langle i_1, j_1 \rangle$ and $\langle i_2, j_2 \rangle$ interfere with each other if the geographic distance between them, $\sqrt{|i_1 - i_2|^2 + |j_1 - j_2|^2}$, is less than or equal to y.

2. Even if the interference range is overestimated, our algorithm works correctly.
3. Even if the communication range is underestimated, as long as it is at least 1, our algorithm works correctly.
4. We can apply our algorithm even if the communication and interference ranges of different sensors vary. We can use the minimum of the communication range of each sensor (underestimate) and the maximum of the interference range of each sensor (overestimate).

4 Application to Time-division Multiplexing (TDM)

In this section, we present an algorithm for time-division multiplexing in sensor networks using the collision-free diffusion algorithm discussed earlier. 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 multiplexing is defined as follows:

Problem Statement: Time-division Multiplexing
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).

Now, we present the algorithm for allotting time slots to the sensors. In Section 4.1, we present the algorithm for TDM in perfect grids. In Section 4.2, we discuss how stabilization is achieved starting from an improperly initialized state.

4.1 Simple TDM Algorithm

In this section, we present our simple TDM algorithm that uses the third version of the diffusion algorithm (cf. Section 3.3) where communication range is 1 and interference range is y. 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 slots in which j (respectively, k) transmits its diffusion message. 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 correctness of the diffusion computation, 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 $c_1, c_2 \geq 0$. 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 $k) + 1$.

In the third version of our algorithm, 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 algorithm for time division multiplexing is as follows:

If sensor j transmits a diffusion message at time slot t_j,
j can transmit at time slots, $\forall c, c \geq 0, t_j + c * ((y+1)^2 + 1)$.

The algorithm assigns time slots for each sensor based on the time at which it transmits the diffusion. 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 4 shows a sample allocation of slots to the sensors.

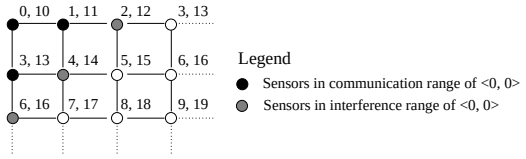

Fig. 4. Sample TDM in networks where a sensor communicates with its distance 1 neighbors and interferes with its distance 2 neighbors. The numbers associated with a sensor shows the slots in which it could transmit.

Theorem 4.1 The above algorithm satisfies the problem specification of TDM. □

4.2 Stabilization of TDM and Diffusion

We now add stabilization to the TDM algorithm discussed in Section 4.1, 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 simple TDM algorithm relies on the collision-free diffusion algorithm discussed earlier (cf. Section 3.3). 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. 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 simple TDM algorithm to transmit messages across different sensors. Moreover, if there are no faults in the network and the links are reliable then no sensor will ever shut down.

Dealing with unreliable links. Now, we show that in the absence of faults, a sensor rarely shuts down even if the link between the neighboring sensors are unreliable. 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 the sensor j will incorrectly shut down is 0.0137.

Observations about our stabilizing fault-tolerant algorithms. We make the following observations about our stabilizing fault-tolerant algorithms:

1. If there are no failures in the network and the links are reliable then no sensor will ever shut down.
2. If there are no failures in the network and the links are unreliable then sensors may shut down rarely. However, the probability that a sensor shuts down incorrectly due to unreliable links can be made as small as possible.

5 Extensions: Dealing with Failed/Sleeping Sensors and Arbitrary Topology

In this section, we discuss extensions that remove some of the assumptions made in Section 2. In Section 5.1, we extend the algorithm to deal with the case where sensors are subject to fail-stop faults. In Section 5.2, we provide an algorithm for collision-free diffusion in the case where the underlying graph is not a two-dimensional grid.

5.1 Diffusion in Imperfect Grids or Grids with Failed Sensors/Links

In this section, we consider the case where sensors can fail, links between sensors can fail, or the grid can be improperly configured (with some sensors missing).

Based on the extension in Section 3.5, without loss of generality, assume that the left-top sensor initiates the diffusion. In the absence of failure of sensors or links between them, the sensors receive the diffusion messages from their north or west neighbors before receiving duplicate messages from their east or south neighbors. Hence, if a sensor receives the diffusion message for the first time from its east or south neighbor, it can conclude that some of the sensors in the network are missing or failed. When a sensor receives such a message from the

south/east neighbor, it updates its clock based on the time information in the diffusion message. Based on its geographic location, it then determines the slot in which it would have transmitted the diffusion if no sensor had failed. Finally, it uses the time-division multiplexing algorithm (cf. Section 4) to find the next slot when it can transmit a message; this slot would be used for retransmitting the diffusion message.

5.2 Diffusion to Other Graphs

Collision-free diffusion in other graphs can be achieved by embedding a (partial) grid in that graph. (Note that the goal of this solution is to show the feasibility of such extension. If additional information about the topology is available then it is possible to improve the efficiency of the diffusion on the transformed graph [2].) To show one approach for embedding such a partial grid, we begin with the observation that, an arbitrary tree can be mapped into a (complete) binary tree. Also, a complete binary tree can be mapped on a 2-dimensional grid with *dilation* $\lceil (k-1)/2 \rceil$ where k is the depth of the tree [3]. If the degree of a node is more than 3, we split that node to construct a binary tree (cf. Figure 5).

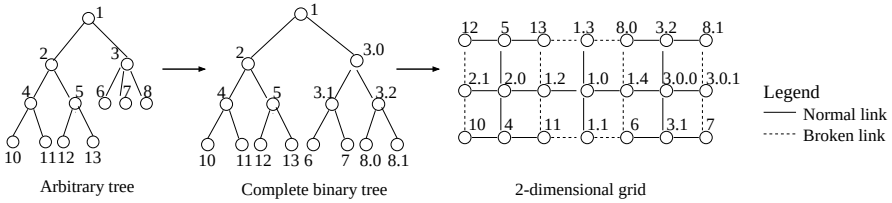

Fig. 5. Mapping an arbitrary tree into a 2-dimensional grid

We can observe from Figure 5 that node 1 is split into 5 nodes, 1.0 . . . 1.4. Hence, node 1 will get 5 different time slots for communication. Also, nodes in the 2-dimensional grid can communicate with the nodes that are at distance 1 (except in the case where the link drawn is a broken link). Some of the nodes in the 2-dimensional grid can communicate with nodes that are at a larger distance. For the purpose of collision-free diffusion, we can treat this communication as interference (e.g., in Figure 5, the communication between 1.1 and 3.1 can be treated as interference). It follows that given an arbitrary tree, we can embed a partial grid in it; in this partial grid, the communication range is 1. And, the interference range is determined based on the way in which nodes of degree more than 3 are split.

Finally, for an arbitrary graph, we can use its spanning tree, and embed a partial grid in it. Then, we can add the remaining edges to this partial grid and treat them as interference-only. With such approach, it is possible to apply the collision-free diffusion algorithm to arbitrary graphs.

6 Related Work

Related work that deals with communication issues in radio/wireless/sensor networks includes [4–6]. In [4], the authors provide a fault-tolerant broadcasting algorithm in radio networks. The model proposed in this paper assumes that the upper bound on the number of faulty nodes is known at start. They also assume that the faults are permanent. The authors do not consider the notion of interference range for nodes. Our paper differs considerably from that in [4]. The assumption about knowledge of the number of faults is not made in our algorithms. Also, unlike [4], we allow sensors (other than the initiator) to fail/recover during computation.

In [5, 6], new time synchronization services are proposed. In [5], the authors propose a time synchronization service for tiny sensor devices like *motes* [7]. This service maintains a tree structure of motes where the root sends a periodic beacon message about its time. Each non-root node gets the best-approximation of the root's time from the neighbor which is closest to the root. In [6], the authors propose a time synchronization service which rely on a third-party node. The nodes normalize their local-time based on the synchronization pulse sent by the third-party node. Based on our observations with motes, collision-freedom is important in these system-wide computations. For this reason, in this paper, we developed a collision-free communication algorithm. Our algorithm can be used for collision-free transmission of the time synchronization messages, thereby enhancing the proposed time synchronization services.

In [8], the authors provide algorithms for completely connected graphs where they consider the difference between a globally synchronous (global clock) and a locally synchronous (local clock with same rate of increase) model with known or unknown network size. In [9], the authors provide algorithms for mobile/ad hoc networks. They provide broadcasting algorithms for a model without collision detection and a model with collision detection. Unlike our algorithms, in [8, 9], the authors assume that the network is fully connected. Also, the algorithms in [8, 9] are not stabilizing fault-tolerant.

Our algorithms differ from Code-division Multiple Access (CDMA) [10]. CDMA requires that the codes used in the system should be orthogonal to ensure minimal interference. Also, it requires expensive operations to encode/decode a message. Our algorithms do not need any specialized codes. Further, to ensure collision-freedom, our algorithm requires only very limited resources, e.g., an addition and a comparison unit.

In [11], the authors have proposed a randomized startup algorithm for TDM. Whenever a collision occurs during startup, exponential backoff is used for determining the time to transmit next. In our approach, we use a deterministic startup algorithm which guarantees collision-freedom and stabilization in case of fail-stop failures. Further, the complexity of the algorithm proposed in [11] is $O(N)$ where N is the number of system nodes, whereas the complexity of our diffusion algorithm is $O(D)$ where D is the diameter of the network. Moreover, the algorithm in [11] optimizes time and communication overhead with increased computation overhead, while our diffusion algorithm optimizes all the

three overheads. The disadvantage of our algorithm is that it has a single point of failure (i.e., initiator of diffusion). In situations where the initiator fails, we can use the startup algorithm from [11] to assign TDM slots.

7 Conclusion and Future Work

In this paper, we presented a stabilizing algorithm for collision-free diffusion in sensor networks and showed how it can be used to provide time-division multiplexing. We presented four versions of our collision-free diffusion algorithm based on the ability of sensors to communicate with each other and their ability to interfere with each other. While the solutions were designed for a grid network, we showed how they could be modified to deal with failed sensors as well as with arbitrary topologies. With these modifications, our solutions deal with commonly occurring difficulties, e.g., failed sensors, sleeping sensors, unidirectional links, and unreliable links, in sensor networks.

Our algorithms permit sensors to save 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. Moreover, our algorithm is stabilizing fault-tolerant [1]. Thus, even if all sensors are deactivated for a long time causing arbitrary clock drift, our algorithm ensures that starting from such an arbitrary state, eventually the diffusion will complete successfully and the time-division multiplexing would be restored.

One of the important issues in our algorithms is to determine the communication and interference range of a sensor. Initially, we can start with the manufacture specification about the ability of sensors to communicate with each other. Then, we can use the biconnectivity experiments by Choi et al [12] to determine the appropriate communication and interference range. One such approach is discussed in [2].

In our solution, it is possible for the initiator of a diffusion to handoff this responsibility to other sensors as the diffusion can be initiated by any sensor as long as only one sensor initiates it. Thus, the current initiator can designate another sensor as subsequent initiator if the current initiator has low battery or if the initiating responsibility is to be shared by multiple sensors.

There are several questions raised by this work: For one, 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 to be the initiators. Alternatively, during deployment of sensors (e.g., by dropping them from a plane), we can keep several potential initiators that communicate with each other directly and use the approach in [8, 9] so that one of them is chosen to be the initiator. Another important concern is how to deal with errors in the location of the sensors. Specifically, we need to analyze the effects of these errors on the collision-free property of our algorithms.

References

1. E. W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Communications of the ACM*, 17(11), 1974.
2. Sandeep S. Kulkarni and Umamaheswaran Arumugam. Collision-free communication in sensor networks. Technical Report MSU-CSE-03-3, Department of Computer Science, Michigan State University, February 2003.
3. J. D. Ullman. *Computational Aspects of VLSI*. Computer Science Press, Rockville, MD, 1984.
4. Evangelos Kranakis, Danny Krizanc, and Andrzej Pelc. Fault-tolerant broadcasting in radio networks. *Journal of Algorithms*, 39(1):47–67, April 2001.
5. Ted Herman. NestArch: Prototype time synchronization service. NEST Challenge Architecture. Available at: `http://www.ai.mit.edu/people/sombrero/nestwiki/index/ComponentTimeSync`, January 2003.
6. Jeremy Elson and Deborah Estrin. Time synchronization for wireless sensor networks. *In Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS), Workshop on Parallel and Distributed Computing Issues in Wireless and Mobile Computing*, April 2001.
7. David E. Culler, Jason Hill, Philip Buonadonna, Robert Szewczyk, and Alec 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.
8. Leszek Gasieniec, Andrzej Pelc, and David Peleg. The wakeup problem in synchronous broadcast systems. *SIAM Journal of Discrete Mathematics*, 14(2):207–222, 2001.
9. Bogdan S. Chlebus, Leszek Gasieniec, Alan Gibbons, Andrzej Pelc, and Wojciech Rytter. Deterministic broadcasting in ad hoc radio networks. *Distributed Computing*, 15(1):27–38, 2002.
10. Andrew J. Viterbi. *CDMA: Principles of Spread Spectrum Communication*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, 1995.
11. Vilgot Claesson, Henrik Lonn, and Neeraj 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.
12. Y. Choi, M. Gouda, M. C. Kim, and A. Arora. The mote connectivity protocol. Technical Report TR03-08, Department of Computer Sciences, The University of Texas at Austin, 2003.