Sprinkler: A Reliable and Energy Efficient Data Dissemination Service for Extreme Scale Wireless Networks of Embedded Devices

Vinayak Naik, *Member*, *IEEE*, Anish Arora, *Member*, *IEEE*,
Prasun Sinha, *Member*, *IEEE*, and Hongwei Zhang, *Member*, *IEEE*

Abstract—We present *Sprinkler*, a reliable data dissemination service for wireless embedded devices which are constrained in energy, processing speed, and memory. Sprinkler embeds a virtual grid over the network whereby it can locally compute a connected dominating set of the devices to avoid redundant transmissions and a transmission schedule to avoid collisions. Sprinkler transmits $O(1)$ times the optimum number of packets in $O(1)$ of the optimum latency; its time complexity is $O(1)$. Sprinkler is tolerant to fail-stop and state corruption faults. Thus, Sprinkler is suitable for resource-constrained wireless embedded devices. We evaluate the performance of Sprinkler in terms of the number of packet transmissions and the latency, both in an outdoor and indoor environment. The outdoor evaluation is based on data from project ExScal, which deployed 203 extreme scale stargates (XSS). Our indoor evaluation is based on an implementation in the Kansei testbed, which houses 210 XSSs whose transmission power is controllable to even low ranges. We compare Sprinkler with the existing reliable data dissemination services, analytically or using simulations also. Our evaluations show that Sprinkler is not only energy efficient as compared to existing schemes, but also has less latency. Further, the energy consumption of nodes and the latency grows linearly as a function of newly added nodes as the network grows larger.

Index Terms—Network protocols, real-time systems and embedded systems, wireless, wireless sensor networks.

1 Introduction

Reprogramming in the field has emerged as a necessary primitive for wireless devices. There are many reasons for this, for instance—resulting from an incomplete knowledge of the deployment environment, planned phases of operation that are instrumented only at runtime or evolution of the operational requirements. Reprogramming necessitates a data dissemination service that is fully reliable since a program must be delivered in entirety. Further, reprogramming must utilize minimum energy so that the lifetime of the network is maximized. In the context of reprogramming, message transmission is an energy expensive operation, as shown in Table 1. Another factor is the number of operations for the microprocessor. Hence, reducing the number of transmissions and the amount of computation are both important goals in addition to reliability.

The problem of reliable data dissemination is widely studied, even in the context of wireless embedded devices. The optimization criteria for one class of well-known existing schemes—viz. Deluge [1], Infuse [2], MNP [3], and PSFQ [4]—are reliability and latency, in that order. Even though some of these schemes do instrument sender suppression techniques, these techniques do not effectively minimize the number of senders and, hence, the number of packet transmissions. The criteria for Sprinkler are reliability, energy, and latency—in that order. To reduce energy consumption, Sprinkler computes a subset of nodes as senders. The subset is connected and every node in the network has a neighbor in the subset. The problem of selecting the minimum number of senders is computing a minimum connected dominating set (MCDS) of the graph induced by the wireless network, which is known to be NP-hard even for a unit disk graph [5]. As a part of Sprinkler, we provide a low complexity CDS algorithm, which is suitable for embedded devices.

The CDS nodes broadcast messages instead of unicasting to reduce the number of transmissions. But broadcast messages can collide due to the hidden terminal effect. Sprinkler avoids hidden terminals by using Time Division Multiple Access (TDMA). The number of TDMA slots determines the latency of a reprogramming operation. The problem of computing a TDMA schedule that avoids hidden terminals with a minimum number of slots in a unit disk graph is equivalent to computing a D-2 vertex coloring [6]. Intuitively, the reason behind distance two is that two nonneighboring nodes u and v interfere with each other if there exists a node w such that there are edges (u, w) and (w, v) in the unit disk graph. We provide a D-2 vertex coloring algorithm of low complexity, which is suitable for embedded devices (henceforth, we use the terms TDMA and D-2 vertex coloring interchangeably). In practice,

- *V. Naik is with the Center for Embedded Networked Sensing, University of California, Los Angeles, 3551 Boelter Hall, 420 Westwood Plaza, Los Angeles, CA 90095-1596. E-mail: naik@cens.ucla.edu.*
- *A. Arora and P. Sinha are with the Department of Computer Science and Engineering, Ohio State University, Room 395, Dreese Labs, 2015 Neil Avenue, Columbus, OH 43210. E-mail: {anish, prasun}@cse.ohio-state.edu.*
- *H. Zhang is with the Department of Computer Science, Wayne State University, 431 State Hall, 5143 Cass Avenue, Detroit, MI 48202. E-mail: hzhang@cs.wayne.edu.*

Manuscript received 11 Mar. 2006; revised 9 Aug. 2006; accepted 17 Oct. 2006; published online 7 Feb. 2007.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-0072-0306.
Digital Object Identifier no. 10.1109/TMC.2007.1013.

TABLE 1
Energy Required by Common Operations

Operation	Current Draw	
	Mote	Stargate
Microprocessor and Idle radio	8mA	330 mA
Packet Reception	16mA	610 mA
Packet Transmission	**24mA**	**980 mA**

node w may or may not interfere with node u depending upon the distance between v and w. Therefore, a D-k vertex coloring, where k is a real number greater than 1 and less than 2, would suffice.

The time complexities of the state-of-the-art CDS construction and D-2 vertex coloring algorithms and the number of message transmissions are functions of the number of nodes [7]. Hence, the existing algorithms are not scalable for extremely large networks, such as those in project ExScal [8]. ExScal consisted of 1,000 extreme scaling motes (XSMs) [9] and 203 XSSs. Such extreme scale networks demand local algorithms of constant time complexities. The networks in ExScal were used for intruder detection, classification, and tracking. One of the services required for this class of applications is a localization service which informs location to each node. Further, to guarantee sensor coverage, the underlying networks are dense. We use these two characteristics, viz. availability of location service and density of networks, to derive local CDS and D-2 coloring algorithms of low time complexities.

1.1 Our Contributions

- Sprinkler effectively reduces the number of senders to a constant factor of the minimum using a local algorithm of time complexity $O(1)$ for constructing a CDS. A performance ratio of a CDS construction algorithm is the maximum ratio of the cardinality of computed CDS for a graph G over that MCDS of G. The performance ratio of our algorithm is $18\frac{2}{3}$. The state-of-the-art distributed CDS algorithm, with constant performance ratio, has $O(\Delta \log^2 n)$ time complexity, where Δ is the maximum degree in the network and n is the number of nodes in the network [7].
- Sprinkler effectively manages the latency by computing a near optimal schedule using a local D-2 coloring algorithm of time complexity $O(1)$ for the above calculated CDS. A performance ratio of a D-2 vertex coloring algorithm is the maximum ratio of the number of computed colors for a graph G over the minimum number of D-2 colors for G. The performance ratio of our algorithm is $1\frac{8}{9}$. The state-of-the-art distributed D-2 coloring algorithm, with constant performance ratio, has time complexity $O(\Delta \log^2 n)$ [7]. Further, both the existing algorithms are randomized, whereas ours are deterministic. The locality and constant time complexity of our algorithms make Sprinkler scalable. A TDMA schedule with few slots results in low latency.
- The number of packet transmissions is uniformly distributed among all the transmitters. This helps in load-balancing across the network. Further, the latency is also uniformly distributed among all the receivers, which means that all the nodes receive a new program at uniformly distributed intervals.
- Sprinkler is tolerant to node addition, fail-stop, and state corruption faults.
- The number of packet transmissions and the latency scale as a linear function of newly added nodes. Further, for a given network, if we increase the density of nodes without increasing the hop counts, then the number of packet transmissions and latency remain approximately the same.

In a nutshell, we divide the problem of reliable data dissemination into three subproblems—CDS construction, D-2 vertex coloring, and a protocol to reliably disseminate data using a CDS and a TDMA schedule. Each of these subproblems is solved separately to yield our reliable data dissemination service.

1.2 Organization of the Paper

We present the system model and the fault model in Section 2. Then, in Section 3, we recall and extend some concepts in graph theory. Following that, we describe our algorithms for CDS construction and D-2 vertex coloring in Section 4. In Section 5, we present our data dissemination protocol Sprinkler, and we prove its fault-tolerance properties in Section 5.1. We adapt Sprinkler to a high-fidelity radio model and evaluate the scalability of Sprinkler in Section 6. Then, we compare the performance of Sprinkler with existing schemes. Finally, we discuss related work in Section 8, summarize our findings in Section 10, and mention future work in Section 11.

2 System Model

2.1 Model

We consider wireless devices which are constrained in energy and processing. The examples include mote and stargate. The devices are embedded in a plane. Let n be the number of devices. In steady state, the devices are static. We allow limited mobility as explained in Section 2.2. Each device knows its location. The location information can be provided using a localization service.

R is the reliable communication radius of the device. Let $p(x)$ be the packet delivery ratio at distance x from a sender and P be the maximum value of p for any x. Let A be the set of distances at which packet delivery ratio is equal to P. Then, R is the maximum distance in A. Intuitively, R is the maximum distance over the set of distances at which the packet delivery ratio is maximum. We use the unit disk radio model for the analytical evaluation of Sprinkler and the probabilistic radio model for experiments and simulations.

The density of the nodes is such that we divide the plane into a virtual grid of equal sized squares where 1) each square has at least one node and 2) a node is able to communicate to all the nodes in each of the four adjoining squares. Formally, our density assumption is:

> If R is the reliable communication radius of the device, then every square of length $R/\sqrt{5}$ contains at least one device.

The density assumption must always be satisfied for use of Sprinkler.

For simplicity, we assume that the network traffic is quiescent at the time of data dissemination. Quiescing data traffic is easy. If a node hears a data dissemination packet, it stops other traffic for the duration of dissemination. This assumption can be made safely in case of dissemination due to reprogramming. Since reprogramming would most probably result in new data traffic, old traffic can even be discarded.

2.2 Fault Model

New nodes can be added, nodes can detectably fail-stop, or the state of nodes can get corrupted to an arbitrary value. A device can be relocated, which is equivalent to a combination of fail-stop and an addition of a new node. In other words, mobile nodes are not assumed to carry their state while moving. In all the above mentioned faults, we assume that the minimum density is preserved.

3 Preliminaries

Consider a set of n equal-sized circles in a plane. The intersection graph of these circles is an n-vertex graph; each vertex corresponds to a circle, and an edge appears between two vertices when the corresponding circles intersect. Such intersection graphs are called *unit disk graphs* [5]. A *dominating set* (DS) of a graph $G = (V, E)$ is a subset V' of V such that every vertex $v \in V$ is either in V' or adjacent to some member of V'. A dominating set is *connected* (CDS) if the subgraph induced by it is connected. A *minimum connected dominating set* (MCDS) is a connected dominating set of minimum cardinality [5]. Let $G(V, E)$ be an undirected graph. A *distance-2 vertex* (*D-2*) *coloring* of a graph is a mapping $f : V \rightarrow \{1, \ldots, k\}$ such that $f(u) \neq f(v)$ whenever there is a path consisting of at most two edges between u and v in G [10]. The D-2 coloring problem is equivalent to the standard minimum vertex coloring on G^2, where G^2 has the same vertex set as G and there is an edge between two vertices of G^2 if and only if there is a path of length at most 2 between the vertices in G [6].

A *bidimensional grid* $B(1)$ of size $r \times c$ has r rows and c columns, indexed, respectively, from 0 to $r - 1$ (from top to bottom) and from 0 to $c - 1$ (from left to right), with $r \geq 1$ and $c \geq 1$. A generic vertex u of B will be denoted by $u = (i, j)$, where i is its row index and j is its column index. Each vertex has degree equal to 4, except for the vertices on the borders. In particular, each vertex (i, j), which does not belong to the grid borders, is adjacent to the vertices $(i - 1, j)$, $(i, j + 1)$, $(i + 1, j)$, and $(i, j - 1)$, as shown in Fig. 1. $B(k)$ is a bidimensional grid such that each vertex (i, j) is adjacent to all the vertices (i', j'), where the euclidean distance between (i, j) and (i', j') is not more than k.

4 Algorithms to Compute CDS and D-2 Vertex Coloring

4.1 CDS Computation

Given a wireless network, let $G = (V, E)$ be its corresponding unit disk graph. We assume that G is enclosed in the smallest rectangular area of length $r(R'/\sqrt{5})$ and breadth $c(R'/\sqrt{5})$, where r, c are positive integers, $3 \leq r \leq c$, and $R' = R$. If $r > c$, then they can be exchanged. Further, the rectangle is divided into square-shaped clusters of length $R'/\sqrt{5}$ and one node is selected from each cluster as a clusterhead. Since $R' = R$, the length of the square is also equal to the reliable communication radius. In Section 6, we will deal with the case where $R' > R$. Only the clusterhead nodes are used to construct a CDS M. At the end of this section, we discuss the case where such clustering is not available.

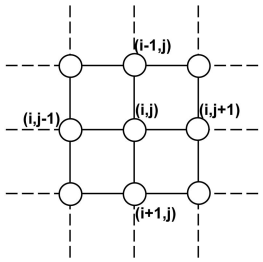


Fig. 1. A bidimensional grid $B(1)$.

A node $u(i, j) \in M$, where $0 \leq i \leq r - 1$ and $0 \leq j \leq c - 1$, if

- $r \bmod 3 \equiv 0$: $[i \bmod 3 \equiv 1] \vee [(i \bmod 3 \not\equiv 1) \wedge (0 < i < r - 1) \wedge (j = 0)]$.
- $r \bmod 3 \equiv 1$: $[i \bmod 3 \equiv 0] \vee [(i \bmod 3 \not\equiv 0) \wedge (j = 0)]$.
- $r \bmod 3 \equiv 2$: $[i \bmod 3 \equiv 1] \vee [(i \bmod 3 \not\equiv 1) \wedge (i \not\equiv 0) \wedge (j = 0)]$.

Fig. 2a illustrates the application the above mentioned algorithm for a network of randomly distributed nodes. The circles represent nodes, gray circles represent selected nodes, and black connected circles represent M. Note that both the rectangle and the grid of squares are virtual. Fig. 2b illustrates M for various cases of r.

Theorem 1. *M is a CDS of G.*

Proof. By construction, M is connected. For a proof of domination, we will first consider the case where $r \bmod 3 \equiv 0$. Consider any node u in cluster (i, j). Let v be a node in cluster

- $(i + 1, j)$ if $i \bmod 3 \equiv 0$,
- (i, j) if $i \bmod 3 \equiv 1$, and
- $(i - 1, j)$ if $i \bmod 3 \equiv 2$.

According to the CDS computation algorithm, v is in M. Since the maximum distance between any two nodes in clusters (i, j) and $(i + 1, j)$ is $R' = R$, node v dominates u. Similarly for the case where v is in cluster $(i - 1, j)$. We can prove the same result for the remaining two cases, viz. $(r \bmod 3 \equiv 1)$ and $(r \bmod 3 \equiv 2)$. □

Lemma 1. $|M| < 2rc/3$.

Proof. We compute the number of nodes in CDS M, returned by our algorithm. We consider four cases, depending upon the number of rows:

- $r \bmod 3 \equiv 0$:
  - $r = 3 : |M| = c < 2rc/3$.
  - $r \neq 3 \quad : \quad |M| = 2(c + 1) + (r/3 - 2)c < 2rc/3$ [since $c \geq r \geq 3$].
- $r \bmod 3 \equiv 1 \quad : \quad |M| = (c + 1) + (r - 4)c/3 + (c + 1) < 2rc/3$ [since $c \geq r \geq 3$].

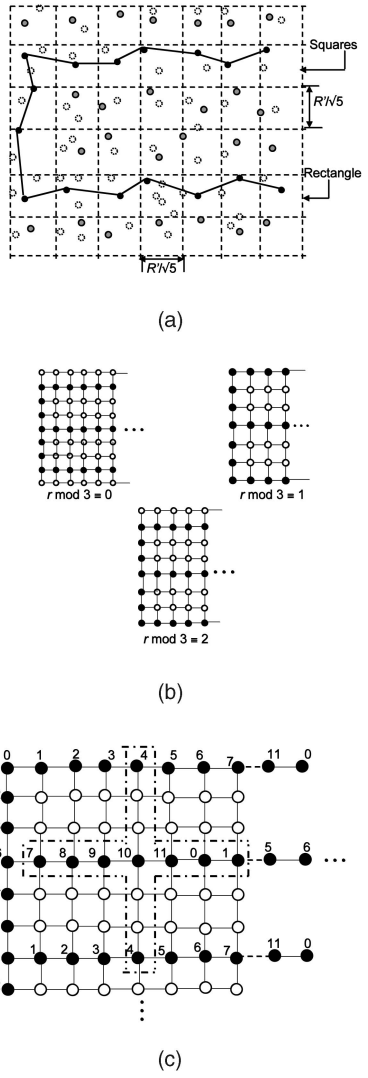


Fig. 2. CDS computation and D-2 vertex coloring. (a) CDS computation for random deployment. (b) CDS of bidimensional grid $B(1)$. (c) A D-2 coloring for M.

- $r \bmod 3 \equiv 2 \ : \ |M| = c + (r-1)(c+2)/3 < 2rc/3$ [since $c \geq r \geq 3$]. □

Theorem 2. *The size of the CDS computed by the algorithm is at most* $18\frac{2}{3}$ *times that of an MCDS.*

Proof. In graph G, the maximum number of neighboring squares within R distance of a dominating node is equal to the maximum number of squares of length $R'/\sqrt{5}$ that can be packed in a circle of radius $R + R/\sqrt{10}$, which is $\pi(R + R/\sqrt{10})^2/(R'/\sqrt{5})^2 < 28$. Hence, a node in $MCDS$ can dominate at the most 28 clusterheads, which means that the $|MCDS| \geq rc/28$. According to Lemma 1, the maximum size of M for a bidimensional grid of size rc is $2rc/3$. Hence, the size of the CDS computed by the algorithm is at most $18\frac{2}{3}$ times that of an MCDS. □

4.2 D-2 Coloring

We only need to compute D-2 coloring for nodes in M since only those will transmit packets. Since $5(R'/\sqrt{5}) > 2R$, no two nodes in M within $5(R/\sqrt{5})$ distance of each other can have the same color, i.e., there exists at least five squares between two CDS nodes. Let $C(i,j)$ denote the D-2 color of a node $u(i,j) \in M$, where $0 \leq i \leq r-1$ and $0 \leq j \leq c-1$. $C(i,j)$ is computed using the following formula:

- $(i \bmod 3 \equiv 0) \wedge (i \bmod 6 \equiv 0)$: $j \bmod 11$.
- $(i \bmod 3 \equiv 0) \wedge (i \bmod 6 \not\equiv 0)$: $(j+6) \bmod 11$.
- $(i \bmod 3 \equiv 1) \wedge (i \bmod 6 \equiv 1) \wedge (j = 0)$: 12.
- $(i \bmod 3 \equiv 1) \wedge (i \bmod 6 \not\equiv 1) \wedge (j = 0)$: 14.
- $(i \bmod 3 \equiv 2) \wedge (j = 0)$: $C(i-1, 0) + 1$.

The number 16 insures that no two nodes within $2R'$ distance of each other have the same color. Fig. 2c illustrates the application of the algorithm. Again, the dark circles represent the nodes in M. The total number of colors is equal to 16. The time complexity of the D-2 coloring algorithm is $O(1)$ [since $c \geq r$].

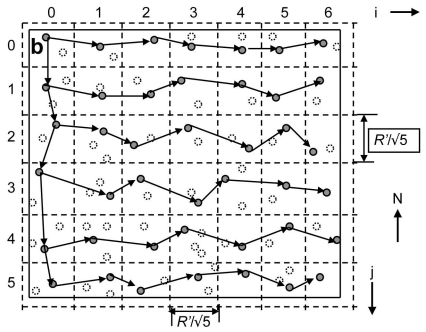


Fig. 3. Distributed cluster formation.

Theorem 3. *The number of colors computed by the algorithm is at most* $1\frac{8}{9}$ *times that for an optimal D-2 coloring.*

Proof. There exists a clique of size nine in M^2, as shown by the dotted region in Fig. 2c. In other words, the maximum distance between any two nodes with the dotted region is less than $2R'$. Therefore, a D-2 coloring of M requires at least nine colors. Sprinkler uses 16 colors. Hence, the number of colors computed by the algorithm is at most $1\frac{8}{9}$ times an optimal D-2 coloring. □

4.3 Cluster Formation

The problem of partitioning the network into nonoverlapping equal-sized squares is equivalent to clustering. Formally, we want to divide the network into square-shaped disjoint clusters of length $R'/\sqrt{5}$. We provide a simple distributed clustering algorithm to form such clusters.

Let b be the base station node that originates broadcast data. In addition to our assumption of minimum density and location information, we assume that b knows the locations of the four corners of the smallest rectangle of length $r(R'/\sqrt{5})$ and breadth $c(R'/\sqrt{5})$ enclosing G, where r and c are positive integers and $3 \leq r \leq c$ as mentioned in Section 4.1. If $r > c$, then they can be exchanged. Any node, after receiving the locations of the four corners of the rectangle, can locally partition the rectangular area into squares of length $R'/\sqrt{5}$ and compute the coordinates of its square. Further, we assume that each node knows the nodes in its one-hop neighborhood. In particular, it knows the ID and the location of its one-hop neighbors.

For simplicity of presentation, we assume that b is located at the northwest corner of the rectangle as shown in

Fig. 3. Note that the complexity of the algorithm does not change if we assume b is at any other location. Following is the algorithm:

if $(ID = b \vee rcv\ \langle locations\ of\ four\ corners \rangle)$
$\wedge \neg sent \rightarrow$ ***then***
  if $j = 0$ ***then***
    select a node u from square $(i+1, 0)$;
      – (A)
    send $\langle locations\ of\ four\ corners \rangle$ to u;
  end if
  select a node v from square $(i, j+1)$;
    – (B)
  send $\langle locations\ of\ four\ corners \rangle$ to v;
  sent := TRUE;
end if
The initial value of the variable sent is FALSE.

Since a node knows the IDs and locations of its one-hop neighbors, it can locally select a node as mentioned in actions (A) and (B). Fig. 3 illustrates the application of the above algorithm for a case where nodes are randomly distributed. Circles represent nodes and gray circles represent selected nodes. The gray node in square $(0, 0)$ is b. An arrow between two nodes represents a transmission of a message.

Theorem 4. *The time complexity of the distributed cluster formation is* $O(1)$ *and the number of messages is* $O(n)$*, where each message is of size* $O(1)$.

Proof. Each node in M sends at most two messages. Hence, the time complexity of the distributed MCDS computation is $O(1)$. Each cluster-head sends at most two messages, and the number of clusters is equal to rc, which is not more than n. Therefore, the total number of messages is $O(n)$. Each message contains the maximum and minimum locations of the nodes. Hence, the message size is $O(1)$. □

5 Data Dissemination Protocol

5.1 Overview

We design a data dissemination protocol that uses CDS and TDMA scheduling for transmissions. In a unit disk model, all nodes should receive data by virtue of the connected dominating set and TDMA scheduling. But, in reality, the radio is more complex. Unlike in a unit disk model, the link reliability has more than two values. Also, link reliability has temporal variation. To deal with packet losses, the protocol must ensure reliability. Again, the objective is to minimize the number of packet transmissions and the latency, in that order.

We divide data dissemination into two phases, viz. *streaming phase* and *recovery phase*. The data to be disseminated is divided into *packets*. Only the nodes in the CDS transmit packets during the streaming phase and the transmissions are scheduled. To recover lost packets, we use piggybacked negative acknowledgments during the streaming phase and separate negative acknowledgment messages during the recovery phase. At the end of the streaming phase, all the nodes in the CDS receive the data completely. And, at the end of the recovery phase, all the non-CDS nodes receive the data completely. Following is the description of the streaming phase and the recovery phase.

5.2 Streaming Phase

Let t_{hop} be the time taken for a packet to traverse one hop and $\mathcal{C}$ be the total number of colors for D-2 coloring of the CDS as mentioned in Section 4.2. t_{hop} includes transmission time. For simplicity, let us assume packet size to be a constant. Then,

$$streaming\ period = t_{hop} * \mathcal{C}. \tag{1}$$

Every packet contains the D-2 color of the sender. We use the synchronous reception property of the wireless medium to achieve time synchronization among the nodes [11]. In particular, when a node in the CDS for the first time hears a packet, it synchronizes its time with that of the sender. Let C_u and C_v be the D-2 color of the nodes u and v, respectively. Let t_0 be the time at u when u hears a packet from v. The smallest time difference ΔC between the transmissions of node u and v is calculated as follows:

$$\Delta C = |(C_u - C_v) \bmod \mathcal{C}| * t_{hop}. \tag{2}$$

Then, the transmission schedule of u is $(t_0 + \Delta C)$,

$$(t_0 + \Delta C + streaming\ period),$$
$$(t_0 + \Delta C + 2 * streaming\ period),$$
$$(t_0 + \Delta C + 3 * streaming\ period),$$

and so on.

We use a local timer at each node to compute the transmission schedule. In practice, the period of a timer has a drift. Further, the drift varies from node to node. Such varied drifts can cause a shift in the transmissions schedules of nodes, hence increasing the number of collisions. To compensate for the drift, a node continuously adjusts its local timer period so that the difference between any two successive transmissions is equal to the streaming period. This ensures that the transmissions follow the global TDMA schedule. Note that Sprinkler uses its data broadcast messages to achieve time synchronization of the required accuracy and does not require an explicit time synchronization service.

Each node, regardless of whether it is in the CDS or not, selects a neighboring node in the CDS as its parent. The objective of selecting a parent is to distribute the number retransmissions for lost data packets uniformly among the CDS nodes. To satisfy this objective, the criteria for parent selection could be as simple as distance or as complex as online link quality measurement. In our experiments, we use the distance. In particular, given a node u, let P_u be the set of CDS nodes, which are closer to the base station than u. Then, the *parent* of u is the closest neighbor of u in the set P_u.

A node in the CDS forwards each newly heard packet. It piggybacks negative acknowledgments for the lost packets and its parent ID while transmitting a packet. The parent retransmits the packet, if it is available, in the next time slot. Therefore, recovery is done on a hop-by-hop basis similar to PSFQ [4]. This avoids downstream propagation of a packet loss, which in turn reduces the number of negative acknowledgments in the recovery phase. Also, the streaming of retransmissions reduces the latency.

Let N be the total number of packets and i be the sequence number of the last heard packet. Then, each node calculates the duration of streaming phase after hearing a packet as follows:

$$duration\ of\ streaming\ phase = (N - i)\ *\ streaming\ period. \quad (3)$$

Therefore, the duration of streaming depends upon the total number of packets and the number of retransmissions.

5.3 Recovery Phase

If a node does not receive all the packets at the end of the streaming phase, it enters the recovery phase. Since all the CDS nodes have received all the packets at the end of streaming phase, only the non-CDS nodes will enter the recovery phase. To recover lost packets, a node unicasts a recovery request message containing a list of missing packets to its parent. In response, its parent unicasts the requested packets, called recovery data messages. Recovery request messages and recovery data messages are sent periodically at certain intervals. These intervals are tuned according to the density of the nodes to avoid network congestion.

The number of packet transmissions in the recovery phase depends upon the number of losses for non-CDS nodes at the end of the streaming phase. By using a proper value of R', we can select a CDS that will minimize the number of losses for non-CDS nodes during the streaming phase. In Section 4, we will describe how to select such an R'.

Although all the nodes are potential transmitters in the recovery phase, the number of transmissions during the recovery phase are few as compared to that of the streaming phase. Therefore, a global transmission schedule for all the nodes will result in wasted time slots. Instead, we use a link level unicast primitive which uses a RTS-CTS-DATA-ACK mechanism for coordination to reliably transmit packets during the recovery phase. Both the recovery request message and the recovery data message are sent via a RTS-CTS-DATA-ACK mechanism.

5.4 Power Management

As mentioned in Table 1, a radio in idle mode also consumes a noticeable amount of energy. In embedded devices, the current draw to switch the radio on or off is about the same as that of one packet transmission or reception, e.g., it takes 14 mA to turn the XSM radio on or off. Hence, putting the radio to sleep saves power when the sleep time is more than that required for two packet receptions. We provide a mechanism to switch off the radios of non-CDS nodes without any loss of data. As mentioned in Section 5.2, each node selects a parent node. Let u be any non-CDS node. When u hears first packet, it calculates TDMA schedule of its parent using (1). Then, during the streaming phase, u keeps its radio off except during time slots when its parent is scheduled to transmit. The periodic switching of radio by a non-CDS node is called *power save* mode.

At the end of streaming phase, if recovery is required, u switches on its radio and keeps it on until it has received all the packets. After recovering all the packets, u again enters power save mode.

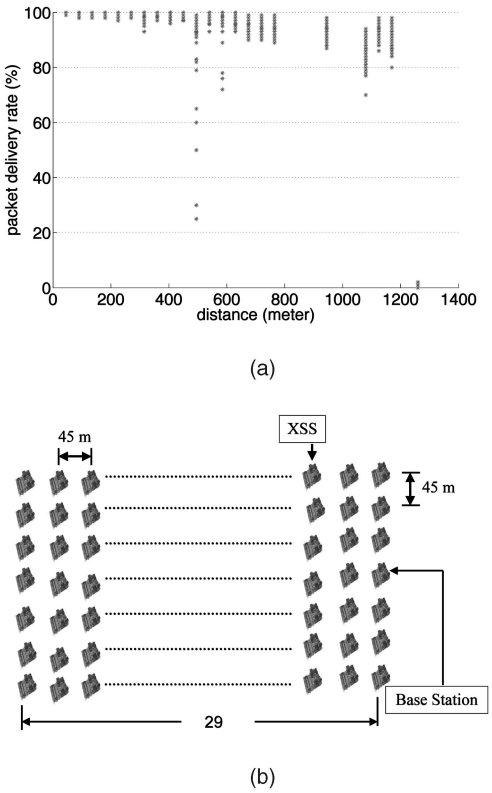


Fig. 4. Outdoor experiment setup. (a) Packet delivery ratio in an open field. (b) Outdoor network topology.

6 Performance of Sprinkler in Practice

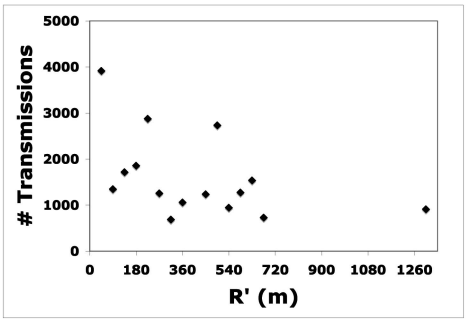
6.1 Adapting a Reliable Transmission Radius

The radio model in practice is more complex than a unit disk model, e.g., the packet delivery ratio for an outside environment is illustrated in Fig. 4a. Commonly used practical radio models are the two ray ground model and the shadow model. Computing a CDS under these models is complex as compared to that of the unit disk model. In this section, we present an adaptation of Sprinkler to the practical radio model.

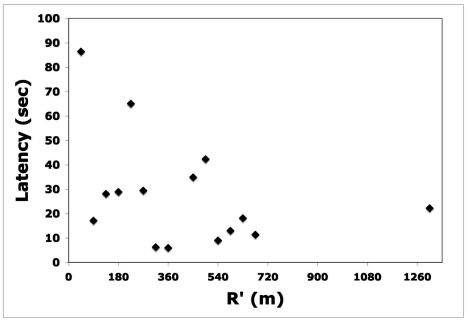
Lemma 2. *Given a graph G, if there exists at least one node in every square of length x in a rectangle enclosing G, then there also exists at least one node in every square of length $y > x$ in that rectangle.*

Proof. Since $y > x$, every square of length y encloses at least one square of length x. Hence, there also exists at least one node in every square of length y. □

According to our model in Section 2.1, each square of length $R/\sqrt{5}$ contains at least one node, where R is a reliable communication range. Therefore, from Lemma 2, every square of length greater than $R/\sqrt{5}$ also contains at least one node. Sprinkler then self-adapts the value of R' greater than or equal to the reliable communication range R such that the total number of transmissions are the minimum for data dissemination. We describe the adaption process for a sample outdoor environment in the following paragraph.



(a)



(b)

Fig. 5. Performance of Sprinkler in an outdoor environment. (a) R' versus number of packet transmissions. (b) R' versus latency.

We use XSS devices equipped with 9 dBi antennae of length 1.82 m for experiments. We deployed 203 XSSs in a grid of 29×7 at a separation of 45 m in a plane open field as shown in Fig. 4b. Sprinkler is implemented in C under EmStar [12] for Linux-based systems such as PC, startgate, iPAQs, etc. The payload consists of 100 data packets. According to the definition of R in Section 2.1, the reliable communication range is 270 m as shown in Fig. 4a. The location information is provided using a GPS device on each XSS. The CDS is a subset of nodes in the middle row. We varied the value of R', thereby resulting in different separation between the two neighboring senders. We find that an R' of 315 m results in the least number of packet transmissions as shown in Fig. 5a. To handle the impact of channel variation, adaption runs of Sprinkler have to execute multiple times. The confidence level behind the selection of R' depends upon number of runs. The number of packet transmissions increases after 315 m due to an extensive recovery phase, which is a consequence of high packet loss in the streaming phase. Fig. 5b shows the latency for various values of R'.

6.2 Scalability of Sprinkler

6.2.1 Network Setup

We use Kansei [13], the testbed of stargates equipped with an IEEE 802.11b ad hoc network, for our experiments. Kansei provides an option to attenuate the transmission power to attain the required power level. We use a network of 49 nodes arranged uniformly in a grid of area 7×7 at a separation of 0.91 m. The node at location $(0, 0)$ is the source of data dissemination as show in Fig. 6. The number of data

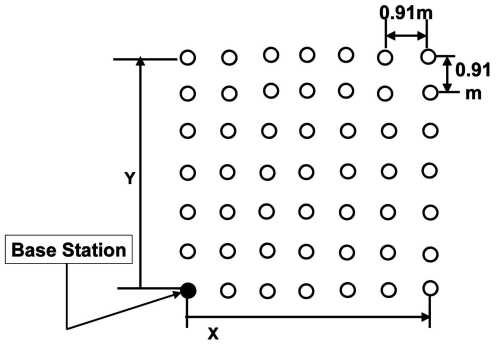


Fig. 6. Network topology.

packets to be disseminated is equal to 240. The mean packet delivery ratios mentioned in Table 2 is used in Kansei. The location is provided to each node using a localization service. According to the definition of R in Section 2.1, the value of R is equal to 0.91 m. Sprinkler adapts R' to a value of 1.83 m using the procedure mentioned in Section 4.

6.2.2 Analysis

We measure the performance of Sprinkler as a function of the number of nodes in the network. We consider two types of distribution, viz. constant density and increasing density. In case of constant density, we add nodes without changing the number of nodes per a square of length $R'/\sqrt{5}$. And, in case of increasing density, we add an equal number of nodes to all the existing squares of length $R'/\sqrt{5}$.

Following are the formulas to compute the number of packet transmissions and the latency:

$$\begin{aligned} &number\ of\ packet\ transmissions \\ &\quad = number\ of\ packets * |CDS|, \end{aligned} \tag{4}$$

$$\begin{aligned} latency = {} & number\ of\ packets \\ & * (number\ of\ hops\ in\ diameter + \mathcal{C}) * t_{hop}. \end{aligned} \tag{5}$$

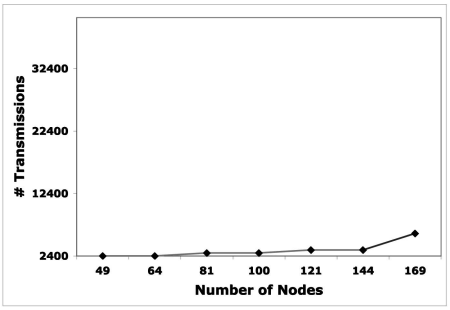
From Lemma 1, the number of nodes in the CDS of a network is a linear function of the number of nodes in the network. Hence, the number of packet transmissions is also a linear function of the number of nodes in the network.

TABLE 2
Mean Packet Delivery Ratios

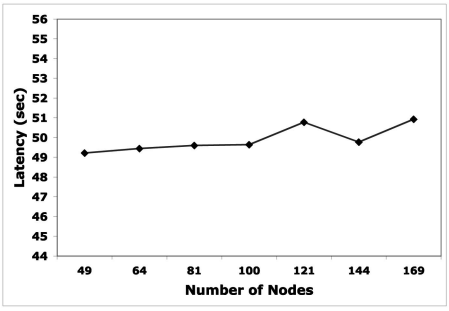
Distance(m)	packet delivery ratio (%)
0.91	98.58
1.83	97.19
2.74	94.63
3.66	89.67
4.57	29.12
5.49	87.69
6.4	12.28
7.32	0.53
8.23	84.98
10.06	0.08
12.8	0.01
13.11	0

TABLE 3
Number of CDS Nodes and Number of Hops in Diameter

# Nodes	$\|CDS\|$	# Hops in Diameter
49	10	6
64	10	6
81	12	7
100	12	7
121	14	8
144	14	8
169	25	12



(a)



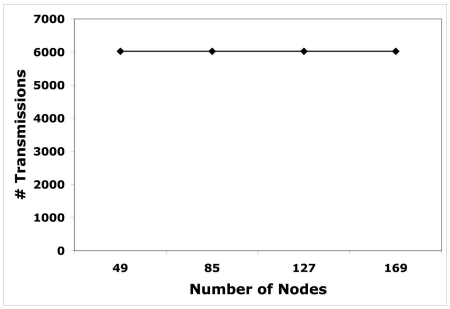
(b)

Fig. 7. Constant density. (a) Number of transmissions. (b) Latency.

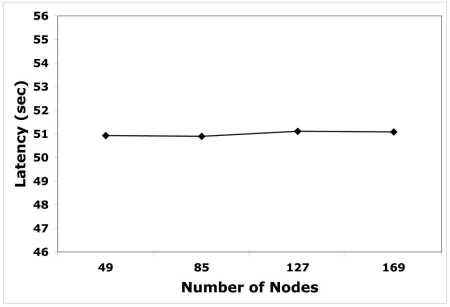
From the algorithm in Section 4.2, the number of D-2 colors is a constant. Hence, the latency is a linear function of the number of packets and the number of hops in the diameter of the network.

As we increase the number of nodes by keeping the density constant, the size of the CDS and the number of hops increase, as shown in Table 3. Hence, as shown in Fig. 7a and Fig. 7b, the number of packet transmissions and the latency increase linearly with the number of nodes. The spike in latency for numbers of nodes equal to 121 is due to the granularity of the period in EmStar's software timer. The granularity of EmStar's timer leads to a drift, which is up to 10 msec. Therefore, the maximum offset in latency for disseminating 240 packets in streaming phase would be 2.4 sec. The spike in Fig. 7b is about 2 sec.

As we increase the number of nodes by increasing the density, the size of the CDS and the number of hops still remain the same. In other words, the new nodes are added to the existing clusters. These nodes receive data by overhearing the broadcast packets sent by the CDS nodes.



(a)



(b)

Fig. 8. Increasing density. (a) Number of transmissions. (b) Latency.

Hence, as shown in Fig. 8a and Fig. 8b, both the number of packet transmissions and the latency are constant.

7 Comparison with Existing Reliable Data Dissemination Schemes

7.1 Comparison with Deluge

Deluge is a reliable data dissemination protocol used for sensor network reprogramming at scale [1]. Deluge does not compute a CDS or a TDMA schedule. It uses heuristics to optimize the number of packet transmissions and latency. Since it is complex to theoretically analyze the performance of Deluge, we do a simulation-based comparison.

Deluge is implemented in NesC [14] under TinyOS [15] for mote platforms. TOSSIM [16] is a simulator for TinyOS-based NesC programs. It has an option to specify the network topology and packet delivery ratio. We used Tython [17], which is a scripting tool, to set up simulation parameters. We used Kansei [13] to conduct experiments with Sprinkler. Under TOSSIM and Kansei, we use the same network setup as mentioned in Section 4.2.1. The values of R and R' are the same as those in Section 4.2.1. We are working on porting Deluge in C, so that, in the future, we can compare performances under a single setting of Kansei. Similarly, we have started porting Sprinkler to NesC which will enable us to compare the performance under TOSSIM.

Fig. 9a and Fig. 9b capture the distribution of the number of data packets transmitted by the nodes in Deluge and Sprinkler, respectively. The size of the circle represents the number of packet transmissions. The reason behind nonuniform separation between nodes is random selection of a clusterhead in a square.

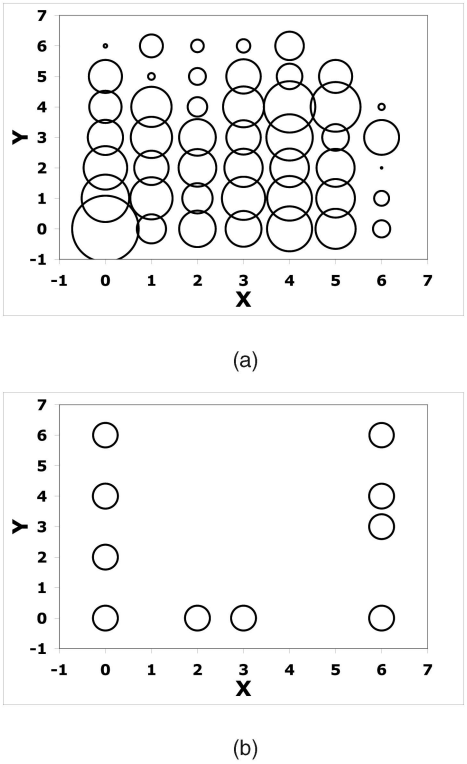


Fig. 9. Number of packet transmissions. (a) Deluge. (b) Sprinkler.

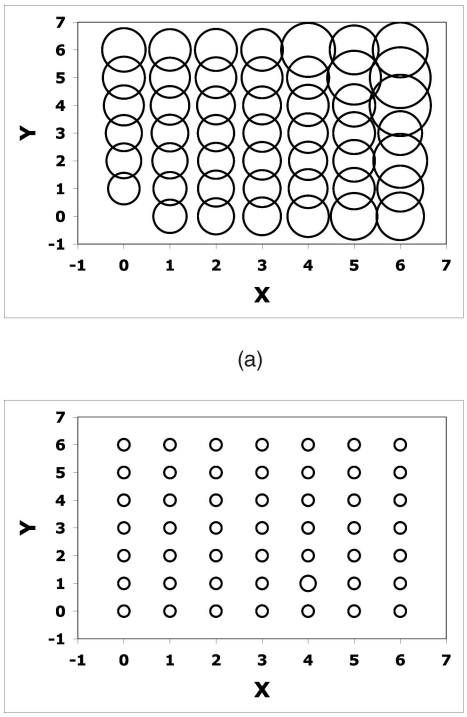


Fig. 10. Latency. (a) Deluge. (b) Sprinkler.

We compare the latency in terms of an abstract time unit, which is equal to the time taken to send a single packet. Fig. 10a and Fig. 10b capture the distribution of latency for the nodes for Deluge and Sprinkler, respectively. The size of the circle represents the latency. Note that the number of packet transmissions and latency is uniformly distributed in case of Sprinkler by the virtue of a regular structure of the computed CDS and TDMA schedule. For a sparse network, while we expect Deluge to have a higher number of data senders, for Sprinkler, we can tune the value of R' to have a number of senders equal to a constant factor of the optimum.

Table 4 gives a comparison in terms of the total number of data senders, packet transmissions, and maximum latency in the network. Sprinkler avoids extraneous transmissions by computing CDS as a function of the transmission radius and scheduling the transmissions to preclude collisions.

7.2 Comparison with Other Schemes

Pump Slowly and Fetch Quickly (*PSFQ*) [4] is a protocol designed to reliably disseminate a large number of data packets. PSFQ operates in two alternating phases, viz. pumping where data is forwarded by source and fetching where destinations recover lost packets using a request-reply handshake. The idea is to perform pump operation at a slow rate and fetch at a fast rate. In contrast, Sprinkler optimizes the rate at which data is transmitted by the sender by calculating an optimal transmission schedule that reduces collisions. Hence, Sprinkler results in lower latency. *Multihop Network Reprogramming* (*MNP*) [3] and Infuse [2] are two reprogramming services for sensor networks. MNP suppresses the number of senders by selecting a few senders in a local manner and pipelines messages for faster delivery, but it does not avoid the hidden terminal problem. Infuse uses TDMA scheduling to avoid collisions but does not optimize the number of senders; hence, it is less energy efficient and requires a higher number of slots.

8 Related Work

In this paper, we address the problem of reliable data dissemination of multiple data packets to all the nodes in a wireless network of energy constrained embedded devices. The reliability has to be 100 percent in terms of both the number of packets and receivers. Further, the protocol has to be energy efficient and simple to be executed on devices with slow processors. We give a brief comparison with the existing data dissemination services.

A naive scheme of the single retransmission of packets results in a *broadcast storm problem* as discussed in [18], which means that collisions and contention hamper reliability. Further, this is not energy efficient. The solution in [18] suppresses transmissions but does not guarantee delivery of all the data packets. The SPIN-RL [19] and Trickle [20] use an advertise-request-reply handshake protocol to disseminate data. But, these protocols are

TABLE 4
Comparison Regarding Packet Transmission and Latency

	Deluge	Sprinkler
# Data Packet Senders	47	10
# Packet Transmissions	38450	2400
Latency	514.31	32.56

designed for a small number of data packets and, hence, are not optimized for bulk data transfer. While [21], [22], and [23] use a CDS to suppress the number of senders, the first two are designed for unicasting a single packet and the latter for a single packet broadcast. Therefore, these schemes address a different problem than that of Sprinkler. The problem of reliable bulk data dissemination and that of single packet dissemination are different. We describe one difference in protocol design. In bulk data dissemination, a bulk datum has to be divided into multiple data packets, each of which could possibly be sent using multiple sessions of a single packet dissemination protocol. Such multiple sessions could either overlap in time or not. While the first case would result in collisions and/or congestion, the latter case would result in increased latency as compared to Sprinkler, which uses pipelining.

Parthasarathy and Gandhi [7] consider the problem of constructing a *virtual backbone*, i.e., a CDS, in an ad hoc network that can be used for broadcast and unicast routing. They also use a D-2 coloring algorithm to calculate TDMA scheduling. The time complexity of their CDS construction algorithm is $O(\log^2 n)$ and that of D-2 coloring is $O(\Delta \log^2 n)$, where Δ is the maximum degree in the network; in the case of Sprinkler, the time complexity of both CDS construction and D-2 coloring is $O(1)$. Recently, Dubhashi et al. [24] proposed a CDS-based scheme of time complexity $O(1)$ for broadcasting. But, the performance factor of Dubhashi's scheme is not proved to be constant and it uses flooding to disseminate the data. While both these schemes, i.e., Parthasarathy and Gandhi's [7] and Dubhashi et al.'s [24], are randomized and give probabilistic guarantees, Sprinkler is deterministic.

Alzoubi [25] considers the problem of distributed construction of a CDS for an ad hoc wireless network. Their algorithm first builds a spanning tree and then forms CDS based on the spanning tree. The performance ratio of the CDS is eight and the time complexity is O(n). Since maintenance of a spanning tree takes O(n) time in the presence of node failures and movement, Alzoubi later proposes an efficient CDS algorithm [26], which does not maintain a spanning tree and locally heals a CDS by maintaining 2-hop neighborhood information. The performance ratio of the algorithm is 192 and the time complexity is O(n). Sprinkler has a performance ratio of $18\frac{2}{3}$, time complexity of O(1), and heals a CDS by maintaining 1-hop neighborhood information.

Parthasarathy and Gandhi and Alboubi's solutions do not assume a minimum density and location information unlike Sprinkler. As discussed earlier, we are working on cluster formation without the location information.

9 Discussion

Constant use of a CDS would result in depletion of the CDS nodes. A load-balancing policy can be enforced in order to guarantee the uniform use of energy across the clusters. After every constant number of packet transmissions, the entire CDS can be shifted by an offset of one horizontal row. Similarly, the CDS can also be shifted by a vertical row. Such a shifting would guarantee uniform load-balacing across all the clusters.

A majority of the large-scale deployments of wireless sensor networks follow bidimensional grid deployment or a variation of it, e.g., ExScal (Extreme Scale) and NESTFE (NEST Final Experiment) [27]. The primary reasons to choose bidimensional gridlike patterns for deployment are 1) simplicity in deployment and 2) high QoS sensing guarantees using a low number of sensors as compared to random or other geometric patterns. Bidimensional grid patterns satisfy Sprinkler's density assumption. Note that Sprinkler does not assume a bidimensional deployment pattern. It makes an assumption about density, which can be satisfied by various deployment patterns including random.

If the distance in the bidimensional grid deployment is $R/\sqrt{5}$, then such a deployment is optimal in the sense that we would need a minimum number of nodes for Sprinkler to work. The total number of deployed sensors can be calculated given the deployment area.

10 Conclusion

We presented Sprinkler, a reliable and energy efficient data dissemination service for wireless embedded devices. A preliminary version of the paper appeared in the *Proceedings of the IEEE Real-Time Systems Symposium (RTSS '05)* [28]. Sprinkler assumes a minimum node distribution density and the knowledge of location information. These assumptions enable us to efficiently construct clusters in the form of squares. We then use the regular grid structure of the clusterheads to efficiently compute a CDS and a corresponding TDMA schedule. As an output of CDS and TDMA algorithms, each node remembers its parent ID, whether it is a CDS node or not, and its time slot number. This is a constant amount of state. As part of the data dissemination protocol, each node maintains the total number of packets in a data dissemination session, the number of packets received, and the recovery requests of its children. Since the number of children is of the order of the degree of a node, the state maintained by Sprinkler during a data dissemination is $O(\Delta)$.

We showed that Sprinkler is tolerant to fail-stop and state corruption faults, which are common in embedded wireless networks. We analytically bounded its performance in terms of the number of packet transmissions and the latency. Further, we compared its performance with Deluge. We found that Sprinkler outperforms Deluge by a factor of 15 for a 7×7 network under simulation. We measured the performance of Sprinkler in an outdoor ad hoc IEEE 802.11 network of 203 devices and suggested an approach to optimize its performance. Sprinkler was successfully used at the backbone nodes of *ExScal*, the largest ever sensor network deployment [8], to broadcast new mote programs, XSS programs, and network management commands.

11 Future Work

Persistent usage of a CDS will result in a faster depletion of energy for the nodes in the CDS. Hence, we will extend the CDS construction algorithm to consider the energy levels of nodes so that the energy usage is balanced across the network. Since Sprinkler uniformly distributes the number

of packet transmissions across the network, it is simple to instrument load balancing. In the current implementation of Sprinkler, non-CDS nodes keep their radio on all the time. We will instrument turning off of radios for non-CDS nodes as described in Section 5.4 and experimentally compare the energy savings of Sprinkler with that of broadcast schemes.

Sprinkler requires location information to form clusters. We are currently working on a clustering algorithm which will guarantee square-shaped clusters without location information. We are also working on weakening the assumption about density. In particular, we will consider the networks where not every square has a node. The formation of square-shaped clusters results in low time complexity algorithms for CDS construction and D-2 coloring. Another regular shape that provides disjoint partitioning of networks like that of a square is a hexagon. We will explore hexagonal clustering to compute CDS and D-2 coloring. In practice, a D-k vertex coloring would suffice, as mentioned in Section 1. A D-k coloring algorithm will result in fewer time slots and, hence, less latency. Currently, we are working on a D-k vertex coloring algorithm of low time complexity.

As mentioned in Section 9, the large-scale sensor network deployments still prefer simplistic bidimensional grid patterns. However, it would be interesting to study deployment characteristics, such as method and number of nodes, for random deployment. It is a research problem in itself. We have not studied this problem. We have included this Section 11 for future work.

APPENDIX

FAULT-TOLERANCE OF SPRINKLER

In this section, we describe the fault-tolerance properties of Sprinkler with respect to the fault model specified in Section 2.2. We provide an intuitive argument. We use the notion of *critical variables* [29] to prove the fault-tolerance properties of Sprinkler. The critical variables are a subset of variables at each node. The correct values of critical variables imply the safety and liveness of a system.

The critical variables for Sprinkler are

- *Coordinates of Square*: Stores the coordinates of the square to which a node belongs.
- *Parent ID*: Stores the node ID of the parent.
- *Child IDs*: Stores the node IDs of the CDS children nodes

A.1 Addition of a New Node

A newly added node assumes the role of a non-CDS node. Hence, the addition of a new node does not affect the critical variables of any other node.

A.2 Node Fail-Stop

We consider two cases depending upon whether a fail-stopped node is a CDS node or not.

- **A non-CDS node**: A node does not maintain any critical state variable about any non-CDS node. Therefore, if a non-CDS node fail-stops, then no critical variable is affected.

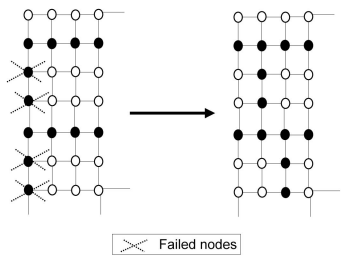


Fig. 11. Repairing of a CDS.

- **A CDS node**: If a CDS node f fail-stops, then the nodes in its transmission radius may not receive packets. In terms of critical variables, the Parent ID variable of all of f's children and Child IDs variable of f''s parent have incorrect values. We use the fault-containment property of clustering to locally repair the fault. In particular, the parent of f selects a new node g in the square of the f node as its child. All the children of f assign g as their new Parent ID.

  It could be the scenario that all the nodes in the child's cluster have fail-stopped. In such a scenario, there are two cases, either the child node belongs to a row in CDS or to a column in CDS. If a child node is a row-node, CDS can be repaired in constant time. The idea is to select a new pair of row-nodes such that CDS is connected. Fig. 11 illustrates a sample scenario, where four failed row-nodes have been repaired. Since, the cardinality of the candidate set of nodes is of the order of length of deployment field, the repairing time is constant. However, if the number of failed nodes is large or the failed child node is a column-node, then a CDS may not be constructed using Sprinkler's CDS construction algorithms. In such cases, standard CDS construction algorithms, which do not assume minimum density, have to be applied. These algorithms do not have a high performance ratio but also have nonconstant time complexities.

  The set $M\backslash\{f\}\bigcup\{g\}$ is a CDS. The color of the newly selected node is the same as that of the failed node. The intuitive proof of these two propositions lies in the fact that the location of the selected node in the square does not have any bearing on Sprinkler's CDS construction and D-2 coloring algorithms.

A.3 State Corruption

The corruption of any of the above mentioned three variables may result in an incorrect CDS or TDMA scheduling. A way to repair the critical variable is to periodically reassign values to them. The period depends upon how frequently the state corruption occurs.

ACKNOWLEDGMENTS

The authors would like to thank Emre Ertin for information related to XSS hardware; Lewis Girod, Jeremy Elson, Nithya Ramanathan, Martin Lukac, and other CENS members for help regarding the EmStar package; Pering Trevor, Epp Edward, Jaidev Prabhu, and Phil Buonadonna for startgate

hardware and system software; Jonathan Hui, Philip Levis, and Joerg Claussen for help in Deluge, TOSSIM, and Tython; Santosh Kumar for giving reference materials for optimization proof techniques and power consumption stats and correcting the proofs of the performance ratios; Sandip Bapat for his initial help in simulations while Sprinkler was a class project; Niranjan Balchandran and Janhavi Joshi for verifying mathematical proofs; and the anonymous reviewers for corrections and comments. This work was sponsored by DARPA NEST contract OSU-RF program F33615-01-C-1901 and by the US National Science Foundation under Grant No. 0341703. Intel sponsored 225 stargates.

REFERENCES

[1] J. Hui and D. Culler, "The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale," *Proc. Second Int'l Conf. Embedded Networked Sensor Systems (SenSys '04),* pp. 81-94, 2004.

[2] M. Arumugam, "Infuse: A TDMA Based Reprogramming Service for Sensor Networks," *Proc. Second Int'l Conf. Embedded Networked Sensor Systems (SenSys '04),* pp. 281-282, 2004.

[3] S. Kulkarni and L. Wang, "MNP: Multihop Network Reprogramming Service for Sensor Networks," *Proc. 25th Int'l Conf. Distributed Computing Systems,* June 2005.

[4] C. Wan, A. Campbell, and L. Krishnamurthy, "PSFG: A Reliable Transport Protocol for Wireless Sensor Networks," *Proc. First ACM Int'l Workshop Wireless Sensor Networks and Applications (WSNA '02),* pp. 1-11, 2002.

[5] B. Clark, C. Colbourn, and D. Johnson, "Unit Disk Graphs," *Discrete Math.,* vol. 86, pp. 165-177, 1990.

[6] S. Krumke, M. Marathe, and S. Ravi, "Models and Approximation Algorithms for Channel Assignment in Radio Networks," *Wireless Networks,* vol. 7, no. 6, pp. 575-584, 2001.

[7] S. Parthasarathy and R. Gandhi, "Fast Distributed Well Connected Dominating Sets for Ad Hoc Networks," Technical Report CS-TR-4559, Computer Science Dept., Univ. of Maryland, http://hdl.handle.net/1903/538, Feb. 2004.

[8] A. Arora, R. Ramnath, P. Sinha, E. Ertin, S. Bapat, V. Naik, V. Kulathumani, H. Zhang, M. Sidharan, S. Kumar, H. Cao, N. Seddon, C. Anderson, T. Herman, C. Zhang, N. Trivedi, M. Gouda, Y. Choi, M. Nesterenko, R. Shah, S. Kulkarni, M. Arumugam, L. Wang, D. Culler, P. Dutta, C. Sharp, G. Tolle, M. Grimmer, B. Ferriera, and K. Parker, "Project Exscal," *Proc. Int'l Conf. Distributed Computing in Sensor Systems,* 2005.

[9] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler, "Design of a Wireless Sensor Network Platform for Detecting Rare, Random, and Ephemeral Events," *Proc. Fourth Int'l Conf. Information Processing in Sensor Networks,* Apr. 2005.

[10] A. Sen and E. Melesinska, "On Approximation Algorithms for Radio Network Scheduling," *Proc. 35th Allerton Conf. Comm., Control, and Computing,* pp. 573-582, 1997.

[11] J. Elson, "Time Synchronization in Wireless Sensor Network," PhD dissertation, Univ. of California, Los Angeles, 2003.

[12] L. Girod, T. Stathopoulos, N. Ramanathan, J. Elson, D. Estrin, E. Osterweil, and T. Schoellhammer, "A System for Simulation, Emulation, and Deployment of Heterogeneous Sensor Networks," *Proc. Second ACM Conf. Embedded Networked Sensor Systems (SenSys '04),* 2004.

[13] V. Naik, S. Bapat, H. Zhang, C. Anderson, G. Fox, J. Wieseman, A. Arora, E. Ertin, and R. Ramnath, "Kansei: Sensor Tesbed for At-Scale Experiments," technical report, Computer Science and Engineering Dept., Ohio State Univ., http://www.cast.cse.ohio-state.edu/exscal/content/CheckpointPresentations/Exscal%20Open%20House/TestbedPoster-2005-01-31.pdf, Feb. 2005.

[14] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler, "The Nesc Language: A Holistic Approach to Networked Embedded Systems," *Proc. ACM SIGPLAN Conf. Programming Language Design and Implementation,* 2003.

[15] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System Architecture Directions for Networked Sensors," *Architectural Support for Programming Languages and Operating Systems,* pp. 93-104, 2000.

[16] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: Accurate and Scalable Simulation of Entire Tinyos Applications," *Proc. First Int'l Conf. Embedded Networked Sensor Systems (SenSys '03),* pp. 126-137, 2003.

[17] M. Demmer, P. Levis, A. Joki, E. Brewer, and D. Culler, "Tython: A Dynamic Simulation Environment for Sensor Networks," Technical Report UCB/CSD-05-1372, Electical Eng. and Computer Science Dept., Univ. of California, Berkeley, 2005.

[18] Y. Tseng, S. Ni, Y. Chen, and J. Sheu, "The Broadcast Storm Problem in a Mobile Ad Hoc Network," *Wireless Networks,* vol. 8, no. 2, pp. 153-167, 2002.

[19] J.W. Heinzelman and H. Balakrishnan, "Negotiation-Based Protocols for Disseminating Information in Wireless Sensor Networks," *Wireless Networks,* vol. 8, no. 2, pp. 169-185, 2002.

[20] P. Levis, N. Patel, S. Shenker, and D. Culler, "Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks," Technical Report UCB/CDS-03-1290, Computer Science Dept., Univ. of California, Berkeley, Mar. 2003.

[21] B. Das and V. Bharghavan, "Routing in Ad-Hoc Networks Using Minimum Connected Dominating Sets," *Proc. IEEE Int'l Conf. Comm.,* pp. 376-380, 1997.

[22] J. Wu and H. Li, "Domination and Its Applications in Ad Hoc Wireless Networks with Unidirectional Links," *Proc. Int'l Conf. Parallel Processing,* pp. 189-200, 2000.

[23] E. Pagani and G. Rossi, "Reliable Broadcast in Mobile Multihop Packet Networks," *Proc. MobiCom,* pp. 34-42, 1997.

[24] D. Dubhashi, O. Häggström, A. Panconsei, and M. Sozio, "Irrigating Ad Hoc Networks in Constant Time," *Proc. 17th ACM Symp. Parallelism in Algorithms and Architectures,* 2005.

[25] K. Alzoubi, "Virtual Backbones in Wireless Ad Hoc Networks," PhD dissertation, Illinois Inst. of Technology, 2002.

[26] K. Alzoubi, P. Wan, and O. Frieder, "Message-Optimal Connected Dominating Sets in Mobile Ad Hoc Networks," *Proc. Third ACM Int'l Symp. Mobile Ad Hoc Networking and Computing,* pp. 157-164, 2002.

[27] http://nest.cs.berkeley.edu/nestfe/index.php/Deployment, July 2005.

[28] V. Naik, A. Arora, P. Sinha, and H. Zhang, "Sprinkler: A Reliable and Energy Efficient Data Dissemination Service for Wireless Embedded Devices," *Proc. 26th IEEE Real-Time Systems Symp.,* Dec. 2005.

[29] M. Gouda, "Elements of Security: Closure, Convergence, and Protection," *Information Processing Letters,* vol. 77, pp. 109-114, 2001.

Vinayak Naik received the PhD degree in computer science from the Ohio State University in 2006 and the BEng degree from VJTI, India, in 1999. Since 2006, he has been a member of the research staff at Center for Embedded Networked Sensing (CENS) at the University of California, Los Angeles. His research focuses on fault-tolerant and secure distributed systems for large-scale wireless networks of embedded devices. He is a member of the IEEE.

Anish Arora received the BTech degree from the Indian Institute of Technology at New Delhi and the MS and PhD degrees from the University of Texas at Austin, all in computer science. He is a professor of computer science at the Ohio State University. His research is on fault tolerance, security, and timelines properties of systems, especially distributed and networked systems of large scale. Recent case studies in his research have centered on sensor networking and home networking, with support from DARPA, the US National Science Foundation, and Microsoft Research. He is a leading expert in self-stabilization and has chaired or cochaired seminars and symposia in this area in 1998, 1999, 2000, and 2002. He is program cochair of the 25th International Conference on Distributed Computer Systems. From 1989 to 1992, he worked at the Microelectronics and Computer Technology Corporation (MCC) in Austin, Texas. He is a member of the IEEE.

Prasun Sinha received the PhD degree from University of Illinois, Urbana-Champaign (UIUC) in 2001, the MS degree from Michigan State University (MSU) in 1997, and the BTech degree from IIT Delhi in 1995. He worked at Bell Labs, Lucent Technologies as a member of the technical staff from 2001 to 2003. Since 2003, he has been an assistant professor in Department of Computer Science and Engineering at the Ohio State University. His research focuses on the design of network protocols for sensor networks and mesh networks. He served on the program committees of various conferences including INFOCOM (2004-2006) and MobiCom (2004-2005). He has won several awards including the Ray Ozzie Fellowship (UIUC, 2000), the Mavis Memorial Scholarship (UIUC, 1999), and the Distinguished Academic Achievement Award (MSU, 1997). He received the prestigious US National Science Foundation CAREER award in 2006. He is a member of the IEEE.

Hongwei Zhang received the PhD degree in computer science from the Ohio State University in 2006. He is an assistant professor in the Department of Computer Science at Wayne State University. His research interests lie in dependable networked and distributed systems with a current focus on wireless systems such as sensor networks and mobile ad hoc networks. His research has been published in premier journals and conferences in computer networking, distributed computing, and dependable systems. His research results are also used in a variety of sensornet systems deployed by the Department of Defense, Motorola Labs, and other organizations. He is a member of the IEEE.