Sprinkler: A Reliable and Energy Efficient Data Dissemination Service for Wireless Embedded Devices*

Vinayak Naik
naik@cse.ohio-state.edu

Anish Arora
anish@cse.ohio-state.edu

Prasun Sinha
prasun@cse.ohio-state.edu

Hongwei Zhang
zhangho@cse.ohio-state.edu

Department of Computer Science and Engineering
The Ohio State University
Columbus, OH-43210

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 $\mathrm{O}(1)$ *times the optimum number of packets in* $\mathrm{O}(1)$ *of the optimum latency; its time complexity is* $\mathrm{O}(1)$*. 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 an indoor environment. Our indoor evaluation is based on an implementation in the Kansei testbed, that 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 have less latency. Further, the energy consumption of nodes and the latency grows linearly as a function of newly added nodes as network grows larger.*

*This work was sponsored by DARPA NEST contract OSU-RF program F33615-01-C-1901 and by the National Science Foundation under Grant No. 0341703. Intel sponsored 225 stargates.

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 relia-

bility.

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 [8], Infuse [4], MNP [10], and PSFQ [14] – 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 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.

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

Table 1. Energy required by common operations

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 determine 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 [9]. Intuitively, the reason behind distance two is that two non-neighboring nodes u and v interfere with each other if there exist 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, 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 [13]. Hence, the existing algorithms are not scalable for extremely large networks, such as those in project ExScal [3]. ExScal consisted of 1000 extreme scaling motes (XSMs) [6] 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 application 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, have $O(\Delta \log^2 n)$ time complexity, where Δ is the maximum degree in the network and n is the number of nodes in the network [13].

- 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 $2\frac{2}{3}$. The stat-of-the-art distributed D-2 coloring algorithm, with constant performance ratio, have time complexity $O(\Delta \log^2 n)$ [13]. 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.

- 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 problem 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 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. 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, and we summarize our findings in Section 9.

2. System 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. 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 rate 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 rate 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 rate is maximum. We use unit disk radio model for analytical evaluation of Sprinkler and probabilistic radio model for experiments and simulations.

The density of the nodes is such that if we divide the plane into a virtual grid of equal sized squares where (a) each square has at least one node and (b) 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.

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 $f : V \rightarrow \{1, ..., k\}$ such that $f(u) \neq f(v)$ whenever there is a path consisting of at most two edges between u and v in G 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 [9].

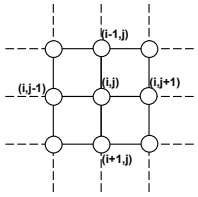


Figure 1. A Bi-dimensional Grid $B(1)$

A *bi-dimensional 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 Figure 1. $B(k)$ is a bi-dimensional 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

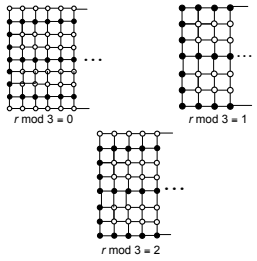


Figure 2. CDS of Bi-dimensional Grid $B(1)$

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 cluster-head. Only the cluster-head nodes are used to construct a CDS M. At the end of this section, we discuss the case where such clustering is not available.

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

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

Figure 2 illustrates M for various cases of r. Figure 3 illustrates an 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.The size of the CDS computed by the algorithm is at most $18\frac{2}{3}$ times that of an MCDS [11].

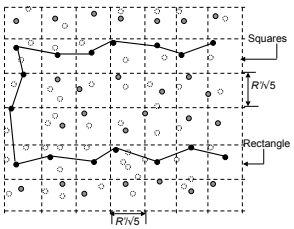


Figure 3. CDS Computation for Random Deployment

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 5 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:

- $(i \bmod 3 \equiv 0) \wedge (j = 0)$: $(i/3)8 \bmod 16$
- $(i \bmod 3 \equiv 0) \wedge (j \neq 0)$: $\{C(i,0) + j\} \bmod 16$
- $(i \bmod 3 \equiv 1) \wedge (j = 0)$: $\{C(i-1,0) + 6\} \bmod 16$
- $(i \bmod 3 \equiv 2) \wedge (j = 0)$: $\{C(i-1,0) + 7\} \bmod 16$

The number 16 insures that no two nodes with $2R'$ distance of each other have the same color. Figure 4 illustrates the application of the algorithm. Again, the dark circles represent in the nodes in M. The total number of colors is equal to 16. The time

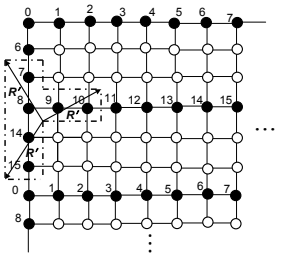


Figure 4. A D-2 Coloring for M

complexity of the D-2 coloring algorithm is $O(1)$. The number of colors computed by the algorithm is at most $2\frac{2}{3}$ times that for an optimal D-2 coloring [11].

5. Data Dissemination Protocol

5.1. Overview

We design a data dissemination protocol that uses a CDS and a 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 streaming phase and the transmissions are scheduled. To recover lost packets, we use piggybacked negative acknowledgments during streaming phase and separate negative acknowledgment messages during 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 the section 4.2. Then,

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

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 [7]. 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, C_v be the D-2 color of the nodes u, 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 transmission schedule. Note that Sprinkler uses its data broadcast messages to achieve time synchronization of 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 [14]. 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:

$$\textit{duration of streaming phase} = (N - i) * \textit{streaming period} \tag{3}$$

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 node have received all the packets at the end of streaming phase, only the non-CDS nodes will enter 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 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 streaming phase.

Although all the nodes are potential transmitters in recovery phase, the number of transmissions during recovery phase are few as compared to that of streaming phase. Therefore, a global transmission schedule for all the nodes will result in wasted time

slots. Instead, we use link level unicast primitive which uses RTS-CTS-DATA-ACK mechanism for coordination to reliably transmit packets during recovery phase.

6. Scalability of Sprinkler

6.1. Adapting Reliable Transmission Radius

The radio model in practice is more complex than a unit disk model.Commonly used practical radio models are two ray ground model and shadow model. Computing a CDS under these models is complex as compared to that of unit disk model. In this section, we present an adaptation of Sprinkler to practical radio model.

Theorem 1 *Given a graph G, if there exist at least one node in every square of length x in a rectangle enclosing G then there also exist a 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 exist a at least one node in every square of length y.

6.2. Network Setup

We use Kansei [12], the testbed of stargates equipped with IEEE 802.11b ad hoc network, for our experiments. Kansei provides an option to attenuate transmission power to attain required power level. We use a network of 49 nodes arranged uniformly in a grid of area 7×7 at a separation of 0.91m. The node at a corner is the base station. The number of data packets to be disseminated is equal to 240. The location is provided to each node using a localization service. According to the definition of R in section 2, the value of R in Kansei is equal to 0.91m. Sprinkler adapts R' to a value of 1.83m using the procedure mentioned in section 6.1.

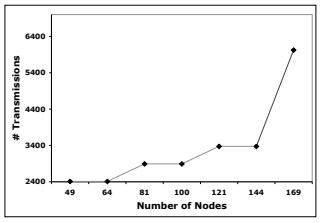
6.3. Analysis

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 equal number of nodes to all the existing squares of length $R'/\sqrt{5}$. Let *packet tx* denote packet transmisions.

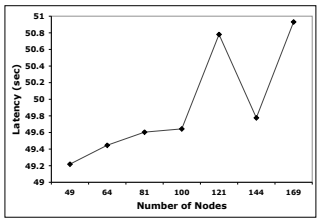
$$\textit{\#packet tx} = \textit{\#packets} * |CDS| \tag{4}$$

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. From 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.

$$\textit{latency} = \textit{\#packets} * (\textit{\#hops in diameter} + \mathcal{C}) * t_{hop} \tag{5}$$



(a) Number of Transmissions

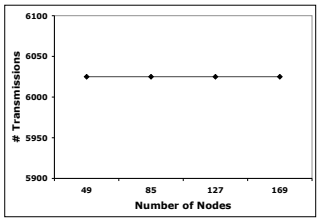


(b) Latency

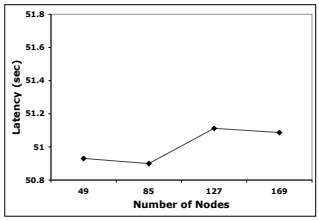
Figure 5. Constant Density

As we increase the number of nodes by keeping the density constant, the size of the CDS and the

number of hops increase. Hence, as shown in Figure 5(a) and 5(b), the number of packet transmissions and the latency increase linearly as the number of nodes. The spike in latency for number of nodes equal to 121 is due to the granularity of the period in EmStar's software timer.



(a) Number of Transmissions



(b) Latency

Figure 6. Increasing Density

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. Hence, as shown in Figure 6(a) and 6(b), both the number of packet transmissions and the latency are constant.

7. Comparison with Existing Reliable Data Dissemination Schemes

7.1. Comparison with Deluge

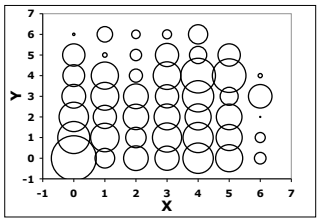
Deluge is a reliable data dissemination protocol used for sensor network reprogramming at scale [8]. Deluge does not compute a CDS or a TDMA schedule. It uses heuristics to optimize the number of packet transmissions and latency. Since it is complex to theoretically analyze the performance of Deluge, we do a simulation-based comparison.

Deluge is implemented in NesC under TinyOS for mote platforms. TOSSIM is a simulator for TinyOS-based NesC programs. It has an option to specify network topology and packet delivery rate. We used Tython , which is a scripting tool, to setup simulation parameters. Sprinkler is implemented in C under for Linux-based systems such as PC, startgate, iPAQs, etc. We use Kansei [12] to conduct experiments with Sprinkler. Under TOSSIM and Kansei, we use the same network setup as mentioned in section 6.2. The values of R and R' are same as those in section 6.2.

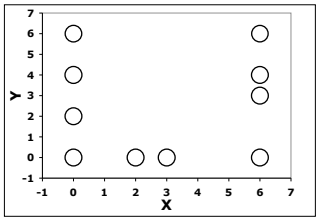
Figure 7(a) and 7(b) capture the distribution of number of data packets transmitted by the nodes in Deluge and Sprinkler respectively. The size of circle represents the number of packet transmissions. The reason behind non-uniform separation between nodes is random selection of a cluster-head in a square. We compare the latency in terms of an abstract time unit, which is equal to the time taken to send a single packet. Figures 8(a) and 8(b) capture the distribution of latency for the nodes for Deluge and Sprinkler respectively. The size of circle represents the latency. Note that the number of packet transmissions and latency is uniformly distributed in case of Sprinkler by the virtue of regular structure of the computed CDS and TDMA schedule. For a sparse network, while we expect Deluge to have more number of data senders, for Sprinkler we can tune the value of R' to have number sender equal to a constant factor of optimum.

7.2. Comparison with other Schemes

Pump Slowly and Fetch Quickly (*PSFQ*) [14] 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 request-reply handshake. The idea is to perform pump operation at a slow

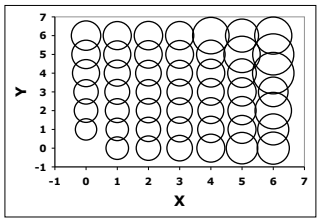


(a) Deluge

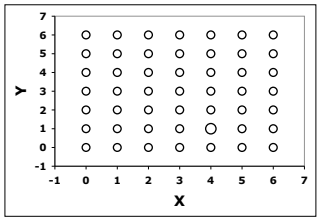


(b) Sprinkler

Figure 7. Number of Packet Transmissions



(a) Deluge



(b) Sprinkler

Figure 8. Latency

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*) [10] and Infuse [4] 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 hidden terminal problem. Infuse uses TDMA scheduling to avoid collisions but does not optimize number of senders, hence it is less energy efficient and requires more number of slots.

8. Related work

Parthasarathy et.al. [13] 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 case of Sprinkler, the time complexity of both CDS construction and D-2 coloring is $O(1)$. While Parthasarathy's scheme is randomized and gives probabilistic guarantees, Sprinkler is deterministic.

Alzoubi et. al. [1] consider the problem of distributed construction of a CDS for 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 8 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 [2], which does not maintain a spanning tree. The performance ratio of the algorithm is 192 and the time complexity is O(n). Sprinkler has a performance ratio of $18\frac{2}{3}$ and time complexity of O(1). Parthasarathy et

.al. and Alboubi et. al.'s solutions do not assume a minimum density and location information unlike Sprinkler.

9. Conclusion

We presented Sprinkler, a reliable and energy efficient data dissemination service for wireless embedded devices. 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 cluster-heads to efficiently compute a CDS and a corresponding TDMA schedule. 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.

References

[1] K. Alzoubi. *Virtual Backbones in Wireless Ad Hoc Networks*. PhD thesis, Illinois Institute of Technology, 2002.

[2] K. Alzoubi, P. Wan, and O. Frieder. Message-optimal connected dominating sets in mobile ad hoc networks. In *Proceedings of the 3rd ACM International Symposium on Mobile Ad hoc Networking & Computing*, pages 157–164. ACM Press, 2002.

[3] 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. In *International Conference on Distributed Computing in Sensor Systems*, 2005.

[4] M. Arumugam. Infuse: a tdma based reprogramming service for sensor networks. In *SenSys '04: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pages 281–282, New York, NY, USA, 2004. ACM Press.

[5] B. Clark, C. Colbourn, and D. Johnson. Unit disk graphs. *Discrete Mathematics*, 86:165–177, 1990.

[6] 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. In *The Fourth International Conference on Information Processing in Sensor Networks*, Apr. 2005.

[7] J. Elson. *Time Synchronization in Wireless Sensor Network*. PhD thesis, UCLA, 2003.

[8] J. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 81–94. ACM Press, 2004.

[9] S. Krumke, M. Marathe, and S. Ravi. Models and approximation algorithms for channel assignment in radio networks. *Wireless Networks*, 7(6):575–584, 2001.

[10] S. Kulkarni and L. Wang. Mnp: Multihop network reprogramming service for sensor networks. In *25th International Conference on Distributed Computing Systems*, June 2005.

[11] V. Naik, A. Arora, P. Sinha, and H. Zhang. Sprinkler: A reliable and energy efficient data dissemination service for wireless embedded devices. Technical Report ExScal-OSU-EN04-2005-05-11, The Ohio State University, Computer Science and Engineering Department, May 2005.

[12] 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, The Ohio State University, Computer Science and Engineering Department, http://www.cse.ohio-state.edu/exscal, Feb. 2005.

[13] S. Parthasarathy and R. Gandhi. Fast distributed well connected dominating sets for ad hoc networks. Technical Report UM Computer Science Department; CS-TR-4559, The University of Maryland, Computer Science Department, http://hdl.handle.net/1903/538, Feb. 2004.

[14] C. Wan, A. Campbell, and L. Krishnamurthy. Psfq: A reliable transport protocol for wireless sensor networks. In *WSNA '02: Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pages 1–11, New York, NY, USA, 2002. ACM Press.