Energy-Conserving Data Cache Placement in Sensor Networks

K. SHASHI PRABH and TAREK F. ABDELZAHER
University of Virginia

Wireless sensor networks hold a very promising future. The nodes of wireless sensor networks (WSN) have a small energy supply and limited bandwidth available. Since radio communication is expensive in terms of energy consumption, the nodes typically spend most of their energy reserve on wireless communication (rather than on CPU processing) for data dissemination and retrieval. Therefore, the role of energy conserving data communication protocols and services in WSN can not be overemphasized. Caching data at locations that minimize packet transmissions in the network reduces the power consumption in the network, and hence extends its lifetime. Finding locations of the nodes for caching data to minimize communication cost corresponds to finding the nodes of a weighted Minimum Steiner tree whose edge weights depend on the edge's Euclidean length and its data traffic rate. We call this tree a Steiner Data Caching Tree (SDCT). We prove that an optimal SDCT is binary, and that at-least two of the three internal angles formed at the Steiner points are equal. We derive expressions that determine the exact location of a Steiner point for a set of three nodes based on their location and their data refresh rate requirements. Based on these (optimality) results, we present a dynamic distributed energy-conserving application-layer service for data caching and asynchronous multicast. We present the results of simulation of our service that verifies its power saving properties.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design; J.7 [**Computers in Other Systems**]

General Terms: Algorithms, Performance, Theory

Additional Key Words and Phrases: Energy and bandwidth management, foundations of sensor networks, asynchronous multicast, data caching, Steiner tree

1. INTRODUCTION

Sensor networks consist of a large number of small devices, called nodes, spread over an area. These devices are equipped with sensors, actuators, memory, a processor, and communication ability. Nodes are deployed in large numbers, and upon deployment they self-organize into an ad-hoc wireless network. Though novel Wireless Sensor Network (WSN) applications are still emerging, WSNs

Authors' address: Department of Computer Science, University of Virginia, 113F Olsson Hall, 151 Engineer's Way, Charlottesville, VA 22904; email: {shashi,zaher}@virginia.edu.

have been used successfully to monitor wildlife [Szewczyk et al. 2004], track objects moving through monitored areas [He et al. 2004], and monitor equipment in a factory [Intel 2004]—beside many others.

These sensing devices have a very limited supply of energy. Thus, the energy consumption of a node must be optimized. Deployment of the nodes in large numbers means that the manufacturing cost of an individual node should be reasonably low for viability of wireless sensor networks. The limitations on their size and cost suggest that available energy will remain limited in the foreseeable future. In general, the energy consumed by a sensor node during its lifetime for communication over radio is much higher than that for computation. Hence, the energy consumption in a WSN that arises from data retrieval and delivery needs to be minimized.

Besides energy, bandwidth available for communication is another scarce resource. Organizing data dissemination paths as a tree with some nodes acting as data caches leads to a more efficient use of bandwidth and energy supply than unicast. Consequently, the network can support a larger number of subscribers for longer durations. We call such a tree a **S**teiner **D**ata **C**aching **T**ree or **SDCT**. In the following, we study the properties and construction of a SDCT in a WSN. For example, in Section 3, we prove that a minimum SDCT (**MSDCT**) is *binary*. For the sake of simplicity of mathematical analysis, this study assumes uniform distribution of nodes spread over a two-dimensional surface. One of our simulation studies, Section 5.6, shows that the energy conserving properties of the SDCT is quite robust in the presence of node distribution irregularities. We do not consider the effects of congestion, and radio and terrain irregularities. These factors may cause time-dependent variations in routing of the data packets, and weaken the energy saving properties of our approach.

The remainder of this article is organized as follows: In the next section, we present the data cache placement problem formulation and the service model. We prove properties of MSDCT in Section 3. Section 4 presents the dynamic data cache placement heuristic, followed by evaluations of the heuristic in Section 5. In Section 6, we present a brief discussion on adaptation of the existing rich collection of polynomial time Euclidean Steiner Tree algorithms to SDCT. Section 7 reviews related work, followed by conclusions in Section 8.

2. PROBLEM FORMULATION AND SERVICE MODEL

In the following, we consider the problem of efficient dissemination and retrieval of data pertaining to some event of interest to multiple subscribers. In WSN applications, such as monitoring, events of interest occur at certain places. These events are then sensed by local sensor nodes. One or more of these nodes send the processed information to the subscribers of the data, also called base stations, or sinks. The node that communicates data about the event to the subscribers can be chosen using some distributed leader election algorithm such as Malpani et al. [2000]. This scenario can be described as a publish-subscribe model, in which the leader node is the publisher of data, and the subscribers belong to the multicast group.

The frequency of data sent to the subscribers depends upon the frequency of data generated by the event, and the requested data refresh rate of the subscribers. *Data Refresh Rate* on a link is the rate at which data pertaining to the event being monitored by the WSN should be delivered to the receiver. This is different from the data generation rate by individual sensors, which is related to the rate of change of event state itself. The data refresh rate, for example, may be set by the receiver in accordance with its maximum tolerated data staleness and it might be lower than the rate of change of data at the source. To optimize energy consumption, update traffic is *asynchronously* multicast from focus locales (where events of interest occur) to subscribers. In contrast to *multicast*, in which the rate at which data is sent to a set of subscribers is the same, in *asynchronous multicast*, the rate at which data is sent to each subscriber is not necessarily the same. Hence, data may need to be buffered at intermediate nodes of the tree and forwarded asynchronously at a different rate from the one at which it was received. We call nodes that buffer and asynchronously forward data, *data caches*.

Given a publisher (with a corresponding data generation rate) and some subscribers (with their respective desired refresh rates), we want to reduce the traffic in the network. As noted earlier, organizing data dissemination paths as a tree leads to a more efficient use of bandwidth and energy supply than unicast. The nodes of the tree are used as caches for buffering and dissemination of data. In a wireless sensor network, since a large number of nodes are scattered in an area of interest, there is considerable freedom in the selection of caches' locations. We pick those nodes for caching data that minimize the cost of the tree. This problem is a variant of the *Steiner Problem*, which is defined as follows: Given a finite set of points, P, in a plane (or in some other metric space), find a network that connects all the points of the set with minimal length [Cieslik 1998]. The solution of this problem is a tree that is called Steiner Minimal Tree (SMT). The tree may contain points other than those contained in the set P; such points are called *Steiner points*. Garey et al. [1977] showed that the SMT problem is NP-hard.

For our problem, to find the locations of caches, we consider Steiner trees whose edges are weighted by the traffic rate *in addition to* their Euclidean length. We refer to these trees as Steiner Data Caching Trees (SDCT). The SDCT that has the minimum cost is called the Minimum Steiner Data Caching Tree (MSDCT). In the absence of channel irregularities, packets are forwarded from one cache to another on a path with a number of hops that is roughly proportional to the Euclidean distance between the two cache locations. Observe that this proportionality assumption does not imply that packets are routed on a minimum-hop path or even on a straight-line path. Individual hop distances along a path between two caches might be much smaller than the radio range (to eliminate poor channels). Hops might not be perfectly aligned in the same direction. However, given a long enough path, it is intuitive that the number of hops traversed will generally increase with increasing distance to destination. Hence, we take Euclidean length as a metric of path cost.

Finally, in the presence of congestion or lossy terrains, packets may need multiple retransmissions or a route that deviates significantly from proportionality to distance (e.g., to route around a hot-spot). In these circumstances, the edges of the SDCT needs to be weighted differently to account for the congestion cost. For example, on lossy paths the edge weights can be expressed as a function of the expected number of retransmissions in addition to the Euclidean length and transmission rate. We delegate this straightforward extension to future work. After a brief description of the data-caching service model, we shall continue our discussion of SDCTs.

2.1 Service Model

Consider a data caching tree where nodes represent cache locations (as well as the locations of the source and final destinations) and edges connect communicating nodes. While each edge is itself a multi-hop path, we find it useful to abstract it by a single path-cost-value that is proportional to the product of its length and rate of data transmission. Observe that all nodes other than the source and final destinations are Steiner points (caches) added to the tree to reduce cost. It is desired to find the most appropriate locations for hosting these caches (i.e., to find the minimum Steiner data caching tree, MSDCT).

In the following, we prove that an MSDCT is binary and at-least two of the three internal angles at any given internal node (Steiner point) are equal. In binary trees each internal node has three neighbors. We derive expressions that determine the exact location of the Steiner point for three nodes. Each data cache uses this result to locate itself optimally with respect to its three neighbors on the binary tree, hence optimizing local cost. Collectively, the overall result is a Steiner tree structure that reduces the cost of asynchronous multicast from the source to all destinations for the given data generation rate and requested refresh rates.

We use these results to develop an application-layer service (Section 4) that conserves energy by caching data at near-optimal locations, and sending asynchronous multicast to the subscribers from dedicated caches. As mentioned above, the publisher, the intermediate cache nodes, and the subscribers form the asynchronous multicast tree. Data caches located at the nodes of the multicast tree are updated in a lazy fashion. Data being communicated is non-accumulative, and hence only its most recent copy is kept in the caches. This data-caching architecture is shown in Figure 1. It provides these main functionalities.

(1) It reconstructs the data caching tree dynamically and in a distributed fashion upon joining and leaving of subscribers or changes in their data refresh rate requests.
(2) It sends data to the subscribers from dedicated caches instead of the original source(s).
(3) It finds near-optimal locations for creating caches that minimize the energy expended in data dissemination.
(4) It sends data to the subscribers at optimal rates.

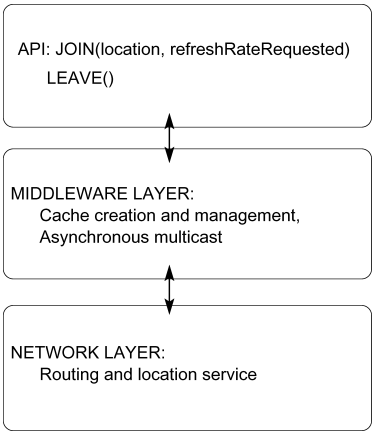


Fig. 1. Middleware architecture.

3. PROPERTIES OF AN OPTIMAL STEINER DATA CACHING TREE

In this section, we study the properties of a minimum Steiner data caching tree (MSDCT) for asynchronous multicast. First, we shall formulate rules to determine the optimum refresh rates on the edges of the tree. As we shall see in the next subsection, an optimal assignment of data refresh rates on the branches of the SDCT leads to elegant symmetries in the tree structure; specifically, two of the three internal angles formed at any node of the tree are always equal, and these angles can be expressed as a function of the refresh rates. We shall then derive an exact expression for the location of an Steiner point for a set of three nodes. At the end of the section, we prove that the MSDCT is binary. In the next section, this result will be used to formulate a heuristic for dynamically constructing and managing a near-optimal SDCT.

3.1 Refresh Rate Rules

As shown in Figure 2, let A represent the sensor node sending data to S. S is the cache that serves B and C, where B and C are either caches or subscribers. Let R_A, R_B, and R_C be the requested data refresh rates of A, B and C respectively.[1]

The optimal data refresh rates on the edges are determined on the basis of the following two observations.

—The maximum refresh rate on any of the branches cannot exceed the rate at which data is generated by the publisher.

—Data need not be sent at a higher rate than the subscriber's requested rate.

The following equations reflect these observations. We base the rest of this article on these equations, and refer to them as the **Refresh Rate Rules**.

[1]We use the following notation in this article:
R_N = Requested data refresh rate of node N.
$R_{N_iN_j}$ = Actual refresh rate on the edge connecting the nodes N_i and N_j.

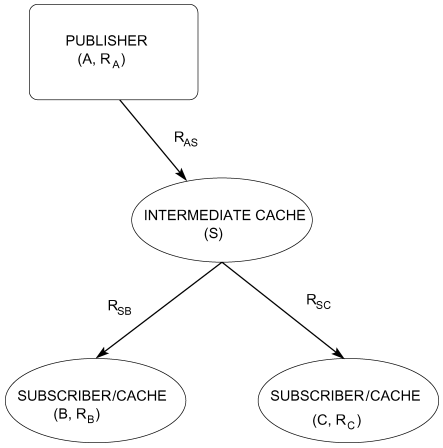


Fig. 2. A cache S serving two nodes B and C. The cache receives data from A.

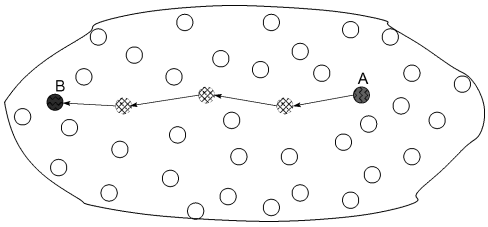


Fig. 3. Data transfer from source A to a subscriber B.

Rule-1: If $R_A \geq R_B \geq R_C$, then $R_{AS} = R_B = R_{SB}$; $R_{SC} = R_C$. (1)

Rule-2: If $R_B \geq R_A \geq R_C$, then $R_{AS} = R_A = R_{SB}$; $R_{SC} = R_C$. (2)

Rule-3: If $R_B \geq R_C \geq R_A$, then $R_{AS} = R_A = R_{SB} = R_{SC}$. (3)

Similarly, for the case when $R_C \geq R_B$. That is, at least two edges always have the same refresh rate. As we shall see in the following theorems, applying these rather simple rules to the data caching problem yields elegant symmetries. Next, we describe the cost metric of SDCTs.

3.2 Cost Metric of SDCT

We set the weight w_{AB} of the edge connecting two nodes A and B as:

$$w_{AB} = d_{AB} * R_{AB}, \tag{4}$$

where d_{AB} is the Euclidean distance between the nodes, and R_{AB} is the data refresh rate on the edge (Figure 3). This equation expresses the cost metric of SDCTs in a network of uniformly distributed nodes, free from congestion or other irregularities. The metric states that the number of transmissions (per unit time) along a path grows with both path length and refresh rate. It does not make explicit assumptions, however, on individual hop length, routing policy, or path shape.

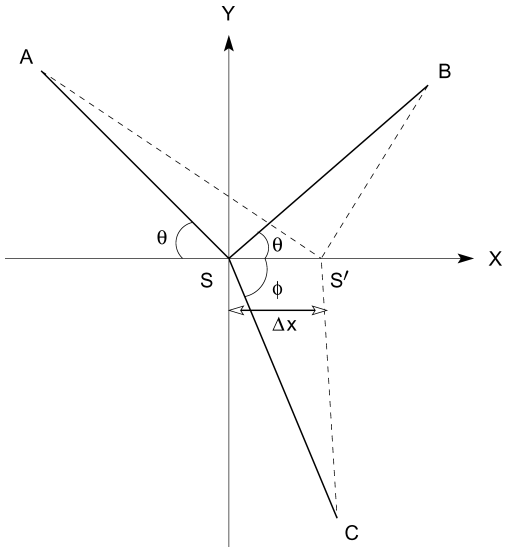


Fig. 4. Construction to obtain $\partial c_{ABC}(S)/\partial x$. We prove that $\phi = \pi/2$.

As mentioned in the previous section, congestion and nonideal radio channels may call for a modification of this metric. In practice, it is good to avoid long hops, since they are usually less reliable and entail more retransmissions. In the following treatment, we assume that long-hop neighbors are black-listed and nonreliable links are eliminated from routing tables. Hence, all remaining links are reliable.

3.3 Internal Angles and the Steiner Point

In this section, we prove that at least two of the three internal angles formed at the Steiner point by a set of three nodes are equal. We follow the proof by deriving the expression for the internal angles in terms of the refresh rates, and use the result to find the Steiner point.

INTERNAL ANGLE THEOREM. *For a given set of three vertices $\{A, B, C\}$ and their corresponding Steiner point S, if the cost metric of an edge is defined as the product of its Euclidean length and data traffic rate, and the data traffic rates are determined by the Refresh Rate Rules, then at least two of the internal angles formed at the Steiner point S are equal.*

PROOF. Let $c_{ABC}(S)$ be the cost of the tree T formed by $\{A, B, C, S\}$, as shown in Figure 4. Then $c_{ABC}(S) = AS * R_{AS} + SB * R_{SB} + SC * R_{SC}$. The three internal angles at S are: $\angle ASB$, $\angle ASC$, and $\angle BSC$. The three refresh rates R_{AS}, R_{SB}, and R_{SC} on the branches are determined using the Refresh Rate Rules (1–3). Without loss of generality, let R_C be the smaller of $\{R_B, R_C\}$. First, we consider the case $R_A \geq R_B \geq R_C$. Application of the rule 1 implies $R_{AS} = R_B = R_{SB}$, $R_{SC} = R_C$. We normalize the rates $R_{AS} = R_{SB}$ to 1. Therefore, $R_{SC} = R_C/R_{SB}$. Next, if $R_B \geq R_A \geq R_C$, then the rule 2 implies $R_{AS} = R_A = R_{SB}$, $R_{SC} = R_C$. After normalizing the rates $R_{AS} = R_{SB}$ to 1, once again we get $R_{SC} = R_C/R_{SB}$. Finally, if $R_B \geq R_C \geq R_A$, all three refresh rates on the

edges are equal, and hence upon normalizing, they all are 1. We denote R_{SC} by r. Clearly, $r \leq 1$.

Let us draw a line through S that bisects the $\angle ASB$, as shown in the figure. We begin with assumption that $\angle S'SC = \phi$, where ϕ is some arbitrary angle. We shall prove in the following that $\angle S'SC = \pi/2$, that is C lies on the bisector of $\angle ASB$.

The cost of T in terms of normalized rates is: $c_{ABC}(S) = AS + SB + rSC$. For the sake of brevity, we shall denote $c_{ABC}(S)$ by $c(S)$ in the following. Since S is the Steiner point, $c(S)$ is also the minimum. Hence, $\delta c(S) = 0$ for infinitesimal displacements around S. Any such displacement can be decomposed into its X and Y components yielding $\frac{\partial c(S)}{\partial x} = 0 = \frac{\partial c(S)}{\partial y}$. For simplifying the proof, we shall set the bisector of the $\angle ASB$ to be the Y-axis. We now consider $\frac{\partial c(S)}{\partial x}$. Let S' be a point on the X-axis such that $SS' = \Delta x$. Let $\angle BSS' = \theta$.

From the cosine law of triangles,

$$\begin{aligned}
AS'^2 &= AS^2 + \Delta x^2 - 2\,AS\ \Delta x\ \cos(\pi - \theta) \\
&= AS^2 + \Delta x^2 + 2\,AS\ \Delta x\ \cos\theta \\
\text{or, } AS' &= AS\left(1 + \frac{2\ \cos\theta}{AS}\ \Delta x + \frac{1}{AS^2}\ \Delta x^2\right)^{1/2} \\
&= AS\left(1 + \frac{\cos\theta}{AS}\ \Delta x + O(\Delta x^2)\right) \text{ using binomial expansion} \\
&\quad O(\Delta x^2) \text{ denotes a series whose terms contain } \Delta x^n, \text{ where } n \geq 2 \\
&= AS + \Delta x \cos\theta + O(\Delta x^2). \qquad (5)
\end{aligned}$$

Similarly,

$$BS' = BS - \Delta x \cos\theta + O(\Delta x^2) \qquad (6)$$

$$CS' = CS - \Delta x \cos\phi + O(\Delta x^2). \qquad (7)$$

Therefore,

$$\begin{aligned}
c(S') &= AS' + BS' + rCS' \\
&= AS + BS + rCS - r\Delta x \cos\phi + O(\Delta x^2) \\
&= c(S) - r\Delta x \cos\phi + O(\Delta x^2) \\
\Rightarrow \frac{\partial c}{\partial x} &= \lim_{\Delta x \to 0} \frac{c(S') - c(S)}{\Delta x} \\
&= \lim_{\Delta x \to 0} [-r\cos\phi + O(\Delta x)] \\
&= -r\cos\phi. \qquad (8)
\end{aligned}$$

Therefore, $\partial c/\partial x = 0$ if $\phi = (2n+1)\pi/2,\ n \in \mathbb{Z}$. By construction, $\phi \in [0, \pi]$. Hence, $n = 0$, and $\phi = \pi/2$ is the only admissible solution. Moreover, since ϕ, the root of $\partial c/\partial x = 0$, is univalued, $\phi = \pi/2$ is also the *global minimum*. Therefore, C lies on the Y-axis, or, $\angle ASC = \angle BSC$. This completes the proof. □

3.3.1 *Expressions for Internal Angles and the Steiner Point.* We now derive expressions for internal angles in terms of the normalized refresh rate ratio r

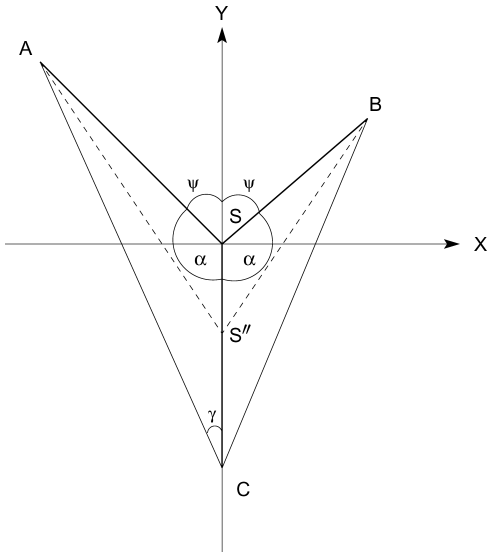


Fig. 5. Finding expression for the internal angles α, and ψ.

and the exact location of the Steiner point, for a set of three nodes. Please refer to Figure 5 for the following derivations. From the Internal Angle Theorem, $\angle ASC = \angle BSC$. Consider a point S'' along the Y-axis at distance Δy from S. Since $c(S) = AS + SB + rSC$ is the minimum, $\partial c(S)/\partial y = 0$.

Similar to Eq. (5),

$$AS'' = AS - \cos\alpha\ \Delta y + O(\Delta y^2) \tag{9}$$

$$BS'' = BS - \cos\alpha\ \Delta y + O(\Delta y^2) \tag{10}$$

$$CS'' = CS - \Delta y. \tag{11}$$

Therefore,

$$\begin{aligned} c(S'') &= AS'' + BS'' + rCS'' \\ &= AS + BS + rCS - 2\cos\alpha\ \Delta y - r\ \Delta y + O(\Delta y^2) \\ &= c(S) - 2\cos\alpha\ \Delta y - r\ \Delta y + O(\Delta y^2) \end{aligned} \tag{12}$$

$$\begin{aligned} \frac{\partial c}{\partial y} &= \lim_{\Delta y \to 0} \frac{c(S'') - c(S)}{\Delta y} \\ &= \lim_{\Delta y \to 0} [-2\cos\alpha - r + O(\Delta y)] \\ &= 2\cos(\pi - \alpha) - r \end{aligned}$$

$$\frac{\partial c}{\partial y} = 0 \Rightarrow \alpha = \pi - \arccos(r/2) \tag{13}$$

$$\text{or, } \psi = \arccos(r/2). \tag{14}$$

Since $0 \le r \le 1$, and S lies inside the triangle ABC

$$\pi/2 \le \alpha \le MIN(2\pi/3,\ \pi - \angle ACB/2). \tag{15}$$

This result is the most distinguishing feature of MSDCT. In contrast to MSDCT, the Euclidean SMT has exactly one value for all the three internal angles.

Note that when $r = 1$, $\alpha = 2\pi/3$; and hence all the three internal angles at S are equal to $2\pi/3$. This corresponds to the Cavalieri's (1647) famous result for the Torricelli points[2] in which he proved that each of the internal angles at a Torricelli point equals $2\pi/3$.

We now find expression for the Steiner point S:
Consider the triangle ABC.

$$\angle BAC = \arccos\left(\frac{b^2 + c^2 - a^2}{2bc}\right) = \theta_A \text{ (say)}, \tag{16}$$

where $a = BC$, $b = AC$, and $c = AB$. Let $\angle SCA = \gamma$, $\angle SBA = \delta$, and $\angle CAS = \beta$. Then,

$$\begin{aligned} \angle SAB + \delta &= 2\alpha - \pi \\ \beta + \gamma &= \pi - \alpha \\ \Rightarrow \delta &= \alpha - \gamma - \theta_A \end{aligned} \tag{17}$$

From $\triangle ASB$ and $\triangle ASC$, using Sine rule:

$$\begin{aligned} c/\sin 2\psi &= AS/\sin\delta \\ b/\sin\alpha &= AS/\sin\gamma \\ \text{or, } \sin\delta &= (2b/c)\sin\gamma\ \cos\psi \\ \Rightarrow \sin(\alpha - \gamma - \theta_A) &= (2b/c)\sin\gamma\ \cos\psi \end{aligned} \tag{18}$$

α and γ together locate S. A fully-worked solution in the Cartesian coordinates can be found in the appendix. In our service, described in Section 4, the optimal coordinates of the caches are determined using these expressions, given the locations and refresh rates of its neighbors. In practice, the sensor node nearest to the optimal location becomes the cache.

3.4 Maximum Degree of an MSDCT Node

In this section, we shall show that the maximum degree of a Minimum Steiner Data Caching Tree (MSDCT) node is equal to 3, that is, an optimal SDCT is binary.

MAXIMUM DEGREE THEOREM. *The maximum degree of a Minimum Steiner Data Caching Tree is 3.*

PROOF. Clearly, the degree of a Steiner node in SDCT is a number greater than 2. Let us consider the case of a node of degree 4. In the following treatment, we shall show that such a node can be split into two nodes, each of degree 3 and lower cost. First, we note that at least one of the angles formed at the node is acute.[3] As shown in Figure 6, suppose that the nodes A and B form an acute angle at S. Without loss of generality, let $R_A \geq R_B$, and $R_C \geq R_D$. We shall use the notation a for AS, a' for AS', b for BS and so on. We now choose another

[2]A Torricelli point is same as an Steiner point when the number of fixed nodes is 3.
[3]We do not consider the rectilinear Steiner trees.

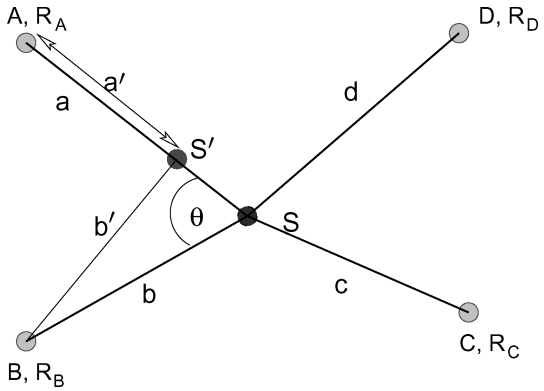


Fig. 6. $\{A, B, S', S, C, D\}$ has lower cost than $\{A, B, S, C, D\}$.

point S' along SA. For simplicity of the proof, we choose $SS' << AS$, which allows us to expand a' and b' in terms of SS'.

$$\begin{aligned} c(S) &= aR_A + bR_B + cR_C + dR_D \\ c(S', S) &= a'R_A + b'R_B + SS'R_{SS'} + cR_C + dR_D \\ \text{or, } c(S', S) &= aR_A - SS'R_A + bR_B - SS'R_B\cos\theta \\ &\quad + O(SS'^2) + SS'R_{SS'} + cR_C + dR_D \\ \text{or, } c(S', S) - c(S) &= SS' * [-R_A - R_B\cos\theta + R_{SS'} + O(SS')]. \end{aligned} \tag{19}$$

$R_{SS'} = R_A$ or R_C depending on whether S feeds S' or vice versa. Thus we have two cases:

Case $(i) : R_{SS'} = R_A$

$$c(S', S) - c(S) = SS' * [-R_A - R_B\cos\theta + R_A + O(SS')]$$

$$\Rightarrow \quad c(S', S) - c(S) = SS' * [-R_B\cos\theta + O(SS')] \tag{20}$$

since $0 \le \theta < \pi/2, \quad -R_B\cos\theta < 0.$

Hence, $\mathbf{c}(\mathbf{S}', \mathbf{S}) < \mathbf{c}(\mathbf{S})$. (21)

Case $(ii) : R_{SS'} = R_C$

Since data is now flowing from S' to S,

$R_C \le R_A$ (Section 3.1)

$$c(S', S) - c(S) = SS' * [-R_A - R_A\cos\theta + R_C + O(SS')]$$

$$\Rightarrow \quad c(S', S) - c(S) \le SS' * [-R_A\cos\theta + O(SS')] \tag{22}$$

once again, since $0 \le \theta < \pi/2, \quad -R_A\cos\theta < 0.$

Hence, $\mathbf{c}(\mathbf{S}', \mathbf{S}) < \mathbf{c}(\mathbf{S})$. (23)

Equations (21) and (23) imply that a Steiner node of degree four can always be reduced to a pair of nodes of degree three and a smaller cost tree. Using similar reasoning, it can be easily shown that a node of degree n, where $n > 4$, can be reduced to a node of degree 3 and another node of degree $n - 1$, and smaller cost. Thus, from the principle of induction, the maximum degree of an MSDCT node is 3 (i.e., the minimum Steiner Data Caching Tree is binary).

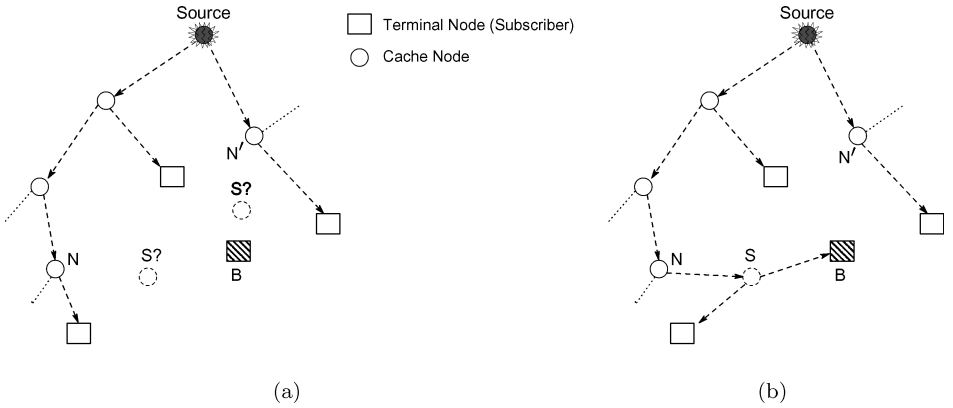


Fig. 7. (a) B requests data from the source. N' is nearer to B than N, but N' has a much lower refresh rate. "S?'s" indicate two possible choices. Edges of the multicast tree are drawn with dashed lines to represent that multiple intermediate nodes route the packets from one node of the tree to another. (b) The multicast tree after B is attached to it. The new cache S serves B and one of the children of N.

Therefore, an optimal multicast tree for n_B subscribers consists of exactly $n_B - 2$ caches. □

We shall use the refresh rate rules, the expression for Steiner points, and the Maximum Degree Theorem—all from this section— to present a heuristic for dynamically constructing and managing a near-optimal SDCT next.

4. DYNAMIC CACHE PLACEMENT HEURISTIC

This section describes a distributed cache placement heuristic that constructs and maintains near-optimal SDCT dynamically. We use the refresh rate rules described in Section 3.1 to determine refresh rates at the edges of the tree, and the expressions derived in Section 3.3.1 to compute the Steiner points. We shall construct *binary* SDCTs in accordance with its optimality property (Section 3.4).

Consider that a new subscriber B (Figure 7(a)) requests data from the source. We want to determine the location of a node S that can be used as a dedicated cache to serve B. Since the new cache will receive data from the existing multicast tree T, selection of the node of T from which the new cache S receives data is influenced by the requested refresh rate and the location of B. Attaching a high refresh rate cache can result in increase of traffic on multiple edges of a branch of T. To illustrate the point, suppose that the node to which S is attached is N, and that S needs to be refreshed at a (partially or fully satisfiable) rate higher than that of N. Then, attaching S to N results in increase of traffic on the branch of T from N upwards to the root, up-to the first cache that is already being refreshed at a rate equal to or higher than that needed by S. Therefore, the best N is not necessarily geographically the nearest. However, the cost does depend on the length of edges as well. Thus, the new cost of T = cost of attaching B to T + increase in the cost of *existing* T. A good heuristic must address both factors.

```
SEND-JOIN-REPLY(subs-node, subs-rr, best-node, min-cost, rr-cost)
    ▷ The pseudo-code is explained in sections 4 and 4.1.
    ▷ subs-node is the subscriber that requests data from the source
    ▷ subs-rr is the refresh rate of the subscriber. All "rr"'s mean refresh rates.
    ▷ best-node is the least cost option if the subscriber were
    to connect to this node of the multicast tree directly
    ▷ min-cost is the cost of connecting the subscriber node to the best-node
    ▷ rr-cost is the part of the cost that comes from difference in refresh rate requirement
 1  if this node is the source node
 2     then min-cost ← ∞
 3          rr-cost ← NIL
 4          message-count ← 1
 5          subs-rr ← (source-rr > subs-rr)? subs-rr : source-rr
 6  child-is-cache ← FALSE
 7  connect-cost = DISTANCE(this-node, subs-node) * MAX(subs-rr, source-rr)
 8  rr-cost += DISTANCE(this-node, parent-node) * MAX(subs-rr − this-cache-rr, 0)
 9  if connect-cost + rr-cost ≤ min-cost
10     then min-cost ← connect-cost + rr-cost
11          best-node ← this-node
12  if both children of this node are caches
13     then message-count ++              ▷ One more join reply message to the subscriber
14  if the left child of this node is a cache
15     then child-is-cache ← TRUE
16          forward the join request with the updated cost and the node information to the left child
            ▷ Only the terminal cache sends the accumulated information
            of the branch to the requesting subscriber
17  if the right child of this node is a cache
18     then child-is-cache ← TRUE
19          forward the join request with updated cost and the node information to the right child
20  if child-is-cache == FALSE            ▷ The last non-subscriber node on this branch
21     then Send the message-count, adjusted subs-rr, min-cost, and best-node
            information to the requesting subscriber
```

Fig. 8. Pseudo-code for processing join() requests.

In the following, we describe a heuristic to place caches in a wireless sensor network that addresses these issues. The heuristic first finds a cache N that offers the smallest increase in the cost of the multicast tree T, as if the requesting subscriber B were to be attached to N directly. A new cache S is then created that serves the requesting node, B, and one of the children of N, the cache to which S attaches itself, thus preserving the binary structure of the tree (Figure 7(b)). Each node on the multicast tree rooted at the publisher node maintains location information of its parent, as well as locations and refresh rates of each of its two children. These location tables route data from the source to the subscribers.

4.1 Joining the Multicast Tree

A subscriber joins the multicast tree by sending a **join()** message to the the source. The message may be either sent explicitly to the source node if known, or to the region of interest where the elected leader eventually gets the message. The message contains the location of the subscriber and its desired data refresh rate. Figures 8 and 9 show the pseudo-codes of two routines, SEND-JOIN-REPLY and PROCESS-JOIN-REPLIES, that send and process messages as a result of join requests. The following description of the

```
PROCESS-JOIN-REPLIES(message-count, subs-rr, best-node, min-cost)
    ▷ The pseudo-code is explained in section 4.1.
    INIT:
 1  max-messages-to-expect ← 1
 2  curr-message-count ← 0
 3  curr-min-cost ← ∞
    ▷ END OF INIT
 4  while max-messages-to-expect ≠ curr-message-count
 5        do curr-message-count++
 6           max-messages-to-expect ← MAX(max-messages-to-expect, message-count)
 7           Adjust the requested refresh rate to subs-rr
 8           if curr-min-cost > min-cost
 9              then curr-min-cost ← min-cost
10                   curr-best-node ← best-node
    ▷ FIND-STEINER-POINT locates the Steiner point based on the results of section 3.3.1
    ▷ A fully-worked calculation in Cartesian coordinates can be found in the appendix A
11  S1 ← FIND-STEINER-POINT(this-subs-node, curr-best-node, curr-best-node.left-child)
12  S2 ← FIND-STEINER-POINT(this-subs-node, curr-best-node, curr-best-node.right-child)
    ▷ Out of S1 and S2, create cache at the location that has smaller local cost.
13  new-cache ← MIN-COST-NODE(S1, S2)
14  create cache at new-cache and inform the neighbors about the new location
15  mark{new-cache.parent, new-cache}
16  while change in the cost of the multicast tree is not within specified tolerance
17        do for all nodes v that are caches and are marked
18           v' ← FIND-STEINER-POINT(v.parent, v.left-child, v.right-child)
19           if v ≠ v'
20              then mark{v'.parent, v', v'.left-child, v'.right-child}
21                   migrate the cache at v to v', and inform the neighbors about the new location
22                   unmark{v}
```

Fig. 9. Pseudo-code for processing join() replies.

heuristic contains references to the lines of the pseudo-code marked by curly braces.

{**8.1–8.11**} Upon receipt of a join() message, the root computes the cost of attaching the new subscriber to itself directly. If none or one of its children are caches, then it sends its node address, cost, and the maximum possible refresh rate to the subscriber. Else, it forwards the message to its children (or, child) *caches*. They compute the increase in cost along the branch connecting them to their parent, and the cost of connecting the subscriber to themselves directly. If this new total cost is larger than the cost received in the parent's message, then the children update cost increase due to refresh rate, and forward the parent's message to their children; otherwise, if the new total cost is smaller, they send their node address, new total cost and the maximum possible refresh rate to their children. {**8.12–8.21**} If both children of a cache are subscribers, then the process of join message forwarding terminates, and the final message from that branch is sent to the subscriber. {**9.4–9.10**} Upon receipt of all join reply messages, the subscriber picks the message that contains the least cost, and {**9.11–9.14**} computes the location of the Steiner point where a new cache will be created. If no node is present at that exact location, which is often the case, the nearest free node is selected.

{**9.15–9.22**} These steps keep the nodes of the tree locally optimal. For example, in Figure 7(b), after addition of S to the tree, the Steiner coordinates

Table I. The cost of a multicast tree as neighboring caches within a given radius are shifted to new locally optimal locations upon joining of a new subscriber

Radius	Tree A	Tree B	Tree C
0	98.4138	74.35	70.8751
1	94.7760	74.1424	70.5885
$\geq$2	94.7760	74.1424	70.4812

The radius within which the caches are shifted (shown in the column 1) is in terms of the number of edges from the newly created cache.

for N based on its new neighbor and the other two existing neighbors might be different than its current value. If the network is dense enough, the nearest sensor node of the new location for N might actually be a different node. One can easily see that migrating N to this new node will yield a tree of smaller cost. As shown in the pseudo-code, these steps are repeated for the neighboring nodes until the nodes get shifted to their locally optimal location. Since the heuristic constructs a *near*-optimal tree, these steps keep the tree's departure from optimality over time under control. Table I shows the savings in the cost of some trees (for the same setting as described in Section 5) as a result of these migrations.

As the concluding step of the join procedure, the subscriber sends messages to the new cache, the old cache, and as applicable—to all other affected nodes about any changes in the location pointers and refresh rates of their parents and children.

{**8.12–8.13, 9.6**} To determine the number of messages that the subscriber should expect to receive, a counter is included in the messages that get propagated from the source in response to the join request. This counter is incremented each time a message reaches a node whose both children are caches. The subscriber keeps track of the maximum of the expected number of responses reported. When the number of replies equals the maximum expected, it executes the process of computing the Steiner point, cache creation, and tree updating.

4.2 Leaving the Multicast Tree

When a subscriber leaves a multicast tree, unless the tree's binary structure is restored, the tree is left with one cache that has only one child. A subscriber leaves the multicast tree by sending a **leave()** message to its parent. The parent then executes the process of computing the new Steiner point for its other child, its parent, and its sibling. It then executes the process of cache creation, restoration of local optimality, and tree updating.

4.3 Change in Refresh Rate

Since the locations of the nodes of the data caching tree depend on the data refresh rates on the branches, the tree needs to be restructured if the refresh rate requirements of a subscriber change. When a subscriber wants to change

its refresh rate, it first sends a leave() message to its parent, and then it sends a join() message to the source.

5. EVALUATION

This section consists of two parts. In the first part, Section 5.1, we evaluate the performance of our SDCT heuristic (Section 4) with respect to MSDCT obtained by exhaustive search. In the second part, Section 5.2–Section 5.6, we evaluate our data caching service using the GloMoSim [Gerla et al. 1999] network simulator. We compared the performance of SDCTs constructed using the SDCT heuristic against unicast, in which the publisher sends updates to the subscribers directly without any caching along the route. Since unicast is the simplest method to disseminate data asynchronously, any other protocol must outperform unicast to merit consideration. We also compared SDCTs against binary greedy data caching trees, in which caches are created greedily instead of at the Steiner points. This greedy protocol is similar to the SDCT heuristic except for the steps {**9.11–9.12**} of the PROCESS-JOIN-REPLIES. Instead of computing the Steiner point, the greedy protocol locates a node that lies one hop away from the best node towards the subscriber on the line joining the two. If the two are closer than one hop, a neighbor of the best node is selected. We used three routing protocols for evaluation—Ad hoc On Demand Distance Vector (AODV) [Perkins and Royer 1999], Greedy Perimeter Stateless Routing (GPSR) [Karp and Kung 2000], and Geographic Forwarding (GF) [Karp and Kung 2000]. GF computes a neighbor table, and updates it periodically. GF forwards the packets to nodes that are geographically nearest to their destination. GPSR improves upon GF by adding capability to route around voids in a network. We used the 802.11 MAC protocol throughout our simulations. The size of the simulated network was 5000 m $\times$ 5000 m, and it had 784 nodes. The nodes were distributed uniformly in the network.

As noted earlier, we carried out the experiments using the GPSR routing algorithm in addition to GF and AODV. In those experiments when the network contained no voids, the plots for GPSR almost overlapped those for GF. Hence, for the sake of clarity of presentation, we have omitted those GPSR plots. Since GF and GPSR send beaconing packets to neighbor nodes periodically to determine their alive neighbors, the number of messages sent was non-zero even when there were no subscribers. We used the following data for MICA-II motes published by its manufacturer Crossbow [Crossbow Technology, Inc. 2004] to calculate the power consumption by the radio unit:

—Current drawn while transmitting: 16 mA
—Current drawn while receiving: 8 mA
—Bandwidth: 38,400 baud
—Packet size: 56 byte

We measured the number of packets at the radio layer, and used the average number of messages sent as the main cost metric. Since, in most cases, variations in the proportionality constant between the number of packets received by the nodes of the network and the number of packets sent was small, and

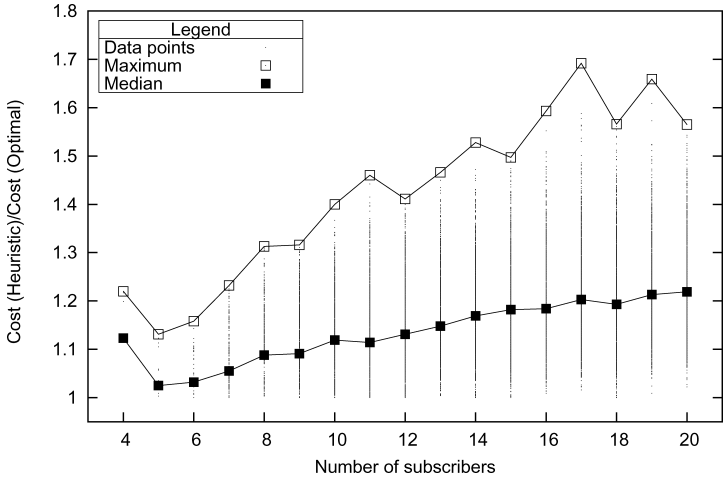
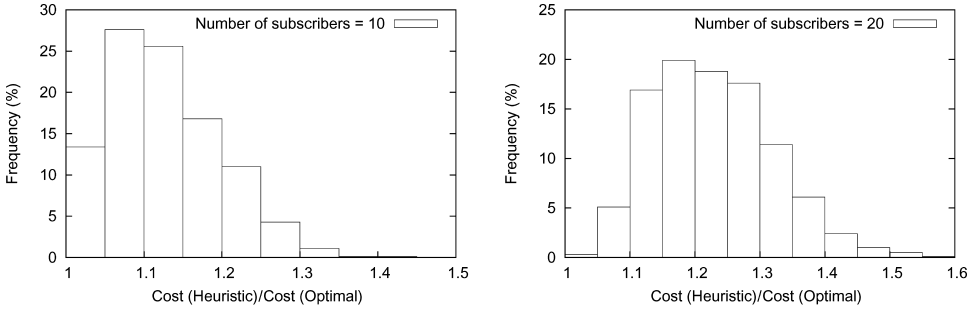


Fig. 10. Performance ratio of the SDCT heuristic vs. number of subscribers.



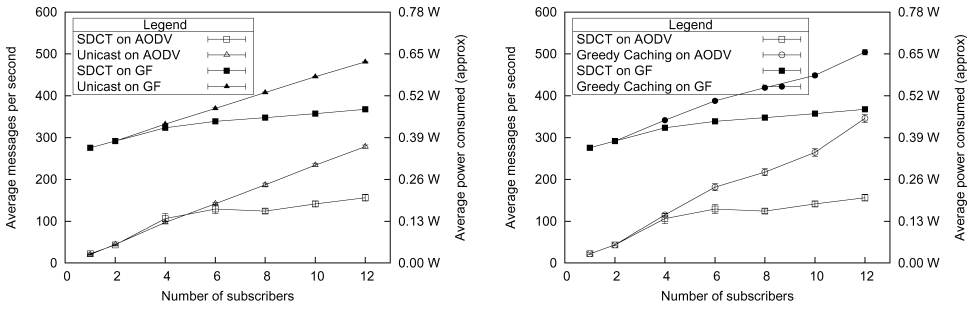
(a) 10 Subscribers (b) 20 Subscribers

Fig. 11. Histograms of performance ratio of the SDCT heuristic.

we also report approximate values of average power consumption on the mirror Y-axis. Bhattacharya et al. [2003] evaluate a heuristic very similar to the near-optimal SDCT heuristic presented in this article on MICA-II motes.

5.1 Performance Ratio of the SDCT Heuristic

In this section, we compare the cost of SDCT produced using the heuristic presented earlier with the cost of minimum SDCT (or MSDCT). It can be easily seen from the pseudo-code of the heuristic that the topology of the SDCT that it constructs depends on the order of arrival and departure of the subscribers. For a set of n subscribers, the total number of arrival orderings is equal to $n!$. Figure 10 shows the performance ratio (defined as the ratio of the cost of the tree constructed by the heuristic to the cost of a minimum tree) of the SDCT heuristic as a function of the number of the subscribers. All subscribers were located within a narrow circular band from the source. We collected data for a maximum of 2000 orderings for each number of subscribers. Thus, the maximum points do not represent the absolute worst cases. However, we plotted the distribution of data points for the cases where the numbers of subscribers



(a) SDCT vs. unicast (b) SDCT vs. greedy protocol

Fig. 12. Average number of messages sent per unit time vs. number of subscribers.

were equal to 10 and 20 respectively (Figure 11). As the histograms show, large performance ratios are distributed exponentially. Thus, although it is possible that the cost of a tree generated by the heuristic can have a higher performance ratio than the maximum shown in Figure 10, the probability of occurrence of such cases is extremely small. The median points of the Figure 10, and the frequency distribution of the histograms assure that in most cases, the cost of the tree generated by the heuristic remains within 2% of the minimum cost *per subscriber*.

5.2 Number of Subscribers

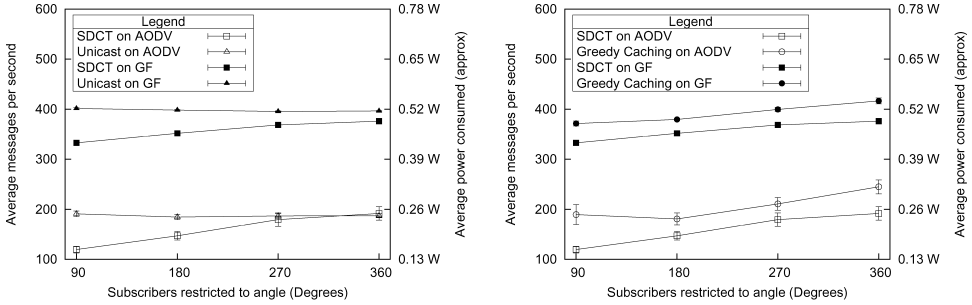
This first simulation experiment studies the performance of the middleware as a function of the number of subscribers. The subscribers were located at approximately the same distance from the source, and were spread out in a region large enough (a region subtending 180° at the source) to not cause excessive collisions. Figure 12 shows the average number of messages sent by the nodes per second as the number of subscribers varied. The error bars in this and all subsequent figures represent 95% confidence interval for the metric on the main Y-axis.

The SDCT and greedy heuristics create a cache when the number of subscribers exceeds 2. Therefore, the plots overlap each other from 1 to 2 subscribers. The gap between AODV and GF curves is in-part due to the periodic beaconings sent out by the GF routing protocol. The (extrapolated) gap at 0 subscribers is purely due to the GF beaconings.

As the plots suggest, the benefit of data caching is small when the number of subscribers is small, but the cost saving gets more and more pronounced as the number of subscribers increases. Compared to unicast and greedy caching trees, SDCTs saved about 50% of the cost of data dissemination when the number of subscribers was 10. The graphs suggest further increase in the energy saving as the number of subscribers increases.

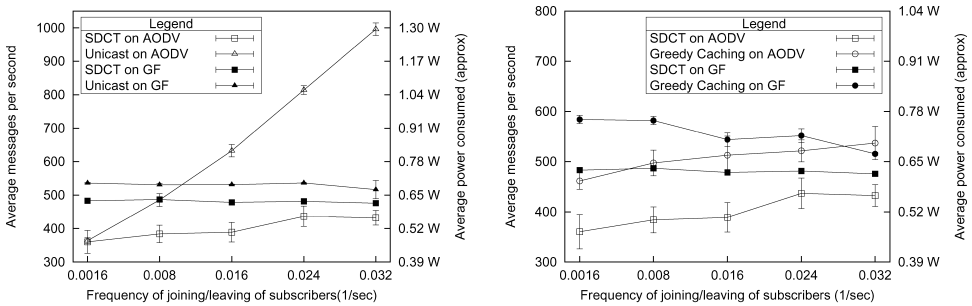
5.3 Spatial Distribution of the Subscribers

Spatial distribution of traffic in the wireless sensor network affects energy consumption. Traffic spread over larger area means larger SDCT, and hence



(a) SDCT vs. unicast (b) SDCT vs. greedy protocol

Fig. 13. Average number of messages sent per unit time vs. clustering of the subscribers.



(a) SDCT vs. unicast (b) SDCT vs. greedy protocol

Fig. 14. Average number of messages sent per unit time vs. frequency of joining and leaving of the subscribers.

increased cost. In this experiment, we simulated the network with varying degree of spatial distribution of the subscribers. The cost of the multicast tree as a function of the spatial distribution of the subscribers is shown in Figure 13. In the figure, x° clustering means that all the subscribers were located in a sector that subtended an angle x° at the source node. Since the number of subscribers was small (10), at higher clustering angles, the multicast tree had similar cost as the unicast tree. Hence, we see the curves for unicast and data caching trees converge as the bases get spread over a larger area.

5.4 Effect of Subscriber Dynamics

Each time a subscriber joins or leaves the network, the SDCT needs to be restructured. In this experiment, we evaluate the overhead of restructuring. Figure 14 shows the effect of frequency of tree restructuring on the average number of messages sent by the nodes per second. Leaving of a subscriber was followed by joining of another in the same neighborhood to keep the cost of the tree unchanged. The curves show that the restructuring overhead of SDCTs is small, and that SDCTs perform better than greedy caching trees. The data for GF is almost immune to the dynamics of the multicast group because it uses static routing tables.

Table II. Partitioning of the Network

Clustering	90°				180°			
Routing	AODV		GF		AODV		GF	
SDCT	Yes	No	Yes	No	Yes	No	Yes	No
T_{SR}	3384 ± 744	633 ± 9	2586 ± 39	613 ± 9	3269 ± 618	644 ± 12	2420 ± 153	620 ± 9
T_P	> 5000	1045 ± 24	1603 ± 138	883 ± 60	> 5000	1023 ± 33	1833 ± 129	987 ± 27

T_{SR} is the time at which the source node died. T_P is the time when the network was partitioned. The number '> 5000' indicates that the network remained unpartitioned till the end of simulation (that ran for 5000s). The errors represent 6σ ($\approx 99.7\%$) confidence interval.

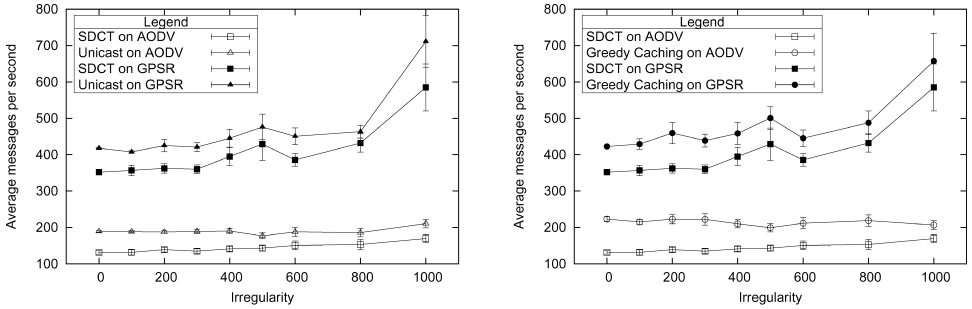
5.5 Partitioning of the Network

Table II shows the performance comparison of SDCT and unicast with respect to the life-time of source nodes and partition-free network. The source nodes were among the first ones to run out of battery. As the source nodes died, we kept replenishing their energy reserve to allow the simulations to continue. In a real deployment, this corresponds to a new leader getting elected at the event location. As expected, the simulations that used GF for routing showed that it takes longer to partition when the traffic is spread over a larger region. However, the partitioning time for unicast was independent of clustering, since in the case of unicast, almost all partitionings arose from the depletion of the energy reserve of the nodes neighboring to the source.

5.6 Irregularities in Node Placement

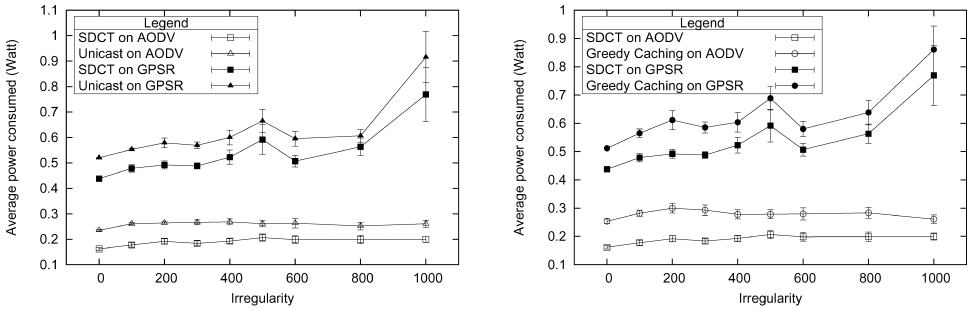
We presented and evaluated the optimality properties of the SDCT heuristic in irregularity free, random, uniformly distributed networks. In reality, as time passes, nodes start to "die" or malfunction. As a result, the distribution of nodes in the network becomes more and more nonuniform. In the regions of sparse or nonuniform node distribution, the probability of finding a node for caching data within a small neighborhood of the Steiner point becomes smaller with increase in the irregularities. Figure 16 shows the performance of our SDCT heuristic in the presence of nonuniform node placement in the network. The nodes were given a random deviation around an ideal grid position in both $\pm x$ and $\pm y$ directions. The numbers on the X-axes of the figures (15, 16) represent the maximum deviation about the ideal position in percentage, for example 100% means that the node would be placed randomly in an area formed by four neighboring squares of the grid centered at the ideal location of the node.

At smaller irregularities, the (routing) cost of the tree edges vary more or less similarly for all the three protocols with the irregularities introduced. Since these costs dominate over change in cost due to deviations from ideal locations in the placement of internal nodes of SDCT or the greedy tree, the difference in the cost remains more-or-less the same at smaller irregularities. As the amount of irregularity increases, performance of all the three protocols become indistinguishable. The plots in Figures 15 and 16 show that SDCT is quite robust up-to a very high level of irregularity introduced in the network. Its performance was consistently better than the unicast and greedy multicast trees in irregular networks.



(a) SDCT vs. unicast (b) SDCT vs. greedy protocol

Fig. 15. Average number of messages sent per unit time vs. irregularity in the placement of nodes. The numbers on the x-axis show the maximum deviation from perfect grid in percentage of grid size.



(a) SDCT vs. unicast (b) SDCT vs. greedy protocol

Fig. 16. Power consumption vs. irregularity in the placement of nodes.

The following list summarizes the factors that influence the energy efficiency of SDCT (as compared to unicast):

—*Number of Subscribers*: Larger ⇒ More effective
—*Data Refresh Rates*: Larger ⇒ More effective
—*Distance between Source and Subscribers*: Few hops only ⇒ Not effective
—*Density of Subscribers*: Larger ⇒ More effective

6. STEINER MINIMAL TREES AND MSDCT

We proved in the Section 3 that the maximum number of degrees of an MSDCT node is three. This is also true for SMT. We also showed that at least two of the internal angles formed at an MSDCT node are equal, where as all the three internal angles of an SMT node are equal, and hence are *exactly* equal to 120°. On the other hand, the internal angles of an MSDCT node have a *range*, given by Eq. (15). Therefore, MSDCT is a more general case of SMT. MSDCT reduces to Steiner Minimal Tree (SMT) when the data refresh rates are equal on all edges of the tree. As mentioned before, the problem of finding the SMT has been shown to be NP-hard. There has been much activity in the direction

of finding approximate polynomial solutions to many variants of the Steiner tree problem [Charikar et al. 1998; Robins and Zelikovsky 2000; Berman and Ramaiyer 1994]. The problem of finding Steiner points for data caching in the wireless sensor networks needs a different approach since the subscribers are not known a priori and global knowledge of the locations of all nodes of the network is also not available (at some centralized location). The locations and timings of requests of the subscribers depend on events, and can't be assumed to be known a priori. In general, we want a distributed solution that manages the tree dynamically.

However, in some applications, like monitoring temperature inside a building or a smart nursing home, where networks do not change much over time, it is possible to put the nodes at predetermined locations. If the refresh rates of the subscribers are more or less the same, one can use some of the polynomial time SMT heuristics, for example the K-SPH heuristic [Bauer and Varma 1996], to precompute the MSDCTs before starting to disseminate data.

7. RELATED WORK

This article presents and proves the properties of a minimum Steiner tree for data caching, namely binary structure and symmetrical internal angles, which to the best of our knowledge, are new results. Bhattacharya et al. [2003] present a heuristic for setting up a multicast tree in WSN. The heuristic presented in Bhattacharya et al. [2003] assumes the cost metric of an edge to be $refresh\ Rate * distance^2$, and does not try to approximate an optimal Steiner Tree.

Intanagonwiwat et al. [2000] proposed Directed Diffusion, a data-centric approach for communication. In Directed Diffusion, intermediate nodes cache data. Stann and Heidemann [2003] proposes a transport layer for Directed Diffusion to provide guaranteed delivery and reassembly. Shenker et al. [2002] and Ratnasamy et al. [2002] propose data-centric storage in WSN. Data is stored as (key, value) pair, and is replicated to avoid overloading. The keys are mapped to geographic locations for caching. Both of these rely on GPSR [Karp and Kung 2000] for routing to a geographic coordinate or its closest neighbor. These efforts do not focus on minimum-energy multicast.

Kim et al. [2003] present SAFE, a protocol for data dissemination in WSN. The SAFE protocol caches the data on *all* intermediate nodes en-route to the destination. Ye et al. [2002] propose data dissemination in a two-tier hierarchy supporting mobility of the sinks. A grid structure is constructed proactively for forwarding data. The subscribers flood their local grid cell with their query. Nodes forming the grid forward the query upstream until data is found, which is then forwarded downstream to the subscriber. Finding a good grid size is crucial to good performance of this scheme. Regardless of size, routing on a grid tends to increase route length and hence power consumption.

Cheng et al. [2003] build a strongly connected minimum energy topology for a WSN by adjusting the transmit power levels of nodes. The metric optimized is the sum of transmit powers of individual nodes. The topology generated by the heuristics presented in the paper can be used for energy efficient routing.

Bauer and Varma [1996] present two heuristics to setup a multicast tree in a (virtual) circuit switched network for a given set of multicast members. The K-SPH heuristic builds the tree by first creating forests, one for each member, and then by merging these forests pair-wise until a single tree is obtained. In the other heuristic, the tree starts to grow at one multicast member, and terminates when all members are included. Unlike our heuristic, that manages the multicast tree *dynamically*, these heuristics *precompute* the multicast tree before starting data dissemination.

8. CONCLUSION AND FUTURE WORK

Motivated by the importance of the need to save the energy spent on communication in wireless sensor networks, we presented optimality properties of Steiner data caching trees, and an energy-conserving application-layer service for data caching and asynchronous multicast. We hope that the theoretical results presented in the article will be useful in future research on data dissemination in wireless sensor networks. The simulation results that we presented in this article verify the importance of data caching and support our case.

This article addressed the scenario where multiple subscribers were receiving data from one source. In another very common WSN application scenario, one subscriber receives data from multiple sources. It is desirable to extend the results and the MSDCT heuristic to address the latter. These two studies can be combined to construct arbitrary energy-conserving data dissemination and retrieval graphs. Addressing the issues of mobility and network irregularities is another important extension of the SDCT heuristic to be carried out in future work.

APPENDIX

A. STEINER POINT IN CARTESIAN COORDINATES

We solve the Eq. (14) and (18) of Section 3.3.1 to obtain expressions for the coordinates of the Steiner point in the Cartesian coordinate system for a set of three given nodes. We obtain the solution by obtaining expressions for the straight lines CS and AS (Figure 17), and then their point of intersection.

$$\theta_A = \arccos\left(\frac{b^2+c^2-a^2}{2bc}\right). \tag{24}$$

From Eq. (14):

$$\psi = \arccos(r/2) \tag{25}$$

$$\alpha = \pi - \psi.$$

From Eq. (18):

$$\gamma = \operatorname{arccot}\left(\frac{\cos(\alpha-\theta_A)+(2b/c)\cos\psi}{\sin(\alpha-\theta_A)}\right). \tag{26}$$

Equation of line AC:

$$y - y_C = m_{AC}(x - x_C)\text{ , where}$$

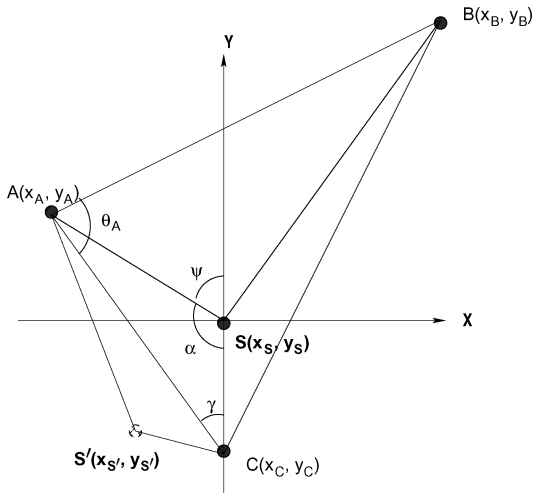


Fig. 17. Calculating the coordinates of S in the Cartesian coordinate system. Out of the two solutions S and S', S is nearer to B.

$$m_{AC} = \frac{y_A - y_C}{x_A - x_C}.$$

Equation of line CS:

$$\begin{aligned} y - y_C &= m_{CS}^{\pm}(x - x_C)\text{ , where} \\ m_{CS}^{\pm} &= \frac{m_{AC} \pm \tan\gamma}{1 \mp m_{AC}\tan\gamma}(x - x_C). \end{aligned} \tag{27}$$

Equation of line AS:

$$\begin{aligned} y - y_C &= m_{CS}^{\pm}(x - x_A)\text{ , where} \\ m_{AS}^{\pm} &= \frac{m_{AC} \pm \tan(\pi - \alpha - \gamma)}{1 \mp m_{AC}\tan(\pi - \alpha - \gamma}(x - x_A) \\ &= \frac{m_{AC} \pm \tan(\psi - \gamma)}{1 \mp m_{AC}\tan(\psi - \gamma)}(x - x_A). \end{aligned} \tag{28}$$

Let us define:

$$\begin{aligned} c_{CS}^{\pm} &= y_C - m_{CS}^{\pm}x_C \\ c_{AS}^{\pm} &= y_A - m_{AS}^{\pm}x_A. \end{aligned}$$

Since S lies at the intersection of CS and AS, we solve Eq. (27) and (28) to get (x_S, y_S). Out of the four solutions corresponding to each combination of the two pairs of m^s, only two lie in the region of interest:

$$S^{(1)}(x_S^{(1)}, y_S^{(1)}):\ \ x_S^{(1)} = \frac{c_{CS}^{+} - c_{AS}^{-}}{m_{AS}^{-} - m_{CS}^{+}},\ y_S^{(1)} = \frac{c_{CS}^{+}m_{AS}^{-} - c_{AS}^{-}m_{CS}^{+}}{m_{AS}^{-} - m_{CS}^{+}} \tag{29}$$

$$S^{(2)}(x_S^{(2)}, y_S^{(2)}):\ \ x_S^{(2)} = \frac{c_{CS}^{-} - c_{AS}^{+}}{m_{AS}^{+} - m_{CS}^{-}},\ y_S^{(2)} = \frac{c_{CS}^{-}m_{AS}^{+} - c_{AS}^{+}m_{CS}^{-}}{m_{AS}^{+} - m_{CS}^{-}}. \tag{30}$$

Between $S^{(1)}$ and $S^{(2)}$, one that is closer to B is S, and the other is S'.

ACKNOWLEDGMENTS

The first author had very useful discussions with Deepak Goel during the initial phase of working on this problem. A great deal of help came from Tian He, who provided his and Brian Blum's implementations of GPSR and GF in GloMoSim. Direct and indirect help of the members of the Control Research Group of our department is gratefully acknowledged. Finally, thanks to Sagnik Bhattacharya for his original conference version that constituted a first draft for this article.

REFERENCES

BAUER, F. AND VARMA, A. 1996. Distributed algorithms for multicast path setup in data networks. *IEEE/ACM Trans. Netw. 4,* 2, 181–191.

BERMAN, P. AND RAMAIYER, V. 1994. Improved approximations for the steiner tree problem. In *Selected Papers from the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms*. Academic Press, New York, 381–408.

BHATTACHARYA, S., KIM, H., PRABH, K. S., AND ABDELZAHER, T. F. 2003. Energy-conserving data placement and asynchronous multicast in wireless sensor networks. In *Proceedings of the 1st International Conference on Mobile Systems, Applications, and Services (MobiSys 2003)*. USENIX, New York.

CHARIKAR, M., CHEKURI, C., YAT CHEUNG, T., DAI, Z., GOEL, A., GUHA, S., AND LI, M. 1998. Approximation algorithms for directed steiner problems. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, Philadelphia, PA, 192–200.

CHENG, X., NARAHARI, B., SIMHA, R., CHENG, M. X., AND LIU, D. 2003. Strong minimum energy topology in wireless sensor networks: Np-completeness and heuristics. *IEEE Trans. Mobile Comput. 2*, 3, 248–256.

CIESLIK, D. 1998. *Steiner Minimal Trees*. Kluwer Academic Publishers, Dordrecht, The Netherlands.

CROSSBOW TECHNOLOGY, INC. 2004. `http://www.xbow.com/`.

GAREY, M. R., GRAHAM, R. L., AND JOHSON, D. S. 1977. The complexity of computing steiner minimal trees. *SIAM J. Applied Math. 32*, 835–859.

GERLA, M., BAJAJ, L., TAKAI, M., AHUJA, R., AND BAGRODIA, R. May 1999. Glomosim: A scalable network simulation environment. Tech. Rep. 990027. Dept. Computer Science. UCLA, Los Angeles, CA.

HE, T., KRISHNAMURTHY, S., STANKOVIC, J. A., ABDELZAHER, T., LUO, L., STOLERU, R., YAN, T., GU, L., HUI, J., AND KROGH, B. 2004. Energy-efficient surveillance system using wireless sensor networks. In *Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services*. ACM, New York, 270–283.

INTANAGONWIWAT, C., GOVINDAN, R., AND ESTRIN, D. 2000. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*. ACM, New York, 56–67.

INTEL. 2004. `http://www.intel.com/research/exploratory/heterogeneous.htm`.

KARP, B. AND KUNG, H. T. 2000. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*. ACM, New York, 243–254.

KIM, S., SON, S. H., STANKOVIC, J. A., LI, S., AND CHOI, Y. 2003. SAFE: A data dissemination protocol for periodic updates in sensor networks. In *Proceedings of the 23rd International Conference on Distributed Computing Systems*. IEEE Computer Society, Press, Los Alamitos, CA, 228.

MALPANI, N., WELCH, J. L., AND VAIDYA, N. 2000. Leader election algorithms for mobile ad hoc networks. In *Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*. ACM, New York, 96–103.

PERKINS, C. E. AND ROYER, E. M. 1999. Ad hoc on demand distance vector (AODV) routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*. IEEE Computer Society Press, Los Alamitos, CA, 90–100.

RATNASAMY, S., KARP, B., YIN, L., YU, F., ESTRIN, D., GOVINDAN, R., AND SHENKER, S. 2002. GHT: A geographic hash table for data-centric storage in sensornets. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*. ACM, New York.

ROBINS, G. AND ZELIKOVSKY, A. 2000. Improved steiner tree approximation in graphs. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, Philadelphia, PA, 770–779.

SHENKER, S., RATNASAMY, S., KARP, B., GOVINDAN, R., AND ESTRIN, D. 2002. Data-centric storage in sensornets. In *Proceedings of the 1st ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets 2002)*. ACM, New York.

STANN, F. AND HEIDEMANN, J. 2003. RMST: Reliable data transport in sensor networks. In *Proceedings of the 1st International Workshop on Sensor Net Protocols and Applications*. IEEE Computer Society Press, Los Alamitos, CA, 102–112.

SZEWCZYK, R., POLASTRE, J., MAINWARING, A., AND CULLER, D. 2004. Lessons from a sensor network expedition. In *Proceedings of the 1st European Workshop on Wireless Sensor Networks (EWSN'04)*. Springer-Verlag, New York.

YE, F., LUO, H., CHENG, J., LU, S., AND ZHANG, L. 2002. A two-tier data dissemination model for large-scale wireless sensor networks. In *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking*. ACM, New York, 148–159.

Received July 2004; revised November 2004 and March 2005; accepted May 2005