Ant-Based Distributed Constrained Steiner Tree Algorithm for Jointly Conserving Energy and Bounding Delay in Ad Hoc Multicast Routing

CHIEN-CHUNG SHEN and KE LI
University of Delaware
CHAIPORN JAIKAEO
Kasetsart University
and
VINAY SRIDHARA
Qualcomm

3

The minimum-energy multicast tree problem aims to construct a multicast tree rooted at the source node and spanning all the destination nodes such that the sum of transmission power at non-leaf nodes is minimized. However, aggressive power assignment at non-leaf nodes, although conserving more energy, results in multicast trees that suffer from higher hop count and jeopardizes delay-sensitive applications, signifying a clear tradeoff between energy efficiency and delay. This article formulates these issues as a *constrained Steiner tree* problem, and describes a distributed constrained Steiner tree algorithm, which jointly conserves energy and bounds delay for multicast routing in ad hoc networks. In particular, the proposed algorithm concurrently constructs a constrained Steiner tree, performs transmission power assignment at non-leaf nodes, and strives to minimize the sum of transmission power of non-leaf nodes, subject to the given maximum hop count constraint. Simulation results validate the effectiveness and reveal the characteristics of the proposed algorithm.

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Routing protocols*

A preliminary version of this article titled "Energy Conserving Multicast for MANET with Swarm Intelligence" appeared in the 2nd IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS). © IEEE 2005.
This work was supported in part by the National Science Foundation under grants CNS-0347460 and CNS-0240398.
Authors' addresses: C.-C. Shen and K. Li, Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716; email: cshen@cis.udel.edu; C. Jaikaeo, Department of Computer Engineering, Kasetsart University, Catuchak Bangkok 10900, Thailand; V. Sridhara, Qualcomm, San Diego, CA 92121.

General Terms: Algorithms, Design, Performance

Additional Key Words and Phrases: Ad hoc networks, constrained Steiner tree, multicast, swarm intelligence

ACM Reference Format:

Shen, C.-C., Li, K., Jaikaeo, C., and Sridhara, V. 2008. Ant-based distributed constrained steiner tree algorithm for jointly conserving energy and bounding delay in ad hoc multicast routing. ACM Trans. Autonom. Adapt. Syst. 3, 1, Article 3 (March 2008), 27 pages. DOI = 10.1145/1342171.1342174 http://doi.acm.org/ 10.1145/1342171.1342174

1. INTRODUCTION

Ad hoc networks are autonomously formed by nodes via multi-hop wireless communications, which are useful in situations where the deployment of any communication infrastructure is difficult or impossible. Multicast communication is an essential capability of ad hoc networks, which allows a source node to communicate with multiple destination nodes (also called multicast group member nodes, or member nodes for short). One way to facilitate multicast communications in ad hoc networks is to extract a connected subset of *forwarding nodes* (or *forwarders* for short), termed a *forwarding set*,[1] which connects the source node with all the destination nodes. Forwarders of a forwarding set are responsible for receiving and forwarding multicast packets by rebroadcasting nonduplicate multicast data packets from the source node to all the destination nodes.

In Shen and Jaikaeo [2005], we described the MANSI multicast routing protocol for mobile ad hoc networks, which employs the mechanism of *swarm intelligence* [Bonabeau et al. 1999] to form a forwarding set connecting a source node with all the destination nodes. In particular, the operations of MANSI assume that all nodes use the same fixed transmission power, and the objective is to minimize the total number of forwarders in a forwarding set.

However, as energy is seriously constrained in ad hoc networks, multicast protocols should strive to conserve this resource. One approach to achieving energy-efficient multicast in ad hoc networks is to jointly form a *Steiner tree* (ST) of forwarders connecting the source node with all the destination nodes, and perform transmission power assignment on the forwarders so that the sum of transmission power of all forwarders in a resulting Steiner tree is minimized. In addition to conserving energy, power assignment also helps reduce interference incurred by forwarding multicast traffic. Figure 1 is an illustrative example, where node A is the source, and nodes B, C, and D are destination nodes. In Figure 1(a), a forwarding set (Steiner tree) of two forwarders is formed with assigned transmission power resulting in a total transmission power of 50 mW. In contrast, Figure 1(b) shows a forwarding set (Steiner tree) of five forwarders with assigned transmission power yielding a total transmission power of 18 mW. In the extreme, the minimum-energy multicast tree problem [Wieselthier et al. 2002; Liang 2006] aims to construct a multicast tree rooted

[1]A forwarding set includes the corresponding source node, and may include destination nodes that also work as forwarders.

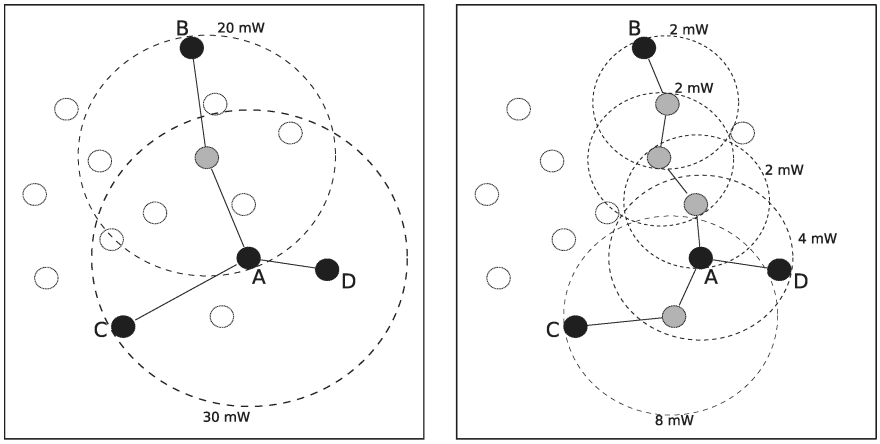


Fig. 1. The number of forwarders and the total energy consumption in an example network: (a) 2 forwarders and 50 mW vs. (b) 5 forwarders and 18 mW.

at the source node and spanning all the destination nodes, such that total transmission power at non-leaf nodes is minimized.

Although aggressive power assignment may conserve more energy, the resulting Steiner tree incurs higher a hop count, which jeopardizes delay-sensitive applications, signifying a clear tradeoff between energy efficiency and delay. Therefore, it is critical that both energy-efficient multicast connectivity and timely data delivery should be jointly considered such that the formation of a Steiner tree strives both to conserve energy as well as satisfy the given maximum hop count constraint.

In this article, we formulate these issues into a variant of the *Constrained Steiner Tree* (CST) problem, which is known to be NP-complete [Cormen et al. 2001]. We describe a distributed constrained Steiner tree algorithm termed CSTMAN, which employs the mechanisms of swarm intelligence to jointly conserve energy, and bound delay (in terms of hop count) for multicast communications in ad hoc networks. In contrast to MANSI, which strives to minimize the total number of forwarders in a forwarding set (the cost function of MANSI), CSTMAN jointly (1) constructs a constrained Steiner tree rooted at the source node and spanning all the destination nodes, (2) performs transmission power assignment on forwarders, and (3) strives to minimize the total of transmission power of forwarders (non-leaf nodes)—the cost function of CSTMAN—subject to the given maximum hop count constraint. While CSTMAN shares certain similarities with MANSI in terms of the swarm intelligence mechanisms and data structures used, modifying the cost function alone is not sufficient to jointly conserve energy and bound delay. The difficulty is caused by the interdependence among (1) the goal of minimizing the sum of transmission power of all forwarders, (2) the transmission power assignment on each forwarder, which in turns depends on how it chooses proper neighboring forwarders and their respective transmission power assignment, and (3) the hop count (delay) constraint imposed on the problem.

The remainder of the article is organized as follows. Section 2 reviews existing constrained Steiner tree algorithms for wired networks and nonconstrained Steiner tree algorithms for wireless networks, respectively, and contrasts CSTMAN to these solutions. In addition, existing solutions to the minimum-energy multicast tree problem are surveyed and contrasted. Section 3 formally defines a CST problem that jointly conserves energy and bounds delay for multicast communications in the context of wireless ad hoc networks, and describes its intrinsic differences from a CST problem applicable to multicast communications in wired networks. Section 4 presents an overview of the CSTMAN algorithm, and its implementation details are described in Section 5. Simulation results are presented in Section 6, which validate the effectiveness and reveal the characteristics of CSTMAN. Section 7 concludes the article.

2. RELATED WORK

Both centralized [Kompella et al. 1993a] and distributed [Kompella et al. 1993b; Wi and Choi 1995; Huang and Lee 2002] heuristic algorithms have been proposed to construct constrained Steiner trees for multicast routing in wired networks. Kompella et al. [1993a] described a centralized algorithm. The basic idea is first to construct a closure graph connecting source and destination nodes, which encapsulates the cost and the delay of the lowest cost delay-bounded path between any pair of nodes, and then to construct a minimum spanning tree (MST) rooted at the source subject to the delay constraint as the approximate solution. Although centralized algorithms produce good results of near-optimal tree cost, it is not suitable for wireless networks due to the high cost of maintaining consistent network states involving all the nodes. Kompella et al. [1993b] also proposed a distributed solution based on a distributed MST algorithm, where control messages are exchanged among nodes to update local information, and two heuristics incorporating cost and delay are used to select edges in the tree. In order to meet delay bounds, each node is assumed to know beforehand the minimum delay to all other nodes. In comparison, Wi and Choi [1995] devised an MST-like distributed algorithm with 20–150% less tree construction time, but 3–10% more tree cost than Kompella et al. [1993b]. Huang et al. [2002] proposed another MST-like distributed algorithm yielding less tree cost than Wi and Choi [1995], subject to the same delay constraint. In their algorithm, at each node, the least delay path and the least cost path to every other node are assumed to be known in advance and to remain unchanged. During the tree construction, a routing token is passed towards a destination member on either the least delay path or the least cost path, and the routing token exchanges information with relaying nodes in the path.

In this article, we address the issues of constructing constrained Steiner trees to balance energy conservation and delay in multicast communications for wireless ad hoc networks. For wired networks and the work previously described, the notion of cost and delay are defined based on wired links, and a node generally needs n transmissions to send the same packet to n different neighbors. In contrast, wireless communications enjoy the *broadcast advantage*, where a single wireless transmission reaches all neighbors within the

transmission range. Therefore, the notion of cost and delay should be defined based on nodes, instead.

In wireless networks, the performance metrics of cost vary depending on the desired properties of the networks. In recent years, tremendous research interest has been put on energy efficiency, where nodes can adjust their transmission power to minimize total energy consumption in ad hoc networks. Several algorithms [Wieselthier et al. 2002; Liang 2006; Cartigny et al. 2003; Pan et al. 2004] have been proposed to construct nonconstrained Steiner trees for minimum-energy broadcast/multicast in wireless networks. Wieselthier et al. [2002] first developed a centralized greedy heuristic called Broadcast Incremental Power (BIP) for constructing energy-efficient broadcast trees. The heuristic is based on Prim's algorithm and takes into account the wireless broadcast advantage. Whereas the inputs to Prim's algorithm are fixed/unchanged link cost, BIP dynamically updates the cost (energy) whenever a new node is added to the tree, so as to reflect the fact that the cost of adding new node is the *incremental power*. To obtain a multicast tree, the broadcast tree derived from BIP is pruned by eliminating all transmissions not needed for reaching the group members. Liang [2006] devised a centralized approximation algorithm for finding minimum-energy multicast trees in symmetric wireless ad hoc networks, with an improved approximation ratio of $4 \ln K$, where K is the number of group members. The idea is to reduce the problem into a fixed/unchanged node cost Steiner tree problem, and then to solve it based on Kruskal's algorithm. The reduction process requires that there should be finite adjustable power levels at each node. Cartigny et al. [2003] presented a distributed protocol for energy-efficient broadcast, where each node requires only the knowledge of its distance to all neighbors and distances among all neighbors. The protocol is based on the use of a Relative Neighborhood Graph and is shown to be experimentally comparable to BIP. For minimum-energy multicasting, Pan et al. [2004] proposed two distributed algorithms (DMIP3S and P-DIP3S) based on the idea of *incremental power with potential power saving* (IP3S). Like BIP, IP3S iteratively grows a tree of covered nodes starting from the source. Differently, IP3S may not grow the tree by adding the node with least incremental power, but would check to make possible power reductions for nodes already in the tree instead, as an effect of the corresponding expansion.

As an alternative to solutions that construct minimum-energy multicast trees, network coding has been proposed to address the minimum-energy multicast problem. For instance, Wu et al. [2005] showed that network coding could potentially attain lower energy consumption than minimum-energy-tree-based solutions. Specifically, the article formulated the minimum-energy multicast problem as a linear optimization problem using Capacity Graphs under a simplified layered model of wireless networks, where the linear optimization problem is solvable in polynomial time using a centralized computation. Lun et al. [2006] gave different formulations for finding minimum-cost subgraphs to support multicast connections over coded packet networks—both wireline and wireless. Decentralized algorithms are proposed for solving the formulated optimization problems in static multicast, where group membership does not change during a multicast session.

In contrast to CSTMAN, these schemes either are centralized, or do not address the tradeoff issue between energy conservation and delay. Furthermore, in comparison with Liang [2006], which devised centralized approximation algorithms with provable approximation guarantees, it is difficult or even infeasible to devise a provable approximation ratio for CSTMAN due to its distributed, asynchronous, and probabilistic nature. In contrast, the goal of CSTMAN is to function as a distributed control protocol that uses localized interactions without any global state information to jointly conserve energy and bound delay in ad hoc multicast routing.

Using the meta-heuristic approach of Ant Colony Optimization (ACO), Gosavi et al. [2003] and Singh et al. [2004] proposed both offline (centralized) and online (distributed) algorithms for constructing nonconstrained Steiner trees in the context of wireless sensor networks. In their online distributed algorithm, the sink must wait for forward ants from all the members to arrive before it can release backward ants, which assumes the sink knows the membership information in advance, and imposes synchronization at the sink. Although using an idea similar to swarm intelligence, in contrast to Gosavi et al. [2003] and Singh et al. [2004], CSTMAN differs in three critical aspects. First, CSTMAN is fully distributed and asynchronous without such synchronization activity nor any knowledge of membership information. Second, CSTMAN addresses the general problem of constrained Steiner tree to bound delay. Last, CSTMAN also performs power control to conserve energy.

3. PROBLEM FORMULATION

In this section, we first review one definition of the CST problem applicable to multicast communications in wired networks, termed CST-*wired*. We then define its variations for multicast communications in the context of wireless networks. CST-*wired* is formulated as follows.

> Given: A directed graph $G(V, E)$, with (1) two weighting functions, termed *cost* (c) and *delay* (d), defined on the edge set E, $c: E \rightarrow R^+$ and $d: E \rightarrow R^+$, (2) a source node $s \in V$ and a set of destination nodes $D \subseteq (V - \{s\})$, and (3) a delay constraint Δ.
> Find: A subgraph G' of G, such that (1) G' is a directed tree that is rooted at the source node s and spans all destination nodes in D, (2) the cumulative delay from the source node s to each destination node satisfies Δ: $\forall$ destination node $t \in D$, $\sum_{(v_i,v_j)\in\mathcal{E}(PATH_{s\rightarrow t})} d((v_i, v_j)) \leq \Delta$, where $PATH_{s\rightarrow t}$ is a subgraph in G' representing the routing path from the source node s to the destination node t, and function $\mathcal{E}$ returns the edge set of an input graph, and (3) the total cost $\sum_{(v_i,v_j)\in G'} c((v_i, v_j))$ is minimized.

To reflect the *unicast* nature of wired communications where a node needs n transmissions to send the same packet to n different neighbors, cost and delay weighting functions are defined on wired links.

In contrast, a single wireless transmission reaches all neighbors within the transmission range. Therefore, in the context of wireless networks, both cost

and delay weighting functions should correctly be defined on nodes, in contrast to links for wired networks, to capture the *broadcast* nature of wireless communications. We term this CST variation CST-*wireless*, which is formulated as follows.

> Given: A graph[2] $G(V, E)$, with (1) the cost and the delay weighting functions defined on the node set V, $c: V \rightarrow R^+$ and $d: V \rightarrow R^+$, (2) a source node $s \in V$ and a set of destination node $D \subseteq (V - \{s\})$, and (3) a delay constraint Δ.
> Find: A subgraph G' of G, such that (1) G' is a tree that is rooted at the source node s and spans all destination nodes in D, (2) the cumulative delay from the source node s to each destination node satisfies Δ: $\forall$ destination node $t \in D$, $\sum_{v \in (\mathcal{V}(PATH_{s \rightarrow t}) - \{t\})} d(v) \leq \Delta$, where $PATH_{s \rightarrow t}$ is a subgraph in G' representing the routing path from the source node s to the destination node t, and function $\mathcal{V}$ returns the node set of an input graph, and (3) the total cost $\sum_{v \in F} c(v)$ is minimized, where F is the subset of all *internal nodes*[3] in G', which forms the forwarding set.

In CST-*wireless*, the cost weighting function (c) maps each node to some fixed value, and different nodes may have different fixed values. For instance, the objective of a CST-*wireless* problem with a constant cost weighting function[4] is to a find a forwarding set with a minimum number of forwarders.

When considering the CST-*wireless* problem for the purpose of energy conservation, the fixed cost mapping corresponds to the scenario where the transmission power is fixed (not adjustable) for all the nodes. However, to conserve energy, nodes in an ad hoc network may reduce transmission power, which results in multicast trees with lower total energy consumption, but more forwarders and higher hop count, as depicted in Figure 1 of two multicast trees with a different number of forwarders and a different total transmission power. In this case, node cost (denoting assigned transmission power) is a variable itself, whose value needs to be determined by a solution to the problem; the solution also performs transmission power assignment in conjunction with finding a constrained Steiner tree with the minimum total transmission power.

We term this CST variation CST-*wireless-ec*, which is formulated as follows.

> Given: A graph $G(V, E)$, where E is the set of edges corresponding to pairs of nodes capable of communicating with given maximum transmission powers,[5] with (1) a delay weighting function defined on each node, $d: V \rightarrow R^+$, (2) a source node $s \in V$ and a set of destination nodes $D \subseteq (V - \{s\})$, and (3) a delay constraint Δ.

[2]Notice that, in practice, the 'graph' representation (or topology) of a wireless network is dynamically determined by factors such as sender's transmission power, receiver's SINR, etc.

[3]An internal node is any node in a tree that has child nodes, and is thus not a leaf node, e.g., the root node. Note that in our problem definition, all leaf nodes are destination nodes, but not all destination nodes are leaf nodes, and some destination nodes may act as internal nodes (or forwarders).

[4]That is, all nodes use the same, fixed transmission power.

[5]Notice that nodes may have different maximum transmission powers.

Find: A subgraph G' of G, where $\mathcal{V}(G') = F \cup D$, and a transmission power assignment $c: F \rightarrow R^+$, which is the cost weighting function defined on forwarding set F, such that (1) G' is a tree that is rooted at the source node s and spans all the destination nodes in D, (2) the cumulative delay from the source node s to each destination node satisfies Δ: $\forall$ destination node t, $\sum_{v \in (\mathcal{V}(PATH_{s \rightarrow t}) - \{t\})} d(v) \leq \Delta$, and (3) the total transmission power $\sum_{v \in F} c(v)$ is minimized.

Note that when $\Delta = \infty$, all of these CST and CST variations degenerate to the corresponding Steiner tree (ST) problems. It can be proved that both CST variations and their ST versions are NP-complete. The proposed CSTMAN algorithm aims to solve the most general CST-*wireless-ec* problem, as motivated by the discussions in Section 1, and can also be adapted for other more specific CST variation and ST problems as we have described. In that sense, CSTMAN is a superset of MANSI, as MANSI only addresses the ST-*wireless* problem with constant cost weighting function. In the following discussion, we concentrate on the CST-*wireless-ec* problem, where CSTMAN strives to minimize the total transmission power of a forwarding set, while constructing a forwarding set and adjusting forwarders' transmission power, subject to the delay constraint of hop count. Simulation results presented in Section 6 also demonstrate CSTMAN's performance of solving the ST-*wireless-ec* problem, of no delay constraint, with the presence of mobility.

4. CSTMAN OVERVIEW

To solve a CST-*wireless-ec* problem instance, CSTMAN constructs a CST by jointly assigning transmission power to nodes and forming a forwarding set with the lowest *forwarding cost* and bounded *path delays*. The forwarding cost is the sum of transmission power of all nodes in a forwarding set, and a path delay of a destination node is the hop count of a routing path from the source node to the destination node via nodes in the forwarding set. A forwarding set is originally constructed by an initialization process, and evolves over time, as depicted in Figure 2.

Forwarding set construction is initiated by the source node (also termed *core*) flooding CORE ANNOUNCE packets over the entire network. Destination nodes send back JOIN REQUEST packets, via the reverse paths, toward the core, as long as the delay constraint is not violated, such that nodes along the reverse paths are selected as forwarders to form an initial forwarding set. The initial forwarding set often consists of separate shortest paths from the core to individual destination nodes, as shown in Figure 2(a). However, forwarding set can be made more energy efficient (lower forwarding cost) by merging individual paths together and incorporating more forwarders with lower assigned transmission power subject to the constraint of path delay bound, as shown in Figure 2(b), which jointly lowers the forwarding cost and satisfies the delay constraint. This is achieved by a forwarding set evolution process as follows.

Once an initial forwarding set is formed, each destination node periodically originates FORWARD ANT control packets. These FORWARD ANT control packets

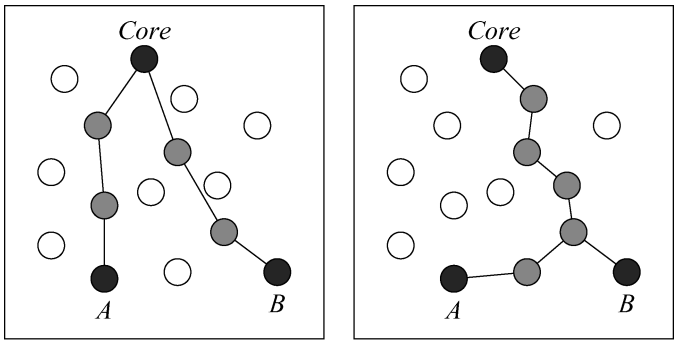


Fig. 2. Forwarding set initialization and evolution with the source (core) node and two destination nodes (shown in black): (a) an initial forwarding set formed immediately after the flooding of CORE ANNOUNCE packets; (b) an evolved forwarding set where forwarding paths are merged and adjacent forwarders are close to each other to conserve energy while satisfying a delay bound for a CST-*wireless-ec* problem instance.

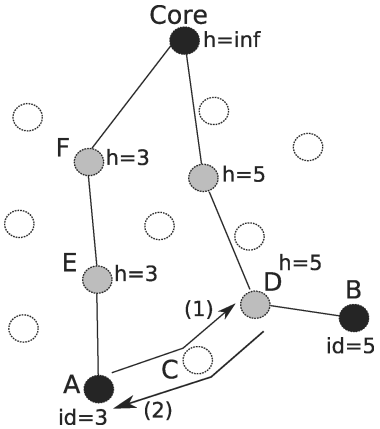


Fig. 3. Behavior of forward and backward ants: (1) a FORWARD ANT deployed from the member A choosing C as the next hop and encountering a forwarding node D, and (2) at D, the total path delay from the core to A via D satisfying the constraint, thus the FORWARD ANT becoming a BACKWARD ANT and following the reverse path back to A while depositing pheromone along the way.

explore the network, and opportunistically find a better path—a path with lower forwarding cost—to connect the destination node to the core within the given delay bound. This exploring process is illustrated in Figure 3. When a FORWARD ANT arrives at a forwarding node that belongs to another group member's path to the core, and the total path delay from the core to its originator via that forwarder[6] satisfies the delay constraint, this FORWARD ANT turns into a BACKWARD ANT and travels back to its originator via the reverse path. On its

[6]Note that the total path delay from the core s to a member, t, via a forwarding node, f, is separated into two parts—the delay from s to f and the delay from f to t, whose values are stored in *delayTab* and *joinDelayTab* tables, respectively, as defined later in this section and formally defined in Section 5.

return trip, a BACKWARD ANT increases the 'desirability' of the nodes along the way to redirect JOIN REQUEST packets to the new path, resulting in a different forwarding set with lower total cost under delay constraint. Using the swarm intelligence metaphor, the desirability of a node being chosen by another node to become a forwarder is represented by the *pheromone intensity* of the link from the requesting node to the requested node. Pheromone intensity on each link decays over time if not reinforced, which allows past history to be gradually forgotten, so that up-to-date information will be more significant, making the algorithm adaptive to network dynamics. For instance in Figure 3, after a BACKWARD ANT comes back to node A via node C with freshly deposited pheromone, node A observes that the pheromone intensity on the link to node C becomes the highest among links to all the neighboring nodes, and will switch to join the group via node C by sending a JOIN REQUEST to node C. Consequently, node C will become a forwarder. Due to the lack of reinforcement by FORWARD ANT and the pheromone evaporation (or decay), nodes E and F cease being forwarders.

To prevent any race condition where members attempt to merge into one another's forwarding path and get disconnected from the core, each forwarder (and member, if serving as a forwarder) is associated with a *height* which is identical to the highest ID of the nodes relying on it to connect to the core. The core has its height set to ∞. Figure 3 also depicts how heights are assigned to forwarders. Besides the satisfaction of the delay constraint, a FORWARD ANT can turn into a BACKWARD ANT only when it encounters a forwarding node with higher height than the ID of the member who originated the ant. That means a member is allowed to connect either to the core directly or via an existing path that belongs to another member with a higher ID, but not vice versa, to assure that the core, whose height is always the highest, will eventually be connected with all the other members.

In contrast to the topology of the initial forwarding set, which resembles a tree consisting of two shortest-path branches rooted at the source node, the topology (snapshot) of an evolving forwarding set resembles a *Steiner tree*. Notice that although the topological structure of an evolving forwarding set resembles a tree, multicast data delivery is carried out in a mesh manner where forwarders may accept non-duplicate multicast data packets arriving from non-parent nodes, and forward them further.

As a distributed algorithm, CSTMAN maintains local data structures at each node, and exchanges information among neighboring nodes through control packets, for example, HELLO, CORE ANNOUNCE, JOIN REQUEST, FORWARD ANT, and BACKWARD ANT. These control packets contain fields to accumulate cost and/or delay, as well as other routing information.

Each node maintains its own delay, and two delay-related tables: (1) *delay-from-core* table (*delayTab*), mapping each neighbor to a delay from the core to the node via that neighbor, and (2) *join-delay* table (*joinDelayTab*), mapping each group member to a delay from the node to that member. The *delayTab* table is updated upon receiving a CORE ANNOUNCE or a BACKWARD ANT packet, and the *joinDelayTab* table is updated upon receiving a JOIN REQUEST packet. To accelerate information dissemination, CSTMAN takes advantage of the broadcast

nature of wireless communication, and uses overheard BACKWARD ANT and JOIN REQUEST control packets to update *delayTab* table, as well as the pheromone table to be introduced in the following. To enforce the delay constraint, CSTMAN consults both tables when choosing the next hop for a JOIN REQUEST packet by checking the satisfaction of the following constraint:

$$delayTab(next) + joinDelayTab(origin) \leq \Delta,$$

where *next* is a next hop to be decided, and *origin* refers to the member who originates the JOIN REQUEST. If no next hop satisfies this constraint, the JOIN REQUEST packet is simply discarded, and the originator of the JOIN REQUEST packet gets disconnected from the core.

To minimize the total cost, each node performs transmission power assignment by computing its node cost as the *largest* required transmission power for reliable communications with its neighbors that are currently selected into the evolving CST as forwarders. To facilitate such computation, each node maintains three cost-related tables: (1) the *neighbor* table (*ntab*), which contains the required transmission power to each neighbor, (2) the *best-cost* table (*bestCost*), which denotes how expensive it is for the node to merge into another group member's path to the core, in terms of the sum of node cost of those nodes along the path from the node to the other group member's path (for instance, the path A→C→D in Figure 3), and (3) the *pheromone* table (τ), which maps each neighbor node to a pheromone value indicating the desirability of choosing that neighbor as a forwarder. The pheromone value is inversely proportional to the sum of the node cost used to go from that neighbor to reach another group member's path to the core. In addition, each entry in τ corresponds to an entry in *delayTab*, and will be updated under the same conditions as for *delayTab*. Therefore, we need to look up τ, as well as *delayTab* and *joinDelayTab*, in order to choose a neighbor with the largest pheromone value whose total path delay satisfies the delay constraint as the next hop to send a JOIN REQUEST packet.

5. CSTMAN PROTOCOL DESCRIPTION

This section describes the CSTMAN algorithm in detail.

5.1 Local Data Structures

Each node i is associated with a cost ($cost_i$) and a delay ($delay_i$). The value of $cost_i$ is the result of the transmission power assignment performed by the CSTMAN protocol. In addition, the following local data structures are maintained at each node i:

—$ntab_i(j)$: denotes a *neighbor* table entry corresponding to neighbor j of node i, which contains the required transmission power (*reqPwr*) for reliable communications from node i to node j.

—$core_i(g)$: indicates the node, believed by node i, to be the current core ID of group g,[7] initially set to *INVALID*.

[7]CSTMAN is capable of constructing parallel CSTs, concurrently supporting different multicast groups.

—$seqNo_i(g)$: keeps track of the sequence number of the latest CORE ANNOUNCE packet.

—$join_i(g)$: denotes the *join* table, which keeps track of nodes requesting node i to be a forwarder of the multicast group g. Table entries are of the form $\langle r, h_r \rangle$, where r is a requesting node's ID, and h_r is its height (described in the following) that it has sent along with its JOIN REQUEST.

—$height_i(g)$: represents the height of node i, which is used to prevent a race condition where two or more destination nodes may attempt to join each other's path and isolate them from the rest of the group. Height is only applicable to forwarders and/or destination nodes, and is identical to the highest ID among the destination nodes, who rely on node i to connect to the core. In addition, $height_i(g)$ is i if node i is a destination node, but not a forwarder of group g, and the core has a height of ∞.

—$\tau_i(g, j, h)$: denotes the *pheromone* table entry representing the pheromone intensity that node i maintains on the link to its neighbor j, with respect to the group g and height h. In other words, it indicates node i's desirability of choosing j as a forwarder for group g, as long as node i is of lower height (w.r.t group g) than h. The allowed range of pheromone intensity is [0, 1].

—$bestCost_i(g, h)$: denotes the *best-cost* table entry keeping track of the currently known best path cost from node i to another forwarder of group g with height h.

—$delayTab_i(g, j)$: denotes the *delay-from-core* table entry corresponding to neighbor node j maintained by node i for group g, which contains the cumulative delay from the *core* to node i via neighbor node j. The entry for neighbor node j also has a corresponding entry in the *pheromone* table, so that the entry is updated whenever the corresponding entry in the pheromone table is updated.

—$joinDelayTab_i(g, t)$: denotes the *join-delay* table entry representing the join-delay at node i with respect to member node t of group g. The entry records the part of the delay from node i to a member node t, measured by a JOIN REQUEST accumulating delays on its trip from its originating member node t to node i. Apparently, the join-delay at node i is 0 if node i is a member node, but not a forwarder, of group g. The join-delay, combined with the corresponding *delay-from-core*, constructs the total path delay that needs to be bound by the delay constraint. Both the *delayTab* and *joinDelayTab* tables need to be consulted to make sure that the delay constraint is not violated whenever choosing a next hop to send a JOIN REQUEST.

5.2 Neighbor Discovery

CSTMAN relies on periodic HELLO packets to allow nodes to learn about their neighbors. In addition, HELLO packets are used to determine the minimum transmission power required by each pair of neighboring nodes for reliable communication between them. HELLO packets are always broadcast with maximum transmission power to ensure that nodes can find all their neighbor nodes who can communicate directly. To support scenarios where nodes may have different

group	coreId	seqNo	accCost	localCost	accDelay

Fig. 4. CORE ANNOUNCE packet format.

the maximum transmission powers, each HELLO includes a field called *txPower*, which is maximum transmission power of the broadcasting node. When hearing a HELLO from node j, node i is able to calculate the transmission power required to reach j using the received signal's power and the marginal reception power, *RX_THRESHOLD*, as follows. Let $txPower_j$ be the transmission power used by node j to send out the HELLO. Let r_{ij} be node j's signal strength measured by node i. Using a simple path loss mode, we have:

$$r_{ij} = txPower_j \times \frac{K}{dist_{ij}{}^{\alpha}}, \tag{1}$$

where $dist_{ij}$ is the distance between nodes i and j, K is a constant, and $2 \leq \alpha \leq 4$ in typical operating environments. In order to communicate with node j reliably, node i's signal needs to be of strength at least *RX_THRESHOLD* when received by node j. Therefore,

$$\begin{aligned} reqPwr_{ij} \times \frac{K}{dist_{ij}^{\alpha}} &= RX\_THRESHOLD \\ reqPwr_{ij} &= RX\_THRESHOLD \times \frac{dist_{ij}^{\alpha}}{K}, \end{aligned} \tag{2}$$

where $reqPwr_{ij}$ is the transmission power used by node i so that node j will receive its signal with the minimum acceptable strength. By combining (1) and (2), we have:

$$reqPwr_{ij} = RX\_THRESHOLD \times \frac{txPower_j}{r_{ij}}. \tag{3}$$

The value $reqPwr_{ij}$ is then used to update $ntab_i(j)$ accordingly. This value indicates the least transmission power node i must use when node j has requested node i to become a forwarder. It is important to note that the calculation of $reqPwr_{ij}$ is *independent* of α, K, and $dist_{ij}$ (the physical distance between nodes i and j).

If a node has not heard a HELLO from its neighbor for the *HELLO_INTERVAL* time period, it removes the entry corresponding to that neighbor from its *ntab*, as well as entries from its *join* table, *pheromone* table, *best-cost* table, and *delay-from-core* table.

5.3 Forwarding Set Initialization

As mentioned earlier, construction of an initial forwarding set is initiated by the core (source node) broadcasting with maximum power, a CORE ANNOUNCE packet, whose format is shown in Figure 4. When node i receives a CORE ANNOUNCE from node j, it updates its $core_g(i)$, $seqNo_g(i)$ and $delayTab_i(g, prev)$, as well as the fields *accCost*, *localCost* and *accDelay* in the packet, as presented in Algorithm 1 (Figure 5). The packet is then rebroadcast. The invocation of the procedure *UpdatePheromoneAndCost* (Algorithm 2, in Figure 6) causes an

Algorithm 1 Node i processing a CORE ANNOUNCE

1: **Input:**
2: $ann \leftarrow$ incoming CORE ANNOUNCE
3: $prev \leftarrow$ the node from which *announce* was received
4: **Begin:**
5: $g \leftarrow ann.group$
6: **if** $coreId_i(g) =$ INVALID_ADDRESS
OR $coreId_i(g) \leq ann.coreId$
OR $seqNo_i(g) \leq ann.seqNo$ **then**
7: Update local information:
$coreId_i(g) \leftarrow ann.coreId$
$seqNo_i(g) \leftarrow ann.seqNo$
$delayTab_i(g, prev) \leftarrow ann.accDelay$
8: Compute the additional cost added to the previous node if requested by i to be a forwarder
$linkCost \leftarrow ntab_i(prev).reqPwr/MAX\_POWER$
$extraCost \leftarrow \max(linkCost - ann.localCost, 0)$
9: Update costs in the announcement packet:
$ann.localCost \leftarrow linkCost$
$ann.accCost \leftarrow ann.accCost + linkCost + extraCost$
10: Update delay in the announcement packet:
$ann.accDelay \leftarrow ann.accDelay + delay_i$
11: Invoke *UpdatePheromoneAndCost*(g, $lastHop, \infty, ann.accCost, FALSE$)
12: Rebroadcast *ann*
13: **end if**

Fig. 5. Node i processing a CORE ANNOUNCE.

update to the *pheromone* and the *best-cost* tables. Specifically, a certain amount of pheromone is added to the entry corresponding to the previous node from which i has received the CORE ANNOUNCE. The field *localCost* allows node i to estimate the extra cost that needs to be added to node j in case i wants to request j to be a forwarder for the group. Note that the cost of a link between a pair of nodes is not the actual transmission power required for both nodes to communicate. Rather, the cost is formulated as a ratio of the required transmission power to the maximum transmission power allowed in the network, denoted as *MAX_POWER*, to limit the range of cost to [0, 1]. The variable $core_i(g)$ is a soft state, which needs to be refreshed every *ANNOUNCE_INTERVAL* time period, or else it is reset back to *INVALID*. Therefore, the node that currently serves as the core of the group is responsible for flooding CORE ANNOUNCE every *ANNOUNCE_INTERVAL*.

Each destination node who maintains a valid core ID must periodically send a JOIN REQUEST packet, also with maximum power, toward the core, to establish a forwarding set. Each JOIN REQUEST contains a list of entries of the form $\langle g, f_g, p_g, h_g, d_g \rangle$, as shown in Figure 7. For node i to broadcast a JOIN REQUEST, it sets the g field to a multicast group address for which i is a destination node or a forwarder. The h_g and p_g fields are set to $height_i(g)$ and the transmission power that node i currently uses for broadcasting data packets, as defined in Equation (6), respectively. The field d_g is set to $joinDelayTab_i(g, h_g)$: the

Algorithm 2 Procedure *UpdatePheromoneAndCost*(g, *next*,*height*,*cost*,*detFlag*) executed by node i

```
1:  Parameters:
2:  g ← multicast group address
3:  next ← neighbor ID indicating which pheromone table entry to be updated
4:  height ← height associated with this update
5:  cost ← cost of choosing next as a forwarder
6:  detFlag ← deterministic flag
7:  Begin:
8:  if τ_i(g,next,height) is not defined then
9:     τ_i(g,next,height) ← 0
10: end if
11: if detFlag = TRUE then
12:    bestCost_i(g,height) ← cost
13:    τ_i(g,next,height) ← τ_i(g,next,height) + 1/(2(cost+ε))
14: else
15:    if bestCost_i(g,height) is not defined OR
          cost < bestCost_i(g,height) then
16:       bestCost_i(g,height) ← cost
17:       τ_i(g,next,height) ← 1.0 /* set intensity to max */
18:    else
19:       τ_i(g,next,height) ← τ_i(g,next,height)
                  +bestCost_i(g,height)/(cost + ε)
20:    end if
21: end if
22: τ_i(g,next,height) ← min{τ_i(g,next,height), 1)}
```

Fig. 6. Procedure *UpdatePheromoneAndCost*(g, *next*, *height*, *cost*, *detFlag*) executed by node i.

$group_1$	$forwarder_1$	$txPower_1$	$height_1$	$joinDelay_1$
...	...	...	...	...
$group_n$	$forwarder_n$	$txPower_n$	$height_n$	$joinDelay_n$

Fig. 7. JOIN REQUEST packet format.

join-delay measured from the destination node h_g to node i. The reason p_g and d_g are included is due to the fact that the JOIN REQUEST is also used by other nodes overhearing the broadcast to update their pheromone and join-delay tables, thus allowing them to estimate the actual joining cost and delay. The field f_g is the neighbor node that i chooses to become a forwarder, which is the node with the highest desirability among i's neighbors that satisfies the delay constraint, with respect to g and h_g. Formally, node i computes the desirability for each of its neighbor by consulting its pheromone table as follows:

$$desire_i(g, h_g, j) = \max_{h>h_g} \frac{\tau_i(g, j, h)}{bestCost_i(g, h) + \epsilon}, \tag{4}$$

where ϵ is a small positive number. Node i then chooses f_g from the one with the highest desirability while satisfying the delay bound as follows:

$$f_g = \text{argmax}_{j \in ntab_i}\, desire_i(g, h_g, j), \tag{5}$$

where $delayTab_i(g, j) + joinDelayTab_i(g, h_g) \leq \Delta$. If none of node i's neighbors satisfies this delay constraint, the JOIN REQUEST packet is simply discarded, and node i gets disconnected from the core.

Node i, who is willing to join a group, should send a request to a neighbor via whom the total path delay does not exceed the delay constraint, and whose goodness was recently confirmed by BACKWARD ANT packets (having high pheromone intensity) and also potentially yields the lowest joining cost. During the initialization process, however, each node has only one entry in its pheromone and delay table, which has been added while processing the first CORE ANNOUNCE packet. As a result, each destination node will choose the node it has received the CORE ANNOUNCE from to be a forwarder, if the accumulated delay does not exceed the delay constraint, by including that node's ID in the JOIN REQUEST it sends out. A node who receives a JOIN REQUEST containing its own ID as a forwarder will add an entry with the sender ID into its join table and follow the same procedure to broadcast its own JOIN REQUEST. The same process continues until the request reaches the core. Nodes with non-empty join tables with respect to group g then become forwarders for group g, and thus an initial forwarding set is formed.

Let D_i be the *dependency set* of node i, which consists of nodes requested by node i as forwarders, as well as nodes that are requesting node i as a forwarder (as listed in the *join* table of node i). The assigned transmission power for node i, denoted as $cost_i$ is computed as follows,

$$cost_i = \max_{j \in D_i} ntab_i(j).reqPwr. \tag{6}$$

Notice that each forwarder uses Equation (6) to compute its transmission power level for transmitting (broadcasting) data packets, while control packets (HELLO and ANT packets) are always transmitted with maximum transmission power level(s).

Since CSTMAN is a reactive protocol, there is no multicast connectivity available to support data delivery when the source becomes active. Hence, the formation of the initial forwarding set is intended to be done with minimal delay while the total cost of the forwarding set is not the main concern during this step. Although every node keeps track of the path cost to the core upon receiving a CORE ANNOUNCE, destination nodes/forwarders simply send a JOIN REQUEST toward a node from which the CORE ANNOUNCE first arrived. They do not take into consideration the resulting cost as yet. It is up to the evolution process to learn new forwarding sets to lower the total forwarding cost over time, as well as keeping track of delays from core.

5.4 Forwarding Set Evolution

The process of deforming the forwarding set starts immediately after destination nodes have heard the first announcement from the core. For every *ANT_INTERVAL*, each destination node releases a FORWARD ANT, whose format is shown in Figure 8(a), to learn a different path that satisfies the delay constraint, hopefully with lower cost, to connect to the core. Since a destination node is allowed to connect to either the core directly or a forwarder of height

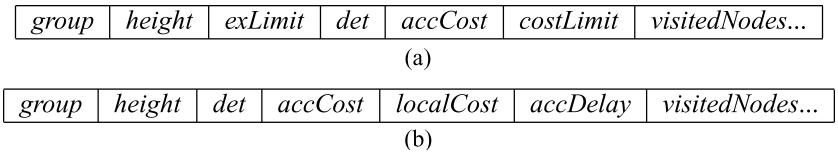


Fig. 8. Ant packet formats: (a) FORWARD ANT and (b) BACKWARD ANT.

higher than its own, every FORWARD ANT carries the height of the destination node that releases it. To avoid loops, IDs of the nodes visited are recorded when each FORWARD ANT traverses the network.

Two types of ants are employed in the process: nondeterministic and deterministic ants. Deterministic ants always follow the next hop with the highest desirability, based on the desirability function in Equation (4). Their objective is to measure the partial cost each destination node is contributing to the total forwarding cost, as well as to ensure the path is still valid and reinforce it. Nondeterministic ants, on the other hand, do not always follow the most desirable path, but probabilistically choose different paths to explore and opportunistically discover a new path with lower cost. A nondeterministic FORWARD ANT, released by a destination node of group g with ID h (its height), determines the probability of each next hop when it is about to leave node i as follows:

$$Prob(j) = \begin{cases} 0 & \text{if } j \in visitedNodes; \\ \dfrac{\frac{desire_i(g,h,j)}{ntab_i(j).reqPwr}+1}{\sum_{j \in ntab_i \setminus visitedNodes}\left[\frac{desire_i(g,h,j)}{ntab_i(j).reqPwr}+1\right]}, & \\ & \text{otherwise,} \end{cases} \tag{7}$$

where $desire_i(g, h, j)$ is the desirability of choosing the next hop defined in Equation (4) and $ntab_i(j).reqPwr$ is the computed transmission power required for reliable communications from node i to node j. By incorporating the transmission power required to reach a particular neighbor, a closer neighbor will have a higher probability to be the next hop for the FORWARD ANT. A base desirability value is added to all neighbors that have not been visited, to increase the chances of visiting a neighbor whose pheromone table entry does not exist. However, ants may get lost very easily when allowed to visit those nodes that have no trace of pheromone, so the *exLimit* field is present to avoid such behavior. Specifically, a nondeterministic FORWARD ANT has a 50% chance to explore a different node rather than the most desirable one, up to the number of times specified in the *exLimit* field. In addition, no nondeterministic ant is allowed to proceed if the accumulated cost it has recorded in the *accCost* field, exceeds the cost limit specified in the *costLimit* field. The cost limit is generally obtained from the *best-cost* table. Similar to how CORE ANNOUNCEs are processed, the cost of a link is the ratio of the transmission power required for that link to *MAX_POWER*. Therefore, as long as the hop count constraint and cost limit are satisfied, FORWARD ANT packets will most likely include more nodes with shorter distances in its exploring path. Note that *exLimit*, *accCost*, and *costLimit* fields are not used by deterministic FORWARD ANT packets. The *det* field in a FORWARD

ANT packet is used to indicate whether the ant is deterministic. Every other ant deployed is a deterministic ant.

Once a FORWARD ANT arrives at a forwarder whose height is higher than the value in its *height* field, it turns into a BACKWARD ANT. Then it returns to its originator using the information in the *visitedNodes* list. Figure 8(b) illustrates the packet format used by BACKWARD ANT packets. The *height* field is set to the height of the forwarder it encounters. The accumulated cost, *accCost* is reset to zero and will be used to measure the cost along its return trip. The use of the *localCost* field is the same as what is used in CORE ANNOUNCE, to allow other nodes to compute the extra cost if the sender of the ant is to be chosen as a forwarder. The accumulated delay, *accDelay* is initialized to be $delayTab_i(g, j)$, to track the accumulated delay from core along its trip back, where i is the forwarder found and j is the next hop from i toward the core determined by Equation (5).

When a BACKWARD ANT, *bant*, broadcast by node i is received or overheard by its neighbor j, the fields *accCost* and *localCost* are updated as follows:

$$\begin{aligned} bant.accCost' &= bant.accCost + linkCost + extraCost, \\ bant.localCost' &= linkCost, \end{aligned}$$

where *linkCost* and *extraCost* are defined as:

$$\begin{aligned} linkCost &= ntab_j(i).reqPwr/MAX\_POWER, \\ extraCost &= \max(linkCost - bant.localCost, 0). \end{aligned}$$

Node j then invokes the procedure *UpdatePheromoneAndCost* (Algorithm 2, in Figure 6) to update the *pheromone* and the *best-cost* tables, using i and $bant.accCost'$ as the values for the arguments *next* and *cost*, respectively. The arguments g, *height*, and *detFlag* are obtained directly from the BACKWARD ANT packet. Node j also updates its *delay-from-core* table using *bant.accDelay*. If node j sees its ID listed at the end of *visitedNodes* in the BACKWARD ANT, it realizes that it is the ant's intended receiver. It then removes its ID from *visitedNodes* and rebroadcasts the ant packet.

It can be observed that similar steps are also performed in Algorithm 1 (Figure 5) to update the *accCost*, *localCost* and *accDelay* fields in a CORE ANNOUNCE packet. In fact, CORE ANNOUNCEs can be interpreted as BACKWARD ANT packets of infinite height released by the core. What the procedure *UpdatePheromoneAndCost* does is to increase the pheromone intensity in the entry corresponding to the previous hop with an amount based on the cost. If the update is deterministic—invoked when processing a deterministic BACKWARD ANT—a small amount of pheromone inversely proportional to the cost is added for the corresponding next hop and height. The best cost for that height is overwritten as well. If the update is nondeterministic and the given cost is lower than what is in the *best-cost* table, the corresponding pheromone intensity is updated to the maximum value. The best cost is also set to the new cost in this case. Otherwise, a small amount of pheromone is just added to the corresponding entry.

Since a node always consults its *pheromone* (τ) and *delay-from-core* (*delayTab*) tables to prepare and send out a JOIN REQUEST message, pheromone

updated by BACKWARD ANT packets will have a direct impact on which neighbor will be chosen as a forwarder. The node then updates its *dependency set* and performs transmission power assignment as defined in Equation 6. Therefore, a BACKWARD ANT coming back with a better cost will immediately trigger a deformation of the forwarding set. If the cost is not better but low enough, pheromone added to the path will likely attract future FORWARD ANT packets to follow and explore areas around that path, hoping to discover a better one.

Pheromone evaporation allows past history to be softly forgotten so that up-to-date information will become more significant, making the protocol adaptive to changing environments. In CSTMAN, in every *DECAY_INTERVAL* time period, pheromone intensities in all pheromone table entries are reduced by *DECAY_FACTOR*, which is a number between 0 and 1. In addition, measurement of link failure frequency allows nodes to estimate the dynamics of their surrounding areas by means of periodic hello messaging. Every certain time interval, each node computes the *normalized link failure frequency*, or *nlff*, which denotes the number of link failures detected per second per neighbor. Whenever a node observes that its *nlff* exceeds *NLFF_THRESHOLD*, it adds an extra entry for each group when sending out a JOIN REQUEST packet. This extra entry is to request the node with the second highest desirability to be a forwarder. Hence, there will be more forwarders in highly dynamic areas, making multicast connectivity to be more robust to link failures.

6. PERFORMANCE EVALUATION

Using QualNet, we devised two groups of simulation experiments to evaluate CSTMAN in solving CST-*wireless-ec* and ST-*wireless-ec* problems, respectively.

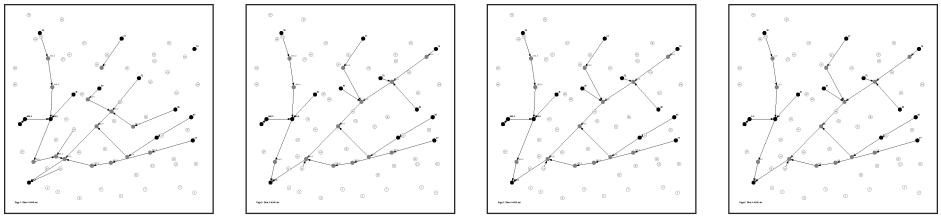
6.1 Experiments for CST-*wireless-ec*

Since there exist neither known optimal solutions to problem instances of *CST-wireless-ec*, nor other algorithms addressing the same problem, it is infeasible to evaluate the efficiency of CSTMAN via comparative studies. To validate the effectiveness and study the characteristics of CSTMAN, we conducted a set of simulations, and present network topology snapshots and forwarding costs (the total transmission power of the forwarding set) of sample CST-*wireless-ec* problems with different hop count constraints.

6.1.1 *Simulation Setup.* Fifty networks were randomly generated with 75 nodes uniformly distributed over a terrain of size 1000×1000 m^2. Each node was equipped with a radio transceiver of adjustable transmission power capable of transmitting signals at the maximum transmission power of 15 dBm over a 2 Mbps wireless channel. This maximum transmission power yields a communication range of up to approximately 250 meters, assuming a two-ray path loss model without fading. We used IEEE 802.11DCF as the MAC layer protocol with promiscuous mode, and IP as network layer. No other routing protocol was running except CSTMAN, which was being evaluated. For each network, a multicast session of one source and 13 group members was

Table I. Simulation Parameters

Terrain dimension	1000×1000 m^2
Number of nodes	75
Communication Range	250 m
Network Bandwidth	2 Mbps
MAC Protocol	IEEE 802.11DCF (broadcast mode)
Multicast group size	13, plus a source
Application Traffic	MCBR (4×512B/s)
HELLO_INTERVAL	3 sec
ANNOUNCE_INTERVAL	10 sec
ANT_INTERVAL	3 sec
EXPLORE_LIMIT	3
DECAY_INTERVAL	2 sec
DECAY_FACTOR	0.05
NLFF_THRESHOLD	0.01
MAX_POWER	15 dBm
RX_THRESHOLD	−81 dBm

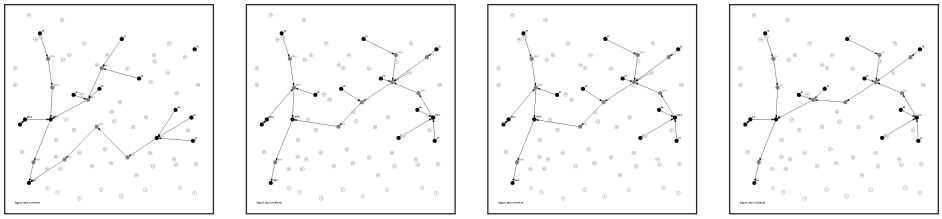


(a) (b) (c) (d)

Fig. 9. Sample network snapshots at the 4th second with Δ of (a) 5, (b) 8, (c) 10, and (d) ∞ hops.

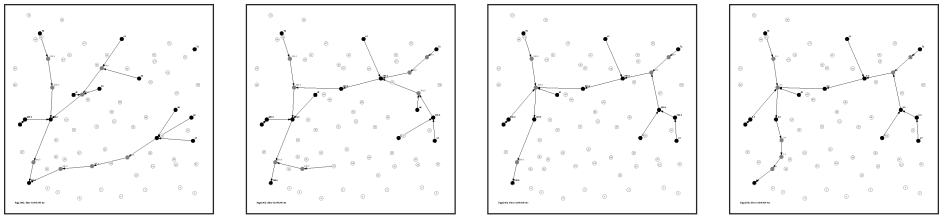
simulated for 1,000 seconds of simulation time. The source generated a multicast constant bit rate (MCBR) traffic of 4 packets/sec starting from the 30th second of the simulation till the end of simulation, and the size of the data payload was 512 bytes. Table I summarizes the parameter settings for networks and CSTMAN in the simulation experiments.

6.1.2 *Results and Discussion.* Figures 9, 10, and 11 present network topology snapshots comparing four forwarding sets formed in one sample network under different Δ of 5, 8, 10, and ∞ hops at the 4th, the 35th, and the 1000th second of simulation time, respectively. Notice that when $\Delta = \infty$, the problem degenerates to the wireless version of the nonconstrained Steiner tree problem. Terminal nodes are shown in black with the source node located at the lower left corner, and other forwarders in gray. Arrows represent the forwarding paths of JOIN REQUEST packets. Under the same Δ, we could observe the evolution of a forwarding set over time by comparing the corresponding vertical figures (for instance, Figures 9(a), 10(a), and 11(a)). It is observed that while the initial forwarding sets at the 4th second resemble shortest path trees from the source to group members, they evolve rapidly (which is also observable in Figures 12 and 13) to topologies (at the 35th second) resembling the topologies at the 1000th second, and eventually evolve into configurations where each



(a) (b) (c) (d)

Fig. 10. Sample network snapshots at the 35th second with Δ of (a) 5, (b) 8, (c) 10, and (d) ∞ hops.



(a) (b) (c) (d)

Fig. 11. Sample network snapshots at the 1000th second with Δ of (a) 5, (b) 8, (c) 10, and (d) ∞ hops.

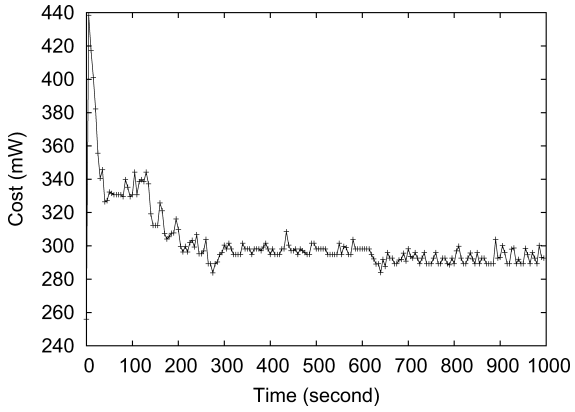


Fig. 12. Forwarding cost for the sample network versus time, with hop count constraint $\Delta = 10$ hops.

group member tends to merge with another member's path with higher height, so as to decrease the total forwarding cost. A *backbone* can be easily identified as the path from the source (at the lower left corner) to the upper left member, who has the highest height, since the general orientation and shape of the path do not change throughout the evolution. In addition, for each Δ, the average hop count of paths $\overline{H}$ from the source to each member increases, due to the power reduction mechanism of CSTMAN, while the largest hop count among all paths, H_{max}, does not exceed the given Δ. At the same time instance, we could observe the evolution of a forwarding set under different Δ's by comparing the corresponding horizontal figures (for instance, Figures 9(a),

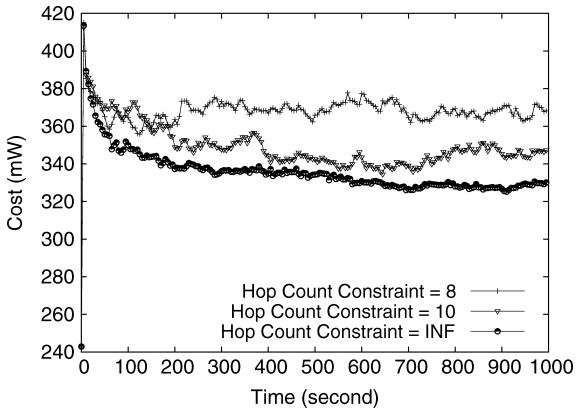


Fig. 13. Average forwarding cost for 50 networks versus time, with hop count constraint $\Delta = 8$, 10, and ∞ hops.

9(b), 9(c), and 9(d)). It is observed that that all initial forwarding sets look exactly the same except for the case of $\Delta = 5$, where the upper rightmost member is disconnected due to the violation of the hop count constraint. Notice that the upper rightmost member remains disconnected throughout the simulation for the case of $\Delta = 5$. The forwarding sets at the 1000th second look different with different Δ's, showing that the bigger the Δ, the larger the $\overline{H}$ and H_{max}. While the snapshots visualize the evolution of forwarding sets and validate the satisfaction of delay constraints qualitatively, Figure 12 quantitatively depicts the total forwarding cost over time for the case of $\Delta = 10$ using the same sample network. It can be observed that the total cost drops dramatically within the first 50 seconds, keeps lowering gradually, and then becomes stable and stays low after about 300 seconds, when possibly no better paths could be found.

Next, we'd like to show the relationship between forwarding cost and delay constraint, Δ. The following figures were made by averaging simulation results from all 50 networks, and plotted with a confidence interval of 90%. Figure 13 compares the average forwarding power over time among Δ of 8, 10, and ∞. They all start at about the same cost, and decrease rapidly with the first 50 seconds. Then they level off at different rates, and finally stay at different cost levels, with $\Delta = 8$, 10, and ∞ in descending order. This shows that the bigger the Δ, the less the total forwarding cost. Figure 14 further quantifies this relationship. We ran CSTMAN over 50 networks with Δ ranging from 5 to 25, and ∞, and averaged the total forwarding cost from 50 networks over the entire simulation period for each Δ. As shown in Figure 14, initially the average cost increases with Δ, because Δ is too restricted to connect all the members, as is the case in Figures 9(a), 10(a), and 11(a). Then, the forwarding cost peaks at the critical point of $\Delta = 7$ when all members can just be reached by source in all networks. After that, the average cost drops rapidly with Δ relaxed with several more hops. Then the reduction slows down, and the average cost stays close to the value of $\Delta = \infty$, which is shown as the starting value of the y-axis.

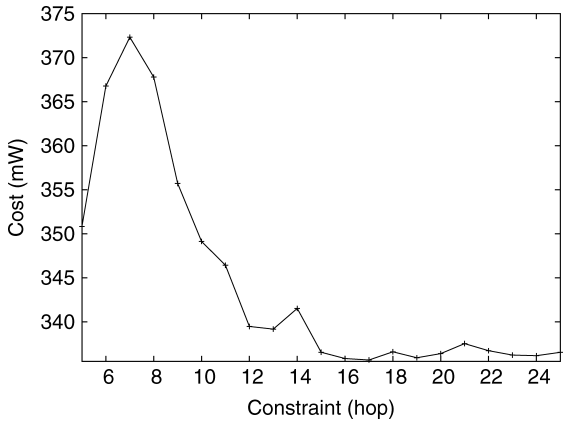


Fig. 14. Average forwarding cost for 50 networks over entire simulation time versus hop count constraint Δ, with the starting value of the y-axis corresponding to the case of $\Delta = \infty$ hops.

6.2 Experiments for ST-*wireless-ec*

As mentioned at the end of Section 3, our algorithm could handle the ST-*wireless-ec* problem when we set $\Delta = \infty$. We evaluated the performance of CSTMAN in solving ST-*wireless-ec* and compared it with ODMRP [Bae et al. 2000]. Performance statistics were collected for various network sizes, mobility speeds, and number of multicast sources.

6.2.1 *Simulation Setup.* In the following experiments, network size changes between 50 and 200, number of sources ranges from 1 to 8, and the number of group members becomes 10. We used the random-waypoint mobility model with speed from 0 to 10 m/s and pause time of 60 seconds. Other simulation parameters are basically the same as those in Table I. For ODMRP, we used its implementation in QualNet, which followed the specification in the Internet Draft `draft-ietf-manet-odmrp-04.txt` [Yi et al. 2002].

The following statistics were collected and used in the comparison. Each measurement was obtained from 20 different randomly generated networks and will be shown with a 95% confidence interval:

—Packet delivery ratio: indicates the fraction of data packets originated that are received by the members. This metric reflects *effectiveness* of a protocol.

—Number of total bytes transmitted per data packet received: indicates the ratio of the number of bytes transmitted over the channel to the number of data packets received by the members. This metric reflects *efficiency* of a protocol in terms of bandwidth utilization.

—Energy consumed per data packet received: indicates the ratio of the amount of energy consumed for transmitting packets to the number of data packets received. Energy consumption is measured from every byte that is transmitted over the channel during an entire simulation. This metric reflects *efficiency* of a protocol in terms of energy usage.

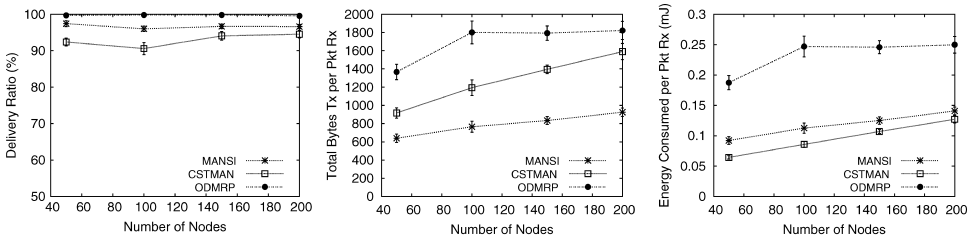


Fig. 15. Comparison among CSTMAN and ODMRP in terms of (a) packet delivery ratio, (b) total bytes transmitted per data packets received, and (c) energy consumed per node, versus network size.

6.2.2 *Results and Discussion.* The first experiment was performed under various network sizes, ranging from 50 nodes to 200 nodes, with one source and no mobility. Figure 15(a) shows that network size does not have a significant impact on the performance of the two protocols in terms of packet delivery. It can be seen that ODMRP can deliver a higher fraction of packets. These results can be explained by the fact that ODMRP does not rely on periodic control packets as much as CSTMAN. Since both protocols employ only 802.11's broadcast mode, where RTS/CTS/ACK are not used, the hidden terminal problem arises and causes packets to drop due to collisions. The problem becomes more visible with periodic messaging. Thus we observe the lower delivery ratio from CSTMAN. Furthermore, since CSTMAN transmits data packets with lower transmission power levels, it is more sensitive to interference. As a result, it yields a lower packet delivery ratio than ODMRP. Nevertheless, it can successfully deliver more than 90% of the packets, which is acceptable. Figure 15(b) compares the total number of bytes transmitted to the channel per data packet received. As we see, CSTMAN transmits fewer bytes than ODMRP, since ODMRP's periodic flooding of data packets incurs high bandwidth usage. It can be observed that, as the network size grows, MANSI's and CSTMAN's overhead increases due to periodic control packets being transmitted by every node. In terms of energy efficiency, CSTMAN shows significantly lower energy consumption per packet received than ODMRP, as evidenced in Figure 15(c). In a 50-node network, it consumes approximately 65% less energy than ODMRP. The differences become smaller as the network size increases, which is most likely caused by the overhead from periodic control packets transmitted at full power.

In the second experiment we fixed the network size to 100 nodes, and varied the number of sources from 1 to 8, still without mobility. As the number of sources in the group increases, the contention built up causes both protocols to drop more and more packets, as presented in Figure 16(a). With one source, CSTMAN can deliver around 91% of packets. Then the packet delivery performance drops gradually with more sources. In contrast, ODMRP's packet delivery ratio starts relatively high, but begins to drop sharply after 4 sources due to very high congestion because every source floods data packets periodically. In fact, ODMRP drops more packets than CSTMAN after 6 sources. In

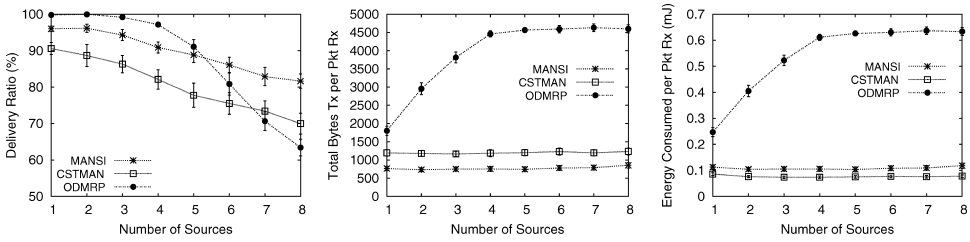


Fig. 16. Comparison between CSTMAN and ODMRP in terms of (a) packet delivery ratio, (b) total bytes transmitted per data packets received, and (c) energy consumed per node, versus number of sources.

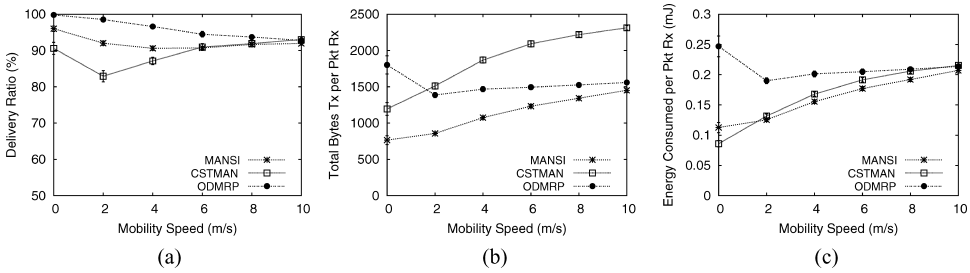


Fig. 17. Comparison between CSTMAN and ODMRP in terms of (a) packet delivery ratio, (b) total bytes transmitted per data packets received, and (c) energy consumed per node, versus mobility speed.

the efficiency aspect, as shown in Figures 16(b) and 16(c), CSTMAN is not affected by an increasing number of sources, as there is still only one source per group designated as the core to flood announcement packets. ODMRP's overhead in both bandwidth usage and energy consumption climbs constantly with the number of sources in the beginning and stops after 4 sources since most forwarding nodes fail to accept and forward packets any further.

Our last experiment evaluated the performance of the protocols with different mobility speeds, varying from 0 to 10 m/s. In this experiment, the network size and the number of sources were set to 100 and 1, respectively. Figure 17(a) shows that the delivery ratio of all three protocols decreases when the network changes from stationary to moving at low speed. As mobility speed increases, ODMRP continues to drop more packets. However, the packet delivery ratio of CSTMAN gradually increases. This is due to the fact that nodes request more of their neighbors to become forwarders when they detect that link failure frequency exceeds the threshold, resulting in more robust connectivity among members. Eventually at a mobility speed of 10 m/s, both protocols yield the same performance in terms of packet delivery. Although CSTMAN is shown Figure 17(b) to transmit the largest number of bytes under mobility, it in fact consumes less energy per packet received than ODMRP, as shown in Figure 17(c). At mobility speeds of 10 m/s, both of them perform equally in terms of energy efficiency. Hence, CSTMAN exhibits a good trade-off between energy efficiency and effectiveness in various mobility conditions.

7. CONCLUSION

In this article, we formally define a constrained Steiner tree problem in the context of wireless networks, and propose a distributed CST algorithm termed CSTMAN, which balances energy conservation and delay of multicast communications in ad hoc networks. By adopting the metaphor of swarm intelligence, a CST evolves from an initial topology into states that yield lower total forwarding cost, subject to the imposed hop count constraint. The design decisions for CSTMAN as a distributed and self-organizing algorithm can be summarized using the paradigms discussed in Prehofer and Bettstetter [2005]: (1) Devise local behavior rules that can achieve global properties. Locally, each destination node opportunistically explores a better path to the source by merging into a path of a destination node with higher height, and choosing a closer forwarder while satisfying the delay constraint. Globally, forwarding cost is decreased. (2) Exploit implicit coordination via overhearing and local inference. The broadcast advantage makes overhearing of control packets practicable. Local tables, such as the pheromone table, best cost table, and delay from core table, are updated upon overhearing Join Request and Backward Ant packets, which helps to keep local information up-to-date and speed up information dissemination so as to accelerate the forwarding set evolution. (3) Keep local information in a soft state. Pheromone in the pheromone table evaporates and other information such as neighbor table and join table are flushed at intervals, so as to adapt to network dynamics. The effectiveness and the characteristics of CSTMAN are validated and revealed via simulation results showing topology snapshots and lowering total forwarding cost over time.

REFERENCES

BAE, S., LEE, S., SU, W., AND GERLA, M. 2000. The design, implementation, and performance evaluation of the on-demand multicast routing protocol in multihop wireless networks. *IEEE Netw. Special Issue on Multicasting Empowering the Next Generation Internet 14,* 1 (Jan./Feb.), 70–77.

BONABEAU, E., DORIGO, M., AND THERAULAZ, G. 1999. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York.

CARTIGNY, J., SIMPLOT, D., AND STOJMENOVIC, I. 2003. Localized minimum-energy broadcasting in ad hoc networks. In *Annual Joint Conference of the IEEE Computer and Communication Societies (INFOCOM 2003)*. San Francisco, CA.

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., AND STEIN, C. 2001. *Introduction to Algorithms, Second Edition*. The MIT Press, Cambridge.

GOSAVI, S., DAS, S., VAZE, S., SINGH, G., AND BUEHLER, E. 2003. Obtaining subtrees from graphs: an ant colony approach. In *Swarm Intelligence Symposium*. Indianapolis, IN.

HUANG, T.-L. AND LEE, D. T. 2002. A distributed multicast routing algorithm for real-time applications in wide area networks. In *Proceedings of the International Symposium of Parallel Architectures, Algorithms and Networks (ISPAN)*, 335–340.

KOMPELLA, V., PASQUALE, J., AND POLYZOS, G. 1993a. Multicast routing for multimedia communication. *IEEE Trans. Netw. 1,* 3 (June), 286–292.

KOMPELLA, V., PASQUALE, J., AND POLYZOS, G. 1993b. Two distributed algorithms for multicasting multimedia information. In *2nd International Conference on Computer Communication (ICCC)*. 343–349.

LIANG, W. 2006. Approximate minimum-energy multicasting in wireless ad hoc networks. *IEEE Trans. Mobile Comput. 5,* 4 (April), 377–387.

LUN, D. S., RATNAKAR, N., MEDARD, M., KOETTER, R., KARGER, D. R., HO, T., AHMED, E., AND ZHAO, F. 2006. Minimum-cost multicast over coded packet networks. *IEEE Trans. Info. Theory 52,* 6 (June), 2608–2623.

PAN, D., MAVINJURVE, P., NGO, H. Q., VERMA, V., AND CHANDAK, A. 2004. DMIP3S: distributed algorithms for power-conserving multicasting in static wireless ad hoc networks. In *IEEE Workshop on High Performance Switching and Routing (HPSR)*. Phoenix, Arizona, 236–240.

PREHOFER, C. AND BETTSTETTER, C. 2005. Self-organization in communication networks: principles and design paradigms. *IEEE Comm. Mag. 43,* 7 (July), 78–85.

SHEN, C.-C. AND JAIKAEO, C. 2005. Ad hoc multicast routing algorithm with swarm intelligence. *ACM Mobile Netw. Appl. J. 10,* 1–2 (Feb.), 47–59.

SINGH, G., DAS, S., GOSAVI, S., AND PUJAR, S. 2004. Ant colony algorithms for Steiner trees: an application to routing in sensor networks. In *Recent Developments in Biologically Inspired Computing*, L. N. de Castro and F. J. von Zuben, Eds. Idea Group Inc., Chapter 6.

WI, S. AND CHOI, Y. 1995. A delay constrained distributed multicast routing algorithm. In *12th International Conference on Computer Communication (ICCC)*. 833–838.

WIESELTHIER, J. E., NGUYEN, G. D., AND EPHREMIDES, A. 2002. Energy-efficient broadcast and multicast trees in wireless Networks. *Mobile Netw. Appl. 7,* 6 (Dec.), 481–492.

WU, Y., CHOU, P. A., AND KUNG, S.-Y. 2005. Minimum-energy multicast in mobile ad hoc networks using network coding. *IEEE Trans. Comm. 53,* 11 (Nov.), 1906–1918.

YI, Y., LEE, S.-J., SU, W., AND GERLA, M. 2002. On-demand multicast routing protocol (ODMRP) for ad hoc networks. http://www.ietf.org/proceedings/02nov/I-D/draft-ietf-manet-odmrp-04.txt.

Received December 2006; revised July 2007; accepted November 2007