© The Author 2011. Published by Oxford University Press on behalf of The British Computer Society. All rights reserved.
For Permissions, please email: journals.permissions@oup.com
Advance Access publication on March 15, 2011
doi:10.1093/comjnl/bxr023

A Depth-optimal Low-complexity Distributed Wireless Multicast Algorithm

A. Sinan Akyurek* and Elif Uysal-Biyikoglu

Electrical and Electronics Engineering Department, METU, Ankara, Turkey
**Corresponding author: asakyurek@aselsan.com.tr*

This paper presents a wireless multicast tree construction algorithm, SWIM (Source-initiated WIreless Multicast). SWIM constructs a tree on which each multicast destination has the minimum possible depth (number of hops from the nearest source). It is proved that SWIM is fully distributed, with a worst case complexity upper-bounded by $O(N^3)$, and an empirically found average complexity of only $O(N^2)$. SWIM forms one shared tree from source(s) to the multicast destinations; yet, as a by-product, it creates a multicast mesh structure by maintaining alternative branches at every tree node, thus providing robustness to link failures. This makes it suitable for both *ad hoc* networks and access networks with multiple gateways. In terms of minimizing the number of forwarding nodes, SWIM is optimal for unicast and competitive with the state of the art for multicast, outperforming the best known distributed approaches from the literature except for the multicast *ad hoc* on demand distance vector (MAODV) algorithm. However, simulations of the MAODV algorithm alongside SWIM on a large set of network instances show that the depth minimality of SWIM leads to lower average delay per multicast destination. It is also shown that the delay performance of SWIM is virtually unaffected by the presence of low mobility in the network.

Keywords: wireless multicast; multicast tree; minimum depth; wireless broadcast problem; greedy set cover; mesh network; number of forwarding nodes

Received 6 July 2010; revised 15 November 2010
Handling editor: Yu-Chee Tseng

1. INTRODUCTION

In multi-hop wireless networks, such as mobile *ad hoc* networks and mesh networks, multicast sessions from a source node, gateway (or set of gateways) to a group of destinations, often occur. Such sessions are sometimes delay sensitive (in the case of, for example, near-real-time applications). Rather than sending the data along unicast routes to each destination, it is desirable, primarily to avoid burdening the network with unnecessary transmissions, to construct a multicast route [1]. A good multicast tree will make efficient use of the bandwidth- or energy-constrained wireless links [2, 3], avoid bottlenecks at multicast source nodes and maximize energy efficiency by keeping the *number of forwarding nodes* (NFN) low, while maintaining a sufficiently low delay for each destination. To keep the delay low, it is of interest to make the depth of each destination on the multicast tree small.

Our aim is to develop an algorithm that keeps the depth of a destination node to a minimum, while keeping the NFN as low as possible. Minimizing NFN requires a solution of the Wireless Multicast Tree (WMT) problem: finding a routing tree with the minimum number of transmissions needed to reach all destinations; the tree needs to dominate, but not necessarily include all destination nodes. An illustration of this is given in Fig. 1.

The WMT problem is an NP-complete problem [4]. To see this, it suffices to show that any instance of the Steiner Tree problem can be converted to an instance of WMT in polynomial time (see Fig. 2): simply add dummy neighbor nodes to each of the nodes to be covered by the Steiner tree (except one, the source), and then solve the wireless multicast problem for the specified source and with the set of dummy nodes as the multicast group. For a detailed survey on the Steiner Tree Problem, refer to [5, 6].

Downloaded from http://comjnl.oxfordjournals.org/ at University of California, San Diego on August 7, 2013

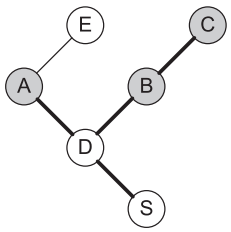


FIGURE 1. The multicast group is $M = \{A, B, C\}$. Node S is the source. For the wired case, the routing tree (*i.e.* the Steiner Tree) consists of the nodes $\{S, A, B, C, D\}$. This is the smallest tree containing $\{S\} \cup M$. In the wireless case, allowing MAC layer broadcast (a.k.a. the wireless multicast advantage,) one transmission from D suffices to reach both A and B. So, the optimal WMT consists only of the nodes $\{S, D, B\}$.

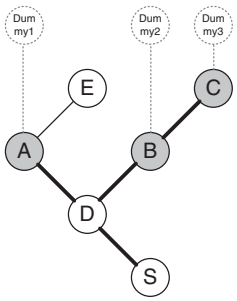


FIGURE 2. Example showing how an instance of the Steiner Tree problem can be turned into an instance of WMT: add dummy neighbors to each of the nodes to be covered by the Steiner tree (except the source), and find the WMT solution: $\{S, A, B, C, D\}$. This is identical to the Steiner Tree for the previous example (Fig. 1).

In this paper, we approach the wireless multicast problem with two goals: the main goal is to obtain a multicast *tree* with minimum depth to decrease the forwarding delay, while keeping NFN as low as possible, in the single-source case. The secondary goal is to obtain a multicast *mesh*, for suitability to the multiple-source scenario and also robustness to link failures and mobility.

The reason for the second goal is our conviction that the multiple-source scenario is important, especially with the emergence of wireless mesh networks (WMN) as access networks for widespread wireless networking, with all the self-organization, self-configuration and self-healing properties of this architecture [7, 8]. In the mesh network scenario, outside access (access to larger networks) is provided by *several gateway nodes* in the network. Clearly, it may be advantageous for different network nodes to be accessing different gateways depending on their respective proximity to these gateways. In this case, insisting on a tree can lead to a poor solution; rather, a multiple-source routing graph needs to be considered.

In many applications of mobile *ad hoc* networks and WMNs, the network contains nodes that may be mobile (or portable), which act both as end users and routers. WMNs, however, also contain mesh routers, which tend to be stationary and act as gateways through which the mesh clients access larger networks. The gateway/bridge functionality of the mesh routers enable the integration of WMNs (and structures such as *ad hoc* and sensor networks) with various networks, including the Internet, cellular networks, WiMAX, etc.

Combining both goals, the problem considered in this paper involves a node (or a set of nodes) acting as a source to a group of other nodes, the *multicast group*, that have requested the same data. The vision is to relieve the bottlenecks forming at sources [2, 9], and be bandwidth and energy efficient.

The main contributions of this paper are the following:

(i) Source-initiated Wireless Multicast (SWIM) is depth optimal by construction in both single- and multiple-source cases: it forms a routing graph where the hop count of each destination to the nearest source is as small as possible. In terms of NFN, SWIM is optimal in the unicast case and exhibits good average performance in multicast and broadcast scenarios. Correct distributed operation without increased complexity is ensured in the multiple-source case, while taking advantage of multiple sources lowers depth and NFN significantly.
(ii) It is observed that the wireless multicast problem inherently contains the problem of finding good Set Covers at various levels from source(s) to destinations. This observation is used to devise a distributed and low-complexity routing algorithm based on a well-known Greedy Set Cover.
(iii) Finally, a method is developed for creating alternative routing trees, which are desirable especially in the case of mobility [10] and in multimedia streaming [11] for reliable transmission.

The rest of this paper is organized as follows: in the next section, we discuss related work reported in the literature. Section 3 introduces the proposed algorithm SWIM. The correctness and complexity analyses of SWIM are presented in Section 4. In Section 6, the performance of SWIM is explored through extensive simulations and compared with several algorithms from the literature. In section 7, a method to use existing table-based unicast algorithms for SWIM tree construction is given. Section 8 presents an extension of the algorithm for dynamic networks. In section 9, the generation of alternative paths is explained. Section 10 presents concluding remarks and outlines future directions.

2. RELATED WORK

Interest in wireless multicast has risen rapidly in the last decade [12]. A number of multicast routing algorithms have been developed, focusing on different priorities such as low latency, energy efficiency and so on [13]. A notable multicast tree formation algorithm, for both weighted and not-weighted

Downloaded from http://comjnl.oxfordjournals.org/ at University of California, San Diego on August 7, 2013

graphs, appears in [14]. This solution is based on merging optimum unicast routes and pruning the resulting subgraph. A different approach is presented in [15]: the objective is to select the minimum number of nodes in the network that are 'on' (and keep others turned 'off'), while keeping a communication path from the source to the destinations, by utilizing information about the geographic position of the nodes in the network. A minimum spanning tree is calculated on the final state to further reduce the number of 'on' nodes. Another notable multicast routing heuristic presented in [16] relies on clustering and a certain medium access control (MAC) protocol [17].

There is a richer literature on wireless *broadcast*, which is a special case of the multicast problem. Much of the recent work on broadcast has considered energy efficiency, and power control [15, 18, 19]. Iterative Maximum-Branch Minimization, proposed in [20], constructs an iterative mechanism for reducing power in a source-initiated wireless broadcast tree, with the objective of minimizing the total required power. Another iterative method with an energy-efficiency objective is presented in [21], where integer programming has been used.

NP-hardness of the minimum energy broadcast problem in metric space was proved in [22] and, later in [23], it was shown under more general conditions that power-optimal broadcast is NP-complete. Wieselthier *et al.* [18] proposed several broadcast tree heuristics: Broadcast Incremental Power, broadcast link-based minimum-cost spanning tree and broadcast least unicast. It is worth noting that multicast versions of these, namely, multicast incremental power, multicast least unicast and multicast link-based minimum-cost spanning tree, have also been proposed. These algorithms have some commonalities with the algorithm proposed in this paper: worst case complexity of $O(N^3)$, optimality in the unicast case and containing a sweeping operation to remove unnecessary transmissions. However, not all of these are distributed and, moreover, their closeness to the optimal tree in terms of the number of transmissions, in other words, NFN was not studied. Moreover, as these algorithms are based on power control, which is out of the scope of our treatment, they are not directly comparable with our solution.

More recently, an algorithm specifically developed for voice multicasting with the purpose of minimizing NFN was proposed in [24]; however, this protocol does not address the multiple-source case.

Minimizing the depth of each destination from the nearest source ideally minimizes the maximum forwarding delay. It is harder to argue about queuing delay, or throughput under imperfect channels, with medium access limitations; however, these are harder to model as they require information about traffic patterns and channel state processes. But realistic performance metrics, such as Delivery Ratio, Average Delay and Maximum Delay, are also studied through a real-time event-based ns3 simulation environment. In addition, issues such as resilience to link failures, distributed implementation, messaging overhead and computational complexity will be addressed as primary concerns in the development of SWIM, which is intended for real-life implementation. The NFN properties of SWIM will also be studied. NFN, in contrast to tree depth, is more directly related to the transmission energy efficiency, rather than delay.

While a full experimental comparison, including realistic channel models and packet arrival models, requires a separate study, one can readily make a conceptual comparison between two well-known protocols and our proposal as follows:

(i) Multicast *ad hoc* on demand distance vector (MAODV) forms a 'shared tree', that is, one tree connecting the source(s) with the multicast destinations, without explicitly optimizing tree depth by taking advantage of the Wireless Multicast Advantage.
(ii) On-demand multicast routing protocol (ODMRP) on the other hand, forms a mesh rather than a tree: redundant routes are kept for reliable transmission in case of a link failure. SWIM contains the redundant route property of ODMRP though the maintenance of alternative routes.
(iii) SWIM not only finds alternative routes from source(s) to destinations, but also finds alternative routes from the intermediate forwarding nodes to the destinations. This implies that SWIM finds at least as many alternative routes as ODMRP does, implying increased reliability.

Simulations have been done to compare MAODV and SWIM. It should be noted that the simulations in this paper are on a static network model and hence are not conclusive as to which is a better solution under mobility.

Our main aim is to find a tree, connecting a source (or set of sources) to a set of destinations, while finding the optimal depth for all destinations and keeping the NFN as low as possible.

We assume that there is a given set of links between nodes, forming a connected graph, and that each node knows about its one-hop neighbors on this graph. An important assumption is that the wireless multicast advantage will be exploited, that is, the MAC layer enables all one-hop neighbors to hear when a node makes a transmission.

3. THE ALGORITHM SWIM

We can now describe the algorithm SWIM (named after 'Source-initiated WIreless Multicast') in some detail. As introduced in Section 1, the main idea of SWIM is to first set hop count levels rooted at the source(s) and then to obtain a single tree using a greedy set cover algorithm. Hence, SWIM works in two phases: (I) level setting and (II) tree formation.

Phase (I) of SWIM could be skipped when the hop count information to the source(s) for all neighbors are already stored by nodes, for example, in proactively created and maintained routing tables of a distance-vector type unicast routing algorithm running in the background. Hence, when SWIM is run on top of such a unicast routing algorithm, it could start directly from Phase (II).

Downloaded from http://comjnl.oxfordjournals.org/ at University of California, San Diego on August 7, 2013

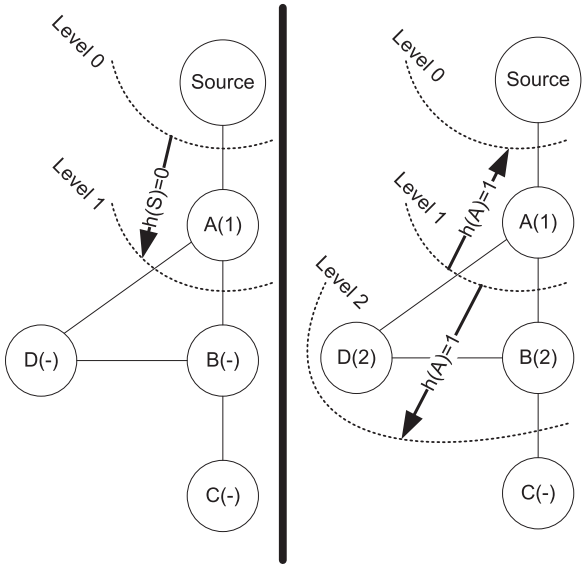


FIGURE 3. Starting from the source (left), nodes broadcast their hop number information(h) to their neighbors(right).

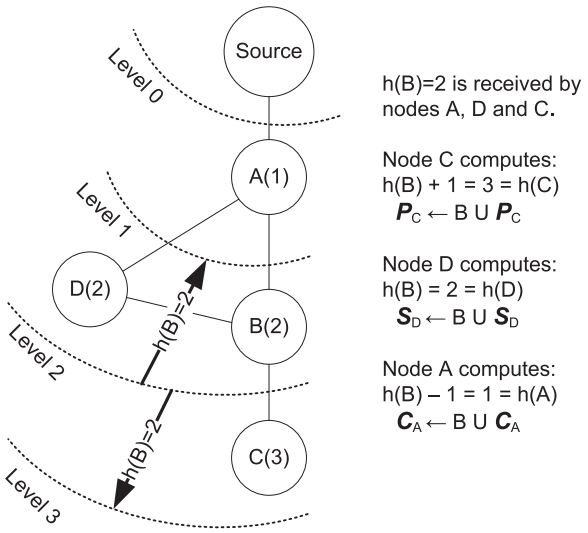


FIGURE 4. Using hop count information received from its neighbors, node B tabulates its neighbors with respect to their distances to the source.

For ease of exposition, we will focus on the single-source case and, later, describe the extension to multiple sources.

SWIM only depends on basic neighbor discovery (Figs 3 and 4), MAC broadcast and a correctly operating link layer that guarantees packet delivery as long as a link is operational. Troubleshooting when a link is broken will also be described as part of the routing protocol, for completeness. We assume that an automatic repeat request (ARQ) mechanism is used to ensure the delivery of control packets, which are used at the construction phase of the tree only.

Local Definitions:	**System Definitions:**
P : Parent Node Set	P_i : Parent Node Set of node i
S : Sibling Node Set	S_i : Sibling Node Set of node i
C : Child Node Set	C_i : Child Node Set of node i
N : Neighbor Node Set	N_i : Neighbor Node Set of node i
M_Seen : Set of Multicast nodes seen	h(i) = d : node i is d hops away from the nearest source

FIGURE 5. Notation used to describe SWIM.

3.1. Definitions

We now pin down the notation to be used to explain SWIM's operation. In the table in Fig. 5, local definitions (about information kept by an individual node) and system-wide definitions are separately listed.

3.2. SWIM Phase I: level setting

The main purpose of this phase is to form levels according to the hop distance information to the source[1]. This phase is initiated by the source[2]. Starting with the source, each node sends (via a MAC layer broadcast) to all of its neighbors its own distance (number of hops) from the source (Figs 6 and 7). The source broadcasts the message '$h(source) = 0$'. Neighbors of the source, upon reception, record their distance as '$h(node) = 1$' and send it as a message to their own neighbors. During this leveling, sequence numbers are checked at every node for packets received from each neighbor to avoid loops and usage of outdated information.

In general, a node will receive distance messages from each neighbor and will record its own distance to the source as the minimum of these. As the distance-setting process starts at the source itself and is based on the hop count, it is expected to progress in 'levels': with the source being on level 0, its one-hop neighbors being on level 1, etc.

3.2.1. Ensuring correct distributed operation

Considering a distributed system with real-life issues, such as excessive link delays and occasional link failures, it should not be assumed that all messages will be received in order of level, or even in order of generation even from the same sender. To ensure correct distributed operation, sequence numbers are used in the implementation of SWIM: older messages received from a given neighbor are discarded in favor of up-to-date ones (see the SWIM pseudo-code in Fig. 8). Similarly, if, because of excessive delays, a node receives such an update from a neighbor after it has already moved on to the next state, it goes back to the

[1]The generalization to multiple sources will follow.

[2]We view the algorithm as source-initiated without loss of generality. When the session request actually originates at a client, this request can be conveyed to the source along a unicast route, following which the source initiates a multicast session as described here.

Downloaded from http://comjnl.oxfordjournals.org/ at University of California, San Diego on August 7, 2013

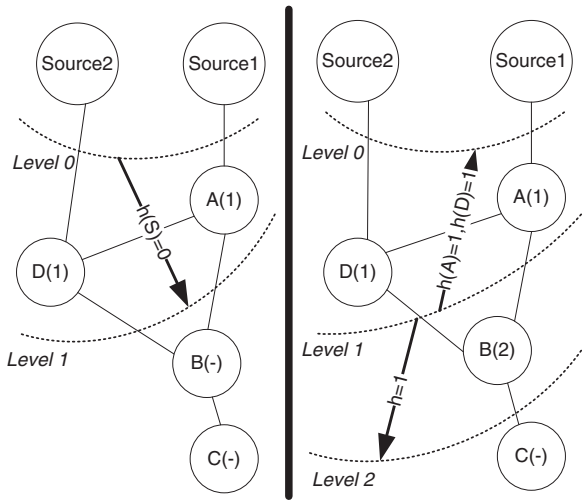


FIGURE 6. Starting from the sources, every node broadcasts its hop distance to the nearest source, to its neighbors.

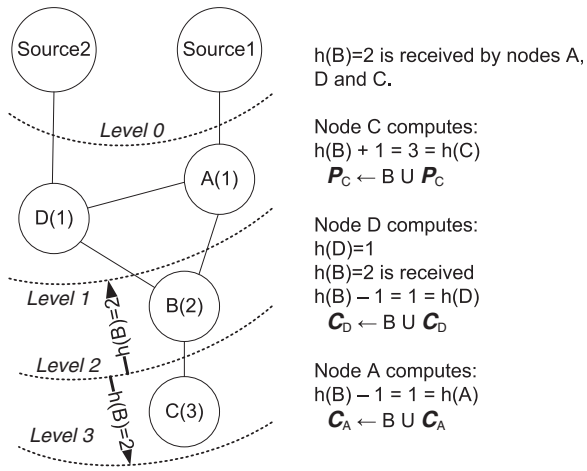


FIGURE 7. Nodes tabulate their neighbors using the hop number information received.

previous state. The counter measure for total link failure is not described in detail, but the idea is simple: if a total link failure happens, the node at the child-end of it will inform neighbors of this through the neighbor discovery procedure.

3.2.2. Neighbor hierarchy

Each node will categorize its immediate neighbors into three groups: neighbors belonging to the immediately higher level (neighbors that are one hop further from the source) are recorded as 'children', those one hop closer to the source as 'parents' and those at the same level as 'siblings' (see Fig. 9 and 10).

3.2.3. Multiple sources

In the multiple-source case, all sources start the process independently. Nodes that receive distance messages simply choose the smallest distance value received so far. Note that nodes can simply keep one distance value for themselves, regardless of which source this distance is from, since to each node its immediate source is its parent.

This process converges, provided that there is a finite number of link failures during this phase, and that these failures do not leave a disconnected network. Once all the child–parent designations are complete, we have a directed connected graph (with links pointing from parent to child) that contains the multicast group. The messages in this phase need to be sent with an ARQ mechanism.

3.3. SWIM Phase II: tree formation

In the beginning of this phase, information about the *Multicast Nodes Seen in Higher Levels* are sent to all nodes, starting from the leaves (nodes furthest from the source) all the way to the source. The pseudo-code for this is given in Fig. 11.

In the tree formation phase, the goal is for each node to tell its children to keep a selection out of all outgoing links, and eliminate the rest. This is related to the problem of choosing a *set cover* in the following way: suppose node a has three children, b, c and d. At the end of Phase I, a knows that b can reach $M_b \subset M$ through its own downstream nodes, c can reach $M_c \subset M$ and d can reach $M_d \subset M$. Suppose that $M_b \cup M_c = M_a = M_b \cup M_c \cup M_d$; that is, between themselves, b and c cover all possible multicast nodes that a is supposed to reach. In this case, a could simply tell d it no longer needs to relay the multicast packets coming from a, thus the link from a to d is taken out of the routing graph, *i.e. pruned*.

Then, the minimum number of child nodes that node a needs to keep in the routing tree, while reaching all elements of M_a, is the solution of a minimum set cover problem. It is well known that the minimum set cover is an NP-complete problem. Fortunately, there is a well-known greedy set cover algorithm (see, for example, [25]) that is at most $\log(d)$ away from optimal, where d is the size of the largest set.

A greedy set cover is illustrated by an example in Fig. 12.

In summary, here is how the routing tree is obtained: starting at leaves, every client lets its parent node know the subset of the multicast group that it can reach through its children. Once this process has progressed to the source, the source now has information on which multicast group members it can reach via which child (focusing on the single-source case for now). The source runs the first greedy set cover and assigns its children, c_i, multicast group subsets M_{c_i} to cover. For all i, child i then runs a greedy set cover among its own children to cover M_{c_i}. This progresses through the graph until no node which has an assignment to cover remains; at that point, all multicast nodes have been covered, and the pruned graph is a tree. A pseudo-code of Phase II appears in Fig. 11. The messages in this phase need to be sent with an ARQ mechanism.

It is straightforward to extend the tree formation phase to the multi-source case: once all sources have gathered information

Downloaded from http://comjnl.oxfordjournals.org/ at University of California, San Diego on August 7, 2013

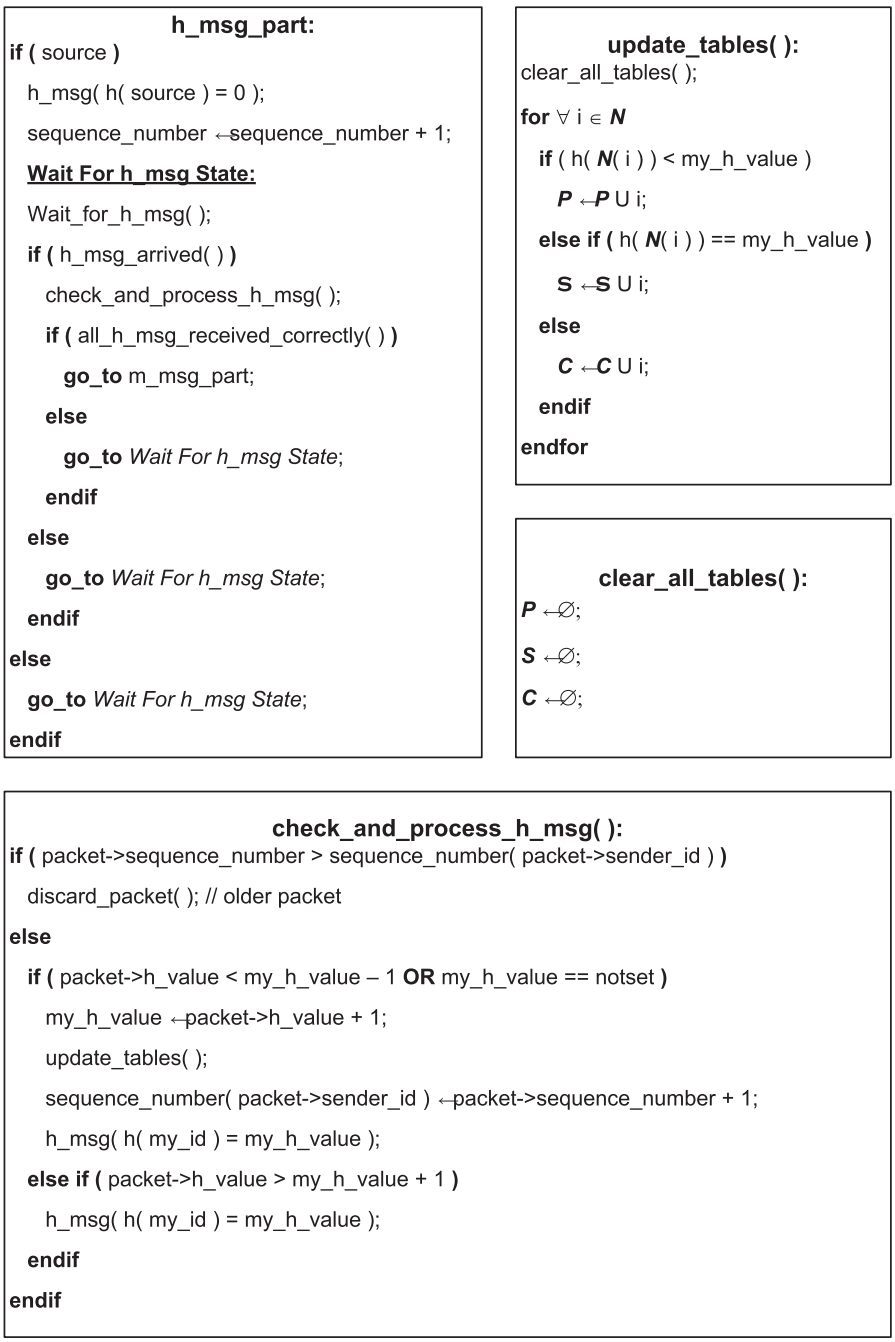


FIGURE 8. Abbreviated algorithm for SWIM Phase I. Essential parts of both the main routine, h-msg and the local subroutines are provided. Note that the sequence number of the last accepted message from any neighbor is stored, to be compared with the sequence number on the next packet (if any) received from that neighbor.

about which multicast clients they can reach, sources then share this information among themselves. One of the sources is selected to run the Prune function and assign other sources with multicast group subsets, after which the sources each form independent multicast trees. By construction of child–parent relationships, multicast trees formed by two different sources will be disjoint.

4. CORRECTNESS AND COMPLEXITY OF SWIM

Clearly, every element of the multicast group M has at least one directed path to the source(s). Furthermore, every link on this directed path is terminated by a parent and a child node on either end. By construction, Phase I of SWIM will simply identify individual shortest paths to each destination, from the nearest

Downloaded from http://comjnl.oxfordjournals.org/ at University of California, San Diego on August 7, 2013

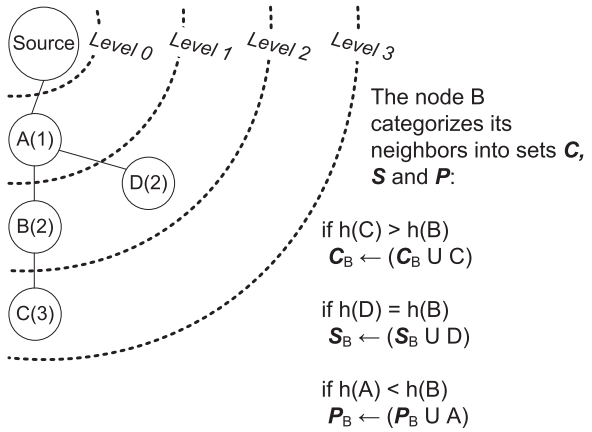


FIGURE 9. Parent, child and sibling designations of a node in the single-source case.

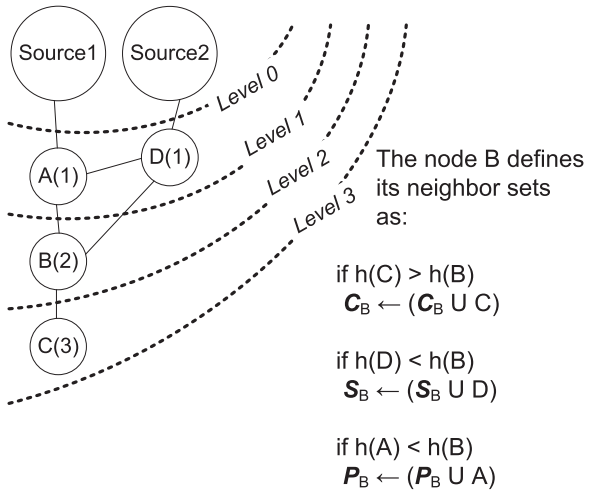


FIGURE 10. Parent, child and sibling designations made by a node in the multiple-source case. Distance $h(\cdot)$ is defined as the distance to the nearest source. Note that $h(\cdot)$ does not carry source ID information. This does not cause a problem, as all nodes see their parent(s) as the immediate source, anyway.

source node. The union of these paths is not necessarily a tree. However, Phase II will ensure that what remains after pruning is a tree (or a separate tree for each source). This can be proved by contradiction: suppose that node X is receiving data from two different parents. Then, each parent must be sending data to a set of multicast nodes that is reached through X. However, this implies that both parents are able to reach these multicast group members, hence the Tree Formation step would have picked exactly one of those parents to reach X. Hence, X will have only one parent in the final graph, and as this is true for all nodes (and parents are of a strictly lower level than children,), there can be no loops.

4.1. Computational complexity of SWIM

The process with the highest order computational complexity is the greedy set cover, which is $O(n^2)$ in terms of the number of sets n [25]. In SWIM, the sets correspond to subsets of the multicast group seen by each child. The number of children of a node cannot exceed the total number of nodes N. So, an upper bound on the computational complexity for a node is $O(N^2)$. Accounting for all nodes, we reach the conclusion that computational complexity scales at most as fast as $O(N^3)$.

However, an extensive set of simulations confirm that average computational complexity (for unicast, multicast and broadcast cases) is in fact $O(N^2)$ (see Fig. 13). In the simulations, computational complexity was computed by counting the number of assembly level instructions for every network instance over a sufficiently large number of instances for the running average of this value to stabilize. This was repeated for various network sizes. Nevertheless, note that the counterexample in Fig. 14 shows that complexity can indeed be $O(N^3)$.

4.2. Messaging overhead in SWIM

Messaging in SWIM is done for the accomplishment of three tasks: the first task is categorization of neighbors as parent, sibling and child. In nominal operation (ignoring link failures), each node makes a single constant-sized packet transmission (all three types of neighbors can receive this packet due to the use of MAC-layer broadcast). Considering all nodes in the network, this task entails N messages in total.

The second task is reporting the 'multicast clients seen' to the parents. Each node that is not a source, sends this information exactly once, yielding a total of $N - S$ transmissions, where S is the number of sources.

The third task entails each node on the Multicast Tree sending to its children the list of Multicast Clients to be reached. The number of messages sent off in this step is $T \leq N$, where T is the size of the multicast tree, *i.e.* NFN.

Overall, the total number of messages that are sent is upper-bounded by $2N + T - S$, which is $O(N)$. The ARQ overhead will only act as a constant scaling factor since the number of maximum retries is limited. An actual count of messages in our SWIM simulations has been plotted in Fig. 15.

The second component of messaging overhead is the length of messages passed. A particular solution for implementation of the messages described above is provided in Fig. 16, together with approximate lengths. The worst case message length is obtained when the nodes form a line, where the complexity will be upper-bounded by $O(N^2 log_2 N)$ due to the request packets.

5. DEPTH OPTIMALITY

By construction, SWIM forms a multicast tree on which the depth of any destination node is minimal. Hence, SWIM is

Downloaded from http://comjnl.oxfordjournals.org/ at University of California, San Diego on August 7, 2013

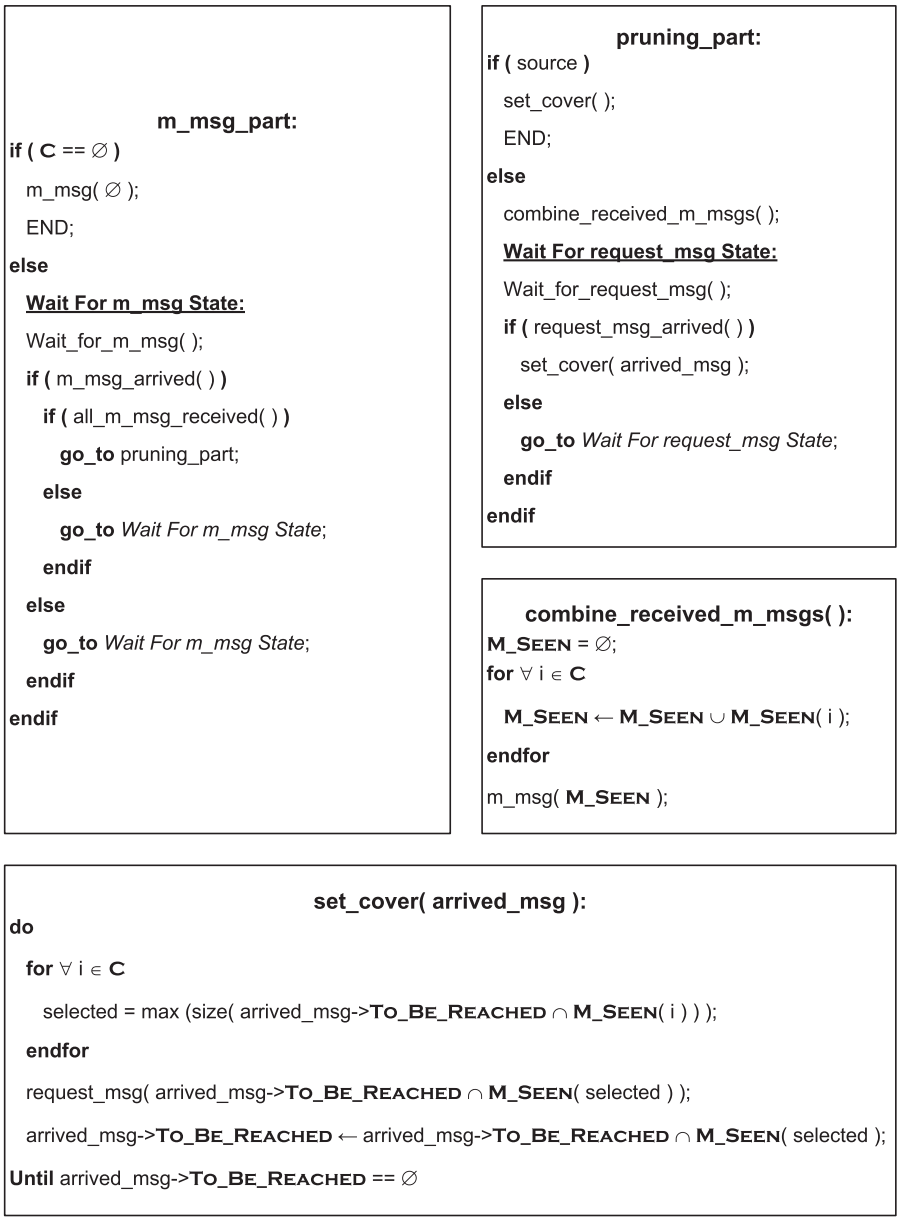


FIGURE 11. Pseudo-code for SWIM Part II: the information-gathering and tree formation part.

optimal in terms of minimizing the hop count of any node from the source, and consequently the maximum depth (that is, tree height). This is made precise in the following proposition:

LEMMA 5.1. *On the wireless multicast tree computed by SWIM with respect to a given source, the depth of every node is minimal.*

Proof. Consider a graph with a multicast group M and source s, and let T be an arbitrary wireless multicast tree rooted at s covering M. Let the depth of multicast node i on T be d_i. This means there is a neighbor of node i that is within distance $d_i - 1$ of the source. Node i does not enter phase II of SWIM before it has heard the h_j announcements from all its neighbors $\{j\}$, upon which it sets its h_i field to one larger than the smallest of the h_js. Therefore, at the start of Phase II, $h_i \leq d_i$. As phase II operates in levels, and node i will be on level h_i, it will have depth exactly h_i, which is smaller than or equal to d_i. □

It follows by a similar argument that in the multiple-source case SWIM minimizes the hop distance of each destination from the nearest source.

Downloaded from http://comjnl.oxfordjournals.org/ at University of California, San Diego on August 7, 2013

Step 1:

Neighbors:	N1	N2	N3	N4	N5
Multicast Clients Seen by Each Neighbor:	M1,M2, M3	M2	M1,M4	M1,M3	M4

Select the Neighbor with the largest set: N1
Send N1 the list to reach: M1,M2,M3
Delete M1,M2,M3

Step 2:

Neighbors:	N1	N2	N3	N4	N5
Multicast Clients Seen by Each Neighbor:	∅	∅	M4	∅	M4

Select the Neighbor with the largest set: N3(could pick N3 or N5 arbitrarily)
Send N3 the list to reach: M4
Delete M4

FIGURE 12. Greedy set cover example: at each step, the largest set is selected. The corresponding child is assigned to the list of Multicast Clients in this set. These multicast clients are then deleted from the sets and the process is repeated.

6. THE PERFORMANCE OF SWIM

The performance of the algorithm will be studied with respect to three different aspects. As a way of judging the performance of SWIM, it has been simulated alongside MAODV, which is one of the most competitive multicast algorithms, and other multicast algorithms reported in the literature. First, the delay

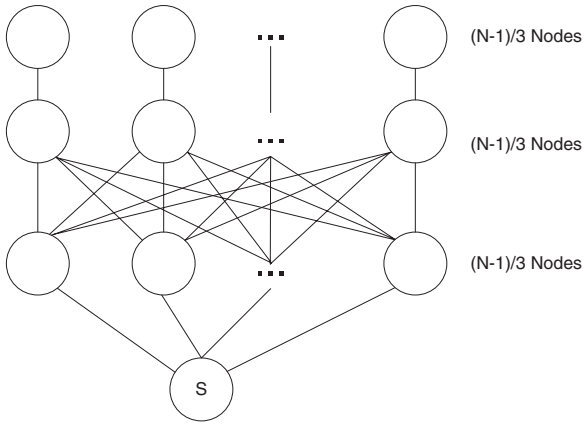


FIGURE 14. An example exhibiting $O(N^3)$ computational complexity: the first hop neighbors of the source need to run the Greedy Set Cover algorithm with complexity $O(N^2)$. Since $(N-1)/3$ nodes run the algorithm, the total computational complexity is $O(N^3)$.

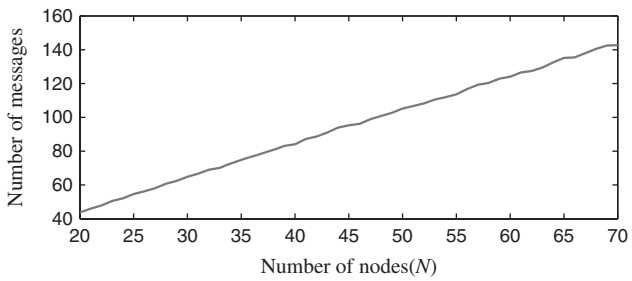


FIGURE 15. Average number of messages that have been passed to build the multicast tree, for various network sizes. For each network size, SWIM was run on 10 000 randomly generated network instances, for the largest possible multicast group in each instance, which is all nodes.

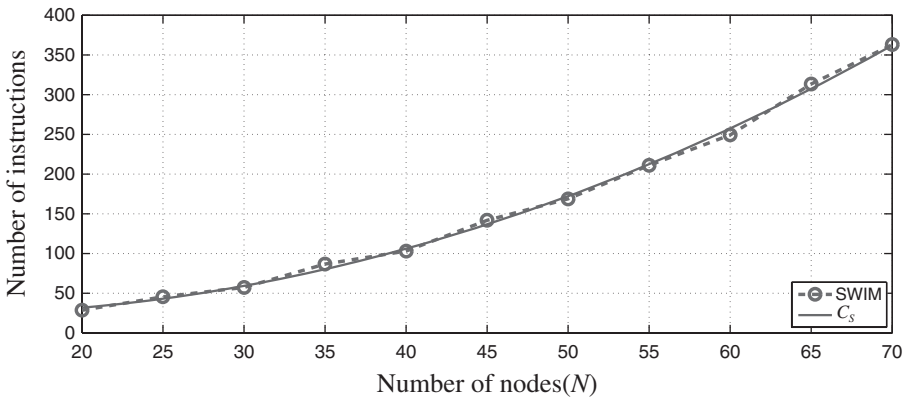


FIGURE 13. $C_S(N) = 0.096N^2 - 2.1N + 35$, is a curve-fitting approximation of the complexity curve observed from the simulation. Computational Complexity Simulation results show that the average computational complexity of SWIM is $O(N^2)$ experimentally.

Downloaded from http://comjnl.oxfordjournals.org/ at University of California, San Diego on August 7, 2013

h_msg Structure : Type00			
Type	**Source address**	**H_value**	**Sequence Number**
2 bits	*$log_2(N)$ bits*	*$log_2(N)$ bits*	*$log_2(N)$ bits*

m_msg Structure : Type01			
Type	**Source address**	**M_Count**	**Seen M addresses**
2 bits	*$log_2(N)$ bits*	*$log_2(N)$ bits*	*$log_2(N)$ bits * M_Count*

request_packet Structure : Type10			
Type	**Destination address**	**M_Count**	**Seen M addresses**
2 bits	*$log_2(N)$ bits*	*$log_2(N)$ bits*	*$log_2(N)$ bits * M_Count*

FIGURE 16. Description of the messages passed by SWIM for all three tasks in Phases I and II.

and delivery ratio performance is simulated in the real-time event-based simulation environment ns3[3]. In the second part, the Depth property of the algorithm is obtained. And in the final part, simulation results related to NFN performance are presented.

6.1. Delay and delivery ratio simulations

Delay (latency) is of significance in many practical applications. In the simulation results reported below, three main delay-related performance metrics have been measured: Average Delay, Maximum Delay and Delivery Ratio. Average Delay is average end-to-end latency over all packets, while the Maximum Delay is the average over all network instances (topologies) of the maximum measured latency over all end-to-end transmissions on each topology. The Delivery Ratio is the successful packet delivery rate averaged over all destination nodes and topologies.

The simulation was performed in ns3, and the setup uses 802.11b WiFi MAC protocol in an *Ad hoc* mode with a 1 Mbps bandwidth. The total number of nodes in the network N is increased from 40 to 70. For each case, the results are averaged over 100 topologies. The source generates a user datagram protocol (UDP) flow with a rate of 100 kb. All the three algorithms are run on the same topology for 150 s and there is no mobility. The propagation loss model is the *Log Distance Model*. The results are presented in the rest of this section.

SWIM is observed to have lower multicast latency than MAODV as expected from its depth minimality. As for the successful delivery ratio, the depth optimality property, which is the strength of SWIM leading to low delay, is also its weakness for delivery ratio: SWIM inherently tries to maximize the number of active outgoing links per a forwarding node. But the larger the number of links, the more points of failure. Hence, SWIM seems to trade-off throughput in favor of minimizing

[3] www.nsnam.org

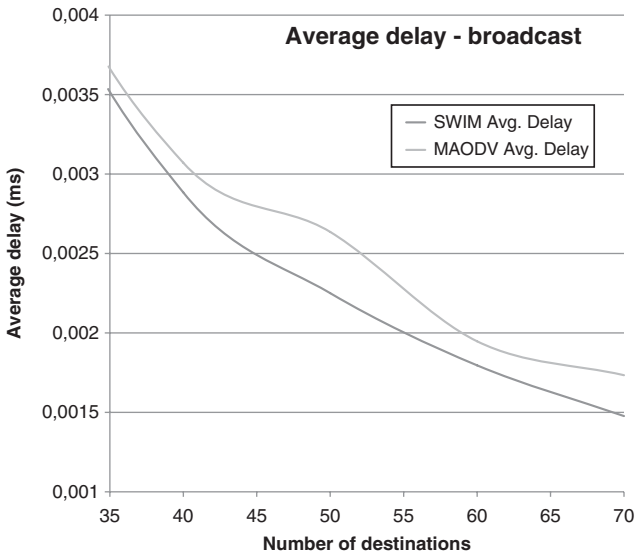


FIGURE 17. Average Delay Comparison of SWIM and MAODV for the broadcast case. SWIM exhibits lower average delay, as expected from its depth optimality.

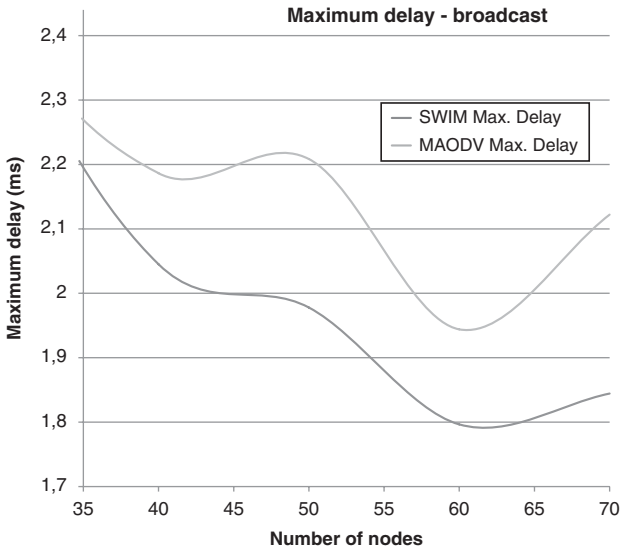


FIGURE 18. Maximum Delay Comparison of SWIM and MAODV for the broadcast case.

delay, and the comparisons are consistent with this (Figs 17, 18 and 19).

However, there is a mechanism to increase throughput in a SWIM implementation: if increased reliability and higher throughput is desired (for example, when link packet success rates are low), SWIM can increase its success rate, by incorporating alternative redundant paths. It should be noted that alternative paths will cause no additional delay, but they may lead to increased energy dissipation, as more nodes will be transmitting.

Downloaded from http://comjnl.oxfordjournals.org/ at University of California, San Diego on August 7, 2013

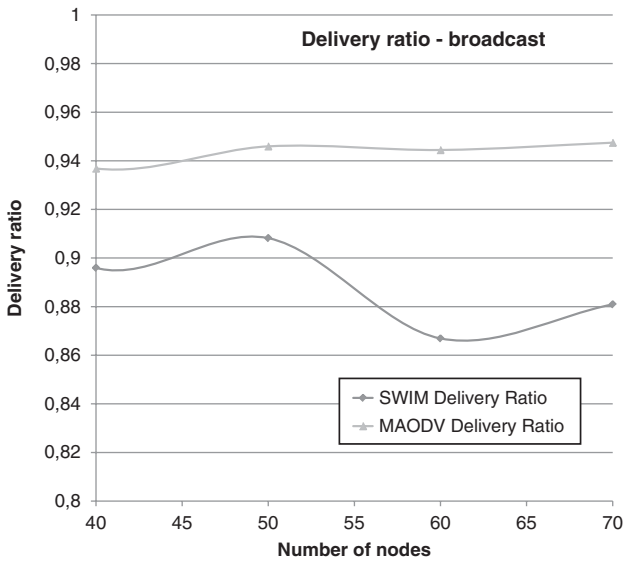


FIGURE 19. Delivery Ratio Comparison of SWIM and MAODV for the broadcast case.

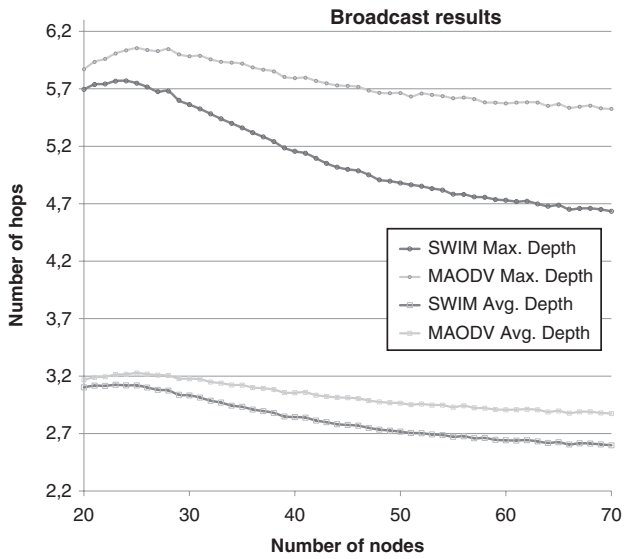


FIGURE 20. Broadcast Simulation results for SWIM and MAODV: maximum as well as average depth over all destinations is plotted as the network size (number of nodes) increases from 20 to 70. SWIM's superiority in terms of the average as well as the maximum depth is exhibited.

6.2. Tree depth simulations

Attaining low depth is where SWIM is expected to have the strongest performance. Comparisons between MAODV and SWIM have been made with respect to Maximum Tree Depth and Average Tree Depth. The algorithms have been run on the same random topologies, generated according to the following simulation settings:

(i) Nodes are placed independently according to the uniform distribution in a unit square region of side length 1.
(ii) Transmission range (the maximum distance between two nodes, such that they are connected) is 0.286[4].

The algorithms were run on the same 10 000 independent randomly created topologies under the first two of the scenarios below. Only SWIM was run for the third scenario:

(i) Broadcast Simulation: The number of nodes (N) is varied from 20 to 70.
(ii) Multicast Simulation: $N = 70$, while the number of multicast clients (m) varies between 1 and 69.
(iii) Multi-Source Simulation: N is increased from 20 to 70 under a broadcast scenario. Results are given for 1, 2 and 3 sources.

The results are reported below (Fig. 20).

[4]This number has been picked because it is the same value that was used in [6], and it corresponds to nodes having a transmission radius of 100 units located in a square area with 350 unit sides.

6.3. NFN simulation

The final performance aspect we consider is NFN, which is roughly related to energy efficiency. Simulation results of SWIM and MAODV will be compared with results obtained under identical simulation settings with the low NFN algorithms

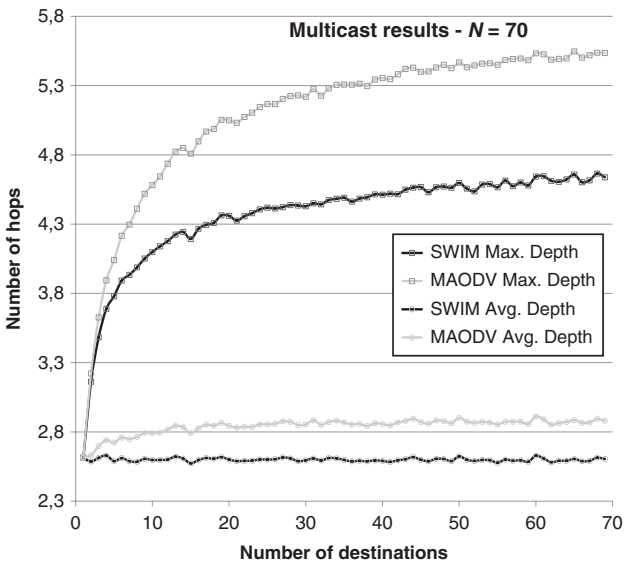


FIGURE 21. Multicast Simulation results for SWIM and MAODV: number of nodes N is fixed at 70, while m increases from 1 to 69. SWIM's superiority in terms of the average as well as the maximum depth is exhibited.

Downloaded from http://comjnl.oxfordjournals.org/ at University of California, San Diego on August 7, 2013

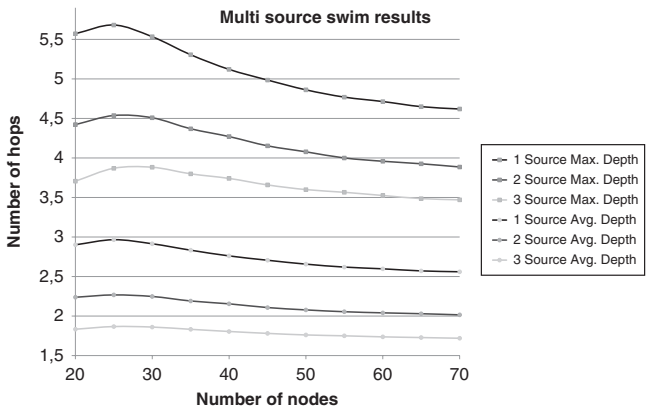


FIGURE 22. Multi-Source simulation results for the broadcast case. SWIM takes advantage of the availability of multiple sources to reduce depth.

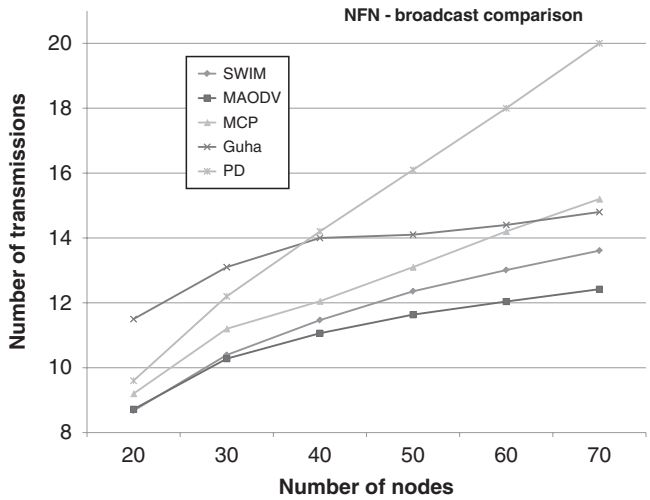


FIGURE 23. SWIM is compared with various algorithms in the literature. Although not the best, SWIM performs better than most of the given algorithms, while being close to MAODV. But since delay was the main aim, energy efficiency is only a secondary concern.

reviewed in [6]. These settings are as follows:

(i) Nodes are placed independently according to the uniform distribution in a unit square region of side length 1.
(ii) The transmission range is 0.286.
(iii) The Number of nodes is changed from 20 to 70.
(iv) Only the broadcast scenario is considered.

The results of the NFN simulation are summarized in Figs. 21, 22 and 23.

7. USE OF AN EXISTING ROUTING TABLE

This section shows that when SWIM runs on an existing proactive routing algorithm that stores and propagates hop count or unit-cost distance vector information, it can use the information stored in routing tables to skip the first phase and directly start tree formation. We assume that any node constructs its table from received neighbor information and the routing algorithm can be modified so that the information received from *all* neighbors (rather than ones with the best distance to each destination) are saved, and that consequently, each node records the hop count of any node to any other node. Conceptual operation of this alternative implementation of the first phase of SWIM is described as follows:

7.1. h-message leveling

The *h-Message* Leveling can be easily sidestepped when routing table information as described above is available: as any node can compute all of its neighbors' distances to the source as well as its own, it can construct the neighbor tables given in Section 3.2.

7.2. Relaying of the information: destinations seen

After levels with respect to the source have been formed, each node i is supposed to send up to its parent its *Seen Destinations*. These are the destination nodes that are further down in the multicast tree from i, in other words, destination nodes that are descendants. We claim that this information can be computed by each node, by referring to the stored hop count vectors. Node i knows that it is on the shortest path (hence 'sees') destination D if D is a child of i itself, or a child of a child j of i. The first condition is provided by the height information, and the latter condition can be checked with the following simple condition: node j is on a shortest path to destination D with respect to source S if

$$\begin{aligned} &\mathrm{d(S,i)} = \mathrm{d(S,j)} - 1, \text{ and} \\ &\mathrm{d(i,j)} = 1, \text{ and} \\ &\mathrm{d(i,D)} = \mathrm{d(j,D)} + 1 \end{aligned}$$

The first and second conditions ensure that j is a child of i, and the last guarantees that j is on the shortest path from i to D. This is made precise in the lemmas 7.1 and 7.2 below.

LEMMA 7.1. *Let $d(i, j)$ denote the shortest distance between i and j. If $d(S, i) = 1$ AND $d(S, D) = d(i, D) + 1$, then i is on one of the shortest paths from S to D* (Fig. 24).

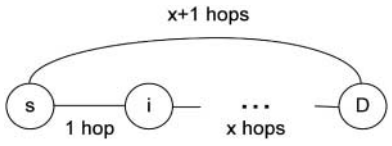


FIGURE 24. Illustration of the proof of Lemma 7.1: i is on the shortest path from S to D.

Downloaded from http://comjnl.oxfordjournals.org/ at University of California, San Diego on August 7, 2013

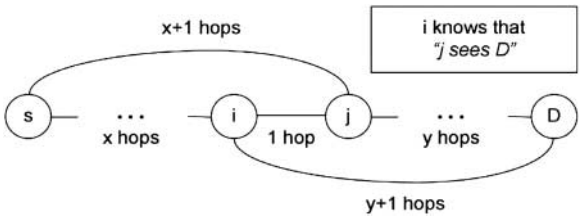


FIGURE 25. Illustration of the proof of Lemma 7.2:i and j are on the shortest path from S to D.

Proof. Let $d(i, D) = x$; then the shortest distance between S and D is $x + 1$. Note that S can reach D over i in $x + 1$ hops. Then i is on a shortest path. □

LEMMA 7.2. *If, for any node i and its neighbor j, $d(i, D) = d(j, D) + 1$ AND $d(S, j) = d(S, i) + 1$, then both i and j are on the shortest path from S to D* (Fig. 25).

Proof. Apply Lemma 7.1 on (S, i, j) and (i, j, D). i is on the shortest path from S to j. Also j is on the shortest path from i to D. Concatenating the shortest path from S to i, with the shortest path from i to D, we find that i and j are on a shortest path from S to D. □

Using Lemma 7.2, any node (i) can compute, which of its neighbors (j) can reach a destination D, by checking the conditions of the Lemma, hence the Tree Formation part of the algorithm can start execution at the source without requiring the exchange of any 'h' or 'destinations seen' messages.

8. EXTENSION FOR DYNAMIC NETWORKS

Changes in network topology, such as link failures, could of course disconnect the tree computed by SWIM, unless the tree is updated with sufficient frequency with respect to the topology dynamics. As a remedy to handle dynamics of network topologies, an extension to SWIM is proposed in this section.

The main idea is to employ topology and tree updates with an appropriately chosen periodicity. To update the topology information in each node in the network, the routing table solution given in Section 7 is used. To update the *tree*, the source periodically restarts SWIM to compute a new tree. All nodes currently on the multicast tree expect to receive periodic updates. Any forwarding node that does not receive any forwarding request updates over a duration of one period concludes that it is no longer on the multicast tree and stops forwarding any multicast packets it happens to receive. As topology information is updated separately, the number of messages to be sent for each tree update is only the NFN.

The update rates can be changed according to the mobility level in the network. Changes in the set of multicast destinations are similarly handled by the periodic tree updates.

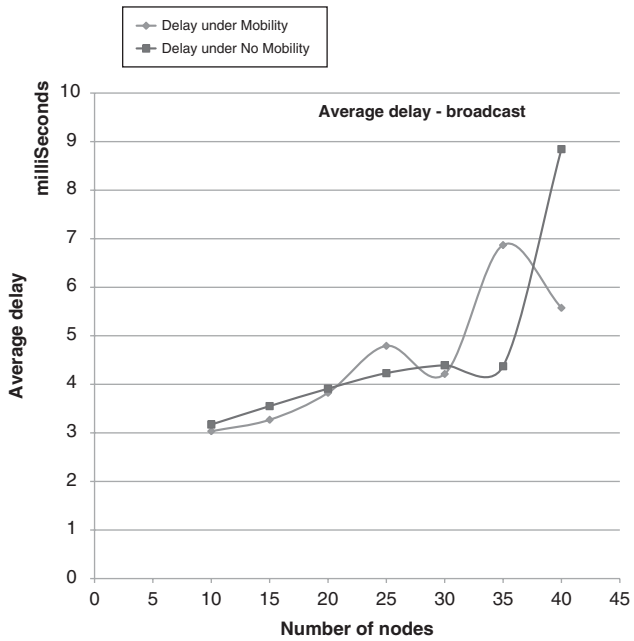


FIGURE 26. Effect of mobility on performance: the effect of mobility on the algorithm performance is seen to be minimal in terms of average delay when the Neighbor and Tree Update Rate pair is selected as (1 s, 1 min).

The effect of mobility on performance has been observed by simulations performed on ns3. The mobility model is the Brownian Motion Model(2d Random Waypoint) with node speed distributed between 3 and 5 m/s. 802.11b WiFi MAC with 1 Mb is used for a UDP traffic of 100 kb. The number of nodes in the network has been varied from 10 to 40, while the multicast group size has been fixed at 9. The performance degradation under mobility can be made arbitrarily small by making the updates sufficiently frequent. Of course, the price paid for this is extra overhead. Consequently, there is a trade-off between good performance in a dynamic environment and overhead, which we study next. First, we find out that delay is virtually unaffected by mobility if one pays a 10 percent overhead (see Fig. 26). Here, overhead is defined as the proportion of the total length (in bytes) of all table and tree formation and update messages sent during simulation runtime among all messages sent, including data.

There are two update rate parameters that affect the dynamic behavior of SWIM:

(i) Neighbor (topology) update rate.
(ii) Tree update rate.

In the square region of 2500 m × 2500 m in our simulations, according to the mobility model, it takes ∼12 min to cross the largest distance in the field. This duration will be our benchmark to select different update periods. In the simulation results

Downloaded from http://comjnl.oxfordjournals.org/ at University of California, San Diego on August 7, 2013

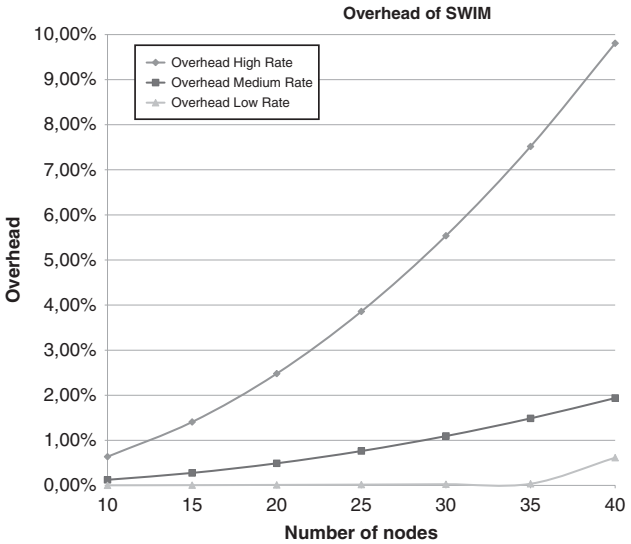


FIGURE 27. Overall protocol overhead of SWIM under three different update rates: low, medium and high. The pair (topology update period, tree update period) changes from low rate (3 min, 30 min) to medium rate (30 s, 6 min), to high rate (1 s, 1 min).

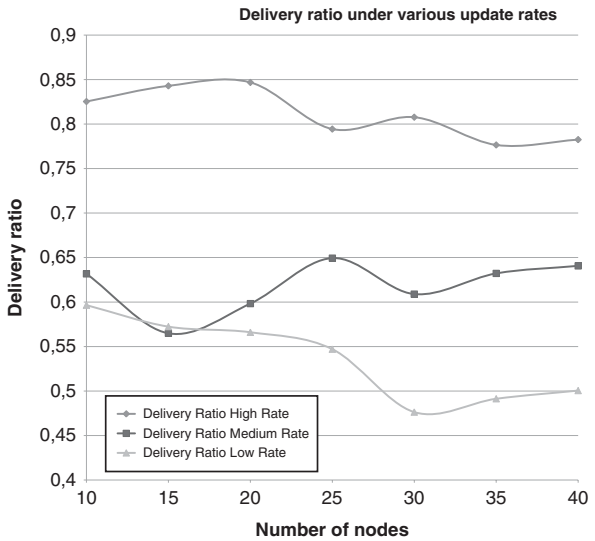


FIGURE 28. Simulation results showing how the delivery ratio of SWIM improves as the pair (topology update period, tree update period) changes from low rate (3 min, 30 min) to medium rate (30 s, 6 min), to high rate (1 s, 1 min).

reported below, the following three different rate pair choices have been used:

(i) Low Rate: neighbor update period = 3 min, tree update period = 30 min
(ii) Medium Rate: neighbor update period = 30 s, tree update period = 6 min
(iii) High Rate: neighbor update period = 1 s, tree update period = 1 min

Step 1:

Neighbors:	N1	N2	N3	N4	N5
Multicast Clients Seen by Each Neighbor:	M1,M2, M3	M2	M1,M4	M1,M3	M4

Discard previously Selected Neighbors:N1,N3

Step 2:

Neighbors:	N2	N4	N5
Multicast Clients Seen by Each Neighbor:	M2	M1,M3	M4

Select Neighbor with largest set: N4
Send N4, the list to reach: M1,M3
Delete M1,M3

Step 3:

Neighbors:	N2	N4	N5
Multicast Clients Seen by Each Neighbor:	M2	∅	M4

Select Neighbor with largest set: N2
Send N2, the list to reach: M2
Delete M2

Step 4:

Neighbors:	N2	N4	N5
Multicast Clients Seen by Each Neighbor:	∅	∅	M4

Select Neighbor with largest set: N5
Send N5, the list to reach: M4
Delete M4

Main Route: N1,N3
Alternative: N2,N4,N5

FIGURE 29. Alternative Route Selection: discard the neighbors selected by the Greedy Set Cover algorithm, which is the main route. Then execute a greedy set cover on the remaining neighbors. If the remaining nodes are sufficient to cover the destination set, they may be used as a redundant path to the same destination nodes.

Downloaded from http://comjnl.oxfordjournals.org/ at University of California, San Diego on August 7, 2013

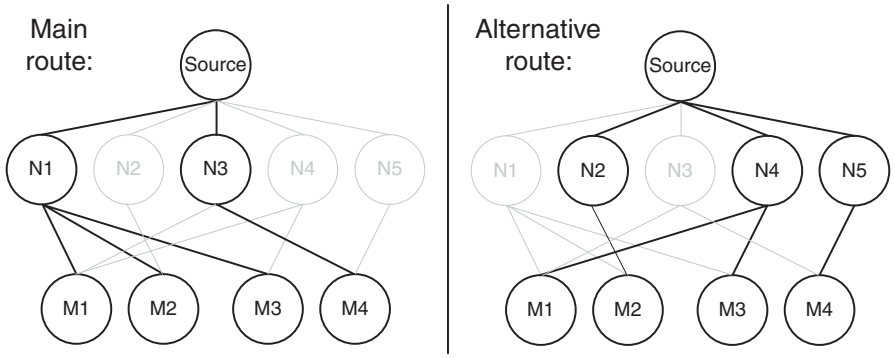


FIGURE 30. Alternative route selection: the graph representation of example Fig. 29. The Main Route is shown on the left and the Alternative Route is shown on the right. In case of a link failure or to increase throughput, the alternative route may be selected.

The results indicate that a reasonable level of overhead in SWIM (Fig. 27) allows quite good adaptation to link failures (Fig. 28).

The effect of mobility on delay has been explored for the high update rate pair only, as for lower rates, there is a non-negligible fraction of dropped packets, and computation of delay over successful ones will be misleading. The simulation result is given in Fig. 26.

9. GENERATION OF ALTERNATIVE PATHS

In the presence of mobile clients, or otherwise significantly fluctuating wireless link quality, some applications (such as streaming multimedia) may be interested in maintaining alternative routing paths in the network layer, so that, when a path momentarily fails, data can continue to flow on an alternative one without interruption. In some cases, applications may get higher throughput through the use of alternative redundant paths.

In this section, we propose an addition to the Tree Formation phase of SWIM that generates an alternative to the original path that, loosely speaking, has minimal overlap with the first path. The main idea is to select, at each step, an alternative set cover that complements the original one. The number of alternative trees that can be produced certainly depends on the topology. The alternative route algorithm is illustrated by Figs. 29 and 30.

Since the usage of alternative paths or the number of alternative paths to be used is a user decision, the user may decide to increase throughput and reliability by using more alternative paths but with a trade-off of energy dissipation. Developing an algorithm to select the number of alternative paths based on a desired throughput goal could be a topic for future studies.

It should be noted that the usage of alternative paths does not affect the maximum or average node depths, thus leaving the forwarding delay unchanged. The trade-off is given from energy dissipation.

10. CONCLUSIONS AND FUTURE DIRECTIONS

We constructed a distributed multicast-tree generation algorithm, SWIM. This algorithm achieves an average complexity of only $O(N^2)$ in the number of nodes N. Despite this low-complexity, SWIM is depth-optimal and obtains a low-average number of forwarding nodes. The key reason for the good performance at low-complexity is that SWIM applies a competitive Greedy Set Cover algorithm at each level from source(s) to destinations. Moreover, the number of messages that need to be exchanged to build the tree is minimal. The power to select the usage of alternative paths increases the throughput and robustness in exchange for energy efficiency. We believe that SWIM offers an implementable solution that will effectively relieve bottlenecks at mesh network gateways, among other applications. With implementation for multimedia streaming in mind, a method is also developed for creating alternative routing trees. While this paper mainly focused on a static network scenario, a repair mechanism for handling topology changes and mobility has been proposed and simulated, with satisfying results under a reasonable amount of mobility. Other future directions include addressing the weighted-graph case (for when links have different qualities or costs) and generalizing the model to allow power control, and developing an algorithm to select the number of alternative graphs based on a given throughput goal.

FUNDING

TUBITAK Career Grant No: 106E119.

REFERENCES

[1] Awerbuch, B. and Azar, Y. (1995) Competitive multicast routing. *Wirel. Netw.*, **1**, 107–114.

[2] Xudong Wang, M.C., Edward Knightly and Ephremides, A. (2007) Guest editorial, a special issue on 'wireless mesh networks'. *Elsevier Ad Hoc Networks*, **5**, 649–651.

Downloaded from http://comjnl.oxfordjournals.org/ at University of California, San Diego on August 7, 2013

[3] Zheng, J. and Lee, M.J. (2007) A resource-efficient and scalable wireless mesh routing protocol. *Elsevier Ad Hoc Networks*, **5**, 704–718.

[4] Karp, R. (1972) Reducibility among combinatorial problems. In Miller, R. and Thatcher, J. (eds), *Complexity of Computer Computations*, pp. 85–103. Plenum Press.

[5] Winter, P. (1987) Steiner problem in networks: a survey. *Networks*, **17**, 129–167.

[6] Guha, S. and Khuller, S. (1996) Approximation Algorithms for Connected Dominating Sets. *ESA*, Lecture Notes in Computer Science, Vol. 1136, pp. 179–194. Springer.

[7] Andersen, D., Balakrishnan, H., Kaashoek, F. and Morris, R. (2001) Resillient Overlay Networks. *Proc. 18th ACM Symp. Operating Systems Principles*, October, Melbourne, FL, USA, pp. 131–145.

[8] Amir, Y., Danilov, C., Musaloiu-E.R. and Rivera, N. (2007) An Inter-Domain Routing Protocol for Multi-homed Wireless Mesh Networks. *IEEE Int. Symp. World of Wireless Mobile and Multimedia Networks WoWMoM 2007*, Helsinki, Finland, pp. 1–10.

[9] Aoun, B., Boutaba, R., Iraqi, Y. and Kenward, G.W. (2006) Gateway placement optimization in wireless mesh networks with qos constraints. *IEEE J. Sel. Area. Comm.*, **24**, 2127–2136.

[10] Langar, R., Bouabdallah, N. and Boutaba, R. (2008) A comprehensive analysis of mobility management in mpls-based wireless access networks. *IEEE/ACM Trans. Netw.*, **16**, 918–931.

[11] Banchs, A., Effelsberg, W., Tschudin, C. and Turau, V. (1998) Multicasting Multimedia Streams with Active Networks. *23rd IEEE Conf. Local Computer Networks (LCN98)*, Lowell, MA, USA, October.

[12] Carlos de Morais Cordeiro, H.G. and Agrawal, D.P. (2003) Multicast over wireless mobile ad hoc networks: present and future directions. *IEEE Netw.*, **17**, 52–59.

[13] Jia, W., Cheng, L. and Xu, G. (2002) Efficient Multicast Routing Algorithms on Mesh Networks. *Proc. 5th Int. Conf. on Algorithms and Architectures for Parallel Processing(ICA3PP'02)*, Beijing, China, pp. 110–117.

[14] Penttinen, A. (2006) Minimum Cost Multicast Trees in ad hoc Networks. *Proc. IEEE ICC 2006*, Istanbul, Turkey, pp. 3676–3681.

[15] Wong, J.L., Veltri, G. and Potkonjak, M. (2002) *System-Level Power Optimization for Wireless Multimedia Communication*. Springer US.

[16] Tavli, B. and Heinzelman, W.B. (2005) Mc-trace: multicasting through time reservation using adaptive control for energy efficiency. *MILCOM*, **4**, 2672–2678.

[17] Tavli, B. and Heinzelman, W.B. (2004) Mh-trace: multi hop time reservation using adaptive control for energy efficiency. *IEEE J. Select. Areas Commun.*, **22**, 942–953.

[18] Wieselthier, J.E., Nguyen, G.D., and Ephremides, A. (2000) On the Construction of Energy-Efficient Broadcast and Multicast Trees in Wireless Networks. *Proc. IEEE Infocom 2000*, Tel Aviv, Israel, 586–594.

[19] Wieselthier, J.E., Nguyen, G.D. and Ephremides, A. (2002) Distributed Algorithms for Energy-Efficient Broadcasting in Ad Hoc Networks. *Proc. IEEE MILCOM'02*, Oct, Anaheim, CA, USA, pp. 820–825.

[20] Li, F. and Nikolaidis, I. (2001) On Minimum-Energy Broadcasting in All-Wireless Networks. *Proc. 26th Annual IEEE Conf. Local Computer Networks (LCN 2001)*, Tampa, FL, USA, November, pp. 193–202.

[21] Min, M., Prokopyev, O. and Pardalos, P. (2006) Optimal solutions to minimum total energy broadcasting problem in wireless ad hoc networks. *J. Comb. Optim.*, **11**, 59–69(11).

[22] Egecioglu, O. and Gonzalez, T. (2001) Minimum-Energy Broadcast in Simple Graphs with Limited Node Power. *Proc. IASTED Int. Conf. Parallel and Distributed Computing and Systems*, Anaheim, Canada, pp. 334–338.

[23] Cagalj, M., Hubaux, J.-P. and Enz, C. (2002) Minimum-Energy Broadcast in All-Wireless Networks: Np-completeness and Distribution Issues. *Proc. 8th Annual Int. Con. Mobile Computing and Networking*, Atlanta, GA, USA, September, pp. 172–182.

[24] Raju, G.V., Reddy, T.B. and Murthy, C.S.R. (2007) A Near Optimal Localized Heuristic for Voice Multicasting Over Ad Hoc Wireless Networks. *ICC*, Glasgow, Scotland, pp. 1648–1653. IEEE.

[25] Johnson, D.S. (1973) Approximation Algorithms for Combinatorial Problems. *STOC '73: Proc. 5th Annual ACM Symp. Theory of Computing*, New York, NY, USA, pp. 38–49. ACM.

Downloaded from http://comjnl.oxfordjournals.org/ at University of California, San Diego on August 7, 2013