Available online at www.sciencedirect.com
ScienceDirect
Ad Hoc Networks 4 (2006) 709–723
Ad Hoc Networks
www.elsevier.com/locate/adhoc

OGHAM: On-demand global hosts for mobile ad-hoc multicast services

Chia-Cheng Hu [a,*], Eric Hsiao-Kuang Wu [b,1], Gen-Huey Chen [a]

[a] *Department of Computer Science and Information Engineering, National Taiwan University, No.1, Sec. 4, Roosevelt Road, Taipei, Taiwan, ROC*
[b] *Department of Computer Science and Information Engineering, National Central University, Chung-Li, Taiwan, ROC*

Received 12 December 2004; received in revised form 30 April 2005; accepted 30 August 2005
Available online 26 September 2005

Abstract

Recent advances in pervasive computing and wireless technologies have enabled novel multicast services anywhere, anytime, such as mobile auctions, advertisement, and e-coupons. Routing/multicast protocols in large-scale ad-hoc networks adopt two-tier infrastructures to accommodate the effectiveness of the flooding scheme and the efficiency of the tree-based scheme. In these protocols, hosts with a maximal number of neighbors are chosen as backbone hosts (BHs) to forward packets. Most likely, these BHs will be traffic concentrations or bottlenecks of the network and spend significant amount of time forwarding packets. In this paper, a distinct strategy is proposed for constructing a two-tier infrastructure in a large-scale ad-hoc network. Hosts with a minimal number of hops to the other hosts rather than those with a maximal number of neighbors will be adopted as BHs in order to obtain shorter multicast routes. The problem of determining BHs can be formulated with linear programming. BHs thus found have the advantages of shorter relay and less concentration. Besides, BHs are selected on-demand and can be globally reused for different multicast groups without flooding again. Simulation results show that the proposed protocol has shorter transmission latency, fewer control/data packets and higher receiving data packet ratios than other existing multicast protocols. Besides, the two-tier infrastructure constructed by the proposed protocol is more stable.

Keywords: Ad-hoc network; Distributed algorithm; Linear programming; Multicast

1. Introduction

With the advent of IP technologies and the tremendous growth in data traffic, novel applications that require multicast support are becoming more and more widespread. Another interesting recent development is the emergence of dynamically wireless ad-hoc networks to interconnect mobile hosts. In contrast to traditional cellular networks, Ad-hoc networks require no fixed infrastructure or centralized administration, and hosts must communicate one another via packet radios in a collaborated manner. Disaster recovery and automated battlefields are

* Corresponding author. Tel.: +886 2 23625336 407; fax: +886 2 23628167.
E-mail address: jjhwu@ksts.seed.net.tw (C.-C. Hu).
[1] Tel.: +886 3 4227151 4514; fax: +886 3 4222681.

1570-8705/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2005.08.003

two typical applications of ad-hoc networks. Recent advances in wireless telecommunication technologies and portable computing are continuing to drive the revolution towards large-scale networks and flexible new generation e-services.

Multicast communication enables a number of novel e-services, such as mobile auctions, game ranking update, audio/video conferencing and distribution of stock quotes. A multicast group in a mobile ad-hoc network (MANET) contains a special host (server) that is responsible for transmitting data packets to the other hosts (clients) in the same group. A typical example of multicast groups is a commander and his soldiers in a battlefield. The soldiers need to keep listening to the commander. There are other examples in which multicast groups need to be established.

Recently, several MANET multicast protocols have been proposed in the literature [1–6]. They can be classified into tree-based protocols [1–4] and mesh-based protocols [5,6]. Tree-based protocols build a tree for each multicast group, whereas mesh-based protocols create a mesh of hosts for forwarding packets between multicast members. For both protocols, adding a new member into an existing multicast group will cause the flooding of a join request message over the entire network. Further, they also trigger the flooding of control packets during the maintenance of the infrastructures. The flooding process is very time-consuming and bandwidth-consuming, especially, for a large-scale MANET.

To avoid the inefficiency of flooding, some routing protocols, e.g., Spine [7], CEDAR [8] and VDBP [9], and multicasting protocols, e.g., MCEDAR [10] and ADB [11], adopt two-tier infrastructures for a large-scale MANET. Some of the hosts, named backbone hosts (BHs for short), are responsible for managing the flooding, maintaining the infrastructures and determining multicast routes. These protocols all select BHs by finding dominating sets with a maximal number of neighbors.

Since most of packets are initiated and processed by BHs, a proper way to select BHs is crucial to a two-tier infrastructure. In MANETs, a host can transmit packets in omni directions, and so each transmission is a local broadcast. Since only one host is allowed to broadcast within a transmission range at a time, the BHs determined in [7–11] are likely to be traffic concentrations or bottlenecks of the networks, i.e., to consume more time in contending radio channels with their neighbors. They also suffer from frequent changes in highly mobile MANETs. Frequent changes of BHs adversely affect the performance of the networks.

In this paper, a novel multicast protocol, named on-demand global hosts for ad-hoc multicast (OGHAM for short), is presented for large-scale MANETs. OGHAM adopts a new methodology to construct a two-tier infrastructure on-demand for ad-hoc multicast services. Once the infrastructure for a particular multicast group is constructed, the selected BHs are globally available for the other ad-hoc multicast groups. Therefore, it is not necessary for follow-up multicast groups to flood again in order to construct additional infrastructures.

In OGHAM, a distinct strategy for selecting BHs is proposed, which minimizes the total number of hops to the other hosts. BHs thus found can determine shorter multicast routes than those determined in [7–11]. To obtain shorter routes is one of the major concerns in most of existing routing/multicast protocols. It can reduce the number of hosts participating in packet forwarding so as to lower bandwidth consumption and shorten transmitting latency from servers to clients. In particular, the issue of shorter routes must be carefully scrutinized for large-scale multicast services, because the BHs attached by servers transmit packets to multiple clients over the networks.

2. Finding backbone hosts

Given a set N of users, the facility location problem is to determine a feasible subset (i.e., facilities) of N so that a predefined objective function is optimized [12]. This problem is known to be NP-hard [13], and it can be formulated as a 0/1 integer linear programming (ILP). The problem of selecting BHs in a two-tier infrastructure is similar to the facility location problem in some aspects. They both intend to determine a feasible subset from a given set so that a predefined objective function is optimized. The major differences between them are the objective functions and the constraints. This motivates us to attack the problem of selecting BHs with an ILP approach. That is, we first attempt to formulate our problem as a 0/1 ILP, and then work out a solution method to the formulated 0/1 ILP.

Suppose that there are n hosts, denoted by $v_1, v_2, \ldots, v_n$, and let $d_{i,j}$ be the number of hops from v_i to v_j, where $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant n$. We have

$d_{i,j} = 0$ if $i = j$, and $d_{i,j} > 0$ an integer if $i \neq j$. We assume $d_{i,j} = d_{j,i}$ for all pairs of i and j. Let $w_i = \sum_{1 \leqslant j \leqslant n} d_{i,j}$ be the total number of hops from v_i to the other $n-1$ hosts. We use $x_i = 1$ ($x_i = 0$) to denote that v_i is (is not) chosen to be a BH, and $y_{i,j} = 1$ ($y_{i,j} = 0$) to denote that host v_i is (is not) attached to BH v_j. The hosts that are not BHs are called NBHs.

The objective is to minimize the total number of hops from each BH to the other $n-1$ hosts, i.e., to minimize $\sum_{1 \leqslant i \leqslant n} x_i w_i$. In a two-tier infrastructure, each host is required to be attached to exactly one BH with at most r hops distant, where $r \geqslant 1$ is a predefined integer. Therefore, two constraints are induced: $\sum_{1 \leqslant j \leqslant n} y_{i,j} = 1$ for all $1 \leqslant i \leqslant n$ and $x_j - y_{i,j} \geqslant 0$ for all $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant n$. Initially, we set $y_{i,j} = 0$ if $d_{i,j} > r$. The 0/1 ILP formulation for our problem is as follows:

$$\text{Minimize} \quad \sum_{1 \leqslant i \leqslant n} x_i w_i$$

$$\text{subject to} \quad \sum_{1 \leqslant j \leqslant n} y_{i,j} = 1 \quad \text{for all } 1 \leqslant i \leqslant n$$

$$x_j - y_{i,j} \geqslant 0 \quad \text{for all } 1 \leqslant i \leqslant n \text{ and } 1 \leqslant j \leqslant n$$

$$x_i \in \{0,1\} \quad \text{for all } 1 \leqslant i \leqslant n$$

$$y_{i,j} = 0 \quad \text{if } d_{i,j} > r \quad \text{and} \quad y_{i,j} \in \{0,1\} \quad \text{if } d_{i,j} \leqslant r \text{ for all } 1 \leqslant i \leqslant n \text{ and } 1 \leqslant j \leqslant n$$

If $x_i \in \{0,1\}$ and $y_{i,j} \in \{0,1\}$ are relaxed to $0 \leqslant x_i \leqslant 1$ and $0 \leqslant y_{i,j} \leqslant 1$, then an LP results, which is polynomial-time solvable. Suppose that x_i^* and $y_{i,j}^*$ are the optimal solution to the LP, where $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant n$. In the following, we present a rounding algorithm that can round x_i^* and $y_{i,j}^*$ to x_i' and $y_{i,j}'$, where $x_i' \in \{0,1\}$ and $y_{i,j}' \in \{0,1\}$ are a feasible solution to the 0/1 ILP. Initially, we set $x_i' = x_i^*$ for all $1 \leqslant i \leqslant n$ and let $X = \{x_i^* | 0 < x_i^* < 1\}$.

The rounding algorithm is executed iteratively until X is empty. In each iteration, a host v_c is selected as a BH if the increment (i.e., Δ^+) of the objective function induced by setting $x_c^* = 1$ is minimum, where $x_c^* \in X$. To offset the increment, we then set $x_j^* = 0$ for all hosts v_j with at most r hops distant from v_c. That is, hosts v_j are determined as NBHs. We use Δ^- to denote the total decrement induced by setting $x_j^* = 0$. If $\Delta^+ - \Delta^- > 0$, the objective function is further decreased by reducing some values x_l^* to fx_l^*, where $0 < f < 1$. We intend to have the objective function as small as possible under constraints. The rounding algorithm is presented below.

1. Determine $x_c^* \in X$ so that $(1 - x_c^*)w_c = \min\{(1 - x_i^*)w_i | x_i^* \in X\}$.
2. Compute $\Delta^+ = (1 - x_c^*)w_c$.
   /*Δ^+ is the increment of the objective function induced by setting $x_c^* = 1$*/.
3. Set $x_c' = x_c^* = 1$ and delete x_c^* from X.
4. Determine $Y = \{x_j^* | x_j^* \in X \text{ and } d_{c,j} \leqslant r\}$.
5. Compute $\Delta^- = \sum_{x_j^* \in Y} x_j^* w_j$.
   /* Δ^- is the decrement of the objective function induced by setting $x_j^* = 0$*/.
6. Set $x_j' = x_j^* = 0$ and delete x_j^* from X for all $x_j^* \in Y$.
   /* Each v_j that is at most r hops distant from v_c is determined as an NBH*/.
7. If $\Delta^+ - \Delta^- > 0$ and X is not empty, determine $0 < f < 1$ satisfying $\sum_{x_l^* \in X}(x_l^* - fx_l^*)w_l \geqslant \Delta^+ - \Delta^-$ and then set $x_l^* = fx_l^*$ for all $x_l^* \in X$.
   /* When $\Delta^+ - \Delta^- > 0$, the objective function is further decreased by reducing x_l^* to fx_l^**/.
8. If X is not empty, go to 1.
9. For each host v_i, determine a BH, say v_p, that is closest to v_i and then set $y_{i,p}' = 1$ and $y_{i,q}' = 0$ for all $1 \leqslant q \leqslant n$ and $q \neq p$.
   /* Each host is attached to the closest BH*/.

As a consequence of step 9, if v_i is a BH, then $y_{i,i}' = 1$ and $y_{i,q}' = 0$ for all $1 \leqslant q \leqslant n$ and $q \neq p$. In other words, when $y_{i,j}' = 1$ ($i \neq j$), we have $x_i' = 0$ and $x_j' = 1$. Suppose that x_i^{**} and $y_{i,j}^{**}$ are the optimal solution to the 0/1 ILP, where $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant n$. Let $S^{**} = \sum_{1 \leqslant i \leqslant n} x_i^{**} w_i$, $S^* = \sum_{1 \leqslant i \leqslant n} x_i^* w_i$ and $S' = \sum_{1 \leqslant i \leqslant n} x_i' w_i$. Clearly, $S^{**} \leqslant S^*$. In the following lemma, we show that the approximation ratio, which is defined as $\frac{S'}{S^{**}}$, is bounded above by $1 + \frac{\max\{w_i | 1 \leqslant i \leqslant n\}}{S^{**}}$.

Lemma 1. $\frac{S'}{S^{**}} \leqslant 1 + \frac{\max\{w_i | 1 \leqslant i \leqslant n\}}{S^{**}}$.

Proof. We assume that n_k iterations are executed by the rounding algorithm, and let Δ_t^+ and Δ_t^- denote

Δ^- and Δ^+, respectively, evaluated at the tth iteration, where $1 \leqslant t \leqslant n_k$. Also, let $S_t = \sum_{1 \leqslant i \leqslant n} x_i^* w_i = \sum_{x_i^* \in X} x_i^* w_i + \sum_{x_j' \notin X} x_j' w_j$ denote the value of the objective function evaluated at the tth iteration. Clearly, $S' = S_{n_k}$.

We first consider $t = 1$. If $\Delta_1^+ - \Delta_1^- \leqslant 0$, then $S_1 = S^* + (\Delta_1^+ - \Delta_1^-) \leqslant S^*$. If $\Delta_1^+ - \Delta_1^- > 0$, then further decrement (i.e., $\sum_{x_l^* \in X}(x_l^* - fx_l^*)w_l$) of the objective function is made in step 7, and so $S_1 = S^* + (\Delta_1^+ - \Delta_1^-) - \sum_{x_l^* \in X}(x_l^* - fx_l^*)w_l \leqslant S^*$. Then, we consider $2 \leqslant t \leqslant n_k - 1$. Similarly, $S_t = S_{t-1} + (\Delta_t^+ - \Delta_t^-) \leqslant S_{t-1}$ if $\Delta_t^+ - \Delta_t^- \leqslant 0$, and $S_t = S_{t-1} + (\Delta_t^+ - \Delta_t^-) - \sum_{x_l^* \in X}(x_l^* - fx_l^*)w_l \leqslant S_{t-1}$ if $\Delta_t^+ - \Delta_t^- > 0$. Hence, $(S^{**} \geqslant) S^* \geqslant S_1 \geqslant S_2 \geqslant \cdots \geqslant S_{n_k - 1}$.

Finally, we consider $t = n_k$. Since X is empty after step 6, we have $S' = S_{n_k} = S_{n_k - 1} + (\Delta_{n_k}^+ - \Delta_{n_k}^-)$. Then, $\frac{S'}{S^{**}} \leqslant \frac{S_{n_k-1} + (\Delta_{n_k}^+ - \Delta_{n_k}^-)}{S_{n_k-1}} \leqslant 1$ if $\Delta_{n_k}^+ - \Delta_{n_k}^- \leqslant 0$. Otherwise, since $S^{**} \geqslant S_{n_k - 1} = S' - (\Delta_{n_k}^+ - \Delta_{n_k}^-)$, we have $S' \leqslant S^{**} + (\Delta_{n_k}^+ - \Delta_{n_k}^-) \leqslant S^{**} + \Delta_{n_k}^+ \leqslant S^{**} + \max\{(1 - x_i^*)w_i | 1 \leqslant i \leqslant n\} \leqslant S^{**} + \max\{w_i | 1 \leqslant i \leqslant n\}$, which implies $\frac{S'}{S^{**}} \leqslant 1 + \frac{\max\{w_i | 1 \leqslant i \leqslant n\}}{S^{**}}$. □

Next, we show that every NBH v_p is attached to one BH v_q that is at most $2r$ hops distant.

Lemma 2. *$d_{p,q} \leqslant 2r$ if $y_{p,q}' = 1$, where $1 \leqslant p \leqslant n$, $1 \leqslant q \leqslant n$, and $p \neq q$.*

Proof. Suppose that v_p is an NBH, i.e., $x_p' = 0$. It suffices to show that there exists a BH that is at most $2r$ hops distant from v_p. According to the execution of the rounding algorithm, we have either $0 < x_p^* < 1$ or $x_p^* = 0$. If $0 < x_p^* < 1$, then $x_p^* \in X$ initially. Assume that there are n_k iterations executed. Then, x_p^* is selected in Y at step 4 of some iteration, say the tth iteration, where $1 \leqslant t \leqslant n_k$. At the same iteration, host v_c is determined as a BH, as a consequence of step 3. We have $d_{c,p} \leqslant r$ as specified at step 4. That is, the distance from v_p to v_c is at most r hops.

If $x_p^* = 0$, let $Z_p = \{v_j | d_{p,j} \leqslant r, x_j^* > 0 \text{ and } 1 \leqslant j \leqslant n\}$, i.e., Z_p is the set of hosts that are candidates to be the attached BH of v_p. We have Z_p nonempty, due to the constraints of the LP. For each $v_z \in Z_p$, we have either $x_z' = 1$ or $x_z' = 0$ at the end of the rounding algorithm. If $x_z' = 1$, then v_z is a BH and its distance to v_p is at most r hops. If $x_z' = 0$, then $0 < x_z^* < 1$. With the same arguments as the previous situation (i.e., $0 < x_p^* < 1$), there exists a BH that is at most r hops distant from v_z. Hence, the distance from v_p to the BH is at most $2r$ hops. □

3. OGHAM protocol

OGHAM constructs a two-tier architecture by selecting BHs on-demand for multicast applications. Each multicast member must be attached to a BH. BHs are responsible for determining multicast routes, forwarding data packets, handling dynamic group membership (the clients can dynamically join or leave the group), and updating multicast routes due to host movement.

All hosts are assumed to use a common wireless channel for communication, and they have the same transmission radius. Two hosts can communicate directly if they are within the transmission range of each other. We assume that the communication is bi-directional. A host can transmit in omni directions, and so each transmission is a local broadcast. Due to the common radio channel, only one host is allowed to broadcast within a transmission range at a time. We assume that there is a supporting MAC protocol that is responsible for channel access (scheduling, queuing, and contention resolution).

When a server (client) v_i attempts to create (join) a multicast group, v_i first tries to find a BH within a region with a radius of $2r$ hops centered at v_i, where $r \geqslant 1$ is a predefined integer. If such a BH can be found, then v_i is attached to it. Otherwise, v_i broadcasts a message in a larger region, called *multicast region*, with a radius of γ hops centered at v_i for collecting neighboring information, where $\gamma \geqslant 2r$ is a predefined integer. Then, v_i selects BHs for the multicast region and determines the attachment from NBHs to BHs by the method of Section 2. Also, v_i sends the list of all BHs and the neighboring information to each BH. A distributed algorithm for selecting BHs in a multicast region is detailed in Appendix A.

For example, suppose that a server s intends to create a multicast group and it cannot find a BH within 2 hops ($r = 1$ for this example). So, s reacts by triggering the selection of BHs in a multicast region with a radius of $\gamma = 4$ hops centered at s. Refer to Fig. 1. First, s broadcasts a message in the multicast region (refer to Fig. 1(a)). Upon receiving the message, hosts in the multicast region reply their neighboring information to s. With the neighboring information, s then selects BHs and attaches NBHs to BHs (refer to Fig. 1(b)).

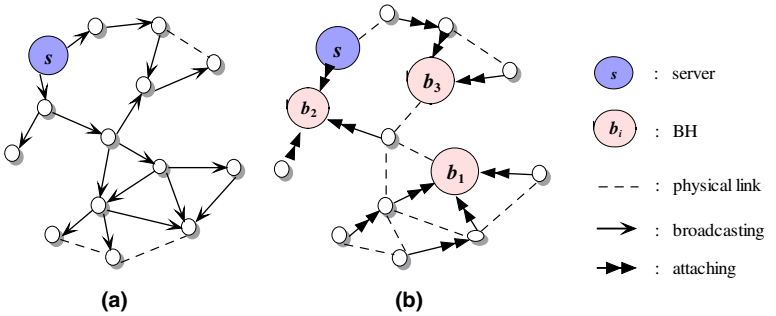


Fig. 1. BH selection in a multicast region. (a) Broadcasting and (b) selecting BHs and attaching NBHs to BHs.

After BHs in the multicast region are selected, clients can join the multicast group by asking the attached BHs to query the location of the server. The BH attached by the server then replies to the queries. Through the round-trip communication (querying and replying), the BH attached by server can determine the multicast routes from the server to the clients. A distributed algorithm for determining multicast routes is detailed in Appendix B.

With the same example of Fig. 1, assume that c_1 and c_2 are two clients. They join the multicast group by attaching themselves to b_1 and b_3, respectively. The server s is attached to b_2. Both b_1 and b_3 can locate s by querying b_2. At the same time, the multicast routes, i.e., s–b_2–f_1–b_1–f_2–c_1 and s–b_2–f_1–b_3–c_2, from s to c_1 and c_2, respectively, can be determined (refer to Fig. 2).

According to Lemma 2, each host is at most $2r$ hops distant from a BH in the multicast region. Hence, follow-up clients in the same group or members of subsequent multicast groups, all in the multicast region, can be attached to BHs within $2r$ hops, without the need of further broadcasting. That is, the overheads to construct additional infrastructures can be avoided.

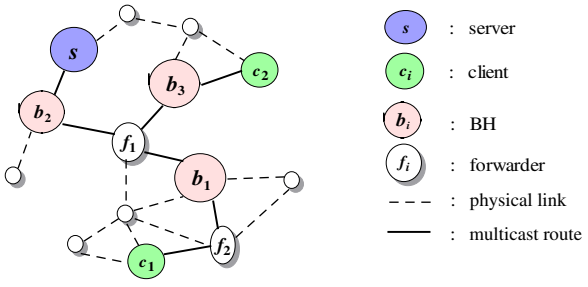


Fig. 2. Determining multicast routes.

As described above, the BHs attached by clients are responsible for querying the location of the server. For a client within a different multicast region from the server, the attached BH fails to locate the server, and so it floods a query message over the entire network, in order to locate the server. Through the flooding, the two multicast regions where the client and the server are positioned can be merged into a larger one. The merging algorithm is detailed in Appendix C. After the merging, a multicast route from the server to the client can be determined by the algorithm of Appendix B.

Following the example of Fig. 2, we assume that there is a client c_3, which is outside the multicast region of Fig. 2, attempting to join the multicast group created by s. The gray portion of Fig. 3 shows the multicast region created by c_3. There are two BHs, i.e., b'_1 and b'_2, selected in the new multicast region and c_3 is attached to b'_1. In order to locate s, b'_1 floods a message. Upon receiving the message, b_2 replies to b'_1. Through the message exchange, a multicast route, i.e., s–b_2–f_1–b_1–f_2–f_4–b'_1–c_3, from s to c_3 is then determined.

In OGHAM, the BHs attached by the members of the multicast groups determine multicast routes by the aid of neighboring information. However, the neighboring information has to be updated as hosts move. There are three mechanisms to relieve the mobility problem. First, a timestamp is stamped onto each entry of the neighboring information. An entry with an outdated timestamp will be removed from the neighboring information. Each host is required to piggyback the up-to-date neighboring information onto the packets which it transmits or forwards. When BHs receive the packets, they update their neighboring information and refresh the timestamps with the piggybacked information. In this way, multicast routes with better availability

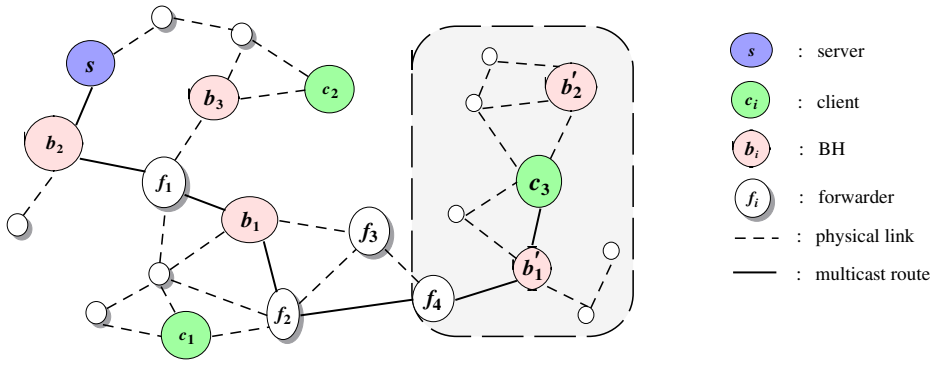


Fig. 3. Determining multicast routes across two multicast regions.

can be determined, and the size of the neighboring information is limited.

Second, when a BH detects a route disconnection, it determines an alternative route by the aid of the neighboring information to replace the disconnected one. Let v_s be the server, v_c be a client, and v_x (v_y) be the BH that v_s (v_c) is attached to. We use $p_{s,x}$ to denote a route from v_s to v_x ($p_{x,y}$ and $p_{y,c}$ have similar meanings). Any route from v_s to v_c has to pass through v_x and v_y. When a host v_i ($\notin\{v_s,v_x\}$) in $p_{s,x}$ detects a disconnected link to the downstream (upstream) host, it transmits a packet to notify v_s (v_x) of this disconnection. The detection of disconnection in $p_{p,q}$ and $p_{c,q}$ is alike. Compared with most of previous routing/multicasting protocols, which determine a new route in the case of disconnection by flooding a message over the network, less frequent flooding is induced and the robustness of the infrastructure is enhanced in OGHAM.

Third, a threshold α is introduced for guaranteeing the transmission quality of the multicast routes. The value of α is predefined, which depends on the quality requirement of the multicast applications. If a BH that is attached by a client counts a greater number of continuously lost data packets than α, i.e., more than α successive data packets transmitted from the server are not received by the BH, then it broadcasts a packet to all BHs for querying the current location of the server. On the other hand, if more than α successive data packets from the attached BH are not received by the client, then the client is attached to a new BH. Hence, by the aid of α, the transmission quality of the multicast routes can be assured.

It should be noted that a highly mobile BH will decline the performance. For such a situation, the second and third mechanisms proposed above can be applied to relieve the problem. When route disconnection occurs due to a highly mobile BH, the multicast members that are attached to it are informed of this disconnection by the second mechanism. Then, with a high probability, they can be attached to new BHs within $2r$ hops. It is revealed by Lemma 2 that the BHs selected by OGHAM are evenly distributed over a multicast region. On the other hand, if the data transmission to a client has a quality lower than α, then the third mechanism forces the client to be attached to a new BH.

4. Related two-tier multicast protocols

MCEDAR [9] and ADB [11] are two representative multicast protocols based on two-tier infrastructures. MCEDAR, which is an extension to CEDAR [8], has only a subset of the network involved in computing multicast routes by extracting a dominating set (cores) from the network to be used for reducing the number of control messages. Each host periodically broadcasts a control message for selecting one of its neighbors with the largest degree as its core.

ADB, which is derived from VDBP [10], selects BHs based on the concept of the dominating set. It relaxes the property of the dominating set and allows a host to be attached to a BH more than one hop away. It creates a forest of varying-depth trees, each of which is rooted at a BH. The selection of the BHs is based on the combination of normalized link failure frequency and neighbor degree. Data forwarding from members of the multicast groups are directed toward the BHs, instead of being broadcast to other irrelevant hosts or the entire network. Then, a flooding mechanism is employed over the network of BHs to disseminate the data packets. Upon receiving data packets from a downstream host, a BH will forward them to

Table 1
Comparison of OGHAM with MCEDAR and ADB

		OGHAM	MCEDAR	ADB
Infrastructures	Strategies to select BHs	Minimal number of hops	Maximal number of neighbors	Maximal number of neighbors and low link failure frequency
	Control messages	Transmitted by multicast members and the attached BHs on-demand	Transmitted by each host periodically	Transmitted by each host periodically
Multicast routes	From a BH to the attached members	Tree	Tree	Tree
	From a BH to the other BHs	Tree	Tree	Flooding to the other BHs

every other BH. Table 1 shows the comparison of OGHAM with MCEDAR and ADB.

5. Performance evaluation

Simulations for evaluating the performance of OGHAM and other multicast protocols are implemented using the Network Simulator 2 package (ns-2) [14]. The simulation environment models a large-scale MANET of 200 hosts distributed randomly over a 1000 m × 1000 m area. The IEEE 802.11 MAC is used as the MAC layer protocol. Each host is equipped with a radio transceiver that is capable of transmitting up to approximately 100 m over a wireless channel. The transmission capability of each network interface is 2 Mbps and the size of data payload is 512 bytes. Ten runs with different seed numbers are conducted for each scenario and collected data are averaged over those runs.

The performance of OGHAM is studied by extensive simulations in three aspects. First, comparisons are made between our strategy and those adopted in other two-tier protocols [7–11] for selecting BHs. Second, performance comparisons are made among OGHAM, MCEDAR and ADB. Third, the performance of OGHAM is investigated by assigning different values of group size, group number and r.

5.1. Comparisons of two-tier infrastructure strategies

Two strategies, i.e., minimum dominating set and maximal number of neighbors, are adopted in previous two-tier protocols [7–11] for selecting BHs. They can be formulated as two 0/1 ILPs. Compared with the 0/1 ILP formulated in Section 2 for OGHAM, they have the same constraints for two-tier infrastructures, but different objective functions.

The strategy of minimum dominating set aims to select a minimum number of BHs that can dominate the entire network. Let $x_i = 1$ ($x_i = 0$) represent that host v_i is (is not) chosen as a BH. The objective is to minimize $\sum_{1 \leqslant i \leqslant n} x_i$. On the other hand, the strategy of maximal number of neighbors aims to select BHs whose total number of neighbors is maximal. The objective is to maximize $\sum_{1 \leqslant i \leqslant n} n_i x_i$, where n_i is the number of neighbors of host v_i. The two resulting 0/1 ILPs can be solved, similar to the 0/1 ILP for OGHAM. During their execution of the rounding algorithm, r is relaxed to $2r$.

To evaluate the qualities of multicast services, simulations are performed regarding the transmission latency from the server to clients, the transmission time between two neighboring hosts, and the number of lost packets. These simulations are carried out based on the infrastructures established by the three 0/1 ILPs mentioned above. Six multicast groups, denoted by $G_1, G_2, \ldots, G_6$, are randomly chosen from 200 hosts. Each group consists of one server and six clients. Each server sends a data packet to its clients every 4 s. The simulations proceed for 1800 s. The servers are scheduled to start data traffics every 300 s. That is, G_1 starts at the first, G_2 starts after 300 s, G_3 starts after 600 s, and so on. Simulation results are collected from G_1 for every interval of 300 s.

Simulation results are exhibited in Figs. 4–6. Two kinds of latency are calculated and compared for delivering each data packet in Figs. 4 and 5. Transmission latency in Fig. 4 indicates the elapsed time to send a packet from the server to a client along the multicast route; transmission time in Fig. 5 indicates the elapsed time to transmit a packet between two neighboring hosts. Figs. 4 and 5 reveal that when the number of multicast groups increases,

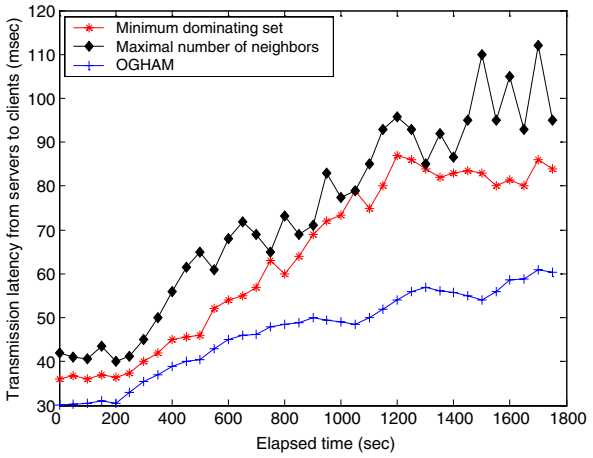


Fig. 4. Transmission latency from servers to clients.

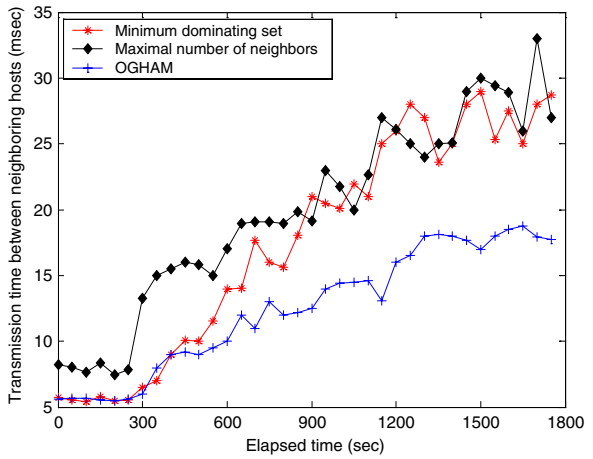


Fig. 5. Transmission time between two neighboring hosts.

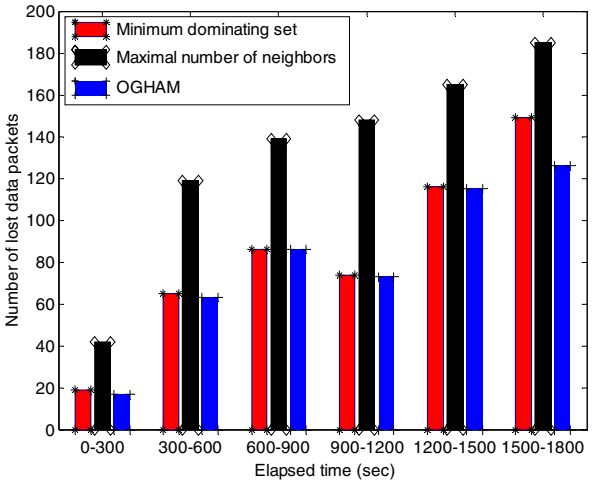


Fig. 6. Numbers of lost data packets.

OGHAM has less transmission latency and transmission time than the other two strategies. OGHAM has less transmission latency because its objective of selecting BHs is to minimize the total number of hops to all hosts. On the other hand, since the BHs selected by the other two strategies have more neighbors, they consume more time in contending radio channels with their neighbors. This explains why OGHAM also has less transmission time.

Fig. 6 shows the numbers of lost data packets. OGHAM and the strategy of minimum dominating set have fewer lost data packets than the strategy of maximal number of neighbors. Since the latter strategy is to maximize the number of neighbors, the selected BHs have higher probabilities of collision during packet transmission.

To sum up, our simulations show that the BH selection strategy adopted in OGHAM is superior to the other two strategies in transmission latency, transmission time and lost data packets. In other words, OGHAM has a more effective traffic distribution than the other two strategies.

5.2. Comparisons of two-tier multicast protocols

One or two multicast groups are randomly chosen from 200 hosts; each consists of one server and seven clients. The simulation proceeds for 300 s and the average speed for host movement varies from 0 to 30 m/s. Table 2 summarizes the parameters and their assigned values for OGHAM, MCEDAR and ADB. The assigned values for MCEDAR and ADB are the same as those assigned in [8,11], respectively.

In order to investigate the effectiveness, attachment stability and receiving data packet ratios of OGHAM, MCEDAR and ADB, six metrics are adopted. They include (1) the number of control packets, (2) the number of data packets transmitted by servers or forwarders, (3) transmission time between two neighboring hosts, (4) transmission latency from servers to clients, (5) the number of messages *init_group* and *join_request* (introduced in Appendix B), and (6) the ratio of receiving data packets (i.e., the ratio of the number of data packets received by clients to the number of data packets delivered from servers). Since *init_group* and *join_request* are issued when multicast members are attached to BHs or re-attached to new BHs, metric (5) can measure the attachment stability of multicast members to BHs. In the simulation, all three proto-

Table 2
Parameters and their assigned values

Parameters	Meanings	Values		
		OGHAM	MCEDAR	ADB
r	Maximal number of hops from an NBH to the attached BH	3	1	3
γ	Radius (number of hops) of multicast regions	7		
maxhopcount	Maximal number of hops for a flooding	20	20	20
α	Number of successively lost data packets	3		
period	Period for each host to broadcast a control packet to its neighbors		1.5 s	1.5 s
R	Robustness factor		2	
Nlff	Time window for computing normalized link failure frequency			3 s
a	Smoothing factor			0.6

cols have the same number of data packets delivered from servers.

We use a simplified instance to illustrate metrics (2) and (6). Consider a multicast group that consists of the server s and two clients c_1, c_2. Assume that the multicast routes are s–f_1–f_2–c_1 and s–f_1–c_2, where f_1 and f_2 are forwarders. For each data packet delivered from s, three broadcasts (by s, f_1 and f_2) will be induced, i.e., three data packets will be generated. If there are five data packets delivered from s, then there are $3 \times 5 = 15$ (the value of metric (2)) data packets generated in the entire network. Further, if four and five data packets are received finally by c_1 and c_2, respectively, then the ratio of receiving data packets is counted as $(4/5 + 5/5)/2 = 90\%$ (the value of metric (6)).

The effectiveness of OGHAM, MCEDAR and ADB is investigated by metrics (1)–(4). Figs. 7 and 8 show the numbers of control packets and data packets, respectively, generated in the network. The data packets counted in Fig. 8 are restricted

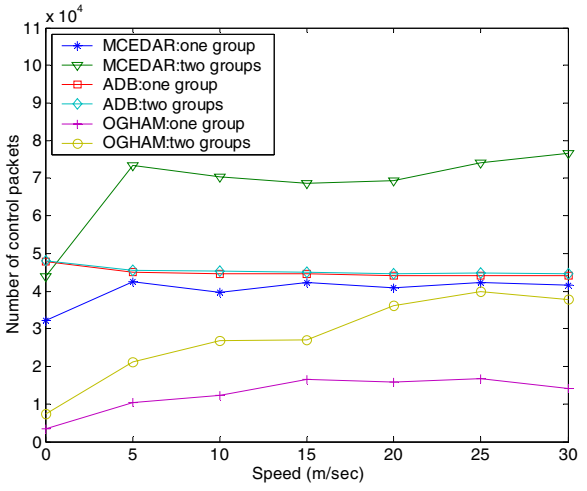


Fig. 7. Numbers of control packets.

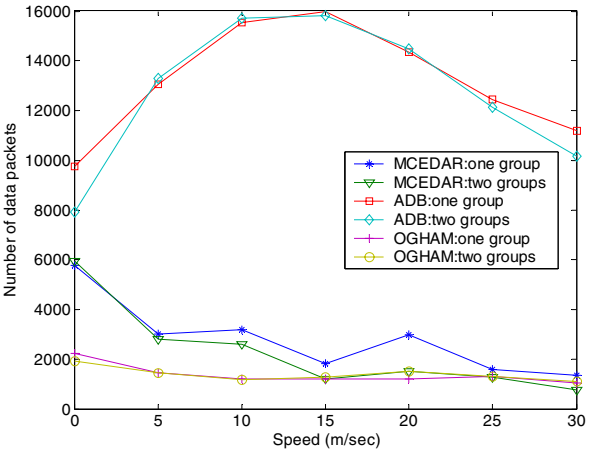


Fig. 8. Numbers of data packets.

to those transmitted by servers or forwarders. In MCEDAR and ADB, all hosts periodically transmit control packets for constructing and maintaining the two-tier infrastructure. In OGHAM, only the multicast members and the attached BHs are required to transmit control packets on-demand. Hence, OGHAM generates fewer control packets. On the other hand, OGHAM also generates fewer data packets, because it selects BHs with the objective of minimizing the total number of hops to all hosts. The objective can result in shorter multicast routes. ADB generates more data packets than MCEDAR because it uses a flooding mechanism to disseminate data packets over the network of BHs.

Fig. 9 shows the transmission time consumed by transmitting a packet between two neighboring hosts. Since MCEDAR and ADB select BHs with a maximal number of neighbors, they spend more transmission time than OGHAM. Also, ADB spends more transmission time than MCEDAR, as a consequence that it generates more data packets

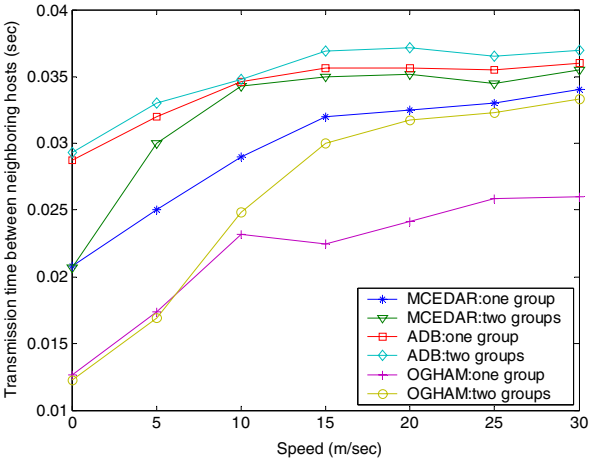


Fig. 9. Transmission time between two neighboring hosts.

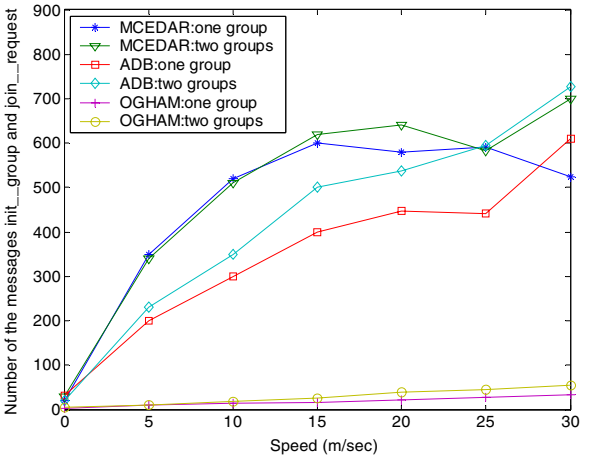


Fig. 11. Numbers of messages *init_group* and *join_request*.

than MCEDAR. Fig. 10 shows the transmission latency induced by transmitting data packets from servers to clients. OGHAM has less transmission latency than MCEDAR and ADB, which is an immediate consequence of shorter multicast routes and less transmission time.

A highly mobile environment will result in a high frequency of BH re-selection, multicast member re-attachment, and multicast route re-construction. Since MCEDAR and ADB select BHs with the strategy of maximal number of neighbors, the neighborhoods of the selected BHs are inclined to change as hosts move. Fig. 11 shows that MCEDAR and ADB have higher frequency of re-attachment than OGHAM by counting the numbers of messages *init_group* and *join_request*.

Fig. 12 shows the ratios of receiving data packets. When the host speed is low, OGHAM has higher ratios than MCEDAR and ADB. The reason is that OGHAM has better effectiveness (fewer packets, shorter transmission time and transmission latency) and more stable attachment. When the host speed increases, the ratios for all three protocols decline, but OGHAM and ADB are superior to MCEDAR. Both OGHAM and ADB have comparable ratios, even though ADB employs a flooding mechanism to disseminate data packets. Flooding of data packets will incur extra overheads and decline the performance and throughput of the entire network.

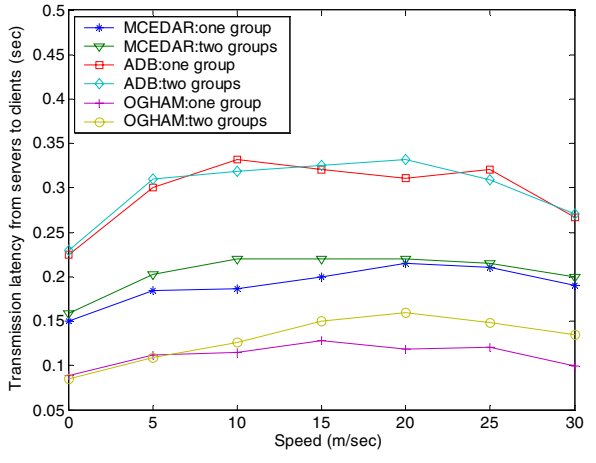


Fig. 10. Transmission latency from servers to clients.

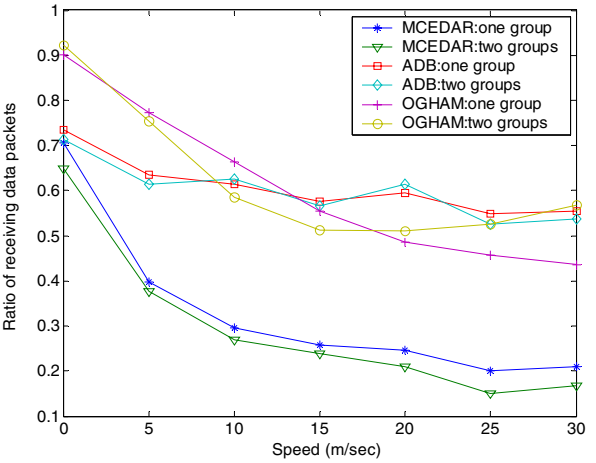


Fig. 12. Ratios of receiving data packets.

5.3. Performance of OGHAM versus group size, group number and r

Since control packets are generated for constructing and maintaining two-tier infrastructures,

they can be used as a measure for the effectiveness of a multicast protocol. An ineffective multicast protocol will generate a large number of control packets. In the simulations of Figs. 13 and 14, OGHAM reconstructs the infrastructure whenever 1000 data packets are delivered from a server to its clients.

Fig. 13 shows the ratios of control packets for different group sizes and different group numbers, which are computed as the number of control packets divided by the total number of control packets and data packets. In OGHAM, since the BHs selected for a multicast group are globally available, it is not necessary for follow-up multicast groups to construct additional infrastructures. Hence, as group size or group number increases, the ratio of control packets declines.

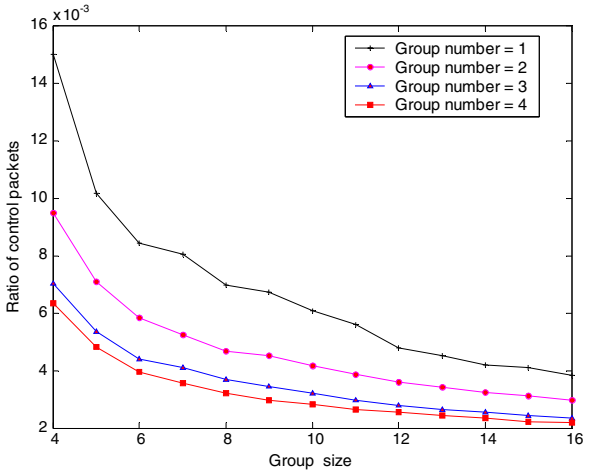


Fig. 13. Ratios of control packets for OGHAM.

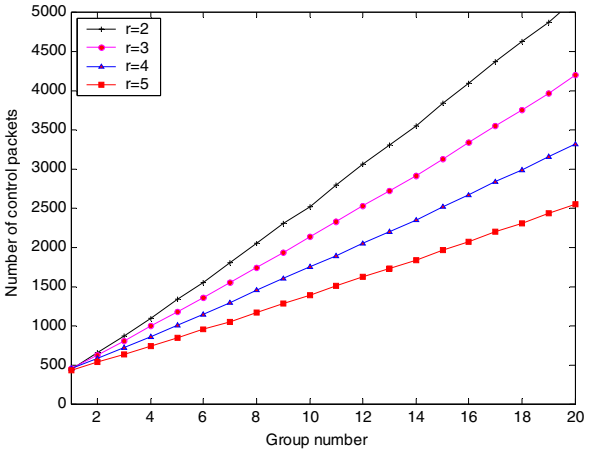


Fig. 14. Numbers of control packets generated by OGHAM.

Fig. 14 shows the numbers of control packets generated by OGHAM for different group numbers and different values of r. Each multicast group consists of one server and six clients. It is observed that OGHAM generates fewer control packets when r increases. The reason is that when the value of r gets greater, OGHAM will select fewer BHs, which causes fewer control packets generated for maintaining the infrastructure.

6. Conclusion

In order to obtain shorter multicasting routes, we have introduced a new strategy for selecting BHs whose objective is to minimize the total number of hops to all hosts. A multicast protocol named OGHAM is proposed to construct a two-tier infrastructure and determine multicast routes. OGHAM has the following advantages over previous two-tier multicast protocols (MCEDAR and ADB).

As a consequence of our strategy of selecting BHs, OGHAM has shorter multicast routes and more stable attachments from multicast members to BHs than MCEDAR and ADB. Stable attachments can reduce the frequencies of BH re-selection, multicast member re-attachment, and multicast route re-determination. Also, the problem of traffic concentrations and network bottlenecks in OGHAM is less serious than in MCEDAR and ADB. On the other hand, compared with MCEDAR and ADB in which all hosts need to broadcast control packets periodically for maintaining the infrastructure, OGHAM asks only members of the multicast groups to construct the infrastructure on-demand. Hence, OGHAM incurs fewer overheads for constructing and maintaining the infrastructure.

The problem of selecting BHs can be formulated as a 0/1 ILP, which is an NP-hard problem. In order to find a feasible solution to the 0/1 ILP, an approximation algorithm whose approximation ratio is bounded above by $1+\frac{\max\{w_i|1\leqslant i\leqslant n\}}{S^{**}}$ is proposed. This guarantees a limited difference between the obtained feasible solution and the optimal solution. Further, since the fraction $\frac{\max\{w_i|1\leqslant i\leqslant n\}}{S^{**}}$ is very small in most instances, the obtained feasible solution is very close to the optimal solution. Therefore, the multicast routes determined by OGHAM are shorter than those determined by the other two-tier multicast

protocols, which was also verified by our simulations.

Acknowledgements

This work was supported by the MediaTek Inc. under the project "Wireless Communication Systems". This work was also supported by National Science Council of Taiwan under the NSC93-2524-S-008-002 Project.

Appendix A. Selecting BHs

We use R_i to denote a multicast region associated with a host v_i, which is a region with a radius of γ hops centered at v_i. In other words, R_i contains all hosts that are at most γ hops distant from v_i. We assume $\gamma \geqslant 2r$. Let V_i be the set of hosts that are located in R_i. A distributed algorithm for selecting BHs within R_i is presented below.

Four variables: t, *hopcount*, N_j and M_j are used in the algorithm. t is the elapsed time required for transmitting a message between two neighboring hosts. *hopcount* counts the number of hops to v_i. N_j is a set containing all neighbors of v_j. M_j is a set containing some N_ks (explained later). Initially, we set $M_j = \{N_j\}$. Two messages: *init* and *echo* are transmitted over the network. *init*, which carries *hopcount*, is initiated by v_i and propagated over R_i. Upon receiving an *init* from a host, say v_α, a host v_j relays it to its neighbors and then waits for a time period to receive *echo*s from its neighbors. If timeout occurs, v_j replies an *echo* to v_α. M_j is carried by *echo*. Upon receiving an *echo* from a host, say v_β, a host v_j augments M_j with M_β. A message is *redundant* to a host if the host received the same message before.

The execution of the algorithm is described as follows:

1. v_i performs the following.
   1.1. Set $hopcount = 0$.
   1.2. Transmit *init*(*hopcount*) to all its neighbors.
   1.3. Wait *echo* messages from the neighbors for a time period of $2(\gamma - 1) \times t$.
2. Upon receiving a nonredundant *init*(*hopcount*) from a host, say v_α, a host v_j ($\neq v_i$) performs the following.
   2.1. Set $M_j = \{N_j\}$.
   2.2. If $hopcount < \gamma - 1$, perform the following.
      2.2.1. Set $hopcount = hopcount + 1$.
      2.2.2. Transmit *init*(*hopcount*) to all its neighbors.
      2.2.3. Wait *echo* messages from the neighbors for a time period of $2(\gamma - 1 - hopcount) \times t$.
   2.3. If $hopcount = \gamma - 1$, reply *echo*(M_j) to v_α.
3. Upon receiving *echo*(M_β) from one, say v_β, of the neighbors, a host v_j ($\neq v_i$) performs the following.
   3.1. Set $M_j = M_j \cup M_\beta$.

If timeout occurs at step 2.2.3, v_j replies *echo*(M_j) to v_α. If timeout occurs at step 1.3, the algorithm terminates and V_i can be determined from M_i as follows: $V_i = \{v_k \mid v_k \in N_j \text{ and } N_j \in M_i\}$. Without loss of generality, suppose $V_i = \{v_1, v_2, \ldots, v_h\}$. An adjacency 0/1 matrix for the hosts of V_i can be obtained from M_i so that a 1 (0) in the (x, y) entry denotes that v_x and v_y are (are not) two neighboring hosts, where $1 \leqslant x \leqslant h$ and $1 \leqslant y \leqslant h$. With this matrix, all $d_{x,y}$'s can be computed by the Dijkstra's algorithm [15]. The hosts in V_i are then classified into two categories: BHs and NBHs by the method of Section 2. Let B_i be the set of all BHs in V_i. Each host in V_i can determine which category it belongs to and compute shortest routes to the other hosts in V_i if v_i broadcasts B_i and M_i over R_i.

Let n be the number of hosts within R_i and l be the number of hosts that are γ hops distant from v_i. There are $n - l$ messages *init* generated at steps 1 and 2, and there are $n - 1$ messages *echo* generated at step 3. Hence, the distributed algorithm generates $2n - l - 1$ messages. On the other hand, the distributed algorithm terminates at step 1.3, and so it takes $2(\gamma - 1) \times t$ time.

Appendix B. Determining multicast routes

We let v_s be the server and v_c be any client. First, v_s and v_c are attached to two BHs, say v_p and v_q, respectively. If v_s is a BH, then $v_p = v_s$. If v_s is an NBH, then v_p is the closest BH to v_s. If there exists a BH whose distance to v_s is within $2r$ hops (v_s is neither a BH nor an NBH), then v_p is the BH. Otherwise, B_s is determined and v_p is the closest BH to v_s which can be determined by the aid of B_s and M_s. v_q can be determined similarly. We assume $v_q \in B_w$ for some $1 \leqslant w \leqslant n$.

In OGHAM, v_p (v_q) acts like a proxy for v_s (v_c). All messages that are initiated from or destined for v_s (v_c) have to be processed by v_p (v_q). When v_c decides to join the multicast group, it sends a joining

request to v_q which is responsible for forwarding the request to v_p. Upon receiving the request, v_p replies to v_q and sends the request to v_s. The route from v_c to v_s has to be kept. A distributed algorithm for constructing a multicast group is presented below.

We use $p_{s,p}$ to represent a route from v_s to v_p. $p_{p,s}$, $p_{p,q}$, $p_{q,p}$, $p_{q,r}$ and $p_{r,q}$ all have similar meanings. These routes can be obtained by the Dijkstra's algorithm and they are included as parts of messages. Four messages: *init_group*, *echo_group*, *join_request* and *join_echo* are transmitted for group initialization. *init_group* and *echo_group* are between v_s and v_p, while *join_request* and *join_echo* are between v_c and v_q. Another four messages: *join_request*, *join_echo*, *join_ack* and *join_nack* are transmitted for joining v_c to the group. They are all between v_c and v_q.

Two messages: *query_server* and *query_ack* are transmitted between v_p and v_q for v_q to query the location of the server. *query_server* also informs the server that v_c is one of the clients. There is a client list, denoted by L_c, kept in the server that stores all the clients. Initially, L_c is empty. In case v_s (v_c) fails to connect with v_p (v_q), two messages: *init_reattach* and *echo_reattach* are transmitted to reattach v_s (v_c) to a new BH. Message flows for determining multicast routes are shown in Fig. 15, where $v_p \in B_w$ is assumed. The execution of the algorithm is described as follows:

4. v_s performs the following for group initialization.
   4.1. Set L_c empty.
   4.2. Transmit *init_group*($p_{s,p}$) to v_p.
   4.3. Wait an *echo_group* message from v_p for a time period of $4r \times t$.
5. Upon receiving *init_group*($p_{s,p}$), v_p performs the following.
   5.1. Transmit *echo_group*($p_{p,s}$) to v_s.
6. Upon receiving *echo_group*($p_{p,s}$), v_s performs the following.
   6.1. v_s finishes the group initialization.
7. v_c performs the following for group initialization.
   7.1. Transmit *join_request*($p_{c,q}$) to v_q.
   7.2. Wait a *join_echo* message from v_q for a time period of $4r \times t$.
8. Upon receiving *join_request*($p_{c,q}$), v_q performs the following.
   8.1. Set $waiting\_time = 2 \times \max\{d_{q,k} \mid \text{for all } v_k \in B_w\} \times t$.
   8.2. Transmit *join_echo*($p_{q,c}$, *waiting_time*) to v_c.
   8.3. Transmit *query_server*(v_c,$p_{q,k}$) to all $v_k \in B_w$.
   8.4. Wait a *query_ack* message from v_p for a time period of *waiting_time*.
9. Upon receiving *join_echo*($p_{q,c}$, *waiting_time*), v_c performs the following.
   9.1. Wait a *join_ack* or *join_nack* message from v_q for a time period of $2r \times t$ plus *waiting_time*.
10. Upon receiving *query_server*(v_c,$p_{q,p}$), v_p performs the following.
    10.1. Transmit *query_server*(v_c,$p_{p,s}$) to v_s.
    10.2. Transmit *query_ack*($p_{q,p}$) to v_q.
11. Upon receiving *query_server*(v_c,$p_{q,k}$), v_k ($\notin \{v_p, v_s\}$) performs the following.
    11.1. Discard the message.
12. Upon receiving *query_server*(v_c,$p_{p,s}$), v_s performs the following.
    12.1. Add v_c to L_c.
13. Upon receiving *query_ack*($p_{q,p}$), v_q performs the following.
    13.1. Transmit *join_ack*($p_{q,c}$) to v_c.
14. Upon receiving *join_ack*($p_{q,c}$), v_c performs the following.
    14.1. v_c finishes the group initialization.
15. Upon receiving *join_nack*($p_{q,c}$), v_c performs the following.
    15.1.Wait a *join_ack* or *join_nack* message from v_q.

/* A *join_nack* message from v_q means that v_p cannot be found in B_w. In this case, another algorithm is invoked to locate v_p. The algorithm will be presented in Appendix C*/.

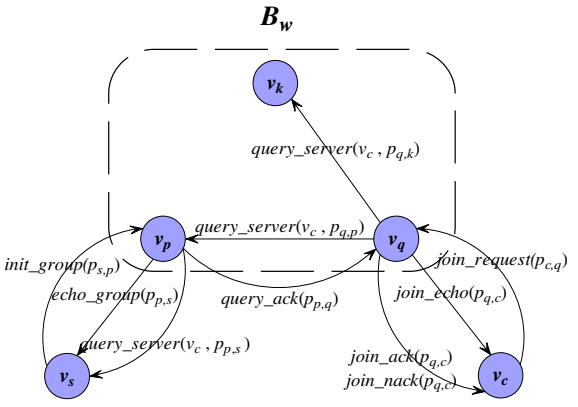


Fig. 15. Message flows.

16. Upon receiving any message above, a host ($\notin\{v_s, v_c, v_p, v_q\}$) forwards it to the next host according to the carried path.

If timeout occurs at step 4.3, v_s transmits an *init_reattach* message to all hosts that are at most $2r$ hops distant from v_s. Upon receiving an *init_reattach*, a BH replies an *echo_ reattach*. After receiving replies from BHs, v_s selects the closest one as new v_p and is re-attached to it (i.e., perform steps 4.2 and 4.3). In case v_s does not receive any *echo_ reattach*, it constructs R_s and then selects new v_p from B_s. If timeout occurs at steps 7.2 or 9.1, v_c does similarly.

If timeout occurs at step 8.4, v_q transmits a *join_ nack* message to inform v_c that v_p cannot be found in B_w. Besides, v_q invokes a distributed algorithm to locate v_p, as detailed in Appendix C. After v_s receives an *echo_group* message from v_p (step 6) and every v_c receives a *join_ack* message from v_q (step 14), the algorithm terminates and a path from v_s to v_c is established.

The distributed algorithm mainly accomplishes two tasks: one is to attach v_s to v_p when v_s attempts to create a multicast group (refer to steps 4–6 and 16), and the other is to attach v_c to v_q and establish a path from v_s to v_c when v_c attempts to join a multicast group (refer to steps 7–16). At most $4r$ messages and at most $8r + \sum_{v_k \in B_w} d_{q,k} + d_{q,p} + \mathrm{msg}_C$ messages are generated for the first and second tasks, respectively, where msg_C is the number of messages generated for the distributed algorithm of Appendix C. Also, $4r \times t$ time and at most $6r \times t + 2 \times \max\{d_{q,k} | \text{for all } v_k \in B_w\} \times t + \mathrm{time}_C$ time are required for the two tasks, respectively, where time_C is the time required for the distributed algorithm of Appendix C.

Appendix C. Merging multicast regions

In case v_q cannot find v_p in B_w, v_q starts a merging algorithm to locate v_p by searching a region with a radius of *maxhopcount* hops centered at v_q, where $maxhopcount > \gamma$ is a constant. Two messages: *global_query* and *global_ack* are transmitted over the region. *global_query*, which carries *hopcount*, v_c, B_w and M_w, is initiated by v_q. Upon receiving a nonredundant *global_ query*, a host v_j ($\notin\{v_p, v_q\}$) relays it to its neighbors. When v_p receives a *global_ query*, it transmits a *global_ack* to v_q. The *global_ ack* carries $p_{p,q}$, B_z and M_z, where $v_p \in B_z$ is assumed. The execution of the algorithm is described as follows:

17. v_q performs the following.
    17.1. Set hopcount = 0.
    17.2. Transmit *global_query*($hopcount, v_c, B_w, M_w$) to all its neighbors.
    17.3. Wait a *global_ack* message for a time period of 2 $maxhopcount \times t$.
18. Upon receiving a nonredundant *global_ query*($hopcount, v_c, B_w, M_w$), a host v_j ($\notin\{v_p, v_q\}$) performs the following.
    18.1. If $hopcount < maxhopcount$, perform the following.
        18.1.1. Set $hopcount = hopcount + 1$.
        18.1.2. Transmit *global_query*($hopcount, v_c, B_w, M_w$) to all its neighbors.
    18.2. If $hopcount = maxhopcount$, perform the following.
        18.2.1. Discard the message.
19. Upon receiving *global_query*($hopcount, v_c, B_w, M_w$), v_p performs the following.
    19.1. Transmit *global_ack*($p_{p,q}, B_z, M_z$) to v_q.
    19.2. Transmit *query_server*($v_c, p_{p,s}$) to v_s.
20. Upon receiving *global_ack*($p_{p,q}, B_z, M_z$), a host v_j ($\notin\{v_p, v_q\}$) forwards it to the next host according to $p_{p,q}$.
21. Upon receiving *global_ack*($p_{p,q}, B_z, M_z$), v_q performs the following.
    21.1. Transmit *join_ack*($p_{q,c}$) to v_c.

A timeout at step 17.3 means that v_p cannot be found in the region. On the other hand, if v_q receives *global_ack*($p_{p,q}, B_z, M_z$), then $v_p \in B_z$ is found. For either case, v_q replies to v_c and the algorithm terminates. A *join_ack* message received at step 15.1, which comes from step 21.1, implies that v_c was successfully added to L_c (at step 12.1). A *join_nack* message received at step 15.1 is a consequence of step 17.3. The waiting time at step 15.1 is set to $(2maxhopcount + r) \times t$. In case $v_p \in B_z$ ($z \neq w$), R_z and R_w are merged (B_z and B_w are merged and M_z and M_w are merged therefore) into a larger multicast region. For any follow-up multicast group in which the server is located in R_w and the clients are located in R_z, a flooding to locate v_p is no longer necessary.

Let m be the number of hosts within a region with a radius of *maxhopcount* hops centered at v_q. The distributed algorithm generates at most $m + maxhopcount$ messages and takes $2maxhopcount \times t$ time.

References

[1] M.S. Corson, S.G. Batsell, A reservation-based multicast (RBM) routing protocol for mobile networks_initial route construction phase, ACM/Baltzer Wireless Networks 1 (4) (1995) 427–450.

[2] E.M. Belding-Royer, C.E. Perkins, Transmission range effects on AODV multicast communication, ACM/Kluwer Mobile Networks and Applications 7 (2002) 455–470.

[3] J. Xie, R.R. Talpade, A. Mcauley, M. Liu, AMRoute: adhoc multicast routing protocol, ACM/Kluwer Mobile Networks and Applications 7 (2002) 429–439.

[4] S.K.S. Gupta, P.K. Srimani, Cored-based tree with forwarding regions (CBT-FR), a protocol for reliable multicasting in mobile ad hoc networks, Journal of Parallel and Distributed Computing 61 (9) (2001) 1249–1277.

[5] S.J. Lee, M. Gerla, On-demand multicast routing protocol in multihop wireless mobile networks, ACM/Kluwer Mobile Networks and Applications 7 (2002) 441–453.

[6] J.J. Garcia-Luna-Aceves, E.L. Madruga, The core-assisted mesh protocol, IEEE Journal on Selected Areas in Communications 17 (1999) 1380–1394.

[7] R. Sivakumar, B. Das, V. Bharghavan, Spine routing in ad-hoc networks, Cluster Computing 1 (2) (1998) 237–248, a special issue on mobile computing.

[8] P. Sinha, R. Sivakumar, V. Bhanghavan, CEDAR: a core-extraction distributed ad-hoc routing algorithm, IEEE Journal on Selected Areas in Communications 17 (1999) 1454–1465.

[9] U.C. Kozat, G. Kondylis, B. Ryu, M.K. Marina, Virtual dynamic backbone for mobile ad-hoc networks, Proceedings of the IEEE International Conference on Communications 1 (2001) 250–255.

[10] P. Sinha, R. Sivakumar, V. Bhanghavan, MCEDAR: multicast core-extraction distributed ad-hoc routing, Proceedings of the IEEE Wireless Communications and Networking Conference (1999) 1313–1317.

[11] C. Jaikaeo, C.C. Shen, Adaptive backbone-based multicast for ad hoc networks, Proceedings of the IEEE International Conference on Communications 5 (2002) 3149–3155.

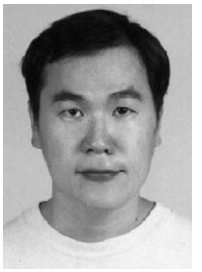
[12] A. Meyerson, Profit-earning facility location, in: Proceedings of the 33th Annual ACM Symposium on Theory of Computing, 2001, pp. 30–36.

[13] V.V. Vazirani, Approximation Algorithms, Springer-Verlag, Berlin Heidelberg, 2001.

[14] Network Simulator (Version 2). Available from: <http://www-mash.cs.berkeley.edu/ns/>.

[15] E.W. Dijkstra, A note on two problems in connection with graphs, Numerische Mathematik (1959) 269–271.

Chia-Cheng Hu received the B.S. degree from Chinese Naval Academy in 1988, and the M.S. degrees in Engineer Science from National Cheng Kung University in 1995, respectively. He is currently a Ph.D. candidate in Department of Computer Science and Information Engineering, National Taiwan University, Taiwan. His research interests include Wireless Networks, Mobile Computing and Combinatorial Optimization.



Eric Hsiao-Kuang Wu received his B.S. degree in Computer Science and Information Engineering from National Taiwan University in 1989. He received his Master and Ph.D. in Computer Science from University of California, Los Angeles (UCLA) in 1993 and 1997. He is an Associate Professor of Computer Science and Information Engineering at National Central University, Taiwan. His primary research interests include Wireless Networks, Mobile Computing and Broadband Networks. He is a member of IICM (Institute of Information and Computing Machinery) and IEEE.

Gen-Huey Chen was born in Taiwan in 1959. He received the B.S. degree in Computer Science from National Taiwan University in June 1981, and the M.S. and Ph.D. degrees in Computer Science from National Tsing Hua University, Taiwan, in June 1983 and January 1987, respectively. He joined the Faculty of the Department of Computer Science and Information Engineering, National Taiwan University, in February 1987, and he has been a Professor since August 1992. He received the Distinguished Research Award from the National Science Council, Taiwan in 1993, 1995, 1997, and 1999. His current research interests include graph theory and combinatorial optimization, graph-theoretic interconnection networks, wireless communication, parallel and distributed computing, and design and analysis of algorithms.