A Hierarchical Approach to Position-Based Multicast for Mobile Ad-hoc Networks

Matthias Transier*
University of Mannheim
Department of Computer Science
68131 Mannheim, Germany
transier@informatik.uni-mannheim.de
phone: +49 (621) 181-2608
fax: +49 (621) 181-2601

Holger Füßler
University of Mannheim
Department of Computer Science
68131 Mannheim, Germany
fuessler@informatik.uni-mannheim.de
phone: +49 (621) 181-2605
fax: +49 (621) 181-2601

Jörg Widmer†
DoCoMo Euro-Labs
Landsbergerstr. 312
80687 Munich, Germany
widmer@docomolabs-euro.com
phone: +49 (89) 56824-236
fax: +49 (89) 56824-300

Martin Mauve
University of Düsseldorf
Department of Computer Science
Universitätsstr. 1
Düsseldorf, Germany
mauve@cs.uni-duesseldorf.de
phone: +49 (211) 81-11636
fax: +49 (211) 81-11638

Wolfgang Effelsberg
University of Mannheim
Department of Computer Science
68131 Mannheim, Germany
effelsberg@informatik.uni-mannheim.de
phone: +49 (621) 181-2600
fax: +49 (621) 181-2601

In this paper we present Scalable Position-Based Multicast (SPBM), a multicast routing protocol for ad-hoc networks. SPBM uses the geographic position of nodes to provide a highly scalable group membership scheme and to forward data packets in a way that is very robust to changes in the topology of the network. SPBM bases the forwarding decision on whether or not there are group members located in a given direction, allowing a hierarchical aggregation of membership information. The farther away a region is from an intermediate node, the higher the level of aggregation for this region should be. Because of aggregation, the overhead for group membership management scales logarithmically with the number of nodes and is independent of the number of multicast senders for a given multicast group. Furthermore, we show that group management overhead is bounded by a constant if the frequency of membership updates is scaled down with the aggregation level. This scaling of the update frequency is reasonable since the higher the level of aggregation is, the lower the number of membership changes for the aggregate will be. The performance of SPBM is investigated by means of simulation, including a comparison with ODMRP, and through mathematical analysis. We also describe an open source kernel implementation of SPBM that has been successfully deployed on hand-held computers.

*Corresponding author

†Part of this work was done while J. Widmer was with the Institute of Communication Systems at EPFL, Switzerland.

1 Introduction

Many applications envisioned for mobile ad-hoc networks rely on group communication. Messaging during disaster relief efforts, networked games, and emergency warnings in vehicular networks are common examples of such applications. As a consequence, multicast routing in mobile ad-hoc networks has attracted significant attention over the recent years.

In this paper we present Scalable Position-Based Multicast (SPBM), an ad-hoc multicast routing protocol that comprises a multicast forwarding strategy and a group management scheme to determine where members of a multicast group are located. The forwarding strategy uses information about the geographic positions of group members to make forwarding decisions. In contrast to existing approaches, it neither requires the maintenance of a distribution structure (i.e., a tree or a mesh) nor resorts to flooding. The group management scheme uses knowledge about geographic positions to hierarchically aggregate membership information.

The forwarding of packets by means of SPBM is a generalization of position-based unicast routing as proposed, e.g., in [3] and [17]. In these protocols, a forwarding node selects one of its neighbors as a next hop in a greedy fashion, such that the packet makes progress towards the geographic position of the destination. It is possible that a node will have no neighbor with progress towards the destination, although a valid route to the destination exists. The packet is then said to have reached a local optimum. In this case, a *recovery strategy* is used to escape the local optimum and to find a path towards the destination. The most important characteristic of position-based routing is that forwarding decisions are based on local knowledge only. It is not necessary to create and maintain a global route from the sender to the destination. Thus, position-based routing is commonly regarded as being highly scalable and very robust against frequent topological changes. In order to extend position-based unicast routing to multicast, SPBM provides an algorithm for duplicating multicast packets at intermediate nodes if destinations for that packet are no longer located in the same direction. This algorithm includes both greedy forwarding and the recovery strategy.

The second important element of SPBM is its group management scheme. It relies on geographic information to achieve scalability: Instead of maintaining a fixed distribution structure, an intermediate node just needs to know whether or not group members are located in a given direction. This allows a hierarchical

aggregation of membership information: The farther away a region is from an intermediate node, the higher the level of aggregation for this region can be. Thus, group membership management can be provided with an overhead that scales logarithmically with the number of nodes and that is independent of the number of multicast senders in a multicast group. A second observation is then used to further reduce this overhead: The higher the level of aggregation is, the lower the frequency of membership changes for the aggregate will be. In SPBM, we therefore propose to scale down the frequency of membership update messages exponentially with the level of aggregation. This results in a constant upper bound on the overhead as the size of the network increases.

The remainder of this paper is structured as follows: In the following section, we discuss related work. We describe the SPBM protocol in Section 3. Section 4 contains simulation results on the performance of SPBM, as well as a comparison to ODMRP. In addition to that, Section 5 describes a Linux kernel implementation and experiences gained using it. Section 6 concludes the paper and gives an outlook on future work.

2 Related Work

Due to the extensive literature and the large number of protocol proposals in the area of mobile ad-hoc networks, we limit our discussion to work closely related to SPBM. We divide the related work into two main groups: topology-based ad-hoc multicast routing protocols (Section 2.1) and position-based ad-hoc routing protocols for unicast and multicast (Section 2.2).

2.1 Topology-Based Multicast Routing Protocols

Topology-based multicast protocols for mobile ad-hoc networks can be categorized into two main classes: tree-based and mesh-based protocols. The tree-based approaches build a data dissemination tree that contains exactly one path from a source to each destination. Topological information is used for its construction. The trees can be sub-classified further into source trees and shared trees. Representatives of the first are ABAM [34], MZR [8], DDM [15], and ADMR [13]. In these protocols, each single source builds its own tree to distribute its packets. In contrast to that, a shared tree is one in which each connected node is able

to send packets to all other nodes using the same tree. Shared trees are built among others by LAM [14], AMRoute [25], MAODV [32], and AMRIS [36]. Tree-based approaches often use local repair mechanisms to shield the distribution structure from link failures caused by mobility.

The second main category is mesh-based approaches, building meshes of data paths to make the multicast routes more stable against topological changes. This comes at the expense of a higher overhead during data delivery. A mesh can contain multiple possible paths from a source to a destination. Members of this class are CAMP [11], ODMRP [22], MCEDAR [33], NSMP [21], SRMP [28], and DCMP [7].

In the performance evaluation in Section 4, we compare our protocol to the On-Demand Multicast Routing Protocol (ODMRP), which has been shown to be a high-performance competitor [24]. ODMRP is a mesh-based protocol that can be regarded as a successor to FGMP [5]. ODMRP uses soft state information to manage forwarding and multicast group membership. Control packets, which optionally can contain a data payload, are periodically flooded through the entire network. An extension to the protocol allows it to exploit position information (if available) to predict node mobility. A distinctive feature is that the protocol can also be used for unicast routing, thus making an additional unicast protocol unnecessary.

2.2 Position-Based Unicast and Multicast Routing Protocols

Exploiting knowledge of a node's geographic position for data packet forwarding was first suggested some time ago [9]. Recently, position-based routing (PBR) has also been investigated for mobile ad-hoc networks and has led to several publications, surveys of which can be found in [27, 12, 19].

The forwarding decisions in position-based routing are usually based on the node's own position, the position of the destination, and the position of the node's direct radio neighbors. Since no global distribution structure—such as a route—is required, position-based routing is considered to be very robust to mobility. It typically performs best when the next-hop node can be found in a greedy manner by simply minimizing the remaining distance to the destination. However, there are situations where this strategy leads to a local optimum, and no neighbor can be found greedily to forward the packet further, although a route exists. In this case, a so-called recovery strategy is invoked. Among the protocols that utilize greedy forwarding and a recovery strategy are GPSR [17], face-2 [3], and GOAFR+ [20]. In addition to these purely position-based

algorithms, there are protocols that are position-aided and make use of position information to improve topology-based routing (e.g., LAR [18]).

Knowledge about the geographical position of nodes has been used in Dynamic Source Multicast (DSM) [1]. In DSM each node floods the network with information about its own position; thus each node knows the position of all other nodes in the ad-hoc network. The sender of a multicast packet then constructs a multicast tree from the position information of all receivers. This tree is encoded in the header of the packet. While DSM uses location information, the resulting distribution tree is determined completely by the sender. This eliminates the most important advantage of position-based routing. Due to periodic flooding of the network, the scalability of this approach is limited.

In [4], the authors report on "Location-Guided Tree Construction Algorithms", using the position of nodes to build an application-level distribution tree. This approach enjoys the benefits of position-based routing, but it is limited to receiver groups small enough to allow inclusion of the address of each destination in each data packet.

A generalization of position-based unicast forwarding is described in [26]. As for the "Location-Guided Tree Construction Algorithms", the sender includes the addresses of all destinations in the header of a multicast packet. In addition, the location of all destinations is included as well. It remains open how the sender is able to obtain the position information, and the scaling limitations seem to be similar to those discussed above.

In contrast to the existing position-based multicast protocols, SPBM retains the advantages of position-based routing while not being restricted to small receiver sets.

3 The Protocol

We now introduce the two building blocks of our algorithm. The *group management scheme* is responsible for the dissemination of the membership information for multicast groups, so that forwarding nodes know in which direction receivers are located. The *multicast forwarding algorithm* is executed by a forwarding node to determine which neighbors should receive a copy of a given multicast packet. This decision is based

Figure 1: Network represented by a quad-tree ($L = 3$)

on the information provided by the group management scheme. In the following, we assume that each node in the network is able to determine its own position, e.g., through the use of GPS.

3.1 Group Management

Position-based multicast requires that the forwarding nodes know the locations of the destinations. Including all of the destinations explicitly in the data packet header does not scale well as the size of the multicast group increases. To improve scalability, our proposal introduces hierarchical management of group memberships.

To this end, the network is subdivided into a quad-tree with a predefined maximum level of aggregation L. Figure 1 shows a quad-tree with four levels. Single squares are identified by their concatenated level-n to level-1 square numbers. In the example, the identifier "442" identifies a level-0 square that is located in the level-3 square comprising the whole network, in the level-2 square "4" and in the level-1 square "44". In level-0 squares, all nodes are within radio range of each other (i.e., level-0 squares have at most a diameter of the radio range).

3.1.1 Algorithm

The membership update mechanism aims to provide each node in the ad-hoc network with an aggregated view of the position of group members. For this purpose, each node maintains a global member table containing entries for the three neighboring squares for each level from level 0 up to level $(L - 1)$. In addition, each node has a local member table for nodes located in the same level-0 square.

Each entry in the global member table consists of the square's identifier and the aggregated membership information for all nodes in that square. Each entry in the local membership table consists of a node ID and the membership information for that node. Membership information is stored and transmitted as membership bit vectors. For the sake of simplicity, we assume that each bit represents one multicast group. A bit that is set to 1 indicates group membership. This encoding enables us to accommodate 256 groups in 32

Table 1: Global and local member table of a node located in square “442”

Square	Groups
1	00011100
2	01000100
3	10100010
41	01010000
42	00010101
43	00100100
441	00000100
443	00010000
444	00100100

Node	Groups
14	00000001
23	01000100
51	00000100

bytes. If there is a need for a much higher number of groups, a Bloom filter [2] scheme could be applied, in which the number of encodable groups is higher than one per bit, trading on the possibility of false positives, i.e., a multicast group could be falsely believed to have receivers in a square.

With the simple bit vector scheme, the amount of state maintained in a node scales logarithmically with the size of the network. Table 1 shows an example of a node located in square “442” with a membership vector length of 8. In this example, the first entry in the global member table can be interpreted as follows: There is at least one multicast receiver for groups 3, 4 and 5 located in the level-2 square “1”. The first entry in the local member table contains the information that node 14 is in the same level-0 square as the node maintaining the table, and that 14 is member of group 7.

A node indicates its group membership status by broadcasting *announce* messages within its level-0 square (i.e., to its direct neighbors). An announce message contains the ID of the node, its position, and a membership vector describing its subscribed groups. Announce messages are broadcast periodically, but do not need to be forwarded by any other node since all nodes within the same level-0 square are within radio range of each other. These messages replace the beacon messages of position-based routing.

A node stores the membership information for all nodes in its level-0 square. Update messages are then used to provide all nodes that are located in a level-1 square with the aggregated membership information for the four level-0 squares contained in the level-1 square. This is done by periodically selecting one node in each level-0 square. For now, we assume that such a selection mechanism is in place. We will show later how it can be realized by means of random timers. The selected node floods the level-1 square

with an update message that includes the ID of the selected node, a membership vector describing the aggregated group membership information, the identifier of the destination square that is to be flooded, and a sequence number for duplicate message detection. The aggregation is done by a bitwise OR-operation on the membership vectors of the nodes located in the level-0 square. In order to perform flooding, each node in the level-1 square forwards this message once. Thus, a total of four update messages will be flooded in each level-1 square per period: one for each level-0 square. In the example, one node in each square "441", "442", "443", and "444" is selected. Those nodes aggregate their level-0 membership information and flood them in an update packet in the level-1 square "44".

The same mechanism is used to aggregate the membership information from an arbitrary level-λ square and flood it in the area of a level-$(\lambda + 1)$ square. In the example, one node in each square "41", "42", "43", and "44" were selected to aggregate their level-1 membership information and flood an update message in square "4". If the node with the membership tables depicted in Table 1 would be selected for square "44", it would perform the aggregation by a bitwise OR-operation on the membership vectors for the individual nodes 14, 23, and 51, and on the aggregated information from the level-0 squares "441", "443", and "444".

Since the size of a square increases exponentially with each level, the likelihood that the aggregated group membership information will change in a given time-span decreases rapidly. We therefore propose to decrease the frequency of flooding membership information exponentially with the level of aggregation. Let f_0 be the frequency of announce messages. Then the frequency f_λ of update messages from a single square on level λ is defined as follows:

$$f_\lambda = q^\lambda \cdot f_0 \qquad \text{for} \qquad \lambda = 1, \ldots, L \quad \text{and} \quad 0 < q \leq 1,$$

where q is a chooseable factor that determines the decrease of the update frequency from one level to the following.

It remains to be shown how one node is selected to send an update message. The selection mechanism is performed by random timers. Every node maintains an update timer for each level. When the timer expires, the node is selected, the update message for the appropriate level is transmitted, and the timer is reset. When a node receives an update message for a square to which it belongs, its timer is reset without sending the

packet, thereby suppressing the transmission of the update message. The main component of each timer is determined by the update frequency of that level. In order to prevent all nodes in a given square from flooding the same update information simultaneously, each timer has also a random exponential element. It was shown in [29] that exponentially distributed timers lead to good suppression ratios, even for large groups of nodes. To generate an exponential distribution, we use the inverse transform of the corresponding distribution function and feed it with random values x which are uniformly distributed between 0 and 1:

$$f(x) = -\frac{T}{\beta} \cdot \ln\left(x \cdot \left(e^{\beta} - 1\right) + 1\right) \qquad \text{for} \qquad 0 \leq x \leq 1, \tag{1}$$

where T determines the interval from which the timers are chosen, and β is a parameter that, while increasing, shifts the weight of the resulting distribution towards the end of the interval. The total runtime of a timer for a given level λ is then chosen as follows:

$$t(x) = \left(\frac{1}{f_0} + \frac{T}{\beta} \cdot \ln\left(x \cdot \left(e^{\beta} - 1\right) + 1\right) - E[M]\right) \cdot \left(\frac{1}{q}\right)^{\lambda} \qquad \text{for} \qquad 0 \leq x \leq 1.$$

The first part of this equation, $\frac{1}{f_0}$, describes the basic update interval at level 0. Added is an exponentially distributed random value between 0 and T. The parameters T and β can be used to tune the probability of collisions. Lower values for both of them lead to tightly chosen timers and thus likely result in collisions, whereas higher values increase the expected value and thus create additional delay. [29] suggests β between 0 and 10, T has been set to half of the basic update interval which is $\frac{1}{2f_0}$. Having chosen the parameters, the expected value of the minimal timer can be estimated as, according to [29],

$$E[M] = T \cdot \int_0^1 \left(1 - \frac{e^{\beta m} - 1}{e^{\beta} - 1}\right)^R dm,$$

where R is the number of competing nodes. The resulting value is then scaled by the factor $\frac{1}{q}$, powered by the level λ. Thus, the exponential part also depends on the level, and therefore on the area in which the nodes should be suppressed. Through the exponential distribution, the probability of having a short timeout value is much less than the probability of a high timeout value. Thus, the vast majority of timers will not expire before an update message from another node has been received. Note that the largest part of the timer

is deterministic. The random component used for the selection process therefore has no significant impact on the frequency with which squares are flooded. Given a constant node density it can be shown that the amount of data transmitted per m^2 by this group management scheme is bounded by a small constant, as the size of the network increases to infinity.

3.1.2 Scalability Analysis

The group management algorithm is proactive; thus, its overhead is independent of actual data traffic and the number of senders in a given multicast group. In the following, we quantify this overhead to examine the algorithm's performance and scaling characteristics.

Let the radio range r be constant. To ensure connectivity within level-0 squares (under the assumption of a unit disk graph), the area A_0 of level-0 squares is:

$$A_0 \leq \frac{r^2}{2}$$

and the area covered by the network with respect to the number of levels can be determined as:

$$A(L) = A_0 \cdot 4^L.$$

We need to determine how often a level-0 square is flooded with update messages from all levels in a fixed amount of time. In a first step, let us consider the case that $q = 1$ and therefore the update frequency is the same for all levels. Then, at level 0, four update messages are generated by four squares which form a level-1 square. These messages are received by each node within the level-1 square. The same holds for each level from 1 up to $L - 1$. Thus, the overhead c linearly depends on the number of levels L. If we quadruple the area of the network, thereby increasing the number of levels by one, each single lowest-level square has to be flooded with four more messages. This means that a multiplication of the size of the network area A stresses only a single node with a constant additional load.

Considering the spatial frequency reuse occurring in a network of growing (area) size, we study the overhead per area. In terms of complexity, the total cost c[1] per area in the network conforms to

$$\frac{c(A(L))}{A(L)} = O\left(\log A(L)\right),$$

which means that the cost per area only grows logarithmically with the size of the area.

More generally, if we allow $0 \leq q \leq 1$, and if n is the number of nodes in the network, then the total cost through update messages in the network c is

$$c(L) = n \cdot f_0 \left(1 + 4 \sum_{\lambda=1}^{L} q^{\lambda}\right), \tag{2}$$

where again f_0 is the frequency of the update messages at level 0 and q the factor which decreases this frequency from one level to the next. For a proof of Equation (2), see appendix.

If $0 < q < 1$, the sum in Equation (2) represents a geometric series which has an upper limit for all values of L. Thus, for $q < 1$, the total cost per area within the network is bounded by a small constant number of update messages per time when growing the area of the network:

$$\frac{c(A(L))}{A(L)} = O(1) \qquad \text{if} \quad q < 1.$$

This is also shown in the appendix.

On the other hand, if $q < 1$, the worst-case join latency grows exponentially with the number of levels, or linearly with the network size. This worst case occurs if a node joins a group where a sender resides in a different level-L square and the following situation arises: The node has just sent an update message at level 0 and, for each other level λ, the update at level $\lambda + 1$ has been sent just before the update at level λ arrives. Thus, at each level a full update period elapses before the membership is propagated the next level up. All timer durations summed up count towards the latency until the join is completed. This worst-case join latency has to be accepted in favor of the good scaling properties of the network load for larger

[1]This cost metric assumes constant-length update messages. If we increase the number of multicast groups, the length of the messages grows linearly.

networks. The average join delay, however, is much shorter since the worst case occurs only if a joining receiver is the first receiver of a certain group in a highest-level square. The membership information has to be propagated only up to the level where other receivers have already subscribed. For dynamic groups this means that high latencies do not occur if there is always a certain number of receivers distributed in the area where the group communication takes place.

The same holds for mobile receivers. A receiver which enters a square without any subscribed nodes has to wait until its membership has been propagated at least to the level where the senders or further receivers of this group are. However, if a receiver crosses a square boundary to a square where other nodes are already subscribed, it will be able to receive multicast messages immediately after sending its first announce beacon.

3.2 Multicast Forwarding

To deliver multicast packets from a source to the subscribed group members, the nodes use the information stored in their member tables. By dividing the network into a quad-tree, geographic regions are built which can be used to aggregate multicast traffic to group members located geographically close to each other.

The forwarding decision is based on information about neighboring nodes. Each node maintains a table of nodes in its transmission range. This is accomplished by having each node periodically broadcast beacon messages containing the ID and position of the node. Beacon messages are not forwarded by the receiving nodes.

Algorithm 1 shows how forwarding works. As an input, the algorithm requires the current node n, the packet p, and the list of neighbors N of n. The packet includes a list of destinations, which initially contains one entry that comprises the whole network, and a group address indicating the group to which the packet is being sent. When the algorithm is invoked, it first checks whether the current node n is a member of the multicast group the packet is being sent to. If this is the case, the packet is delivered.

In the next step, the algorithm looks at each entry in the list of destinations in the packet: If the global or the local membership tables contain a de-aggregation of the entry, then the entry is subdivided into those squares of the next lower level that include members for the group the packet is being transmitted to. At

Algorithm 1 The forwarding algorithm

Require: node n, packet p, list of neighbors N
  if $n \in receivers(group(p))$ **then**
    $deliver(p)$
  end if
  $D \leftarrow \emptyset$
  for all $d \in destinations(p)$ **do** {for all destinations packet p is addressed to}
    if $mysquare \subseteq d$ **then** {$mysquare$ is the current level-0 square of node n}
      $D \leftarrow D \cup subdivide(d)$ {replace destination d by its subsquares}
    else
      $D \leftarrow D \cup \{d\}$
    end if
  end for
  $F[N] \leftarrow \emptyset$ {stores the destinations for a given next hop}
  for all $d \in D$ **do**
    $\nu \leftarrow \emptyset$ {stores the next hop}
    if $recover(d)$ **then** {is destination d in recovery mode?}
      $\nu \leftarrow rightHand(prevHop,\ d)$
    else
      $\nu \leftarrow forwardGreedy(N,\ d)$
    end if
    if $\nu = \emptyset$ **then**
      $\nu \leftarrow rightHand(n,\ d)$
      if $\nu = \emptyset$ **then**
        $drop(d)$
      end if
    end if
    $F[\nu] \leftarrow F[\nu] \cup \{d\}$
  end for
  for all $\nu \in N$ **do**
    if $F[\nu] \neq \emptyset$ **then**
      $send(p,\ \nu,\ F[\nu])$ {send packet p to neighbor ν for destinations $F[\nu]$}
    end if
  end for

Figure 2: Forwarding on the quad-tree

level-0, a de-aggregation is performed by replacing the square with the IDs of the nodes that are group members.

Consider, for example, the situation where a node in square "442" (see Figure 1) sends a multicast packet to the group number 1. It initializes the packet with the whole network as the single destination area and sets the multicast address to 1. The packet is then handed to the forwarding algorithm. After checking whether the current node is a receiver of multicast group 1, the destinations are de-aggregated. Based on the membership tables given in Table 1 for multicast group 1, the complete network can be de-aggregated into the level-2 square "2" (since bit 1 of the membership vector is set), the level-1 square "41", and the individual node 23 in the same level-0 square as the forwarding node.

After de-aggregation of the destinations, it is checked which neighbor is best suited to forward the packet to each destination. This is done in a fashion similar to position-based unicast routing (see [27]): In order to determine the most suitable next hop for a packet and a given destination, the source compares the geographic progress for each of the neighbors in respect to the destination and picks the neighbor with the greatest progress. If the destination is a square, the position of the nearest point in that square is used as the destination position.

After finding the next hop for each destination, the current node n makes a copy of the data packet for each of these next hops. In the list of destinations, it enters a list of the destinations which shall be reached through this specific next hop, and sends the packet to the next hop by using unicast transmission. The use of unicast increases the reliability of data delivery at the expense of bandwidth utilization, as each copy of the packet will be acknowledged on the MAC layer, but at the cost of multiple messages.[2]

Figure 2 shows an example of the forwarding procedure.[3] Node A wants to send a packet to the group of which nodes C, E and F are members. Thus A's member table contains the information that there is at least one receiver in square "4". It sends the packet in this direction, and node B is the first node located in the level-2 square "4". Consequently, it has the information that there are nodes subscribed to the group in the

[2]This is a design decision; depending on the application and the environment of the ad-hoc network one may choose to transmit the packet using broadcast.

[3]The figure only depicts nodes which are involved in the process of refining the destination square information.

level-1 squares “43” and “44”. It therefore updates the information in the packet header accordingly. Node C is the first forwarding node in square “43”. Besides delivering the packet, it checks its member table and recognizes that it does not need to forward the packet to any additional receivers in square “43”. In square “44”, node D replaces square “44” in the packet header with the level-0 squares “441” and “444”. After receiving the packet, nodes E and F replace their square with potential additional destination nodes in this square. If there are any, the packets will now be sent directly to the receivers since the radio ranges of E and F cover the complete squares “441” and “444”, respectively.

3.2.1 Recovery from Greedy Failures

If, for one or more destinations, a forwarding node does not find a next hop that yields geographic progress, a recovery strategy has to be employed. Similar to position-based unicast routing [17, 3], SPBM uses a distributed planarization of the network graph combined with the right-hand rule to route around void regions. If there is a destination with no suitable next hop, the algorithm first planarizes the surrounding network graph. Then, the node determines the angles counter-clockwise between the line from the node to the destination, and the line from the node to each remaining neighbor. The neighbor with the smallest angle is chosen as the next hop. This destination is marked as a *recovery destination*, and the current position is stored in the packet in order to inform the following hops about the position where the recovery mechanism started. The chosen next hop is then handled as a normal destination.

A node which receives a packet containing a recovery destination first checks whether it itself is located closer to the destination than the position which is stored in the packet as the recovery starting point. The destination is always known by every node in the network since the recovery mode is only needed for destination *squares*, whose positions are known by definition. In this case, regular forwarding is resumed. If this is not the case and the node is located farther away from the destination than the recovery starting point, the node has to continue the recovery process. After performing planarization, it chooses the next hop by using the right-hand rule.

Figure 3: Recovery from a greedy failure

The recovery strategy works independently of the grid structure. As long as a destination is marked as a recovery destination, it is not necessary to change or replace it, because only the nodes at the destination have enough information to refine the destination square.

Figure 3 shows an example of the case in which one destination cannot be reached using a greedy strategy. In this example, a packet arriving at node C is addressed to the three destination squares I, II, and III, which are depicted by the shaded areas. Direct communication is only possible where the nodes are connected by a line. Thus, node C selects B as a forwarder to destination III, and D as a forwarder to destination I. Since there is no node with geographical progress towards destination II, this destination has to be handled as a recovery destination. By applying the right-hand rule, node D is selected as the next hop to destination II. In the following, C sends a packet to node D addressed to the destinations I and II, where II is marked as a recovery destination. Destination III is reached in a greedy fashion and shall not be part of this example from now on. Node D now checks whether it is closer to the recovery destination II than the node which put the destination into this mode. Since it is not, destination II remains in recovery mode. The next hop according to the right-hand rule is node F, whereas destination I can be reached greedily via node E. The packet from D to F contains only the recovery destination II. Arriving at node F, the packet is closer to the destination than at the point where the recovery mode was started. Hence, it can be switched back into greedy mode and routed to K via H.

4 Simulations

4.1 Simulation Setup

The simulations were performed using the network simulator ns-2 [30]. As a reference, the ODMRP implementation from [31] was chosen and ported to ns-2.27. We discovered and rectified some misbehaviors of this implementation: First, the calculation of the header sizes was corrected; second, the delay for join queries was changed so as to contain a constant part in addition to the random back-off. This reduces the overhead by about 10% without affecting the packet delivery ratio. Third, according to the ODMRP

draft [23], duplicate join queries and join replies are suppressed, further reducing the overhead of the protocol.

The MAC layer in all simulations was IEEE 802.11, with a maximum bandwidth of 2 MBit/s. The transmission power resulted in a radio range of 250 meters. Since the transmitted packets were relatively small, the use of RTS/CTS was disabled. All runs were simulated 20 times with different random seed values and in different movement scenarios; we report on the average of those runs. A run represents the simulated time of 180 seconds, where nodes joined at the beginning of the simulation, and the first data packet was sent after 60 seconds in order to give the group management enough time to initialize. The data payload had a size of 64 bytes per packet, and each source transmitted one packet per second.

The protocol-specific parameters of SPBM were: The basic group membership update frequency for an order-0 square f_0 was set to $\frac{1}{3\ s}$. The value for the timeout of entries in the member table was 2.5 times the corresponding update interval. ODMRP's protocol-specific parameters were: a join refresh interval of 3 seconds, an acknowledgment timeout for join table messages of 25 milliseconds, and a maximum number of join table transmissions of 3. To improve comparability, all these protocol-specific parameters were kept constant throughout all simulations.

Some other simulation parameters were varied to investigate their influence on the results. During each series of simulation runs, only one parameter was changed, leaving the others constant. In particular the varied parameters were: The modeled scenarios were squares of 350 meters by 350 meters to 2800 meters by 2800 meters, where 100 nodes per square kilometer moved according to the random way-point model [16], with a pause time of 10 seconds and a minimum speed of 1 meter per second. Mobility varied from 0 to 15 meters per second. The number of senders ranged from 1 to 15 and the number of receivers from 5 to 25. While all senders and receivers belonged to one multicast group, they were disjoint.

4.2 Performance Metrics

The metrics used to evaluate the protocol performance are packet delivery ratio, overhead, and delay. The *packet delivery ratio* (PDR) is defined as the sum of all unique data packets received, divided by the sum of all data packets that should have been delivered (sum of sent packets multiplied by the number of receivers).

The *overhead* is the total number of bytes transmitted at the MAC layer, including acknowledgments in the case of unicast transmissions. To measure the overhead on the MAC layer, it is necessary to capture MAC layer retries induced by mobility or packet collisions. These effects would be invisible if the overhead were counted at the network layer.

The *delay* is defined as the interval that elapses between the time a packet is sent and the time at which the packet is successfully delivered. This value is averaged over all packets and all receivers.

4.3 Results

4.3.1 Number of Senders

(a) Packet delivery ratio

(b) Bytes transmitted on MAC layer

(c) Delay

Figure 4: Performance w.r.t. number of senders (25 receivers, 1 Pkt/s, 5 m/s, 1400 m $\times$ 1400 m, 100 nodes/km^2)

Figure 4 shows the respective PDR, overhead, and delay for an increasing number of senders. The other parameters were kept constant in this setup. While the PDR of SPBM is quite stable for different numbers of senders (up to 15 in these experiments), ODMRP suffers from the load generated by the additional senders. This is due to the fact that each sender floods the entire network with data and control packets at regular intervals in order to build its forwarding group. The group management of SPBM is independent of the existing multicast sources. If only one sender is active, the network load induced by ODMRP is lower than in SPBM. This is because the proactive group management of SPBM is responsible for a certain constant overhead. For ODMRP, the high increase in load is accompanied by a high decrease in the ratio of delivered packets.

SPBM, in contrast, sustains a satisfactory packet delivery ratio. The increase in overhead is mainly due to the increased number of data-forwarding operations for the data packets of the additional senders. The

proactive group management overhead of SPBM remains constant. A similar result was achieved when varying the number of receivers while keeping the number of senders constant. In this case, ODMRP quickly saturates the network, resulting in a constantly high network load, while SPBM still operates with a satisfactory packet delivery ratio with a load increase caused mainly by the higher number of forwarding operations.

Regarding the end-to-end-delay (Figure 4(c)), the results show that ODMRP performs slightly better than SPBM for a small number of senders. Since ODMRP's forwarding algorithm is a form of scoped flooding, and the delay is measured as the first copy of a certain packet arrives, ODMRP is able to use the direct route from source to each destination. At the same time, the overhead introduced through the scoped flooding leads to a steep increase in the delay once the network becomes saturated due to the increase in senders.

4.3.2 Node Mobility

(a) Packet delivery ratio

(b) Bytes transmitted on MAC layer

Figure 5: Performance w.r.t. maximum movement speed (15 senders, 1 Pkt/s, 25 receivers, 100 nodes/km^2)

Figure 5 shows the impact of node mobility on the packet delivery ratio and the bytes transmitted at the MAC layer. While SPBM performs very well for low-to-medium node mobility, the packed delivery ratio drops significantly at high node speeds. Further investigation revealed two reasons for this behavior: (1) When group members cross square "boundaries" into a square that did not previously contain a group member, they will not receive packets until the group management scheme has spread the new information. (2) When node mobility increases, forwarding failures appear that are induced by discrepancies in the neighbor table used for the next-hop selection. If a node is selected as a forwarder but has moved out of radio range, the current forwarder has to wait for four unsuccessful retransmissions followed by a link layer notification

before it is able to select a different node[4]. This reduces the packet delivery ratio and increases the number of MAC packets transmitted. To avoid this problem, we have conducted preliminary experiments to adapt the ideas of contention-based forwarding, as described in [10], to SPBM. Moreover, we specifically managed situations where nodes crossed square boundaries. These experiments indicate that the modifications will make both the delivery rate and the number of transmitted packets largely independent of node mobility.

4.3.3 Network Size

(a) Packet delivery ratio

(b) Bytes transmitted on MAC layer

Figure 6: Performance w.r.t. network size (3 senders, 1 Pkt/s, 10 receivers, 100 nodes/km^2)

In these experiments we varied the size of the network from $350m \times 350m$ to $2800m \times 2800m$. This corresponds to L (numbers of levels) from 1 to 4. The node density was left constant, as was the number of senders and receivers. Thus, the number of nodes ranged from 13 for $L = 1$ to 784 for $L = 4$. Figure 6(a) indicates that the packet delivery ratio is independent of the number of levels (or the network area). As expected, the network load grows linearly with the area of the network (see Figure 6(b)).

5 Linux Implementation

In order to perform experiments with a real system, we implemented SPBM as a Linux kernel module [35]. To receive incoming packets, the module registers a new layer-3 protocol defining a protocol number for SPBM. Every incoming packet containing the proper protocol number in its protocol field is directly delivered to the SPBM module.

[4]This effect has been extensively described in [10].

Figure 7: Setup for the real world test

Outgoing packets generated at the local host are captured via the netfilter interface at the `NF_IP_POST_ROUTING` hook (see [6]) in order to analyze their destination. If a packet is addressed to a multicast group, it is directed to the SPBM module.

There are three subtypes of SPBM packets: beacons, update messages, and data packets. If a node receiving an SPBM data packet is a member of the destination group, the module injects the packet back into the protocol stack as if it had been received directly from the network interface.

The module uses the proc interface of the kernel to communicate with programs in the user space. Within the directory `/proc/spbm`, there are different virtual files through which the user or program can control the behavior of the module. Table 2 lists these files and their functions.

Table 2: Entries in the /proc/spbm directory

Entry	Read from file	Write to file
pos	get position	set position
join	get subscribed groups	join a group
leave	–	leave a group
neighbortable	get current neighbors	–
membertable	get current member table	–

Within the module, a virtual coordinate system is used. It extends to 16 bits in x-direction and 16 bits in y-direction. The current position has to be fed to the module via the `/proc/spbm/pos` file as a string of two space-separated 16-bit hexadecimal values.

The mapping of real to virtual coordinates is done by a user-space positioning daemon. This gives a high grade of flexibility regarding the positioning system used. The daemons on each node have to be configured to provide a consistent coordinate system. E.g., the GPS coordinates have to be mapped to identical virtual coordinates on every node.

The SPBM implementation has been installed and tested both on laptop computers and iPaq hand-held computers. It is available for download from our website at http://www.informatik.uni-mannheim.de/pi4/projects/pbm/.

In the experiment, the nodes were "virtually" located as depicted in Figure 7. In order to enable reproducible experiments, the physical location of the nodes was directly next to each other. The topology was enforced by filtering packets from nodes with a virtual position beyond the transmission range, as depicted by circles in Figure 7. This setup increased the congestion level of the network since all nodes are in interference range of each other.

During each experiment, we transmitted packets from node A to a multicast group that was joined by all other nodes. Group membership management, beaconing, and data forwarding were performed according to the SPBM algorithms as defined above. The sending rate of node A was limited only by the rate accepted by the MAC of node A; the size of the data payload was set to 1000 bytes; IEEE802.11b was set to 11 MBit/s, thus about 2.2 MBit/s gross for each link in Figure 7. The experiment was conducted 10 times. As a result, all nodes B through F, which were iPaq 3660 devices, received data at the rate of 408 kBit/s on the average, whithout any packet loss. The latter was to be expected since there was no node mobility and all transmissions of data packets were performed using unicast and MAC-level retransmissions. We are currently planning more experiments, including node mobility.

6 Conclusions and Future Work

We described in this paper a novel position-based ad-hoc multicast routing protocol. It differs significantly from previous work in that it introduces a hierarchical organization of nodes for membership management, as well as for packet forwarding. By means of simulation we demonstrated that SPBM performs very well, particularly as the number of multicast senders and receivers increases.

Our main priority for future work is to integrate contention-based forwarding and a management scheme for nodes crossing square boundaries into SPBM. First results show that these mechanisms are able to eliminate the impact of very high node mobility on the performance of SPBM.

To summarize, we are convinced that a hierarchical approach to position-based multicast is a very promising solution if the protocol is intended to scale to a reasonable number of nodes.

References

[1] S. Basagni, I. Chlamtac, and V. R. Syrotiuk. Location aware, dependable multicast for mobile ad hoc networks. *Computer Networks*, 36(5–6):659–670, August 2001.

[2] B. H. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM*, 13(7):422–426, July 1970.

[3] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc Wireless Networks. In *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications (DIAL-M '99)*, pages 48–55, Seattle, WS, August 1999.

[4] K. Chen and K. Nahrstedt. Effective Location-Guided Tree Construction Algorithms for Small Group Multicast in MANET. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, pages 1192–1201, New York City, New York, June 2002.

[5] C.-C. Chiang, M. Gerla, and L. Zhang. Forwarding Group Multicast Protocol (FGMP) for Multihop, Mobile Wireless Networks. *ACM-Baltzer Journal of Cluster Computing: Special Issue on Mobile Computing*, 1(2):187–196, December 1998.

[6] J. Crowcroft and I. Phillips. *TCP/IP and Linux Protocol Implementation*. John Wiley & Sons, 2002.

[7] S. K. Das, B. S. Manoj, and C. S. R. Murthy. A Dynamic Core Based Multicast Routing Protocol for Ad hoc Wireless Networks. In *Proceedings of the third ACM international symposium on Mobile and ad hoc networking & computing (MobiHoc '02)*, pages 24–35, Lausanne, Switzerland, June 2002.

[8] V. Devarapalli and D. Sidhu. MZR: A Multicast Protocol for Mobile Ad Hoc Networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, volume 3, pages 886–891, Helsinki, Finland, June 2001.

[9] G. G. Finn. Routing and addressing problems in large metropolitan-scale internetworks. Technical Report ISI/RR-87-180, ISI, March 1987.

[10] H. Füßler, J. Widmer, M. Käsemann, M. Mauve, and H. Hartenstein. Contention-Based Forwarding for Mobile Ad-Hoc Networks. *Elsevier's Ad Hoc Networks*, 1(4):351–369, 2003.

[11] J. J. Garcia-Luna-Aceves and E. L. Madruga. A Multicast Routing Protocol for Ad-Hoc Networks. In *Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 1999)*, pages 784–792, New York City, New York, March 1999.

[12] S. Giordano, I. Stojmenovic, and L. Blažević. Position-Based Routing Algorithms for AdHoc Networks: A Taxonomy. In D.-Z. Du, editor, *Ad Hoc Wireless Networking*. Kluwer Academic Publishers, November 2003.

[13] J. G. Jetcheva and D. B. Johnson. Adaptive Demand-Driven Multicast Routing in Multi-Hop Wireless Ad Hoc Networks. In *Proceedings of the second ACM international symposium on Mobile and ad hoc networking & computing (MobiHoc '01)*, pages 33–44, Long Beach, California, October 2001.

[14] L. Ji and M. S. Corson. A Lightweight Adaptive Multicast Algorithm. In *Proceedings of IEEE Global Telecommunications Conference (Globecom1998)*, pages 1036–1042, Sydney, Australia, November 1998.

[15] L. Ji and M. S. Corson. Differential Destination Multicast—A MANET Multicast Routing Protocol for Small Groups. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001)*, pages 1192–1202, Anchorage, Alaska, April 2001.

[16] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In T. Imielinski and H. Korth, editors, *Mobile Computing*, volume 353, pages 153–181. Kluwer Academic Publishers, 1996.

[17] B. N. Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *Proceedings of the sixth annual ACM/IEEE International Conference on Mobile computing and networking (MobiCom '00)*, pages 243–254, Boston, Massachusetts, August 2000.

[18] Y.-B. Ko and N. H. Vaidya. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. In *Proceedings of the fourth annual ACM/IEEE International Conference on Mobile computing and networking (MobiCom '98)*, pages 66–75, Dallas, Texas, October 1998.

[19] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric Ad-Hoc Routing: Of Theory and Practice. In *Proceedings of the 22nd ACM Annual Symposium on Principles of Distributed Computing (PODC '03)*, pages 63–72, Boston, MA, July 2003.

[20] F. Kuhn, R. Wattenhofer, and A. Zollinger. Worst-Case optimal and average-case efficient geometric ad-hoc routing. In *Proceedings of the fourth ACM international symposium on Mobile and ad hoc networking & computing (MobiHoc '03)*, pages 267–278, Annapolis, Maryland, June 2003.

[21] S. Lee and C. Kim. Neighbor Supporting Ad hoc Multicast Routing Protocol. In *Proceedings of the first ACM international symposium on Mobile and ad hoc networking & computing (MobiHoc '00)*, pages 37–44, Boston, Massachusetts, August 2000.

[22] S.-J. Lee, M. Gerla, and C.-C. Chiang. On-Demand Multicast Routing Protocol. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC '99)*, pages 1298–1302, New Orleans, LA, September 1999.

[23] S.-J. Lee, W. Su, and M. Gerla. On-Demand Multicast routing Protocol (ODMRP) for Ad Hoc Networks. Internet Draft, draft-ietf-manet-odmrp-02.txt, work in progress, 2000.

[24] S.-J. Lee, W. Su, J. Hsu, M. Gerla, and R. Bagrodia. A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols. In *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2000)*, pages 565–574, Tel Aviv, Israel, March 2000.

[25] M. Liu, R. R. Talpade, A. McAuley, and E. Bommaiah. AMRoute: Adhoc Multicast Routing Protocol. Technical report, CSHCN, University of Maryland, 1999.

[26] M. Mauve, H. Füßler, J. Widmer, and T. Lang. Position-Based Multicast Routing for Mobile Ad-Hoc Networks. Technical Report TR-03-004, Department of Computer Science, University of Mannheim, 2003.

[27] M. Mauve, J. Widmer, and H. Hartenstein. A Survey on Position-Based Routing in Mobile Ad-Hoc Networks. *IEEE Network*, 15(6):30–39, November/December 2001.

[28] H. Moustafa and H. Labiod. SRMP: A Mesh-based Protocol for Multicast Communication in ad hoc networks. In *2002 International Conference on Third Generation Wireless and Beyond*, pages 43–48, San Francisco, CA, May 2002.

[29] J. Nonnenmacher and E. W. Biersack. Scalable Feedback for Large Groups. *IEEE/ACM Transactions on Networking (TON)*, 7(3):375–386, 1999.

[30] The ns-2 network simulator. http://www.isi.edu/nsnam/ns/.

[31] Wireless multicast extensions for ns-2.1b8. http://www.monarch.cs.rice.edu/multicast_extensions.html.

[32] E. M. Royer and C. E. Perkins. Multicast Operation of the Ad-hoc On-Demand Distance Vector Routing Protocol. In *Proceedings of the fifth annual ACM/IEEE International Conference on Mobile computing and networking (MobiCom '99)*, pages 207–218, Seattle, Washington, August 1999.

[33] P. Sinha, R. Sivakumar, and V. Bharghavan. MCEDAR: Multicast Core-Extraction Distributed Ad hoc Routing. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC '99)*, pages 1313–1317, New Orleans, LA, September 1999.

[34] C.-K. Toh, G. Guichal, and S. Bunchua. ABAM: On-Demand Associativity-Based Multicast Routing for Ad Hoc Mobile Networks. In *Proceedings of the IEEE 50th Vehicular Technology Conference (VTC'00)*, pages 987–993, Boston, MA, September 2000.

[35] M. Transier, H. Füßler, T. Butter, and W. Effelsberg. Implementing Scalable Position-Based Multicast for the Linux Kernel. In *Proceedings of the 2nd German Workshop on Mobile Ad-hoc Networking (WMAN 2004)*, Ulm, Germany, September 2004.

[36] C. W. Wu and Y. C. Tay. AMRIS: A Multicast Protocol for Ad Hoc Wireless Networks. In *Proc. of IEEE Military Communications Conference (MILCOM)*, pages 25–29, Atlantic City, NJ, November 1999.

Appendix

Theorem 1 (Cost Function)**.** *Consider an ad-hoc network of square geometry and n the number of nodes. Let L be the maximum hierarchy level, $0 < q \leq 1$ be the timer frequency coefficient, and f_0 the smallest-square frequency. Then the average number of (proactive) radio transmissions per time of the SPBM group management protocol is given as*

$$\begin{aligned} c &= nf_0 \left(1 + 4\sum_{\lambda=1}^{L} q^{\lambda}\right) \\ &= \begin{cases} nf_0\,(1+4L) & q = 1 \\ nf_0\left(1 + 4\left(\frac{1-q^L}{1-q}\right)\right) & 0 < q < 1. \end{cases} \end{aligned} \tag{3}$$

Proof. Be c_λ the average number of transmissions per second on level λ ($\lambda = 0,\ \ldots,\ L$) for the whole network. At level 0, each node sends f_0 packets per second. Thus

$$c_0 = nf_0.$$

At every higher level λ ($\lambda = 1,\ \ldots,\ L$) $4^{L-\lambda}$ squares exist, each with $\frac{n}{4^{L-\lambda}}$ nodes on average. At a frequency of f_λ, one of the nodes of each square at level λ sends update packets. Each of these packets is relayed by all nodes in the 4 adjacent squares of level λ which belong to the same square of level $\lambda + 1$. This induces $\left(4\frac{n}{4^{L-\lambda}}\right)$ packet transmissions for each square of level λ:

$$c_\lambda = 4^{L-\lambda} \cdot 4 \cdot \frac{n}{4^{L-\lambda}} \cdot f_\lambda = 4 \cdot n \cdot f_\lambda \qquad (\lambda = 1,\ \ldots,\ L)$$

Aggregating the cost on all levels, we have

$$\begin{aligned} c &= c_0 + \sum_{\lambda=1}^{L} c_\lambda \\ &= nf_0 + \sum_{\lambda=1}^{L} 4nf_\lambda \\ &= n\left(f_0 + 4\sum_{\lambda=1}^{L} f_\lambda\right). \end{aligned}$$

Incorporating the definition of the frequencies $f_\lambda = q^\lambda f_0$ gives

$$c = nf_0\left(1 + 4\sum_{\lambda=1}^{L} q^\lambda\right)$$

leading directly to the theorem. □

Corollary 1 (Cost Complexity)**.** *With the definitions of Theorem 1, the SPBM group management protocol overhead per area and time has a complexity of* $O(1)$ *for* $q < 1$ *and* $O(\log A)$ *for* $q = 1$ *with respect to the total size of the network area* A*.*

Proof. Let us assume that we have a network consisting of only one square of size A_0. We further assume that we have a limited node density d denoting the number of nodes per A_0 area. The number of nodes in the complete network is then given as $n = dA$, where A is a multiple of A_0. Whenever the network area increases, we quadruple the network area by increasing the hierarchy level by one, such that the new area is covered by the new square, i.e., the number of hierarchies is calculated as

$$L(A) = \lceil \log_4 A \rceil < 1 + A_0 \log_4 A. \tag{4}$$

With the increase of the area, the possibility of spatial frequency reuse grows linearly. Thus, we consider the cost per area c_A. Following Equation (3), the average overhead cost per time and area is

$$\begin{aligned} c_A &= \frac{c}{A} \\ &= df_0 \left(1 + 4 \sum_{\lambda=1}^{L} q^\lambda\right). \end{aligned}$$

With Equation (4), an upper bound $\overline{c_A}$ for the cost per area can be specified:

$$\overline{c_A} = df_0 \left(1 + 4 \sum_{\lambda=1}^{1+\log_4 A} q^\lambda\right)$$

Considering the case $q = 1$, this upper bound results in

$$\overline{c_A} = df_0 \left(5 + 4 \log_4 A\right)$$

which conforms to $O(\log A)$.

With $0 < q < 1$, the geometric row converges and is bounded:

$$\begin{aligned} c_A &= df_0 \left(1 + 4 \sum_{\lambda=1}^{L} q^\lambda\right) \\ &< df_0 \left(1 + 4 \frac{q}{1-q}\right), \end{aligned}$$

which is independent of the chosen area size or the maximum level, respectively. Thus, the complexity is $O(1)$. □