Building Multicast Trees in Ad-hoc Networks

Raphaël Kummer
University of Neuchâtel
Computer Science Department
Emile-Argand 11, CP 158
CH-2009 Neuchâtel, Switzerland
raphael.kummer@unine.ch

Peter Kropf
University of Neuchâtel
Computer Science Department
Emile-Argand 11, CP 158
CH-2009 Neuchâtel, Switzerland
peter.kropf@unine.ch

Pascal Felber
University of Neuchâtel
Computer Science Department
Emile-Argand 11, CP 158
CH-2009 Neuchâtel, Switzerland
pascal.felber@unine.ch

ABSTRACT

Multicast trees are used in a variety of applications, such as publish/subscribe systems or content distribution networks. Existing algorithms for ad-hoc networks typically produce multicast trees requiring many nodes to act as relays even though they are not part of the multicast group. In this paper, we propose an algorithm for building efficient multicast trees that strives to minimize the number of non-member relay nodes and the number of transmissions required to reach all the group members, and to balance the degree of members when acting as internal nodes of the multicast tree. Our algorithm relies upon a lightweight distributed hash table (DHT) to construct and optimize the multicast trees. We evaluate the efficiency and scalability of our algorithm by simulations with various network configurations and sizes.

Categories and Subject Descriptors

C.2.2 [**Computer - Communication Networks**]: Network Protocols

General Terms

Multicast Trees, MANETs

Keywords

Mobile Networking, Wireless Networks Protocol, Ad-hoc Wireless Networks, Mobile Ad-hoc Computing, Multicast trees

1. INTRODUCTION

Multicast communication is widely used in distributed applications for efficiently delivering data from a source to a potentially large number of destinations. Multicast algorithms typically create and maintain distribution trees spanning all destinations in a way that messages are transmitted over each link of the network exactly once.

AUTONOMICS 2008, September 23-25, Turin, Italy

DOI 10.4108/ICST.AUTONOMICS2008.4648

In mobile ad-hoc networks (MANETs), communication between remote nodes usually requires multiple hops via relay nodes, thus complicating the task of building efficient multicast trees. Because relaying messages consumes scarce CPU cycles and energy, one must minimize the number of nodes that act as relays.

Centralized tree construction algorithms do not scale well and offer but limited reliability. We thus investigate decentralized strategies where nodes are organized in a peer-to-peer (P2P) overlay configuration.

A basic, unstructured approach consists in flooding the network with data. Alas, this method highly overloads the network and makes only inefficient use of the (limited) resources.

Superimposing a structured overlay on top of the physical network can help to construct efficient multicast trees. This approach has been used, for instance, by Scribe [11]: multicast trees are rooted at "rendez-vous nodes" managed by an underlying distributed hash table (DHT). Nodes interested in joining the multicast group route a request via the DHT and connect to the first member reached on the way to the rendez-vous node. While this strategy is effective in wired networks, it cannot be easily transposed to mobile ad-hoc networks where communication is multi-hop and physical proximity is an essential consideration.

In this paper we propose a novel algorithm for building multicast trees in MANETs relying on a lightweight DHT overlay [7]. The ad-hoc DHT provides efficient lookup in a logical space by exploiting the physical proximity of nodes and the properties of the wireless broadcast communication medium. As in Scribe, the DHT is used for localizing the source of the multicast and, one tree is created per source. We use several techniques to connect joining nodes to an existing member that is physically close and in the direction of the source. Various refinements are proposed to reduce the number of relay nodes involved in the relaying of messages but which are not members of the multicast group.

We have performed extensive simulations of our algorithm using different scenarios. Results indicate that our algorithm does produce multicast trees with only a limited number of non-member nodes relaying the multicasts, and that it scales well to large networks.

The remainder of this paper is organized as follows. Section 2 discusses related work, and then Section 3 gives an overview of the system background. We detail our algorithm in Section 4 and present results from experimental evaluation in Section 5. Section 6 concludes.

2. RELATED WORK

Different logical structures can facilitate appropriate tree construction in ad-hoc networks. MZR [4] relies upon the Zone Routing Protocol [5] to build a multicast tree. The nodes in ZRP define a zone around them and maintain proactively routes to all nodes within that zone. A reactive route discovery protocol is used when the destination is outside the sender's zone. When a source has data to multicast, it advertises it to all the nodes in its zone, and then extends the tree to nodes at the border of other zones. An interested node has to answer to the source and, when the message reaches a multicast group member, a branch is created. While the zone structure constrains the flooding necessary to build the tree, it still floods the whole network zone by zone. Its bandwidth and energy requirements are thus significant. In addition, this protocol doesn't provide generic lookup facilities as it is the case for our DHT based algorithm.

Similarly to our approach, XScribe [8] and Georendezvous [2] use a DHT to support the multicast tree creation. XScribe is based on CrossROAD [3], a cross-layer DHT for ad-hoc networks providing the same interface and functionalities as Pastry [10], but based on a proactive routing protocol with lower bandwidth requirements. XScribe exploits the DHT routing capacities to distribute multicast messages. Unlike our proposal, this system scales poorly as each source has to know all the members of the group and sends the multicast messages directly to each member by unicasting messages, thus generating high communication load on the source. It does neither try to optimize resource consumption, such as minimizing the number of relay nodes nor the number of transmission required.

Georendezvous relies on CHR [1], a specialized ad-hoc DHT that groups nodes in clusters according to their physical location. The DHT is used to efficiently localize the cell responsible for a group. The nodes in this cell manage the membership for the group and forward the multicast to all the members. Membership management is centralized in a cell containing multiple nodes, which are also responsible for distributing multicast messages. Our approach is expected to produce more efficient multicast trees, with lower bandwidth requirements, and to offer better scalability compared to centralized approaches like shortest path trees.

3. BACKGROUND

As our algorithm builds source-based multicast trees [1], it requires an efficient way to locate the node responsible for a group. Thus, as in Scribe [11], we use a DHT specifically adapted for mobile ad hoc networks [7] to locate the source without flooding the network. The DHT provides scalability, efficient lookup and reliability as detailed in section 4.

3.1 The Underlying DHT

The ad-hoc DHT used by our multicast tree construction algorithm consists of a minimalist logical overlay where all the nodes of the ad-hoc network are organized in a ring (see Figure 1). Each node is assigned a random identifier in the logical space, e.g., by hashing the node's IP address using a cryptographic hash function such as SHA-1. The

[1]The source initiates the content distribution and is partially responsible for membership management. See section 4 for a precise description.

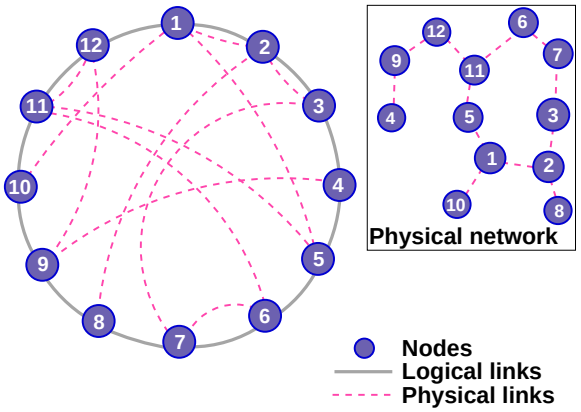

Figure 1: Illustration of the DHT model for an ad-hoc network.

nodes are ordered in the logical ring using a proximity metric based on their unique identifiers. The DHT maps keys to nodes in the P2P overlay, and the logically nearest node on the ring (i.e., in the identifier space) is responsible for the key. Each peer needs to keep track at all times of one successor and one predecessor, thus building a logical overlay without long-range neighbors in the logical space. Unlike most other DHTs (e.g., Pastry [10] or Chord [12]), the long-range neighbors are spontaneously provided by the physical neighborhoods of the nodes traversed during the lookup as illustrated in Figure 1 with an ad-hoc network and its logical overlay links and the logical shortcuts (physical links).

At each traversed node, the algorithm searches among its physical and logical neighbors, the current destination and itself which node is logically closest to the searched key. If the current node is the closest one, it is thus responsible for the key and the lookup ends. Otherwise, the message is forwarded to the selected node, either through a multi-step path to a logical neighbor or directly delivered to the designed physical neighbor (long-range link). Figure 2 shows an example of a path along logical and physical links.

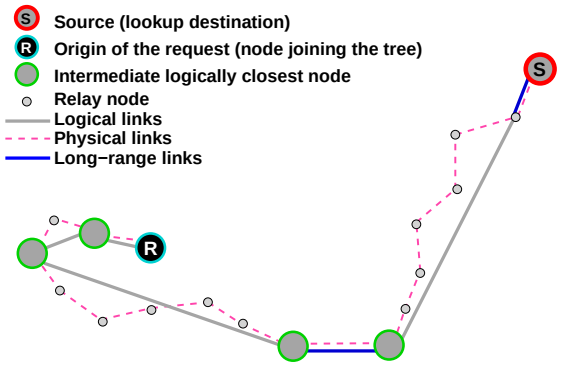

Figure 2: Path of a request routed by the DHT.

We extended this basic algorithm with two improvements. The first one considers the neighbors of the physical neighbors to extend the visibility and thus providing more long-range possibilities to the lookup algorithm. In the second

extension, the nodes keep track of the traffic seen but not intended to them in a cache. Consequently, the cached information is used by the extended algorithm to increase its set of available long-range neighbors thus providing a larger number of shortcuts in the DHT's logical space.

For more details, we refer the interested reader to [7].

3.2 Experimental Model

We consider that our algorithm runs on simple, resource limited devices as used for instance in public spaces or office buildings for personal or professional communication and data exchange.

We thus consider that all participating devices (*nodes*) are uniformly distributed in a Cartesian space where they can independently decide to move during a finite period of time t with a speed S randomly chosen in the interval $0 \leq S \leq 2\ m/s$ in arbitrary directions to reflect human displacements. At the end of the period t, a node decides whether to stay or to move on.

The speed of the nodes may be altered if necessary to continue their movements into the defined area when reaching a border. In our experiments, the nodes have on average 13 to 16 direct neighbors. These communicate together by broadcasting messages (i.e., each node within the communication range is then able to listen to them) hence forming a connected network supported by an implementation of the ad-hoc on-demand distance vector ($AODV$) protocol [9] for multi-hop communication.

We consider that unidirectional communications can be detected and hidden at the network layer (i.e., all the links are bidirectional).

We do not consider unexpected departures of nodes. However, our algorithm may use the underlying DHT lookup facilities to easily and efficiently find a new parent in case of unexpected disconnection. Therefore, no additional control traffic will be generated by the multicast algorithm to deal with churn.

Finally, we assume that only a small subset of the nodes in the network is interested in the multicasted content (no more than 10%). Indeed, if the proportion of multicast members is too big, broadcasting techniques are better adapted.

3.3 Terminology

In order to facilitate the reading of the following sections, we fix here the terminology we use to describe the different roles of nodes running the multicast algorithm. The terms we use are hierarchically structured and their definitions are presented in table 1. Additionally, we illustrated the different terms relatively to their position in a sample multicast tree in figure 3 for better comprehension.

Table 1: Terminology

Term	Definition
Nodes	All the nodes in the system.
Members	All the nodes interested in a multicast group.
Non-members	All the nodes not member of a multicast group.
Internal members	Members inside the tree helping to distribute multicast messages (i.e., members with children).
Leaf members	Members at the end of a branch of the multicast tree (i.e., members without children).
Relay nodes	Non-member nodes relaying multicast messages.
Non-relay nodes	Non-member nodes not included in multicast tree activities.

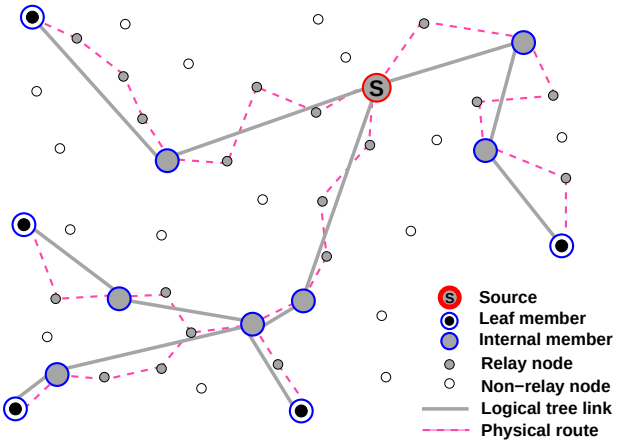

Figure 3: Sample tree describing the used terminology

4. TREE CONSTRUCTION ALGORITHM

In ad-hoc networks, multicasting messages to a group of nodes can be essentially achieved in two different ways. The first approach relies on flooding the network to discover a source or to build a tree, the second one uses directed search methods as done for instance in Scribe [11]. As flooding tends to overload the network, involves many uninterested nodes and scales poorly, it is preferable to use the second approach which is more bandwidth and energy efficient.

Thus, to construct a multicast tree, we first lookup the data source, which acts as a rendez-vous point (i.e., tree root) for the multicast group. Hence, we use a DHT [7] to efficiently locate the source without flooding the network. The DHT also provides support to our multicast algorithm for handling the unexpected departure of an internal member. Indeed, a member can easily and efficiently find a new parent by looking up the DHT with the multicast group key.

Our algorithm strives to construct multicast trees that use internal members as relays. Membership is handled in a decentralized way as a joining node might connect to the tree without the source being aware of it. Consequently, the load on the source is reduced and we can avoid bottlenecks. One tree is constructed per active data source. We assume "best effort" delivery for multicast messages; additional mechanisms could be incorporated to implement reliable delivery.

4.1 The Basic Connection Algorithm

The core of the algorithm (called *basic*) consists in building a multicast tree using DHT lookups, and then applying various extensions to improve the tree.

To join the multicast tree, a node routes a request to the identifier associated with the source of the multicast (group

identifier) using a DHT lookup. As for the underlying DHT, all the nodes are running the multicast algorithm, but they are not necessarily members. Thus, they can be used to provide information on the network state by adding some knowledge in relayed messages.

When receiving a join request on its way to the source, a node checks if it is a member of the group. If so, it replies to the joining node and proposes itself as parent in the multicast tree. Otherwise, it simply forwards the request to the next node towards the source (according to the DHT lookup protocol). Thus, the joining node receives, most of the time, several potential parents (but at least one as the request is always routed towards the source).

To join the distribution tree as soon as possible, the requester connects to the first parent it receives. Thereafter, if it receives further responses from potential parents, it changes only if (1) it has not already received a multicast from its parent and; either (2a) the distance to the new parent is shorter than to the old parent and the new distance to the root is no more than twice the old distance; or (2b) the new parent is at the same distance as the old one but the distance to the root has shortened. Distances are computed according to the number of physical hops. At the end of the process, the node is connected to the multicast tree with the member that it considers as being the best (shortest distance) parent among the received proposals. This parent selection method has been designed empirically so as to perform well in various scenarios, as shown in Section 5.

With this straightforward algorithm, many members are connected to the source with direct paths and many relay nodes are located on the paths from the source to the members. In order to improve the tree structure and to reduce the number of relay nodes involved in multicasting, we propose a number of extensions presented below. The first one is applied during lookup, while the second and third ones rely on information added by nodes to the multicast messages for reorganizing the tree. Finally, the last one listens to wireless communications for finding potential children. These extensions are always cumulated when applied (i.e., an extension also incorporates the former ones).

4.2 Finding More Potential Parents

The first extension (Ext. #1) of our algorithm takes advantage of the broadcast communication feature inherent to MANETs. Indeed, the nodes that are not part of a communication but within the communication range of the sender may listen to it and use the information gathered. In particular, a node listening to a join message will send a response to the requester if it is a member of the multicast group. Listening to communications is typically a cheap operation as it does not generate extra messages, yet it often allows the improvement of the tree structure. Moreover, listening to messages avoids pathological situations where two multihop requests cross, but do not traverse a common node.

The requester can accept such connection proposals as long as (1) it has not received its first multicast message, and (2) it has no children. Indeed, as soon as it is integrated into the tree, the risk to partition it becomes too high: a situation to be avoided.

4.3 Finding Better Parents

Our second extension (*Ext. #2*) exploits the multi-hop path of a multicast message which usually traverses several

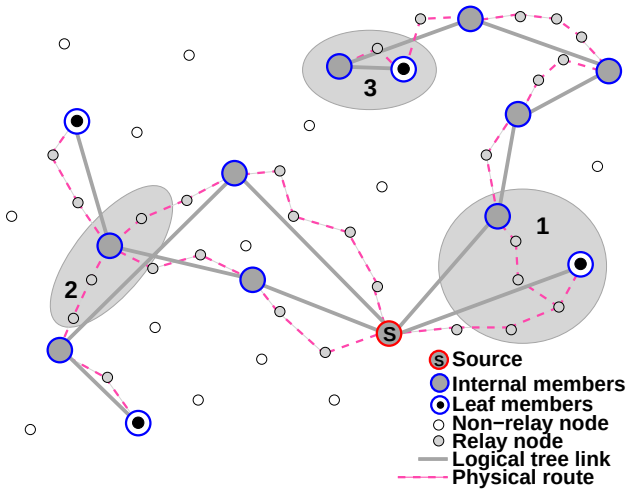

Figure 4: Sample scenarios where the tree is not organized optimally.

relay nodes between a parent and a child in the logical multicast tree. Two messages from one member to two different members may traverse a number of common relay nodes as in scenario 1 presented in Figure 4 (grey circle labeled 1). In fact, if two messages follow the same physical path, their destinations are likely to be in the same area of the network. Consequently, a node can inform one of the members that the other one is possibly a better child or parent in the multicast tree. This extension is interesting because, by shortening the distance between parents and children in the tree, we also reduce the network load.

We propose a solution where no additional messages are needed. When a relay node relays a multicast message (uniquely identified by a group and a message identifier) it memorizes the group, the message identifier, and the destination address. If a relay node receives the same message (same group and message identifier) intended for another destination, the relay node adds the previously memorized address to the message before forwarding it. If more than one relay node has an address to add, only the last one is kept in the message; hence space overhead is negligible.

The receiver of a multicast message checks if the address of a member has been added to the message. If so, the receiver sends a message to that member and proposes to become its parent. A reconfiguration takes place only if (1) the new parent is closer than the current parent (i.e., tree structure improvement); (2) the resulting configuration is a valid tree (i.e., tree with no disconnected parts or cycles).

This strict validity check avoids partitioning the tree and losing the connection with the source. The position in the tree is given by a tag built by the multicast messages along their path to the leaf nodes. This tag, together with a timestamp provides global tree structure information to the node with a freshness indication.

By not applying such an integrity check, a descendent node might propose itself as new parent for one of its ancestors thus creating a disconnected cycle.

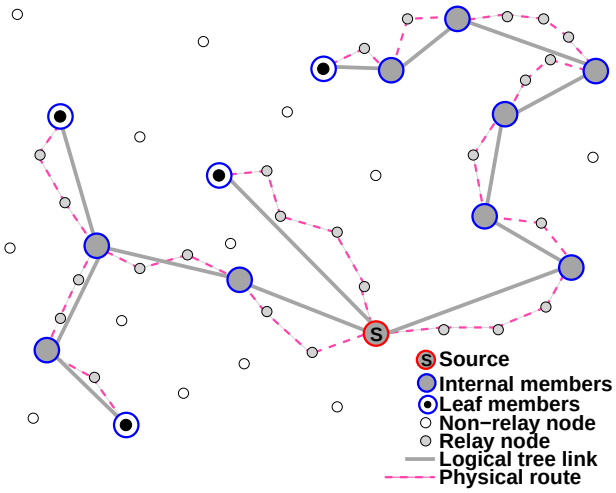

Figure 5: Scenarios of Figure 4 after applying extensions.

4.4 Removing Redundant Parents

The next extension (*Ext. #3*) considers the multi-hop connections between a member and its children in the multicast tree. It may happen that a node is both a member of the logical tree as well as a relay node along a multi-hop path between a member and one of its children as presented in scenario 2 of Figure 4 (grey circle labeled 2). Obviously, the resulting structure is a sub-optimal situation that should be avoided as the affected member receives the same message from more than one parent. To deal with this situation, a member that receives the same message from multiple paths (both as a relay node and as a member) keeps only one connection with the closer parent and discards the others.

If the member is on a multi-hop path to its parent, as presented in scenario 3 of Figure 4 (grey circle labeled 3), it may need to take its parent as a child, or it may need to promote itself as new inner node of the tree along that path (i.e., change from leaf member to internal member).

In either case, one physical path is discarded or rearranged, and the number of relay nodes is reduced. Figure 5 exposes the effect of these last two extensions on the tree presented in Figure 4. One can notice that the tree better fits to the underlying topology and multicast efficiency is improved.

4.5 Multicast Message Gathering

Finally, the last extension (*Ext. #4*) aims, as the first one, to take advantage of the nodes' operation by broadcast communication. Indeed, during their distribution, multicast messages may travel near a group member without being actually received or relayed by it. As communications are broadcast, members can however gather the information contained in those messages by simply listening to them.

The source decides to intermittently allow the members from a selected branch to use this grabbed messages by setting a flag in the multicast message sent to this branch. Each member from the selected branch tries to find the worst case it can identify in the subset of valid messages allowed to be used. The message considered as the worst case is the message having the largest distance with its sender.

A message is considered valid if (1) it has the same time stamp as the last multicast message received by the current member; (2) the current member has never considered the destination as the worst case and; (3a) its destination is outside the current branch or; (3b) the destination is a descendant of the current member.

When the candidate message is found in accordance with the above constraints, the current member proposes to the addressee to become its parent.

By not applying this method continuously, but intermittently only, the tree structure can stabilize. Thus, when less improvement possibilities exist, our algorithm tries to provide fresh enhancement opportunities by correcting a highly unfavorable situation with the gathered information. Moreover, by opting for an unhandled candidate message only, a node issues one single message to a new potential child, hence limiting the network load and avoiding to repeatedly proposing an identical solution.

5. EXPERIMENTATIONS

5.1 Setup

To evaluate our multicast tree construction algorithm, we have extended the experimental system already used for testing our ad-hoc DHT [7]. The simulator is divided into three layers: the routing, the DHT, and the multicast layer. These layers communicate together through dedicated interface methods. We assume that the routing protocol (*AODV*) is able to process messages faster than the upper layers thus avoiding performance interferences.

In this paper, we only simulate simple scenarios based on the experimental model presented in section 3.2. Our study presents results issued from static and mobile networks.

DHT identifiers are randomly assigned to nodes, and a group identifier (mapped to a single source) is randomly selected in the same logical space. We let randomly selected nodes to join that group. We first "warm up" the underlying DHT by performing several lookups (100 in our tests); this allows the nodes to populate their routing tables (see [7] for details). Then, between one and three non-members join the tree at each simulation step until the desired number of members is reached. Concurrently, multicast messages are sent continuously every one to five simulation step throughout the tree. We stop the simulation once the tree structure stabilizes (i.e., when the algorithm stops producing changes to the tree structure).

We experimented our algorithm in networks of $1,000$ and $5,000$ static and mobile nodes in accordance with the experimental model presented in section 3.2. Results are averages of ten experiments.

Using these configurations, we have evaluated the different versions of the multicast tree construction algorithm:

- *Basic:* the construction algorithm with no extension; (Section 4.1)
- *Ext. #1:* during join, members listen and propose themselves as parent when applicable; (Section 4.2)
- *Ext. #2:* when sending messages, we try to identify common sub-paths and reconnect members to better parents; (Section 4.3)

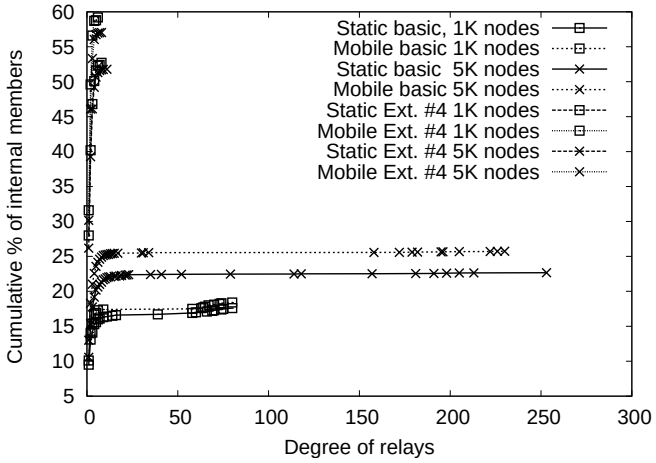

Figure 6: Cumulative percentage of members acting as internal members as a function of their degree (mobile-static comparison).

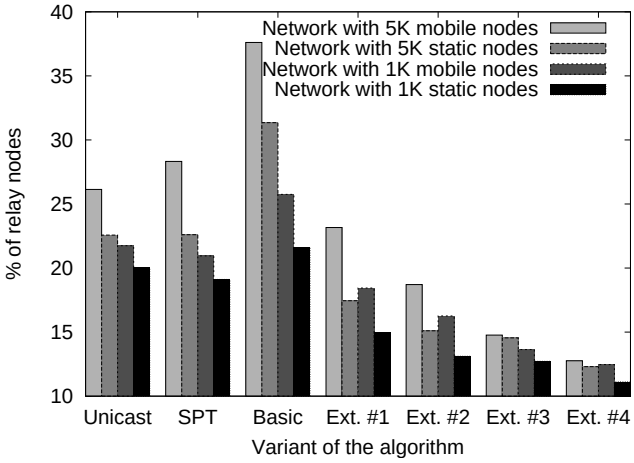

Figure 7: Percentage of relay nodes involved (mobile-static comparison).

- *Ext. #3:* we prevent nodes to receive duplicate messages when acting both as member and relay node in a multicast tree; (Section 4.4)
- *Ext. #4:* we listen multicast messages to propose to the members with the biggest physical distance to its actual parent to become its new parent. (Section 4.5)

Extensions are always cumulated in the presented results (i.e., extension n also incorporates extension $m < n$).

We compared the performances of our algorithm with two other well-known approaches: a unicast information distribution (*Unicast*) method and a shortest path tree (*SPT*).

In both approaches, the source controls the membership and the multicast message distribution. The *unicast* messages are sent separately to their destination while the *SPT* messages are grouped as long as they have a common next hop thus forming the shortest path tree.

We are essentially interested in evaluating the tree structure, the degree of internal members, the number of relay nodes required to route messages, as well as the relative distance between two members, the total transmission costs required to reach all members, and the general multicast cost.

For the general multicast cost measurement, we use the metric defined by Jaquet and Rodolakis in [6] as

$$R(n) = \frac{multicast\ cost}{average\ unicast\ cost}$$

where the normalized multicast cost is expressed in number of hops (= number of transmissions) in the multicast tree and the average unicast cost is the average route length from the source to a random member. To compute the average cost of unicast in a defined network size, we consider that the members are all directly connected to the source, then we compute the average unicast cost in number of transmission (= number of hops) as unicast cost reference.

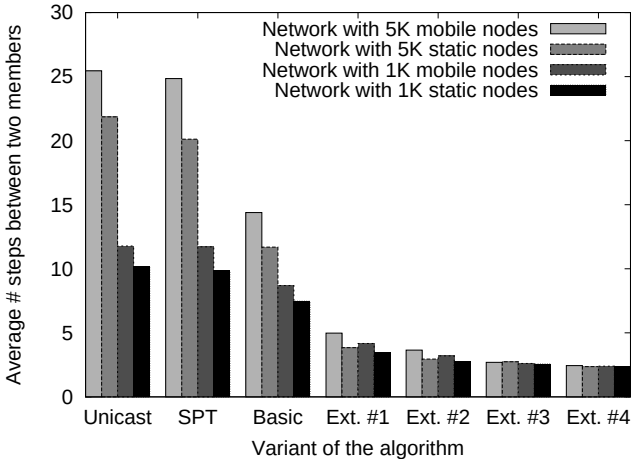

Figure 8: Average number of physical steps separating two members (mobile-static comparison).

5.2 Results

5.2.1 Degree of Member Nodes

Figure 6 compares the number of internal members with their degree. A small Y value means that the tree includes only a few internal members. In the same way, a small X value indicates that the nodes have a small degree.

The ideal case would be many internal members having a small degree. In the tree produced by the basic algorithm, for both static and mobile configurations less than 30% of the members participate in the message distribution (i.e., more than 60% are only leaf members) and some of them show quite a high degree (most notably the source). In contrast, when the tree structure has been improved by the different methods presented in Section 4, more than 50% of the members help to distribute content, thus act as internal members. The degree of these internal members does not exceed 7 for all the considered network sizes. Hence,

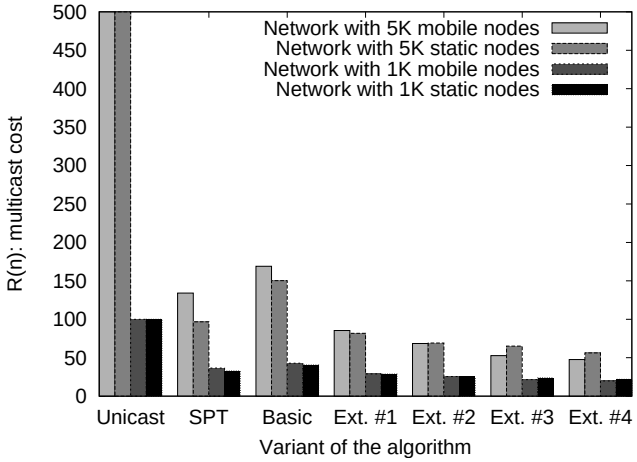

Figure 9: Multicast cost comparison between different network sizes (mobile-static comparison).

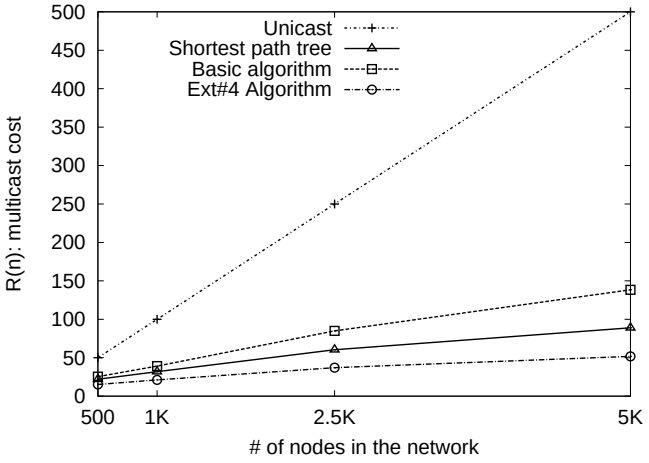

Figure 11: Multicast cost evolution (static configuration).

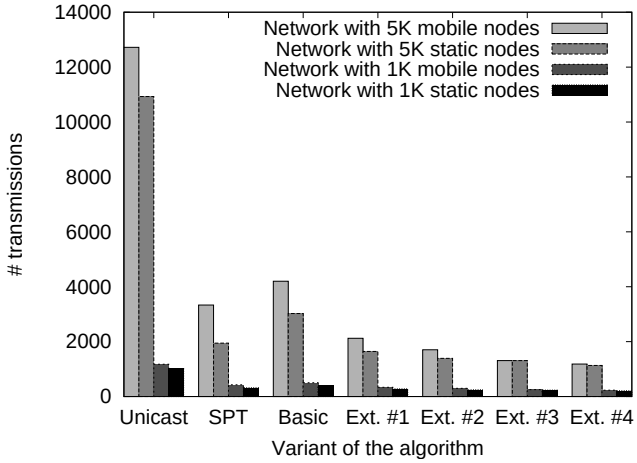

Figure 10: Number of transmission comparison between mobile and static configurations.

our algorithm contributes to evenly distribute the multicast and membership management load between the members. Moreover, for both, the basic and Ext. #4 version, the mobile scenario performs better while including more internal members. As nodes are moving, they potentially get more optimization solutions because requests have a higher probability to pass next to or through a member during the connection process.

5.2.2 Non-member Relays

As a result of the increased number of internal members relaying messages, the number of relay nodes decreases. There is a significant improvement (from almost 40% relay nodes down to less than 13%) as a result of our extensions for all network sizes and scenarios (see Figure 7). The shortest path tree and unicast distribution methods use less relay nodes than our basic version because they use many times the same routes to reach the members. In contrast, our basic algorithm uses internal members to distribute the multicasts hence reducing the transmission load of the source but adding some new paths and relay nodes to support them. This is confirmed by the transmission cost (see Figure 10) that is only slightly higher for our basic algorithm than for the shortest path tree and noticeably lower than for the unicast distribution method.

Finally, in Figure 7, it can be noticed that the basic algorithm in the mobile context needs more relay nodes. However, the tree built with all the extensions in the mobile scenario needs roughly the same number of relay nodes as the static experiments, which is a remarkable result. Recall that we let the simulation running after the tree has been built. For the static situation, it allows for a stabilization of the tree structure. For the mobile situation, as the nodes continue to move and as we evaluate the tree at the end of the simulation only, it demonstrates that our algorithm is not only able to build efficient multicast trees, but also to maintain the structure. This interesting fact is confirmed by the other measurements presented in Figures 6 and 8.

5.2.3 Average Distance Between Member Nodes

Our algorithm builds a tree in which members that are physically close can discover each other and connect by just passively gathering information as messages are broadcast. Thus, the physical path length between two consecutive members is small, as shown in Figure 8, and in all cases smaller than the one in the shortest path tree and the unicast solutions. This is not surprising as all the shortest path tree members are leaf members directly connected to the source. Therefore, the average distance between two members (i.e., the average distance to the source) is maximal since no connection to a physically close member is envisaged. Consequently, the tree built by our algorithm is closely mapped to the underlying physical topology in mobile and static environments by connecting physically close members together. This is a major advantage especially in mobile ad-hoc networks, because while reducing the physical distance between two members, the time to rebuild a broken route is diminished, and the probability that a route breaks is reduced.

5.2.4 Multicast Cost

Figure 9 illustrates that our algorithm is cheaper in distributing content than unicasting a message to each member. The improved algorithm also outperforms the shortest path tree method. Only our basic algorithm has a cost slightly higher than the shortest path tree. In fact, as already explained above, our basic algorithm constructs longer paths than the shortest path algorithm while using internal members to participate in multicast distribution. Those nodes are not necessarily on the optimal route to the leaf members, thus requiring some more transmissions as shown in Figure 10. Nevertheless, our algorithm requires fewer transmissions than the shortest path tree or the unicast in all other cases, thus avoiding unnecessary energy consumption and increasing the battery lifetime.

5.2.5 Scalability

One should note that when applying the four extensions, the physical distance between two members, the number of involved relay nodes and the degree of internal members are nearly the same for all the considered network sizes. Moreover, the experimentations with the mobile scenarios have shown that our algorithm not only builds good multicast trees, but is also able to maintain its good structure for all the tested network sizes. These two results support the claim for the scalability of our approach. This is confirmed by Figure 11, where the multicast cost of our improved algorithm is not only lower but also grows slower than the multicast cost of shortest path tree and unicast.

5.2.6 Variability of the results

For several experiments presented here we evaluated the coefficient of variability (COV). This represents the ratio of standard deviation to the mean. This allows scale-free comparisons out of variability consideration, as opposed to variance measurements.

The average number of steps between two members (Figure 8) has a maximum COV of 0.7% for all network sizes and configurations with the extended algorithm. This means that there are no significant differences between the conducted experiments. Similarly, considering the number of relay nodes (Figure 7), we obtain a maximum COV of 8%. Only the number of transmissions varies more, what is a consequence of the applicability of routing algorithm optimizations relative to the network topologies.

6. CONCLUSION

Although much work has been done on the problem of multicast in MANETs, most of the solutions use some form of flooding or centralized solutions that are not scalable. In this paper, we presented an algorithm for the construction of efficient multicast trees using an underlying ad-hoc DHT overlay. Our algorithm strives to create trees that involve as few relay nodes as possible, requiring a limited amount of transmissions, with short inter-members paths and good scalability. Simulation results indicate that our algorithm meets these objectives in the considered network settings and also maintains a good structure in a mobile environment.

7. ACKNOWLEDGEMENT

This research was partially supported by the Swiss National Science Foundation under grant number 5005-67322 (NCCR-MICS) and by the MiNEMA Research Network.

8. REFERENCES

[1] F. Araujo, L. Rodrigues, J. Kaiser, C. Liu, and C. Mitidieri. CHR: A distributed hash table for wireless ad hoc networks. In *ICDCSW '05*, pages 407–413, Washington, DC, USA, 2005. IEEE Computer Society.

[2] N. Carvalho, F. Araujo, and L. Rodrigues. Reducing latency in rendezvous-based publish-subscribe systems for wireless ad hoc networks. In *ICDCSW '06*, page 28, Washington, DC, USA, 2006. IEEE Computer Society.

[3] F. Delmastro. From pastry to CrossROAD: CROSS-layer ring overlay for AD hoc networks. In *PERCOMW '05*, pages 60–64, Washington, DC, USA, 2005. IEEE Computer Society.

[4] V. Devarapalli and D. Sidhu. MZR: a multicast protocol for mobile ad hoc networks. volume 3, pages 886 – 891, 2001.

[5] Z. Haas. A new routing protocol for the reconfigurable wireless networks. In *IEEE 6th International Conference on Universal Personal Communications Record*, volume 2, pages 562–566, San Diego, CA, USA, 1997.

[6] P. Jacquet and G. Rodolakis. Multicast scaling properties in massively dense ad hoc networks. *11th International Conference on Parallel and Distributed Systems*, Volume 2:93 – 97, July 2005.

[7] R. Kummer, P. Kropf, and P. Felber. Distributed lookup in structured peer-to-peer ad-hoc networks. In R. Meersman and Z. Tari, editors, *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE*, volume 4276 of *Lecture Notes in Computer Science*, pages 1541–1554. Springer Berlin / Heidelberg, 2006.

[8] A. Passarella, F. Delmastro, and M. Conti. XScribe: a stateless, cross-layer approach to P2P multicast in multi-hop ad hoc networks. In *MobiShare '06*, pages 6–11, New York, NY, USA, 2006. ACM Press.

[9] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. *MILCOM '97*, 1997.

[10] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware 2001*, volume 2218 of *Lecture Notes in Computer Science*, pages 329–350. Springer Berlin / Heidelberg, 2001.

[11] A. Rowstron, A.-M. Kermarrec, M. Castro, and P. Druschel. SCRIBE: The design of a large-scale event notification infrastructure. In *Networked Group Communication*, pages 30–43, 2001.

[12] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01*, pages 149–160, New York, USA, 2001. ACM Press.