Multicast Overlay Spanning Tree Protocol for Ad Hoc Networks

Georgios Rodolakis, Amina Meraihi Naimi, and Anis Laouiti*

INRIA and Ecole Polytechnique, France
Georges.Rodolakis@inria.fr, Amina.Naimi@inria.fr, Anis.Laouiti@inria.fr

Abstract. In this paper we present an extension to the OLSR unicast routing protocol to support multicast routing in mobile ad hoc networks. The proposed protocol is based on Multicast Overlay Spanning Trees (MOST). The main benefits of this approach are twofold. Firstly, it implies that only nodes interested in taking part in the multicast communication need to participate in the protocol operation, which is transparent to other OLSR nodes. In addition, the MOST approach scaling properties achieve the theoretical performance bounds concerning the capacity of multicast communication in massive ad hoc networks. We perform simulations of the MOST protocol under the ns-2 simulator to compare with the theoretical results, and we present a fully working implementation for real network environments.

1 Introduction

Multicast offers an elegant way to establish group communication between users by using the concept of multicast groups, which are defined by their corresponding address. Interested clients can join and leave those groups in order to send and/or receive data from other group members. Moreover, the mechanisms which enable multicast communication ensure that an efficient strategy is used to deliver the data packets to all the members simultaneously. Therefore, multicast communication is adequate for a large class of applications, such as video-conferences, multi-player games, streaming applications *etc.* The previously described requirements make multicast routing an important and difficult challenge in the Internet, and even more so in ad hoc networks. In fact, mainly due to the dynamic nature of the routes, multicast protocols developed for wired networks cannot operate in the harsher mobile environment. This creates a need for protocols which are specially adapted to ad hoc networks.

Multicast ad hoc protocols can be classified according to the underlying routing structure to tree-based protocols and mesh-based protocols. The routing structure can be either group shared, or source dependent. Some tree-based protocols are MAODV [16] which is an extension to the unicast routing protocol AODV [15] based on a group shared tree, MOLSR [12] which is an extension to OLSR unicast routing protocol [6], based on a Dijkstra tree, and Adaptive

* Anis Laouiti is currently with GET/INT, France.

F. Boavida et al. (Eds.): WWIC 2007, LNCS 4517, pp. 290–301, 2007.

Demand-driven Multicast Routing Protocol (ADMR) [11]. A protocol which is based on overlay trees is AMRoute [14]. As an example of mesh-based routing protocols we mention On-Demand Multicast Routing Protocol (ODMRP) [13] and Core-Assisted Mesh Protocol (CAMP) [8].

In contrast to the previously described protocols our work is motivated by analytical results on the achievable capacity of multicast communication in ad hoc networks. We show how to use minimum spanning trees to perform efficient multicast routing. The performance of minimum spanning tree multicast in ad hoc networks was studied in [10] and it was shown via analysis to be nearly optimal in case the number of multicast clients of each group is small compared to the network size. The advantage of this approach is that only multicast nodes need to participate in the multicast protocol. We present an overlay group shared tree based routing protocol called MOST which works in conjunction with the unicast OLSR protocol. The MOST protocol being based on overlay trees guarantees robustness, while it achieves good performance since OLSR provides topology information allowing to construct an optimal multicast tree. In addition, the MOST protocol is designed with the aim to achieve the theoretical capacity bounds.

The rest of the paper is organized as follows. In Section 2 we summarize the theoretical results concerning multicast scaling in ad hoc networks. In Section 3 we present the MOST protocol specification. A description of the protocol implementation is provided in Section 4. In Section 5 we present ns-2 simulations of the proposed protocol, which are compared to the theoretical analysis.

2 Asymptotic Multicast Properties in Ad Hoc Networks

One of the advantages of multicast routing is that it reduces the total bandwidth required to communicate with all group destinations, since some links can be common to several destinations. In wired networks, the gain of multicast communication has been studied in [4,5], by estimating the ratio of the number of links in a multicast tree to n destinations over the average unicast hop distance between two random nodes. The resulting normalized multicast cost has been found experimentally to scale in $n^{0.8}$. The gain of multicast is reflected by how far the normalized multicast cost deviates from linear growth. However, the topology of mobile ad hoc networks is significantly different and one would expect a much different scaling law. Indeed the average unicast hop distance in wired networks is usually of the order $\log N$, where N is the total number of nodes in the network, while in ad hoc networks the average distance grows proportionally to $\sqrt{N/\log N}$, since the optimal neighbor degree increases in $O(\log N)$ when the capacity increases [9].

In [10], performance bounds are established on the expected size of multicast trees as a function of the number of multicast destinations n, both via analytical methods and via simulation. In random mobile ad hoc networks, the gain of multicast communication compared to unicast is significantly larger than in wired networks. For instance, a scaling law in $O(\sqrt{n})$ holds for the normalized

multicast cost. The implications of this scaling law consist in a significant increase of the total capacity of the network for data delivery, which will be proportional to $\sqrt{\frac{nN}{\log N}}$. It is shown that when the number of multicast clients for each group is small compared to the network size, minimum spanning trees lead to asymptotically optimal performance in terms of network bandwidth utilization and achieve the theoretical capacity bounds. Therefore, the theoretical analysis can be applied in the design of an efficient multicast protocol based on the overlay minimum spanning tree approach.

3 Multicast Overlay Spanning Tree (MOST) Protocol

In this section, we present an extension to the OLSR unicast routing protocol, called MOST (Multicast Overlay Spanning Tree), in which we take into consideration the previously described theoretical results. First we present the main algorithms used for the multicast tree construction, followed by the protocol description and specification.

3.1 Overlay Tree Construction Algorithms

As we saw previously, it is more efficient to consider minimum spanning trees. We discuss two algorithms for the overlay tree construction, which achieve optimal normalized multicast cost. The algorithms do not require any more information than what is provided by a link state unicast routing protocol, like OLSR.

Algorithm 1. Basic Minimum Spanning Tree Algorithm

***Input*:** Network graph.
***Output*:** Overlay tree.

1. Find shortest paths between all pairs of multicast nodes.
2. Build complete graph on multicast nodes with costs $c_{ij} = \{$length of shortest path between i and $j\}$.
3. Build minimum spanning tree on the complete graph, rooted at the source node.

In algorithm 1 the construction of the minimum spanning tree (step 3) can be implemented using Prim's algorithm. The resulting tree is an overlay multicast tree, since it consists only of multicast nodes and its links are in fact tunnels in the actual network. Multicasting is achieved when each node forwards multicast packets to its successors in the overlay tree. It must be noted that in a fully distributed protocol, each node must be able to compute the same minimum spanning tree independently from the others. Therefore, we impose an ordering to the multicast nodes based on their IP addresses when executing Prim's algorithm. The computed minimum spanning tree will be directed and rooted to the node with the smallest IP address. However, this fact has no practical importance in the protocol's operation, where the tree will be treated as a shared tree

with no root. Step 1 corresponds to n Dijkstra algorithm iterations. Therefore, the total complexity is $O(n(M + N \log N))$, where n is the multicast group size, N and M are the number of nodes and edges in the network, respectively.

The algorithm's expected complexity can be improved because it is not necessary in practice to compute all shortest paths from each node to all other nodes to build the minimum spanning tree. We propose Algorithm 2 as a faster alternative to compute minimum spanning overlay trees. The algorithm is essentially equivalent to Algorithm 1, but the shortest paths are calculated in conjunction with the minimum spanning tree. Hence, it is not necessary to compute shortest paths between all pairs of multicast nodes. In fact, according to tests in wireless network topologies, this algorithm has an average running time comparable to a Dijkstra algorithm, even when the number of clients increases.

Algorithm 2. Efficient Minimum Spanning Tree Algorithm

***Input*:** Weighted Graph $G(V, E, w)$, Multicast Node Set S, Root Node s.
***Output*:** Predecessor Table π.

```
1. for all (v ∈ V) { d[v] ← ∞; pred[v] ← NIL; }
2. d[s] ← 0;
3. Q ← V;
4. F ← S;
5. while (F ≠ ∅) {
6.       u ←EXTRACT-MIN(Q);
7.       if (u ∈ S) {
8.            d[u] ← 0;
9.            DEL(F, u);}
10.      for each (v ∈ adj[u]) {
11.           if (d[v] > d[u] + w(u, v)) {
12.                d[v] ← d[u] + w(u, v);
13.                if(v ∉ Q) { INSERT(Q, v); }
14.                if (u ∈ S) π[v] ← u;
15.                else π[v] ← π[u]; } } }
```

We denote $G(V, E, w)$ the network graph, where V is the node set, E is the edge set, and each edge e is associated with a cost $w(e)$. We also denote S the set of multicast nodes. The array d associates each node with a distance to the multicast overlay tree, *i.e.*, $d[v]$ corresponds to the minimum distance of node v to the multicast nodes that are already part of the tree. This distance is initialized to 0 for the root node and to ∞ for all other nodes. The array π associates each node with a predecessor multicast node. When this table has been computed, it contains the information needed to represent the overlay tree, since each multicast node will be associated with another multicast node (except from the root). The predecessors of the other nodes in the graph need only be maintained during the computations.

The algorithm manages a set F of multicast nodes that have not been covered yet by the tree, and a min-priority queue Q which includes all nodes, with the

priority attribute being equal to their distance d. In each iteration the algorithm chooses a node with the smallest distance to the overlay tree (step 6), and checks whether it is a multicast node (step 7). In this case, the node's distance is updated to 0 (because the node is added to the overlay tree) and it is removed from the set F. Afterwards, for each chosen node, steps 11−15 check its adjacent nodes on whether their distance can be improved, and update the predecessors appropriately, similarly to Dijkstra's algorithm. However, in this case there are two important differences: the overlay predecessors can only be multicast nodes, hence steps 14−15 perform an additional check; moreover, previously extracted non-multicast nodes might be re-inserted in the priority queue in case their distance to the tree has improved due to the addition of new overlay nodes. The iteration ends when multicast nodes have been covered, hence the improvement in the average case complexity.

3.2 Specification of MOST Protocol

We now present the MOST multicast routing protocol which is based on overlay group shared trees. The protocol must be used in conjunction with a link state protocol, hence we choose to develop an extension to OLSR. One of the advantages of the overlay approach is that only the multicast nodes need to participate in the construction of the multicast tree, while the other nodes serve merely as relays and are not necessarily aware of the multicast communication. This fact facilitates the development of a peer to peer protocol which can be run only by the participating multicast nodes, hence it could be downloaded dynamically by a node whenever it decides to join a multicast communication.

The Optimized Link State Routing protocol [6] is a proactive table-driven routing protocol. An OLSR node uses Hello messages in order to detect neighbors and informs the entire network of its local topology by broadcasting TC messages. Broadcast traffic is relayed via Multi-Point Relay (MPR) nodes. MPR nodes are elected by their neighbors because they cover their two-hop neighborhood. That way broadcast traffic consumes less resources in order to be forwarded to all destinations. A node can either advertize the full neighbor link set, or the advertized link set can be limited to MPR links, *i.e.*, the neighbors that have elected this node as an MPR, while still guaranteing that the shortest paths can be computed.

In a fully distributed spanning tree design, the tree computation is performed independently by each group member. To proceed to the correct computation of the overlay tree, multicast nodes need to know the membership of their multicast group. Therefore, when a node wants to join or leave a group, it broadcasts a Join or Leave message to the entire network, via the optimized MPR-flooding mechanism used in OLSR. Join messages are sent periodically according to the *Join_Interval*, which is set by default to be equal to the TC message interval, *i.e.*, 5 seconds. The total protocol overhead is limited in these messages, independently of the number of groups and sources. In fact, MOST is well suited for managing numerous groups of small size, with arbitrary sources. Each group member must compute periodically the multicast tree to discover and maintain

its overlay neighbors. In order to compute the multicast tree, the node needs information about network topology which is delivered by OLSR. The overlay neighbor set is a subset of clients sharing the same tree with that member, which are linked to it via unicast tunnels. The distance between overlay neighbors can be of one or several hops. Each client receives/retransmits multicast data from/to its overlay neighbors. The computed tree is a group shared tree, hence it must always be the same for all the clients. However, because of changes in the network topology and group membership, there is no guarantee that all clients hold the same tree. Consequently, to avoid loops there is a need to maintain a duplicate table in each client node. Moreover, some redundancy is introduced in data forwarding after tree updates to avoid packet losses.

Managing Join and Leave Messages. The Join message contains the multicast group(s) address(es) that the sender has joined. The message format is the OLSR message format [6]. The Join message contains a list of multicast group addresses. Each multicast node maintains a membership table with the members of all the groups it belongs to. Upon receipt of a new Join message, each concerned node adds the new client to its membership table. The entries in the membership table are also associated with an expiration time, which is determined by the *VTime* field in the Join message header. After this time period the entries are removed, unless another Join message is received, in which case the expiration time is updated. When a node wants to leave a group, it broadcasts a Leave message to the entire network but keeps acting as a group member if it receives data for a predefined transition period. Beyond this period, received packets are not retransmitted. Leave messages follow the same format as Join messages, hence the message contains the group addresses that the originator wishes to leave. We note that in case there is a multicast node failure or disconnection from the network, this event will be accounted for in the OLSR topology table. Therefore, other multicast nodes will act accordingly, as if the problematic node had left all multicast groups, and the multicast communication will not be affected.

Tree Computation and Maintenance. For each group, each client computes periodically the corresponding overlay tree, according to its membership table. Therefore, the client needs to maintain a table with its overlay neighbors. The update period is set by default to the Hello interval, *i.e.*, 2 seconds, which has been found empirically to provide optimal performance. If a new node joins the group or a client leaves the group, the tree is updated immediately. Whenever the overlay neighbors change after an update, the client considers both new, as well as older overlay neighbors for a transition period of 1 second, that is half the update period. The transition period is introduced to make it possible for all the clients to take into account tree changes, and to improve the packet delivery ratio. After that period, only new neighbors are considered. Finally, in order to be able to perform the overlay tree construction according to the previously described algorithm, nodes switch to full-OLSR mode and start advertizing their complete neighbor set whenever they join a multicast group. The reason for this

is that it must be possible to compute the distance between each pair of group members.

Transmission and Forwarding of Data Packets. Unlike common multicast protocols where data packets are transmitted in a broadcast mode, in MOST data packets are encapsulated in unicast packets before being forwarded to the overlay tunnels. Unicast transmissions present important additional benefits when the subnetwork layer in use is IEEE 802.11 [3], as is the case in most actual wireless networks. Firstly, the packet delivery ratio is significantly improved, since packets are retransmitted in case of collisions, while this is not the case for broadcast (or multicast) packets. Secondly, in most 802.11 variants (including b and g), the broadcast frames are transmitted by default at a lower rate than unicast frames. For instance, when 802.11b operates at a unicast data rate of 11 Mbps, the default broadcast rate is 2 Mbps, and most wireless card drivers do not offer the possibility to change it. Therefore, unicast tunnels can actually increase the available bandwidth, since data is sent at a much higher rate. Under these MAC constraints, MOST outperforms protocols using broadcast frames.

When a client receives a multicast data packet[1], it checks whether the packet has already been received. If this is the case, the duplicate packet is dropped. Otherwise, the client forwards the packet to each of its overlay neighbors, except the one from which the packet was received. Source nodes act also as group members, hence they simply send their data in unicast to their overlay neighbors.

4 Implementation Overview

In this section we outline our complete implementation of the MOST protocol for Linux. An overview of the architecture is depicted in Figure 1. The implementation consists of two modules: MDFP and OOLSR.

MDFP (Multicast Data Forwarding Protocol) is a forwarding protocol that enables point to multipoint data transfer. Multicast packets are captured and encapsulated in order to be forwarded inside a multicast tree. This module was developed for use with the MOLSR multicast protocol [1], and we adapted it to also support MOST. OOLSR (Object oriented OLSR) is INRIA's implementation of the OLSR protocol in $C++$ [2]. The core of MOST was implemented as an extension inside this module.

The OOLSR module with MOST extension is in charge of sending and processing Join and Leave messages, as well as computing and maintaining the overlay multicast tree, based on the network topology. The MDFP module is in charge of the actual forwarding of multicast data packets in the overlay tree. For this purpose, it performs encapsulation and decapsulation of data packets, and maintains a table in order to detect duplicate packet receptions. As shown in Figure 1, the two modules constantly exchange information. The OOLSR daemon provides MDFP with up to date overlay neighbors information, which is

[1] In fact all packets are received in unicast, so we refer here to the encapsulated multicast content.

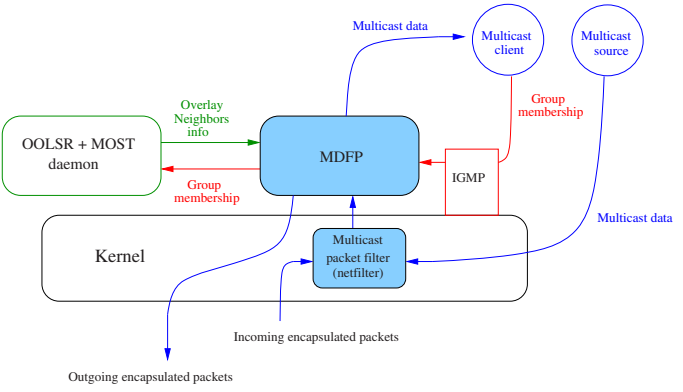


Fig. 1. Overview of multicast implementation

all that is needed to perform the transmission and forwarding of multicast data. Conversely, MDFP communicates to the OOLSR daemon the group membership information concerning the node's OLSR interfaces. In fact, multicast client applications update the interfaces' IGMP information (*cf.* Internet Group Management Protocol [7]), and this information is interpreted by MDFP. Incoming multicast data packets are captured by the *netfilter* module in the kernel, following predetermined rules (such as a predetermined UDP port number). MDFP decapsulates the packets and passes them to the client applications, while it re-encapsulates them in order to forward them to the overlay neighbors. Similarly, data transmitted by a local multicast source is also captured by netfilter and processed by MDFP. Finally, we note that the OOLSR module (including the MOST extension) can be loaded as a plugin in ns-2, hence the simulator shares the same source code as the real implementation.

5 Simulation Results

In this section we perform ns-2 simulations in various scenarios aiming to verify the theoretical minimum spanning tree analysis. We also present some protocol performance measures in a mobile ad hoc network environment. In Table 1, we summarize the parameters that are common for all our simulations.

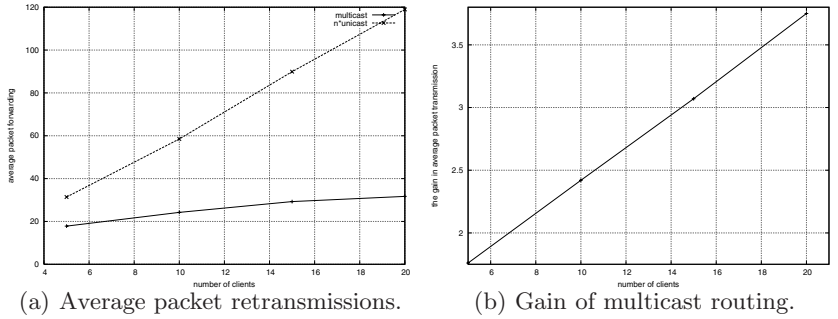
Comparison of Multicast and Unicast Performance. We measure the average multicast cost for various group sizes, by counting the total number of times each packet is relayed to reach all destinations, using MOST. The unicast cost is determined in the same manner, by repeating the same simulations and considering OLSR unicast transmissions between each source-client pair.

The simulation environment consists of a randomly generated topology of 100 wireless nodes forming an ad hoc network, in an area of $1500m \times 1500m$. We consider group sizes ranging from 5 to 20 nodes (not including the source). Multicast groups and sources are chosen at random. The source node sends to

Table 1. Common simulation parameters

MAC Protocol	IEEE 802.11b (11Mb rate)
Propagation model	Two ray ground
Transmission range	250m
Packet size	1200 bytes
Traffic type	CBR
Number of iterations	5

the group CBR traffic of 64kbps, with packets of 1200 bytes, for 150 seconds of simulated time. To obtain reliable results, simulations are conducted several times with 5 different seeds. The mean results are depicted in Figure 2.



(a) Average packet retransmissions. (b) Gain of multicast routing.

Fig. 2. Comparison of multicast versus unicast to all destinations

The analysis in [10] allows to compute an upper bound on the number of multicast packet retransmissions as a function of the number of group clients n. Namely, an upper bound for this parameter is: $\sqrt{2n} \times d_u$, where d_u is the average unicast distance between two nodes in hops. In Figure 3, we compare the average number of retransmissions measured through simulations to the theoretical bound (where border effects are ignored). Although the analysis is performed in an asymptotic setting, we notice that the upper bound is also valid in these simulations.

Multicast Performance versus Throughput and Group Size. Simulations are conducted to determine the maximum source rate that maintains an acceptable delivery ratio in a multicast group. By acceptable we mean that its value is higher than 95%. We consider static topologies again since the goal is to find the saturation point of the network. We consider a 200 wireless nodes network in a $1800m \times 1800m$ area, with one multicast group. We vary the number of clients as well as the source bit rate and we measure the packet delivery ratio, as shown in Figure 4. We notice that the source node can transmit with a rate of up to 200kbps with a delivery ratio higher than 99%. From a 250kbps rate, the performance remains good for small groups but it decreases for large group sizes.

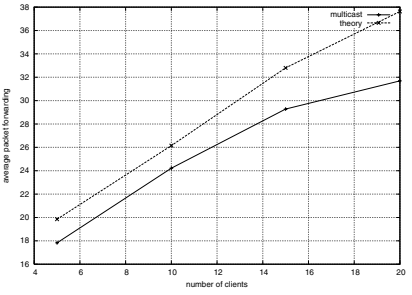


Fig. 3. Simulation results vs theoretical upper bound

We also run other simulations by fixing the number of clients to 10 and varying the number of active multicast groups in a 300 nodes network. In each group a source is transmitting CBR traffic with a rate of 64kbps. As shown in Figure 5 we notice that the delivery ratio is very high until network saturation, with 8 groups of 10 clients.

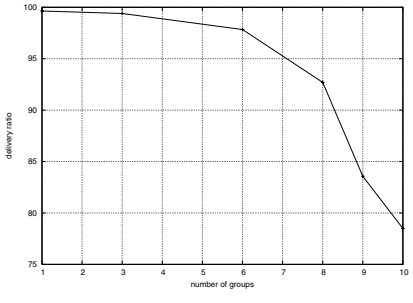


Fig. 4. Delivery ratio (%) vs number of groups in the network

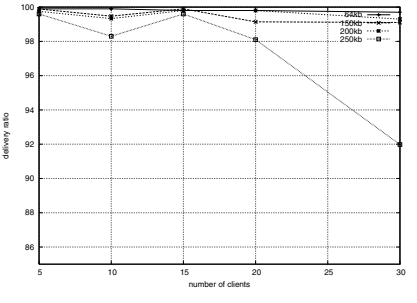


Fig. 5. Delivery ratio (%) vs group size for different source rates

Multicast Performance with Mobility. To evaluate the protocol performance with mobility, we consider a scenario consisting of a randomly generated topology of 200 wireless nodes in an area of $1850m \times 1850m$, and one multicast group in which an arbitrary source node sends a CBR traffic of 64kbps for 300 seconds. We run simulations in which we vary the group members, the number of clients (from 5 to 20 nodes, not including the source) and the maximum mobility speed (from $1m/s$ to $10m/s$). The mobility model is the Random Way-point model with a pause time of 10 seconds: nodes choose a random point in the network area and move to it with a constant speed chosen at random between $1m/s$ and the maximum defined value; after they reach their destination, they remain idle for a period equal to the pause interval and then the same procedure is repeated. Moreover, we consider the following OLSR parameters: a Hello interval of 1 second and TC interval of 5 seconds; we note that the performance of MOST with mobility can be improved by using smaller intervals, at the cost

of higher OLSR control message overhead. The simulations are again repeated several times with 5 different seeds. We measure the multicast packet delivery ratio and the traffic load caused by duplicate packets, and we depict the obtained results in Figures 6 and 7.

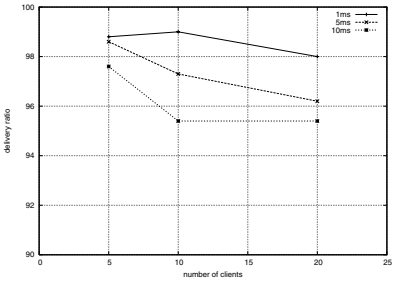


Fig. 6. Delivery ratio (%) vs group size with different mobility speeds

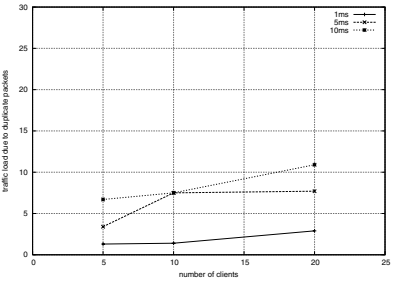


Fig. 7. Duplicate traffic load (%) vs group size with different speeds

As we can see, by varying the speed from $1m/s$, $5m/s$ and up to $10m/s$, the delivery ratio remains acceptable, *i.e.*, higher than 95 % for all groups of up to 20 clients. On the other hand, traffic load due to duplicate packets is higher when the mobility speed or the group size increase. This can be explained by the fact that any change in the topology due to mobility can affect the shared tree. In fact, each client is aware of these changes since it uses the OLSR protocol, and when it recalculates the overlay tree it enters a transition period, during which old and new overlay tree neighbors are maintained. A compromise can be found between the overhead due to duplicate packets and packet delivery ratio by setting a suitable transition period length. We notice that the duplicate traffic load in the network remains small compared to the total traffic load (10% in the worst case), hence the important advantage of improving the packet delivery ratio comes only with a modest performance cost.

6 Conclusion

In this paper we presented MOST, a Multicast Overlay Spanning Tree routing protocol which is an extension to OLSR ad hoc routing. The protocol is fully distributed in the sense that each group member computes and maintains the shared multicast tree independently. The MOST protocol being based on overlay trees guarantees robustness while it achieves good performance, with additional important advantages in the case of IEEE 802.11 networks. The protocol was tested through ns-2 simulations, which verify the corresponding analytical results stating that multicasting can reduce the overall network load by a factor $O(\sqrt{n})$, for n multicast group members. Furthermore, MOST has been tested in real network environments, hence we intend to provide measurement studies of the protocol performance in future work.

References

1. MDFP, http://hipercom.inria.fr/smolsr-molsr/.
2. OOLSR, http://hipercom.inria.fr/oolsr/.
3. IEEE 802.11 Standard, Wireless LAN Medium Access Control and Physical layer Specifications, 1997.
4. C. Adjih, L. Georgiadis, P. Jacquet, and W. Szpankowski. Is the internet fractal: The multicast power law revisited. In *SODA*, 2002.
5. J. C.-I. Chuang and M. A. Sirbu. Pricing multicast communication: A cost-based approach. *Telecommunication Systems*, 17(3):281–297, 2001.
6. T. Clausen and P. Jacquet (editors). Optimized link state routing protocol (OLSR). RFC 3626, October 2003. Network Working Group.
7. W. Fenner. Internet Group Management Protocol, Version 2, RFC 2236, 1997.
8. J. Garcia-Luna-Aceves and E. L. Madruga. The core-assisted mesh protocol. *Journal on Selected Areas in Communications*, 17(8):1380–1294, August 1999.
9. P. Gupta and P. R. Kumar. Capacity of wireless networks. *IEEE Transactions on Information Theory*, IT-46(2):388–404, 2000.
10. P. Jacquet and G. Rodolakis. Multicast scaling properties in massively dense ad hoc networks. In *SaNSO, Fukuoka, Japan*, 2005.
11. J. Jetcheva and D. B. Johnson. Adaptive demand-driven multicast routing in multi-hop wireless ad hoc networks. In *Proceedings of the Second Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001)*, 2001.
12. A. Laouiti, P. Jacquet, P. Minet, L. Viennot, T. Clausen, and C. Adjih. Multicast Optimized Link State Routing, INRIA research report RR-4721, 2003.
13. S. Lee, W. Su, and M. Gerla. On demand multicast routing protocol in multihop wireless mobile networks. *ACM/Baltzer Mobile Networks and Applications, special issue on Multipoint Communication in Wireless Mobile Networks*, 2000.
14. M. Liu, R. R. Talpade, A. McAuley, and E. Bommaiah. AMRoute: Adhoc Multicast Routing Protocol. *UMD TechReport 99-8*.
15. C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (AODV) routing, RFC 3561, 2003.
16. E. Royer and C. Perkins. Multicast Ad hoc On-Demand Distance Vector (MAODV) Routing, IETF, Intemet Draft: draft- ietf-manet-maodv-00.txt, 2000.