Topology-Aware Group Communication Middleware for MANETs

Pedro Garcia Lopez, Raúl Gracia,
Marc Espelt, Gerard Paris, Marcel Arrufat
Universitat Rovira i Virgili
C/Escorxador S/N
43003 Tarragona, Spain
pedro.garcia@urv.cat

Roc Messeguer
Universitat Politècnica de Catalunya
C/Jordi Girona, 1-3
08034 Barcelona, Spain
messeguer@ac.upc.edu

ABSTRACT

We believe that any MANET middleware should be aware of the underlying multi-hop routing protocol to improve communication efficiency. In general, existing MANET middleware either ignore the underlying routing protocol or create specific cross-layer solutions that break the strict layering of the network stack. This problem is even more severe in the case of traditional group communication middleware (GC) where membership protocols, failure detection mechanisms or flow control layers can considerably harm the overall performance of the network.

We propose to move the routing logic to the application layer in order to achieve a smooth and clean integration between the middleware and the underlying MANET topology.

In this line, we have modified a well-known GC toolkit (JGroups) in order to adapt membership protocols, failure detectors and flow control mechanisms to the underlying MANET topology. We have implemented the MANET OLSR protocol in the application layer using UDP (jOLSR). On top of it, we have developed an overlay Multicast protocol (OMOLSR) that directly benefits from the OLSR protocol to improve communication efficiency. As a consequence, in our middleware group membership is obtained from OMOLSR, failure detection from the jOLSR protocol, and our modified flow control protocol benefits from jOLSR topology information. We validate our approach in a real test-bed to demonstrate the feasibility and efficiency of our middleware.

Categories and Subject Descriptors

C2.4 [**Distributed Systems**] Distributed Applications

C2.2 [**Network Protocols**] Routing Protocols

General Terms

Reliability

Keywords

Group Communication, Mobile Ad-hoc Networks, Middleware

1. INTRODUCTION

Mobile Ad-Hoc Networks (MANETs) represent an interesting substrate for many types of applications that do not require a fixed network infrastructure (Access Points). When every participant is a routing node in the network, several interesting multi-hop scenarios may arise. For example, spontaneous collaborative applications are of particular interest for settings where many users can interact with close-by participants (conferences, campus, stadium, popular events, city, and games). In these settings, it is very important for MANET middleware and applications to be aware of the underlying routing protocols and topology. If the middleware just uses the underlying transport protocol as a black box, it can incur in communication inefficiencies due to the multi-hop nature of the medium.

This problem is more severe in the case of traditional GC toolkits working over IP multicast like Ensemble or JGroups. Such toolkits provide message reliability and ordering, group membership, failure detection and flow control among other functionalities. But they are not designed for multi-hop environments and they can really incur in a strong overhead for the underlying network. These toolkits cannot just work transparently over a MANET multicast protocol due to their current design.

A typical solution has been to implement cross-layer solutions enabling the construction of more efficient middleware (using topology information). However, we believe that this approach is flawed because it implies dirty ad-hoc hacks that break the clean network layering model. These approaches solve specific problems but are difficult to port or adapt to different scenarios.

We believe that in this setting is completely justified to move the routing logic to the application layer. In fact, peer-to-peer technologies are a classic example of moving routing logic to the application layer. In this line, content distribution networks or application layer multicast are peer-to-peer technologies created to overcome the limitations of the transport layer. In the case of MANET settings, there exist several reasons to move the routing logic to the application layer:

- Clean interaction between middleware and routing layer
- Flexibility and adaptation to specific requirements
- Simplicity of development, testing and deployment
- Topology-Aware overlay

In the context of the European project IST-POPEYE (Peer to Peer Collaborative Working Environments over Mobile AdHoc Networks), we developed a Java implementation of the OLSR protocol using UDP sockets (jOLSR). To address the group communication issues of the collaborative middleware, we have created OMOLSR, an overlay multicast channel that directly benefits from the jOLSR routing protocol. Finally, using jOLSR and OMOLSR we have modified an existing group communication middleware (JGroups) to support MANET settings.

In particular, group membership is obtained from the OMOLSR protocol, failure detection from the jOLSR protocol, and our modified flow control protocol benefits from jOLSR topology information. We will demonstrate by simulation and in a real test-bed that the system is communication efficient and that it benefits from the knowledge of the underlying topology.

The structure of this paper is as follows: in section 2 we present related work, then in section 3 we present some design decisions and in section 4 we describe the overall middleware architecture. In section 5 we present validation results using both simulation and experimentation, and finally we draw some conclusions in section 6.

2. RELATED WORK

It is widely accepted in MANET settings that cross-layer solutions are required for achieving performance and adaptation in many scenarios [1],[2],[3]. Nevertheless, many researchers also recognize that strict layering enables controlled interaction among layers (each layer is independent developed and maintained). Besides, many cross-layer solutions generate spaghetti-like code that is impossible to maintain efficiently because unexpected dependencies between layers may arise.

To avoid this coupled cross-layer solutions, authors in [1] proposed a vertical component named Network Status that would avoid nasty inter-layer communication. All the layers (MAC, Network, Transport, Middleware and Application) would then communicate directly with Network status and thus avoiding direct coupling between layers. This design is more elegant than the classical cross-layer solutions but still face important problems. On the one hand, it is not a standardized operating system component so it is difficult to port and maintain. The community should agree to such transversal solution with an open standard. This is clearly difficult to achieve in the short term. On the other hand, network status seems appropriate for offering information between layers, but not as a clear medium for upcalls and downcalls. In specific settings, a tight coupling between some layers can be even desirable to achieve optimal performance.

If we focus in the middleware and application layers, it is also obvious that cross-layer solutions clearly present better performance. Both [4] and [5] modify existing peer-to-peer middleware (FreePastry, Gnutella) to benefit from a cross-layer communication with underlying MANET protocols. It is quite clear that if middleware ignores the underlying topology, it can even hurt or degrade the overall performance of other nodes. In conclusion, we can summarize the existing approaches to cross-layer in the following groups: including information in packet headers, using the Internet Control Message Protocol, offering device or kernel ad-hoc connections, creating transversal components for all layers [1], standardizing interfaces and loopback addresses [6], or even allowing direct signaling among layers [7]. Some projects have even considered a complete rethinking of the network stack for MANET environments [8].

As we will explain in the following sections, our approach for MANET middleware is simpler: moving the routing logic to the application layer. This is probably not a solution for all kind of problems and we neither aim to replace existing transport layer MANET protocols. But we will explain how this simple solution can be appropriate for many scenarios where cross-layer information flows are required by middleware and applications. In fact, other research works in MANET middleware already followed this approach. In [9] authors present COMAN, a protocol to organize the nodes of a MANET in a tree-shaped network providing content-based routing functionalities. COMAN is implemented at the application layer and it benefits from topology information to self-repair and reconfigure the middleware. In [10] authors also present a content-based publish/subscribe system for MANETs that extends the ODMRP (On-Demand Multicast Routing Protocol) to construct an optimized dynamic dissemination mesh. Authors in [10] stress the importance of cooperation between the middleware-tier and network components.

Regarding GC services over MANETs we must outline two research works: JazzEnsemble [11] and MobileMan [12]. The authors of JazzEnsemble specifically addressed the problems of developing a GC middleware over MANETs. They studied in diferent works [11],[13],[14] problems regarding group membership protocols, failure detection and flow protocols over MANETs. They devised lightweight membership protocols based on fuzzy membership and Random Walks, gossip-based failure detection protocols, and they also adapted flow protocols to MANET environments. Nevertheless, they did not construct their GC primitives (membership, failure detection) over standard MANET unicast and multicast routing protocols (OLSR, OMOLSR) like us. As we will explain later, our decision entails important communication savings.

Finally, in [12], authors constructed very basic GC services with their cross-layer solution based on Pastry Distributed Hash Table (DHT) and Scribe Application Layer Multicast over the OLSR MANET protocol. They did not specifically addressed problems like membership, failure detection or flow control and their proposal was mainly a prototype. Furthermore, it is questionable if Distributed Hash Tables are the right solution for a reduced MANET setting. Furthermore, the Scribe multicast protocol is not designed for MANETs, like other MANET multicast protocols (MOLSR, OMOLSR). It is obvious that Pastry and Scribe will then generate more communication overhead than native MANET protocols like OLSR or OMOLSR.

3. DESIGN DECISIONS

Before delving into our main technical contribution, we first justify our decision of moving routing logic to the application layer, and then we explain the design decisions behind the chosen unicast and multicast routing protocols.

3.1 Advantages and drawbacks of application-layer MANET routing

The idea of moving routing logic to the application layer is not new. The peer-to-peer paradigm is frequently used to overcome the limitations of the transport layer. For example, Application

Layer Multicast (ALM) solutions have emerged due to the problems with IP multicast in the Internet. ALM solutions create overlay topologies that permit efficient one-to-many communications using unicast transport communications. One of the major issues with these overlay networks is to correctly reflect the underlying topology to be efficient in terms of latency. In this line, the routing latency between two nodes on the overlay network can be different from the unicast latency between those two nodes on the underlying network. The ratio of these two quantities is called the latency stretch. If this ratio is high, there is a considerable penalty in using the overlay. In fact, it is still an open problem in peer-to-peer research to create topology-aware overlays.

Hereafter, we will present the main advantages of our approach:

Topology-Aware overlay. In MANET environments, nodes are connected in a peer-to-peer network created in the transport layer using protocols like OLSR, AODV or DYMO. If we just move these protocols to the application layer (OLSR for example), the topology does not change. The application layer peer-to-peer network will then reflect exactly the physical connections. Furthermore, the application layer protocol will route packets using UDP unicast connections in the same way that the transport protocol would operate. This is a remarkable advantage of MANET application layer routing and avoids many problems found on Internet topology-unaware overlays.

Clean interaction between middleware and routing protocols. When MANET routing is just another middleware component we can use well-established software engineering practices to specify module interaction. With standardized APIs in the application layer it is not necessary to break the strict layering of the network stack for MANET middleware.

Flexibility and adaptation to specific requirements. Different applications may have completely different requirements for the routing layer and underlying topology. If it is a matter of changing a software component, it is feasible to develop specific adaptive routing layers for different settings. Emergency scenarios, military applications or collaborative work settings may develop specific underlying routing components adapted to their particular scenarios.

Simplicity of development, testing and deployment. It is easier to develop and test routing protocols in the application layer than in the transport layer. The middleware can be created transparent from the operative system and thus simplifying kernel compilation, library dependencies and hardware settings. Furthermore, the key point is easy deployment and portability to different platforms. Mobile phones could easily install an application that creates its own MANET adapted to its specific requirements. It is far more difficult to motivate users to install new transport protocols.

On the contrary, we find these major drawbacks:

Specific routing service. At the time present, our approach does not offer a generic service for all the applications running in a OS (like transport protocols do). Only the application using this middleware will benefit from the MANET routing services. If we have for example existing videoconferencing or desktop sharing tools using standard transport protocols, they won't be able to benefit from our middleware. In any case, this problem could be overcome in the same way that IP over Peer-to-Peer Networks (IPOP) are working today. If necessary, the middleware component could then even provide standard transport level services to legacy applications.

Performance. It is obvious that working in the user-space implies a penalty over kernel transport protocols. Besides this, the implementation in Java may also damage the performance of the protocol compared to native kernel implementations. We justify in the validation that our middleware achieves performance numbers that are reasonable for a large number of applications.

3.2 MANET Protocols

Here we include a brief summary of recent unicast and multicast protocols considered in our design. Concerning unicast protocols, we considered two recent protocols (one reactive and one proactive):

DYMO (Dynamic MANET On-Demand) [15] is a reactive unicast protocol, successor of the popular AODV (Ad-Hoc On-Demand Distance Vector) and shares many of its functionalities. Routes are created on-demand by sending request and response control packets. In consequence, no global topology information is available. On the other hand, this means that when nodes stop sending messages, there is no overhead traffic in the network. DYMO seems more suitable for sparse communications, but may not work as well in case of congestion.

OLSR [16] is a proactive unicast protocol, so it maintains routing table information up to date continuously. Topology information is exchanged by means of controlled flooding of topology messages (TC messages). Hello messages provide information about the two-hop neighborhood in a way that each node selects a neighbor as MPR (Multi-Point Relay). These MPRs are in charge of sending topology messages to the entire network performing controlled flooding. With the topology information, each node can build the routing table in order to be able to send messages to the other nodes. OLSR performs well in small-medium sized networks where node density is relatively high. The knowledge of the topology, together with its good performance with dense communication patterns turns OLSR in a good candidate for performing group communication in MANETs.

Concerning MANET multicast protocols, we also evaluated several network multicast and application level multicast protocols. Let us compare the more important systems studied:

ALMA (Application Layer Multicast Algorithm) [17] creates a tree of logical links between the group members. The aim of this protocol is to reduce the cost of each link in the tree by reconfiguring the tree under mobility and congestion situations. When a node joins the network it must select a node as a parent, so as to become part of the tree. If tree performance drops below a defined threshold, the node must reconfigure the tree by switching the parent or freeing children. This mechanism leads to a complex loop avoiding and detection system, since synchronous switching can occur. ALMA also considers the existence of a rendezvous host for obtaining the structure of the logical tree as well as neighbor information in the bootstrapping process.

PAST-DM (Progressively Adaptive Subtree in Dynamic Mesh) [18] is an overlay multicast protocol based on the construction of a dynamic virtual mesh. The mesh is maintained dynamically through the exchange of link state packets, thus adapting to network topology changes. These packets provide link state table information, that is, a partial view of the network. With the topology information extracted from the mesh, nodes compute a source-based Steiner tree to deliver information to all members in the multicast group. Logical and physical hop distances are used as heuristics to compute the Steiner tree. Each child of the source tree is responsible for delivering the multicast message to all nodes in the subtree. This process is repeated through every node until the subtree becomes empty. The decision of packet delivery path is computed at each receiver, so path selection is performed always with the most up-to-date information. Although this is an efficient way of delivering data, some packets may be lost if nodes change location, once the source node has computed its corresponding subtree.

In conclusion, both protocols, as most application level protocols, need to send control packets to keep their structures up to date: using periodic exchange of link packets or communicating with other nodes in the tree. In both cases, the protocols ignore the underlying unicast network routing protocol so they perform redundant communication and, in consequence, bandwidth misuse.

Because our group communication middleware must provide both unicast and multicast services, we believe that it will be optimal to use multicast protocols that benefit from the OLSR unicast protocol. This leaves us with two main protocols: SMOLSR and MOLSR:

SMOLSR (Simple Multicast OLSR) is a straightforward multicast protocol that simply floods the network using the OLSR Multipoint Relays (MPR). Since this protocol does not build any tree, all the nodes receive all the messages even if they are not in the multicast group. Hence, SMOLSR is not efficient for multi-group networks although it does not send any control message to the network. MOLSR (Multicast extension for the Optimized Link State Routing protocol) [19] is a multicast routing protocol for MANETs designed to work in top of OLSR nodes. It is a source tree based protocol that maintains one multicast tree per tuple (source, multicast group). This stateful protocol provides multicast routing that benefits from the topology knowledge of OLSR (MPRs). Although MOLSR is an interesting alternative, we finally decided to design our own application level multicast tailored for group communication on top of OLSR. We will further explain our OMOLSR protocol in the following section.

4. M-CHANNEL: A TOPOLOGY-AWARE GROUP COMMUNICATION MIDDLEWARE FOR MANETS

Instead of constructing an entire GC toolkit from scratch we decided to modify a well-known Java toolkit like JGroups [20]. JGroups is a toolkit for reliable multicast communication that offers a JChannel abstraction providing group membership (methods to connect, disconnect, and obtain the current members of a group), methods to send messages to one or to all members in a group, and event listeners about the channel state (member joins, leaves and received messages). Furthermore, JGroups offers a flexible protocol stack with many existing protocols for group membership, failure detection, flow control, fragmentation of packets, and message ordering and reliability over IP multicast.

Our M-Channel abstraction extends the JGroups JChannel to incorporate the information from the underlying MANET topology. As depicted in Figure 1, the multi-hop channel (MChannel) uses both unicast and multicast functionalities provided by the two underlying protocols: jOLSR and OMOLSR. In first place, OMOLSR computes multicast routing thanks to the information received via events coming from jOLSR. These events allow OMOLSR to update the OMOLSR Network Graph (ONG), which will be used for routing computation as well as for providing membership information. As we can see, both jOLSR and OMOLSR have been integrated as standard protocols in the JGroups stack.

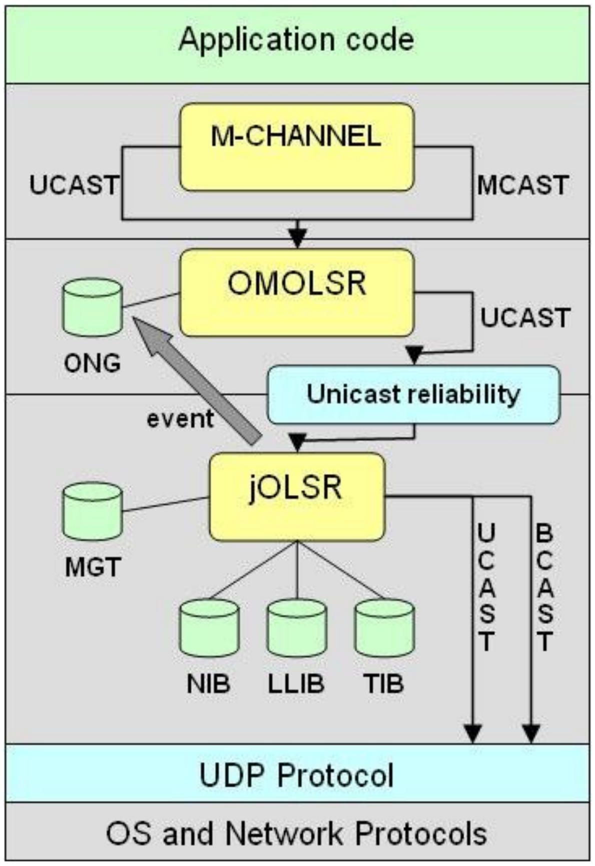

Figure 1. Middleware architecture

To explain the overall architecture, we will proceed now bottom-up: we will first describe our jOLSR unicast MANET implementation, then the OMOLSR overlay multicast, and finally the services provided by the GC toolkit (M-Channel).

4.1 JOLSR

jOLSR is an application level implementation of the OLSR routing protocol [16] written in Java. Of course, the goal of jOLSR is not to be a fully compliant implementation of the standard, but an implementation which follows its basic functionalities. In any case, jOLSR implements nearly all components of the core functionality of OLSR. Although the core functionality also includes support for multiple interface addresses, this feature is not provided in the current version of jOLSR in order to simplify the implementation.

jOLSR stores network information in different tables similarly to OLSR specification: Neighborhood information base (NIB) stores neighbor information; Local Link Information Base (LLIB) keeps updated information about the state of links to the neighbors; Topology Information Base (TIB) maintains information of the network topology to perform routing calculation.

Some modifications have been added to the basic specification of OLSR to provide topology and group membership information to the upper multicast protocol. The multicast protocol, OMOLSR, will benefit from this information in order to avoid flooding the network with unnecessary control packets. The main changes are the following:

- TC messages now include a list of multicast groups that are joined by the sending node.
- A new table called Multicast Groups Table stores the information about multicast groups received in TC messages. With this information, a partial representation of the network, similar to a graph, is computed when a change is detected in the different tables.

TC Message Modification. OLSR and jOLSR send two different types of control messages, HELLO and TC (Topology Control) messages. HELLO messages permit a node to know its one-hop and two-hop neighbors, since each node sends information about its local neighborhood. Based on this information, the node can select its multipoint relays (MPR) which will be in charge of performing controlled flooding. TC Messages are sent to all the nodes in the network thanks to this controlled flooding mechanism, and disseminate topology information of the local node to all nodes in the network.

In both OLSR and jOLSR, TC messages are sent periodically from one node to the rest of the network, so all nodes can compute its topology table. We have realized that by adding little information in TC messages, we can disseminate information easily to all nodes in the network. This is in fact really useful since the multicast routing protocol needs information about the multicast groups joined by each node.

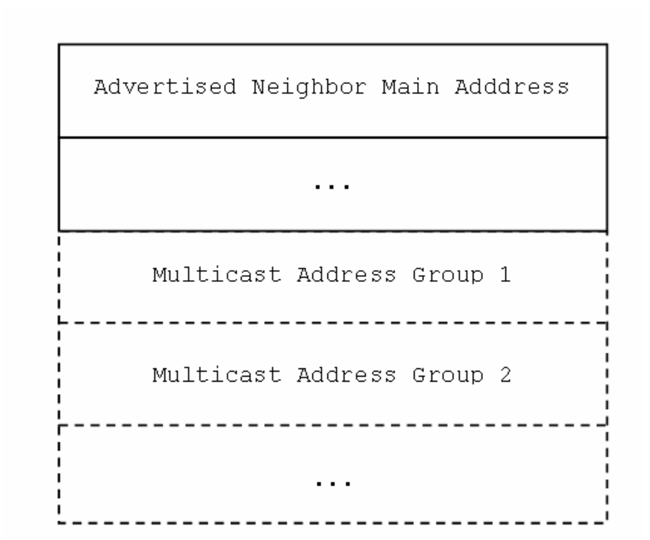
Advertised Neighbor Main Adddress
...
Multicast Address Group 1
Multicast Address Group 2
...

Figure 2. TC message modification

Therefore, we propose to attach the multicast address of the groups joined by the local node in TC messages, as we can see in Figure 2. The attached information about the groups is retrieved from the multicast group table.

Multicast Group Table. The Multicast Group Table keeps the information about the multicast groups joined by each node. This information is updated when the local node decides to join a new group or when it receives a new TC Message. The table keeps a set of the joined multicast groups for each node in the network. The information in this table is used by the upper multicast routing protocol and changes in the table are reported as membership events to the application. When a change is detected in the neighbor table, in the topology table or in the multicast groups table, a graph containing the members of the group is computed. The multicast protocol will receive a new event with the information of this graph.

In order to calculate the graph with the members of the group, we obtain an approximate representation of the network by creating a network graph from the information stored in the topology table. Then we check in the Multicast Group Table which nodes belong to which group, so we create an event for each different group in the table. The graph can then be used by the multicast protocol for retrieving updated membership information.

4.2 OMOLSR

OMOLSR (Overlay Multicast over OLSR) is a new application level multicast routing protocol, designed to work on top of jOLSR. OMOLSR computes locally minimum spanning trees by benefiting from the topology information gathered by jOLSR. The main characteristic of OMOLSR is that it does not need to send additional control packets to perform multicast delivery. The unicast routing protocol already provides all necessary information. Besides, this application level multicast protocol is designed for group communications because it benefits from its close interaction with the undelying unicast protocol (jOLSR) to provide a lightweight membership protocol.

Basic Operations. The basic operations of OMOLSR are the dynamic computation of the Minimum Spanning Tree and the routing of multicast packets. In first place, OMOLSR computes a virtual mesh that connects all members of the multicast group. This mesh is a contraction of the network graph that jOLSR generates from the Neighbor Table and the Topology Table. By using this contracted graph, which contains only the members of the group, the local node can easily know which members are located at one hop in the membership. These nodes located at one logical hop are also known as virtual neighbors of the local node. The procedure of computing the contraction of the graph is performed for each multicast group the local node is member of.

Then, once the graph is ready, we compute a minimum spanning tree with the local node as the source of the tree, as depicted in Figure 3. The tree will now be used for routing the packet to all the members of the group.

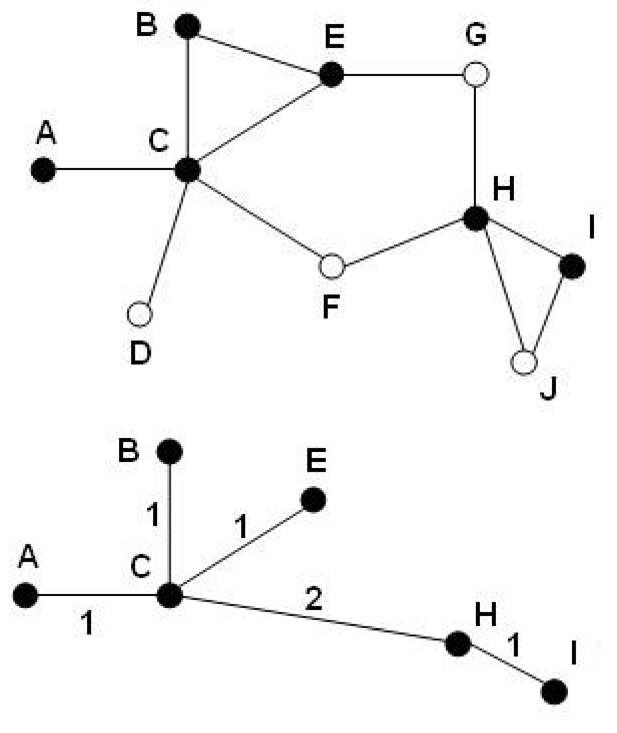

Figure 3. The spanning tree for the node A. Colored nodes belong to the multicast group. Edges

Multicast routing. In order to route multicast packets, OMOLSR uses an explicit multi-unicast scheme. When the application generates a new multicast packet, OMOLSR routes the message based on the tree computed for that multicast group. A copy of the packet is sent to each virtual neighbor, which is responsible for delivering the message to a certain subset of nodes. This subset is defined in each node by using the source-created tree and consists of all the nodes that are in the subtree of each virtual neighbor. This information is then attached to the header of the data packet. When the virtual neighbor receives the message, it computes a tree with the subset of nodes contained in the header. Again, it sends a copy of the data message to its virtual neighbors with new header content. The process is repeated until the subset which must receive the message is empty.

Reliability and integration. In order to ease and clarify the development of both jOLSR and OMOLSR, we have extended an existing toolkit for reliable communication: JGroups. The key feature of JGroups is its flexible protocol stack, which can be configured and extended depending on the communication needs. Each protocol in the stack provides different functionalities: ordering, reliability, membership, state transfer, etc. In our case, we have implemented both routing protocols (jOLSR and OMOLSR) as JGroups protocols so we can benefit from unicast reliability and ordering by adding the UNICAST protocol to our stack. This unicast reliability layer uses an acknowledgement scheme to provide lossless transmission of unicast messages. Therefore, multicast reliability is also ensured by adding this layer between both protocols: OMOLSR splits each multicast packet in several unicast packets that will be sent under the acknowledgement scheme.

The flexibility of the stack is also useful for validation purposes. For instance, we have also changed the lowest JGroups protocol that creates UDP sockets for another one that virtualizes communication so we could easily create an emulation layer.

4.3 M-Channel

On top of the routing protocols we have developed a channel which enables flexible group communication over mobile ad-hoc networks: the M-Channel. The main characteristic of the MChannel is that users can send messages to a single member or to all the members in the group even if they are not in range. In consequence, a MChannel is bound to a single group, so if we want to communicate in two groups, we should create two different channels.

When we designed the GC middleware for manet settings we had to cope with three main issues: group membership, failure detection and flow control.

Group Membership: GC toolkits like JGroups maintain membership and failure detection pinging frequently or using keep-alives to all members in the JChannel. Whereas this approach works fine in local area networks, it can severely harm the overall MANET network creating unnecessary traffic. This happens because JGroups is unaware of the multi-hop nature of the medium. Our approach is to provide a lightweight group membership protocol that directly benefits from the group information of the OMOLSR multicast tree. In our case, membership changes in OMOLSR are injected as JGroup membership events to the protocol stack.

Failure Detection: As explained before, failure detection implies pings or keep-alives to all group members. Again, this works in flat groups, but it causes a real burden in a multi-hop network. Our solution is to rely on the JOLSR topology detection algorithms. jOLSR is already checking the availability of nodes and continuously repairing the topology graph. Because of that, it is optimal to benefit from this information to detect leaving parterns of failing nodes. It is nonsense to duplicate the communication overhead if jOLSR is already doing that job in an efficient and decentralized way.

Flow Control: Sending messages to the network without any flow control causes congestion and degrades the network throughput. This is even worse in a multi-hop network where throughput decays as the number of hops increases. In fact, throughput degradation due to hop count has been well studied by Gupta et al [21].

JGroups already provides a simple flow control protocol based on a credit system. Each sender has a number of credits (bytes to send) and when the credits have been exhausted, the sender blocks. Each receiver also keeps track of how many credits it has received from a sender. When credits for a sender fall below a threshold, the receiver sends more credits to the sender. Again, the existing algorithm does not take into account the underlying multi-hop setting so data flows are not optimized for the underlying network.

We have modified the JavaGroup FC (flow control) protocol to benefit from topology information. Our strategy is to assign credits to nodes in different proportions depending if they are at one hop, two hops or more. We assign credits to nodes in a proportion that depends in its decay of throughput due to the multi-hop setting. The closest node (one hop) will obtain more credits whereas distant nodes (more hops) will get less tickets. With this decision we aim to adjust the flow to the available throughput between nodes.

Apart from these functionalities, the M-Channel could offer more complex services in order to build MANET applications: quality of service (QoS) considering multiple parameters at a time (bandwidth, battery, CPU,...) to perform routing, hence providing adaptive middleware; scoped multicast delivery enabling TTL parameters related to MANET routing hops. In this way, multicast messages could be restricted to a certain groups of (closer) nodes; and MANET anycast services enabling filtering of events in nodes.

We believe that the definition of these services could help to build a useful library for collaborative MANET applications. Thanks to the simplicity of our communication middleware for MANETs, we believe that many future applications could use it and benefit from its services. We offer a working prototype, with clear APIs, and integrated in an existing well-known group communication middleware (JGroups). Finally, the interesting point is that our middleware is self-contained: we do not rely on any installed MANET transport protocol. In our case, the application creates the MANET network.

5. VALIDATION

The work presented in this paper is not an architectural draft of a middleware for communication in MANETs, but a real implementation of this middleware. Thus, our objective is not to show performance simulations of the application layer routing protocols showing improvements over other protocols. Our main goal is to provide a ready-to-use middleware to build group applications in MANET networks. Nevertheless, we need to validate our approach both in emulation and real environments, to verify that our middleware is able to be the basis of group applications over ad-hoc networks.

5.1 Emulation

As aforementioned before, an emulation transport layer was implemented in order to validate our approach. We replace the UDP layer of the JGroups protocol stack for a virtualized layer: socket message delivery is replaced by local message queues. It is worth saying that we used the final implementation of the middleware with only a change in the transport layer, so we could validate the middleware under emulated and real environments without changing a line of code.

The emulator works as following:

1. It loads a dynamic network graph from a text file. This file follows the Pajek format [22], describing the nodes of the network and the edges that link the nodes.
2. The emulator applies the connectivity constraints in each simulation step.
3. Then, the emulation layer simply adds all outgoing unicast packets (from the jOLSR protocol) to the receiver queue of the destination node of the message if this node has a direct connection with the source node. Similarly, broadcast packets are delivered to the queues of all nodes that have a direct connection with the source node in that moment.

The emulation layer can also simulate the loss of broadcast packets, a useful feature to test the convergence of the routing protocols when broadcast control packets are not received.

We tested both protocols (jOLSR and OMOLSR) in different topologies, varying the number of nodes and the density of the network. The applications used in the validation generate unicast traffic to all the nodes in the network and multicast traffic for different multicast groups. We could see that the performance was acceptable and the data delivery ratio was almost the optimum thanks to the unicast retransmission layer.

5.2 Real Test-Bed

As pointed out in [23], there is a serious lack of real-world experiences in MANET research. Usually, researchers focus in solving problems that are not observed in real world experiences. However, since our middleware is intended to be used by real applications, we believe that a real-world test should be done to verify the performance of the routing protocols.

In order to test our middleware, 40 computer science students were told to participate in a MANET test with their laptops (Figure 4). They were distributed in several groups along the campus, establishing different ad-hoc networks of maximum diameter Ø = 4. Each node runs a test application that was able to join a chat with the other members of the group. The application kept track of all sent and received packets as well as periodical pictures of the underlying topology graph. The test involved heterogeneous devices with different O.S. (Windows XP, Vista and Linux) and different wireless cards. Test results were successful, showing that both unicast and multicast traffic was sent and received correctly. In our tests we achieved that 20 laptops participated in the same multicast group with a network diameter of 4 hops.

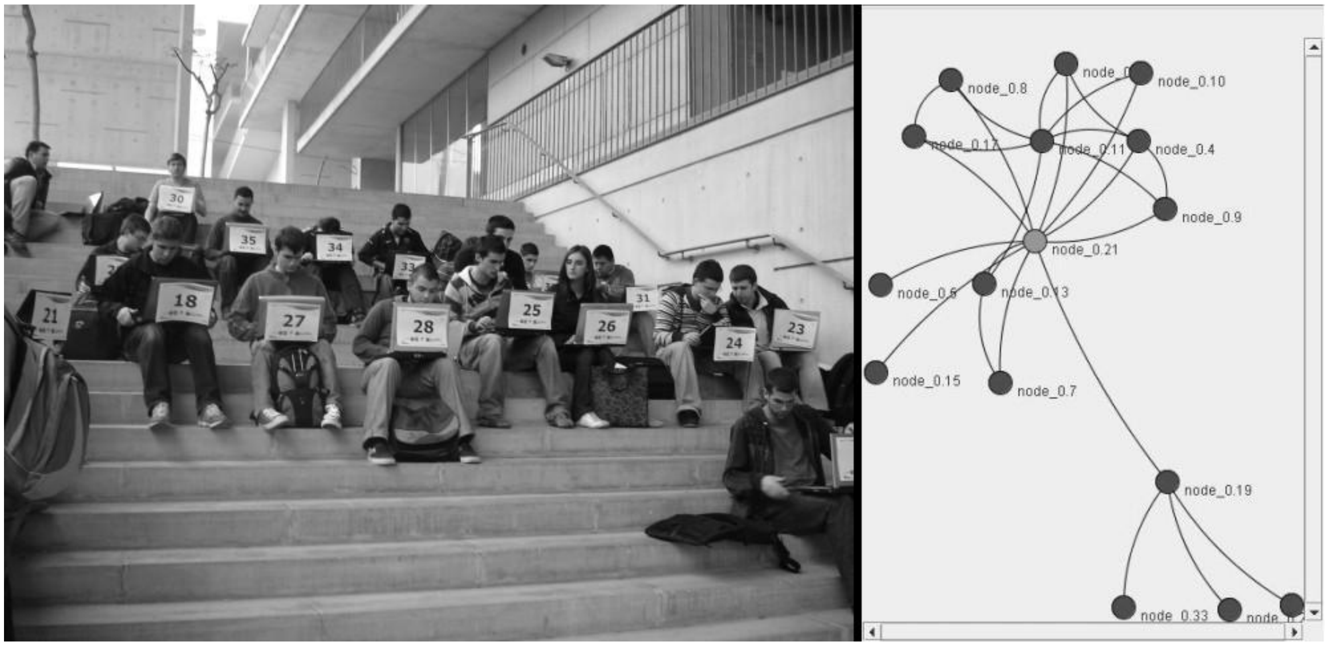

Figure 4. Picture of the real test and screenshot of the network graph as seen by one of the participant nodes.

To verify the feasibility of our middleware, we sent a file (15MB size) from one node to other nodes at different hops and we measured the throughput of the channel. The test was performed with nodes at one, two, and three hops to validate the decay in throughput while transferring the file. Note that we used our modified Flow Control protocol and a JGroups Unicast reliability protocol that send retransmissions if the packet does not arrive to the destination. For this reason, our packet delivery ratio is 100% thanks to retransmissions, but we can observe in red the proportion of retransmissions in each hop. As we can see in the Figure 5, we obtain a reasonable performance ranging from almost 1MB per second at one hop, and decaying until around 200KB per second at four hops.

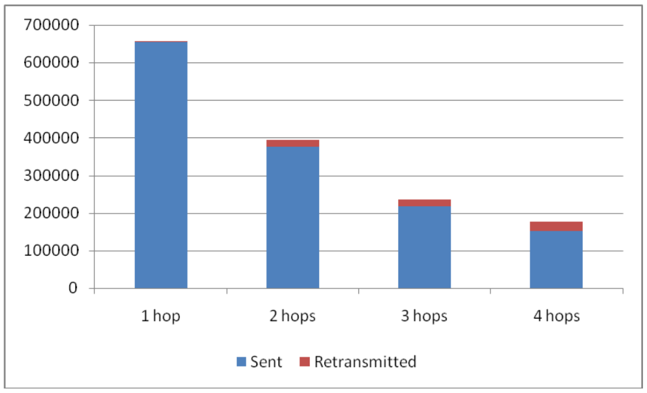

Figure 5. Bytes/s transmitted in different number of hops

We can see that in this case the proportion of retransmissions is low in proportion. These numbers make our middleware feasible for a large number of synchronous applications like shared whiteboards, tele-pointers or file sharing. Furthermore, this throughput is reasonable for voice communication applications and even more if we remove the reliability protocol (unicast retransmissions) and the overhead that it imposes. We must also note that Java itself is imposing a high overhead and it limits the maximum throughput of the channel. We are still trying to improve these limits configuring the Java Virtual Machine.

Finally, we also tested the resiliency of the middleware to node failures. We sent a file from one node to another one with one intermediate hop (node 3). While transferring the file, we halted the intermediate node 3 to verify that the system would self-repair. As we can see in the figure 6, when the intermediate node halts, the first node sends some retransmissions, but then jOLSR detects the failing node and updates the topology graph. In that moment (few seconds before) the traffic to the end node is rerouted through node 2. This path is a less powerful node with more interferences but finally all data arrives to the destination.

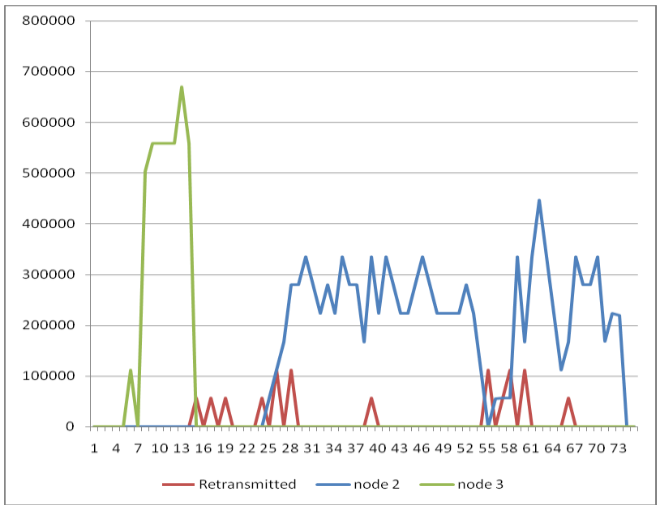

Figure 6. Bytes/s retransmitted by the sender and received coming from node 2 and 3

Although the overall system is working fine, we are still trying to improve the flow control protocols among nodes. A severe problem happens when powerful nodes flood with traffic less powerful nodes and thus causing many retransmissions. We are now designing adaptive and self-adjusting flow control protocols that try to adjust credits to the available throughput between each pair of nodes. We could also try to arbitrate traffic flows at the application layer to avoid collapsing the network with unnecessary competing data. We also foresee interesting optimizations in the interaction of the routing protocols and the middleware and applications.

6. CONCLUSIONS

We have presented a novel topology-aware group communication toolkit for mobile ad-hoc networks. We argue in this paper that any GC middleware should be aware of the underlying multi-hop routing protocol to improve communication efficiency. For that reason, and avoiding dirty cross-layer solutions, we decided to implement unicast (OLSR) and multicast (OMOLSR) MANET protocols in the application layer. As a consequence, our middleware directly benefits from the topology information of the protocol and we avoid unnecessary traffic at a higher level.

We have modified a well-known GC toolkit (JGroups) in order to adapt membership protocols, failure detectors and flow control mechanisms to the underlying MANET topology. In our middleware, group membership is obtained from OMOLSR (overlay multicast), failure detection from the jOLSR protocol, and our modified flow control protocol benefits from jOLSR topology information.

To the best of our knowledge, we do not know any other group communication middleware for MANETs built on top of existing unicast and multicast MANET protocols. We offer a working prototype, with clear APIs, and integrated in a existing well-known group communication middleware (Jgroups). Furthermore, our middleware is self-contained and we do not rely on any installed MANET transport protocol.

We validated our middleware by emulation and in a real test-bed (40 campus users). We have proven that the protocol is consistent, fault-tolerant and scalable to at least forty nodes with a maximum of four hops. The protocol self-repairs the network in a scalable way, and we proved that the system is resilient to failures. In our test, one route was broken, and the system just rerouted the traffic to other existing intermediate node. Furthermore, thanks to our modified Flow Control protocol we obtained a reasonable throughput when sending large files to different nodes. Our performance numbers enable the development of many synchronous applications including voice transmission, tele-pointers, shared whiteboards and one-to-many presenter tools over the MANET.

We foresee more work in this line to create specific routing middleware depending on application requirements and scenarios. We also believe that mobile devices will clearly benefit from such application layer approaches that are neither intrusive nor dependent with the underlying mobile device OS. Every application will then be able to construct a MANET network adapted to its communication requirements.

All the source code is freely available for download at http://ast-deim.urv.cat/mchannel. In this site is also available a youtube video of the MANET experiment.

7. ACKNOWLEDGEMENTS

This work was partially funded by the Spanish research project P2PGRID (TIN2007-68050-C03-03) of the Spanish "Ministerio de Ciencia e Innovacion".

8. REFERENCES

[1] Conti, M., Maselli, G., Turi, G., Giordano, S.: Cross-Layering in Mobile Ad Hoc Network Design. IEEE Computer Vol. 37, Num. 2, pp 48-51, Feb. 2004.

[2] Goldsmith, A.J., Wicker, S.B.: Design Challenges for Energy-Constrained Ad Hoc Wireless Networks. IEEE Wireless Comm., vol. 9, no. 4, 2002, pp. 8-27.

[3] Lin, X., Shroff, N.B., Srikant, R.: A Tutorial on Cross-Layer Optimization in Wireless Networks. IEEE Journal on Selected Areas in Communications. Aug. 2006. Volume: 24.

[4] Delmastro, F.: From Pastry to CrossROAD: CROSS-layer Ring Overlay for AD hoc networks. Conf. on Pervasive Computing and Communications Workshops (PerCom 2005 Workshops).

[5] Turi, G., Conti, M., Gregori, E.: A Cross Layer Optimization of Gnutella for Mobile Ad hoc Networks. Proc. ACM MobiHoc Symposium, Urbana-Champain, May 2005.

[6] Stine, J.A.: Cross-Layer Design of MANETs: The Only Option. Military Communications Conference, 2006. MILCOM 2006, pp. 1-7. 23-25 Oct. 2006.

[7] Wang, Q., Abu-Rgheff, M.A.: Cross-layer signalling for next generation wireless systems. In: IEEE Wireless Commun. And Net. Conf., 2003, pp. 1084 -1089.

[8] BAA05-42, PIP for DARPA ATO, Control-Based Mobile Ad-Hoc Networking Program.

[9] Mottola L., Cugola G., Picco G.P. A Self-Repairing Tree Topology Enabling Content-Based Routing in Mobile Ad Hoc Networks. IEEE Transactions on Mobile Computing. August 2008 (vol. 7 no. 8) pp. 946-960. http://home.dei.polimi.it/cugola/Papers/topologyManagement.pdf

[10] Yoneki E. and Bacon J. Distributed Multicast Grouping for Publish/Subscribe over Mobile Ad Hoc Network. IEEE Wireless and Communications and Networking Conference 200. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=01424873

[11] Friedman R. JazzEnsemble: A Group Communication Middleware for MANETs. Minema 2004 http://www.minema.di.fc.ul.pt/secondworkshop/RFriedman.ppt

[12] Conti M, Crowcroft J., Delmastro F., Passarella A. P2P support for Group-Communication Applications: a Cross-Layer Approach for MANET Environments. INFOCOM 2006 http://www.comsoc.org/confs/infocom/2006/Posters/1568980817_P2P%20support%20for%20Group-Communication.pdf

[13] Bar-Yossef Z, Friedman R, and Kliot G. Evaluating Failure Detection in Mobile Ad-Hoc Networks. International Journal of Wireless and Mobile Computing Vol. 1 No. 8, 2005

[14] Bar-Yossef Z, Friedman R, and Kliot G. RaWMS - Random Walk based Lightweight Membership Service for Wireless Ad Hoc Networks http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/2006/CS/CS-2006-05.pdf

[15] Chakeres, I., Perkins, C.: Dynamic MANET On-demand (DYMO) Routing. Internet draft: draft-ietf-manet-dymo-12.

[16] Clausen, T., Jacquet, P.: Optimized Link State Routing Protocol (OLSR). Request for Comments 3626, October 2003.

[17] Krishnamurthy G, Faloutsos, M.: Application versus network layer multicasting in ad hoc networks: the ALMA routing protocol. Elsevier Ad Hoc Networks Journal, vol. 4, no. 2, pp. 283-300, 2006.

[18] Gui, C., Mohapatra, P.: Efficient overlay multicast for mobile ad hoc networks. In: The Wireless Communications and Networking Conference (WCNC), New Orleans, Louisiana, USA, Mar. 2003

[19] Jacquet P., et al.: Multicast Optimized Link State Routing. Internet-Draft, draft-ietf-manet-olsr-molsr-01.txt, November 2001.

[20] JGroups Toolkit. http://www.jgroups.org/.

[21] P. Gupta and P.R. Kumar, “The capacity of wireless networks,” IEEE Transactions on Information Theory, vol. 46, no. 2, pp. 288–404, March 2000.

[22] Batagelj, V., Mrvar, A.: PAJEK: Program for large network analysis. Connections, 21:47-57, 1998.

[23] Gunningberg, P., Lundgren, H., Nordstrom, E., Tschudin, C.: Lessons from Experimental MANET Research. Ad Hoc Networks Journal, special issue on “Ad Hoc Networking for Pervasive Systems”, Vol. 3, Number 2, March 2005.

[24] Kawadia, V., Kumar, P.R.: A cautionary perspective on cross-layer design. Wireless Communications, IEEE Volume 12, Issue 1, Feb. 2005.

[25] Zhou, B. et al.: A Cross-Layer Route Discovery Framework for Mobile Ad Hoc Networks. EURASIP Journal on Wireless Communications and Networking, 2005.

[26] Srivastava, V., Motani, M.: Cross-layer design: a survey and the road ahead. Communications Magazine, IEEE Volume 43. 2005.