A Self-Repairing Tree Topology Enabling Content-Based Routing in Mobile Ad Hoc Networks

Luca Mottola, Gianpaolo Cugola, and Gian Pietro Picco

***Abstract*— Content-based routing (CBR) provides a powerful and flexible foundation for distributed applications. Its communication model, based on implicit addressing, fosters decoupling among the communicating components, therefore meeting the needs of many dynamic scenarios, including mobile ad hoc networks (MANETs). Unfortunately, the characteristics of the CBR *model* are only rarely met by available *systems*, which typically assume that application-level routers are organized in a tree-shaped network with a fixed topology.**

In this paper we present COMAN, a protocol to organize the nodes of a MANET in a tree-shaped network able to *i)* self-repair to tolerate the frequent topological reconfigurations typical of MANETs; *ii)* achieve this goal through repair strategies that minimize the changes that may impact the CBR layer exploiting the tree. COMAN is implemented and publicly available. Here we report about its performance in simulated scenarios as well as in real-world experiments. The results confirm that its characteristics enable reliable and efficient CBR on MANETs.

***Index Terms*— Content-based routing, publish-subscribe, query-advertise, mobile ad hoc network.**

I. Introduction

Content-based routing (CBR) differs from classical routing in that messages are addressed based on their content instead of their destination. In conventional systems, the sender explicitly specifies the intended message recipients using a unicast or multicast address. Instead, in CBR the sender simply injects the message in the network, which determines how to route it according to the nodes' interests. These identify the relevant classes of messages based on their content, e.g., using key-value pairs or regular expressions. Therefore, in CBR it is the receiver that determines message delivery, not the sender.

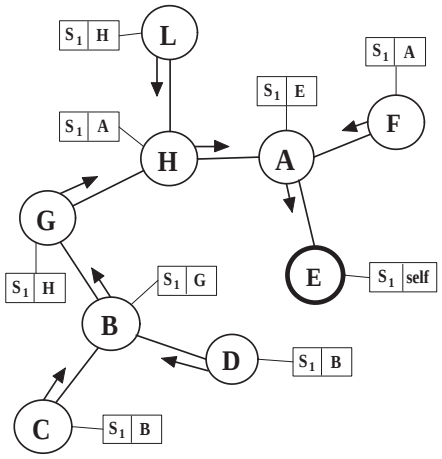
This ability is useful in many application scenarios. For instance, in a stock quote application data producers can *publish* stock updates on a CBR network, which routes them only towards the consumers who *subscribed* to receive such updates. Similarly, in a data sharing application repositories can *advertise* the attributes of the data they hold; a *query* from a user node is then routed by the CBR network only towards the repositories containing attributes involved in the query. CBR is at the core of many systems, including event notification [22], distributed databases [8], file sharing [27], and data collection in wireless sensor networks [30]. Hereafter, we adopt the terminology made popular by content-based publish-subscribe, and refer to interests as *subscriptions*.

Although it enables multi-point communication, CBR is *not* simply multicast. In network-level (e.g., IP) or application-level (e.g., topic-based publish-subscribe [22]) multicast, the address of the multicast groups (or topics) must be defined a priori and made globally known or available. Moreover, a component joined to a given group receives *all* the messages addressed to that group—and only those. If messages are to be received from multiple groups, the component must join all of them. Indeed, the messages are conceptually partitioned in classes, and the binding between a message and its class is established by the sender. Instead, in CBR message consumers define their own message classes. These select *only* the desired messages, need not be known to other components, and can be arbitrarily overlapping.

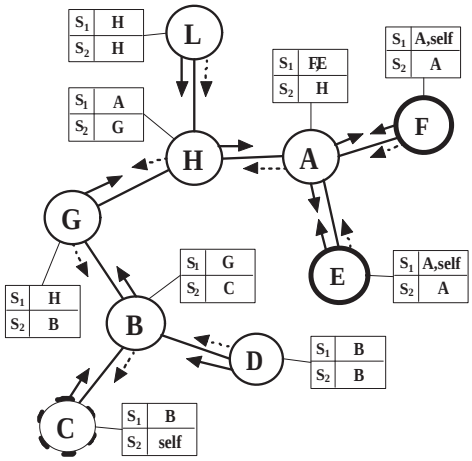
The difference between multicast and CBR is reflected also in the routing mechanisms enabling them. Multicast approaches typically propagate messages along a tree defined on a *per-group* basis, spanning all the receivers for that group. Therefore, a message addressed to multiple groups is typically duplicated at the sender, and each copy is routed independently. In CBR, the absence of an explicit and a priori definition of groups, along with the fact that every message can be addressed to a different set of components, discourage the use of separate per-group trees. Instead, most systems connect all the *brokers* (the CBR application-level routers) in a single tree-shaped network [37]. This *broker tree* is exploited to forward messages. Usually, these are not flooded to the entire tree, but routed towards the interested components according to the message content and the subscriptions stored at tree nodes. Notably, the same arguments justify also why existing multicast services cannot be directly used to implement CBR, as discussed in [38]. Indeed, this would require either a single multicast group, delivering all messages to all members regardless of their subscriptions, or a group per each possible set of recipients, placing the burden of determining the target group for each message on the publishing node. In both cases, the very benefits of CBR would be lost.

As a concrete illustration of CBR and as a reference for the rest of the paper, Figure 1 describes the *subscription forwarding* routing strategy [10], perhaps the most widely used among the CBR systems available so far. Each broker holds a *subscription table* used to decide how to forward messages along the broker tree. Such tables are populated by propagating subscriptions along the broker tree. As an example, Figure 1(a) shows the content of the brokers' subscription tables after subscription S_1, coming from E, propagated to the entire tree. The arrows summarize the content of subscription tables, showing the route followed by messages matching S_1. In Figure 1(b) the same broker tree is shown after node F issued the same subscription S_1 above, while node C subscribed to S_2. Subscription S_1 from F reached A, which updated its subscription table and forwarded S_1 only towards E. Indeed, the subtree including H and its descendants already received the subscription. Therefore, their subscription tables need not to be updated. Instead, subscription S_2 from C, appearing for the first time, propagates to the entire tree,

L. Mottola and G. Cugola are with the Dipartimento di Elettronica ed Informazione, Politecnico di Milano, P.zza L. da Vinci, 32, 20129 Milano, Italy. E-mail: {mottola,cugola}@elet.polimi.it. G.P. Picco is with the Dipartimento di Ingegneria e Scienza dell'Informazione, Università di Trento, Italy. E-mail: gianpietro.picco@unitn.it.

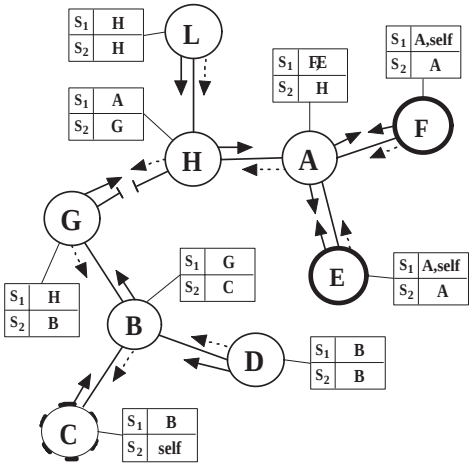


(a) A broker tree in the presence of a single subscriber, node E.

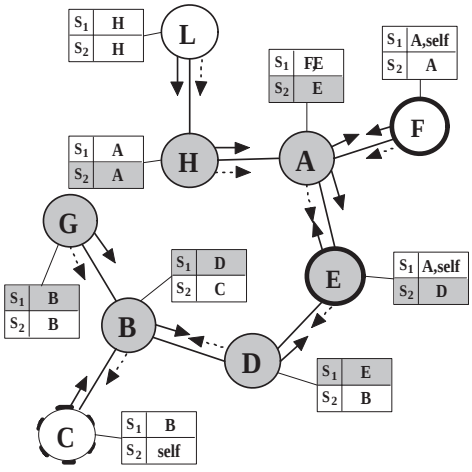


(b) The same broker tree after F and C subscribed.

Fig. 1. Content-based routing using *subscription forwarding*.



(a) A link on the broker tree fails.



(b) The broker tree is repaired.

Fig. 2. The impact of tree reconfiguration on CBR information.

populating the subscription tables of every broker. As shown by the arrows in the figure (solid for S_1 and dashed for S_2) this mechanism defines the minimal routes followed by messages in reaching the subscribers along the broker tree.

Motivation. CBR fosters a form of *implicit* communication that breaks the coupling between senders and receivers. Senders no longer need to determine the address of communication parties. Similarly, receivers do not know who is the sender of a message, unless this information is somehow encoded in the message itself. The sharp decoupling induced by this form of communication enables one to easily add, remove, or change components at runtime with little impact on the overall architecture.

These characteristics of CBR are an asset for developing distributed applications in dynamic environments, notably including mobile ad hoc networks (MANETs). In this scenario, CBR can be used effectively to coordinate mobile users, e.g., in a disaster recovery application [15], [32]. Members of a rescue team can publish damage assessment information; other members can selectively query or subscribe to the information germane to their task. The CBR network matches the published messages against the dynamically changing interests and routes them to the appropriate recipients.

Unfortunately, the advantages provided by this interaction *model* are not supported by the state of the art of implemented *systems*. Indeed, the majority of available CBR systems address scalability and ease of implementation by realizing the broker tree as an overlay network, whose topology is assumed to be stable—a requirement that clashes with the reality of dynamic scenarios like MANETs. Therefore, this situation leaves the potential of CBR largely unexploited precisely in the application scenarios where it would make a huge difference.

Contribution. This paper overcomes the limitation above by achieving the following goal:

> *defining a protocol to organize the nodes of a mobile ad hoc network in a single, self-repairing tree that efficiently supports content-based routing.*

The goal of supporting CBR explains the rationale behind the choice of a tree topology. As already mentioned, most of currently available CBR protocols adopt this topology for interconnecting brokers, but assume that the tree does not change. Therefore, our self-repairing tree *enables* the reuse of mainstream CBR protocols in the dynamic scenario characterizing MANETs, by leveraging off the consistent body of results related to tree-based CBR. At the same time, providing a tree able to self-repair upon changes in the physical topology of a MANET is only our minimal (and obvious) target. Our ultimate goal is to design a protocol whose characteristics *simplify* the operations of the CBR layer operating

on it. To clarify the exact meaning of this claim, Figure 2 shows the same CBR network of Figure 1 before and after a reconfiguration where the link G-H breaks and is replaced by the link D-E. In this case, we observe that the brokers requiring an update of their subscription tables, whose changed entries are highlighted in gray in Figure 2(b), are not only those directly involved (i.e., G, H, D, and E), but also those lying on the "*reconfiguration path*" [18] connecting the broken and new links, i.e., the nodes G, B, D, E, A, H shown in gray in Figure 2(b). A broker tree that keeps the reconfiguration path as short as possible (e.g., by selecting H-B as a replacement link instead of D-E, provided H is also in range of B) makes CBR more efficient, reducing the effort required to rearrange the routes.

In achieving our goal, we were largely inspired by the Multicast Ad Hoc On Demand Distance Vector (MAODV) [41], [42] protocol for multicast over MANETs. Indeed, MAODV organizes the members of each multicast group in a single tree without relying on any underlying multi-hop unicast solution. Moreover, the link repair process of MAODV is localized around one of the two endpoints of the broken link. This limits the impact of reconfiguration to a small portion of the system, and therefore intrinsically resonates with our goal of minimizing the reconfiguration path.

Nevertheless, in this paper we do *not* simply reuse MAODV as is. In particular, we do *not* rely on MAODV for *routing* messages because, as already mentioned, multicast routing protocols are not suited to support CBR [38]. Instead, we borrow from MAODV the logic concerned with maintenance of the tree topology, and adapt, extend, and optimize it in a context MAODV was not designed for—content-based routing. The result, and main contribution of this paper, is COMAN (COntent-based routing for Mobile Ad hoc Networks), a protocol for maintaining a CBR broker tree in a MANET environment. Differently from mainstream CBR approaches, which interconnects broker through a tree overlay, COMAN provides a tree topology directly at the network level. Moreover, COMAN does not rely on any lower-level network protocol and assumes only that local, one-hop unicast and broadcast communication is available—a fundamental and always satisfied assumption in MANETs.

To verify the feasibility of the approach, we implemented COMAN as a stand-alone module, publicly available as open source [2]. The module is currently used by the REDS content-based publish-subscribe middleware [4], [19] we developed, but its simple interface arguably enables seamless integration in other CBR systems. Using our implementation, we evaluated the performance of COMAN in small-scale real-world experiments. In addition, we carried out larger-scale synthetic experiments through simulation. Both kinds of experiments confirmed that COMAN indeed supports CBR over MANETs by efficiently maintaining a tree-shaped network with a short reconfiguration path. As for its relationship with the tree maintenance protocol of MAODV, our evaluation shows that: *i)* the performance of COMAN is clearly superior in a CBR setting; *ii)* as a by-product of our research, COMAN provides better performance even when its tree topology is used for plain multicast routing (as in MAODV) instead of CBR.

Thanks to our choice of focusing on tree-based approaches, and to the independence of COMAN from the CBR protocols relying on it, our results are directly applicable to the vast majority of CBR proposals. In the context of our own research, this paper completes previous work on tree-based CBR by our group addressing the complementary problems of efficiently rearranging the CBR routing tables upon topological changes [18], [40] and recovering lost messages [16].

Road-map. The rest of the paper is organized as follows. Section II summarizes the relevant aspects of MAODV. Section III discusses how we adapted its tree maintenance protocol, yielding the COMAN protocol. Section IV evaluates COMAN through simulation, showing that the tree topology *i)* is maintained efficiently in the face of mobility; *ii)* is repaired in a way that minimizes the impact on the routing layer; *iii)* the same performance cannot be achieved by MAODV's tree maintenance strategy. Section V corroborates these findings by reporting about small-scale real-world experiments leveraging off our implementation of COMAN. Section VI shows that COMAN also improves over MAODV's tree maintenance protocol even when used for plain multicast routing instead of CBR. Section VII places our work in the context of related research efforts. Finally, Section VIII ends the paper with brief concluding remarks.

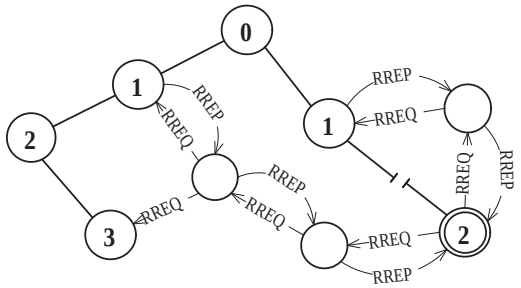
II. Multicast Ad-Hoc On-Demand Distance Vector (MAODV)

Before delving into our main technical contribution, we present a concise summary of MAODV. Our presentation does not cover all the details of the protocol, as our purpose here is simply to provide enough background to understand our contribution. Further details about MAODV can be found in [41], [42].

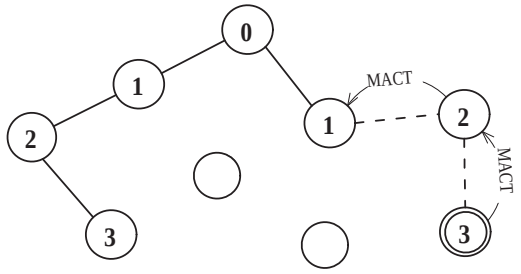
MAODV (Multicast Ad-Hoc On-Demand Distance Vector) is the multicast counterpart of the AODV protocol [39]. All nodes belonging to the same multicast group, along with nodes (called *forwarders*) required to forward messages among group members, are organized in a tree used to propagate messages addressed to that group. Each multicast group results in a different tree. To maintain such trees, MAODV leverages off the same route discovery protocol used in AODV with the addition of special nodes, called *multicast group leaders*, in charge of providing consistency information within each connected partition of a multicast group's tree. Specifically, each leader periodically distributes a monotonically increasing *group sequence number* in its own connected partition. This measures the "freshness" of the multicast group information stored at a node.

The protocol exploits four kinds of messages:

- **Route request (RREQ):** is broadcast by a node willing to join a specific multicast group, repair a branch of the tree, or merge two network partitions. It contains the identifier of the target group and the most recent sequence number known for it. When used to repair the tree, RREQ contains also the last measured hop distance from the leader to the sender.
- **Route reply (RREP):** is unicast towards a node that previously broadcast a RREQ, to inform that its request can be satisfied. RREP contains the identifier of the target group, its most recent sequence number known at the responding node, the identifier of the leader, and the current distance between the RREP sender and the leader. This information, along with the number of hops traveled by the RREP, is used to infer the new distance of the requesting node from the leader.
- **Multicast activation (MACT):** is unicast to explicitly activate a particular route towards the multicast tree. Furthermore, specific flags are used to implement operations such as



(a) After a disconnection, the double circled node issues a route request (containing a distance of 2 from the group leader) and receives two replies.



(b) The disconnected node selects a reply for activation among the ones received.

Fig. 3. An example of MAODV route request and activation. Numbers in circles denote the distance of each node from the group leader.

identifying a new group leader after a failed repair, pruning a node from the tree, and updating the nodes distance from the current leader. Node pruning is required when a forwarder node becomes a leaf. Instead, the nodes distance from the leader must be updated whenever the tree topology changes (e.g., when a broken link is replaced by a new one).

- **Group hello (GRPH):** is periodically broadcast by each group leader and rebroadcast across the whole network. Its main purpose is to disseminate the group sequence number and let each group member verify its *distance* (in hops) from the leader. It is also used to update information at group members in case the group leader has changed, using a proper flag.

We now briefly sketch how MAODV handles link breakages and network partitions.

Link breakages. As shown in Figure 3(a), when a link between two nodes on the tree fails, the node downstream in the tree (i.e., the node with the greater hop count from the leader) acts as the *initiator* of the repair process, started by broadcasting a RREQ message. When a non-tree member receives the RREQ it updates a local table, later used to determine the *reverse path* to be followed by RREPs, and rebroadcasts the RREQ. Tree members operate differently. Upon receiving a RREQ, a tree member checks if: *i)* it is not itself involved in a repair process; *ii)* its routing table stores a sequence number for the target group greater than or equal to the one in the RREQ; *iii)* its distance from the leader is less than or equal to the one in the RREQ. If all these conditions hold, the node replies with a RREP, otherwise it silently drops the RREQ.

Similarly to RREQs, RREP messages set up a *forward path* while traveling toward their destination. This path might be later followed in the opposite direction by a MACT message that effectively activates that route. Indeed, as shown in Figure 3(a), the initiator may receive multiple RREPs for a RREQ, each representing a viable route to the tree. The initiator completes the repair process by explicitly activating one of these routes using a MACT message, as illustrated in Figure 3(b).

In case the initiator, after a given number of retries, does not receive any RREP, it either elects itself as the new group leader or selects one of its descendants to become so. The first choice is adopted by group members, while the latter is taken when the initiator is a pure forwarder. In the latter case, the decision is communicated to the chosen descendant with a special MACT message. When the receiving node is a forwarder as well, the same process is repeated.

Partition merging. The opportunity to merge two partitions is detected when a group leader GL_1 hears a GRPH message addressed to its group but originated by a different group leader GL_2. Indeed, since GRPHs are broadcast by tree members, this situation implies the existence of a path connecting two leaders in different network partitions. Upon receiving such a GRPH, GL_1 sends a RREQ with a specific flag set to the group leader GL_2, by following the reverse path established by the GRPH message. Upon receiving the RREQ, GL_2 elects itself as the new leader of the merged partition and responds with a RREP. As this propagates toward GL_1 the forwarding nodes activate the links required to connect the two partitions, while the members of GL_1's partition update their group leader information to reflect GL_2 as the new leader.

Observe that this procedure works even in presence of more than two partitions. Indeed, it is also used to build the tree at start-up, when every node acts as the leader of a partition containing only the node itself. Accordingly, the procedure covers also the (very rare) cases when the tree needs to be rebuilt from scratch.

III. Tree Maintenance for Content-Based Routing

This section describes how we build upon and extend MAODV's tree topology maintenance mechanisms to address the challenges of our target domain, i.e., CBR on a MANET.

A. Minimization of Route Changes

Problem. When a change in the broker tree occurs, the routing information used to perform CBR must be updated accordingly. In particular, as Figure 2 illustrated, this requires a modification of the routing tables at the nodes along the reconfiguration path. To improve the overall CBR performance, it is crucial to minimize the number of such nodes, as this enables a more efficient route update and therefore faster convergence to a newly stable situation.

Solution. To address this problem, we modified the policy used by MAODV for selecting a RREP for activation. Specifically, we choose the reply resulting in the shortest reconfiguration path. However, this CBR information is not available in MAODV.

To overcome this limitation, we store at each node an *ancestor list* containing the identifiers of all of its tree ancestors, starting from the group leader. For instance, with reference to Figure 4 the ancestor list of node A is $\langle L, H \rangle$. Moreover, we extend RREPs to include a list of node identifiers, initially set to the ancestor list of the responding node. As the RREP travels towards the initiator, the traversed nodes append their identifier to this list. At the end of this process the list contains the new ancestors of the

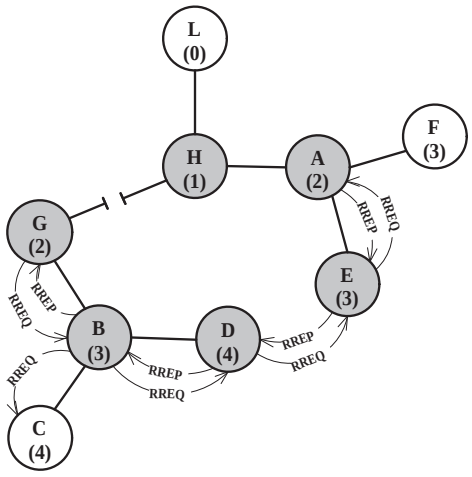


Fig. 4. The example of Figure 2, now highlighting the messages needed to repair the tree. Dark grey nodes are those on the reconfiguration path.

initiator in case that particular RREP were chosen for activation. By comparing this list with the one available before link breakage, the initiator can easily determine the number of nodes potentially affected by the reconfiguration. For instance, as illustrated in Figure 4, when the RREP coming from A reaches the initiator G, it holds the list of nodes $\langle L, H, A, E, D, B\rangle$. By comparing this list with its former ancestor list $\langle L, H\rangle$, the initiator can determine the 6 nodes G, B, D, E, A, and H as those potentially affected by the reconfiguration.

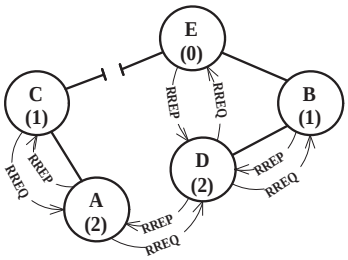
The processing above requires each node to maintain its ancestor list even in the presence of changes in the tree topology. In case of a successful tree repair we use MACT messages, originally used to update the nodes' distance from the leader, to update also their ancestor list. Similarly, to address network partitions we use special GRPH messages that, instead of being broadcast as usual, are propagated along the tree links. When a partition occurs and a new group leader is elected, we use them to update the ancestor list of the nodes in the subtree of the new group leaders. These GRPH messages are sent periodically, and interleaved with the broadcast of normal GRPH messages.

Notice that our technique does not require additional messages, the only (arguably small) increase in traffic overhead is due to an increase in their size.

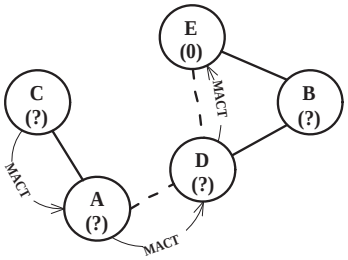
B. Request Propagation

Problem. MAODV is designed to operate in scenarios where each node belongs only to a few multicast groups and each multicast tree spans only the nodes belonging to the same group, plus some forwarder nodes. In CBR scenarios, instead, it is typical (see e.g., [29]) to assume that all the nodes run a broker of the CBR network, which routes messages for the application components running on the same host. A single tree must be built and maintained to span all the nodes.

To understand the impact of this aspect, let us recall that MAODV's RREQs do not propagate among tree members. Consequently, in a situation where every node is a tree member, the only possibility for a RREQ to be answered is to find a replying node within the range of the initiator itself. If this is not the case, the repair process fails (after several retries), the tree becomes partitioned, a new leader is elected, and the tree merging procedure must begin and complete successfully before the two partitions can reconnect. This results in a long delay to repair the broken links, hindering communication for a long time.



(a) RREQ propagation among members of a tree. Node E is currently acting as the group leader.

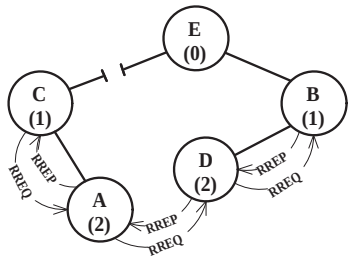


(b) Path activation after a free RREQ propagation, forming a cycle involving D, B and E.

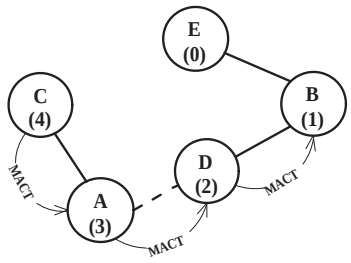
Fig. 5. MAODV repair procedure when RREQ messages are allowed to propagate among tree members. The current distance from the group leader is shown in parentheses at each node.

To overcome this limitation, let us first analyze why MAODV adopts this rule. Consider the situation in Figure 5(a). In this case, node C moves outside the communication range of E. Both nodes experience a link breakage and C, being the downstream node, triggers the repair procedure by broadcasting a new RREQ. Node A receives the RREQ, but it cannot respond because of its greater distance from the current group leader E w.r.t. the initiator C. Now, suppose that A, instead of ignoring the RREQ as required by MAODV, rebroadcasts the RREQ, which now reaches D. Suppose also that D, which is in the same situation as node A, rebroadcasts the RREQ as well. Both nodes E and B are in the communication range of D and able to respond with RREPs, which are received by C along the reverse path set up by the previous RREQs. At this point, if the reply from E were chosen, as in Figure 5(b), the MACT message would activate the link D–E, resulting in a loop involving nodes D, B, and E.

Solution. Looking at the situation in Figure 5, we observe that the issues arising from an uncontrolled propagation of RREQs come from the fact that these messages are allowed to propagate off the tree for more than a single hop. If we force each RREQ to jump off the tree at most once we prevent loops and, at the same time, increase the chances that a disconnected node immediately reconnects. To this end, we extend RREQ with an additional flag called *onTree*, and modify the RREQ forwarding rule accordingly. The flag is initially set to false and remains so while the RREQ travels along the links of the subtree rooted at the initiator. It becomes true only after the first hop off the tree. Upon receiving



(a) RREQs are now allowed to jump off the tree at most once.



(b) After the replacement link is activated, the distances from the group leader are updated.

Fig. 6. RREQ propagation using our modified forwarding rule.

a RREQ with the *onTree* flag set, a node unable to reply stops broadcasting the message. Moreover, if its ancestor list contains the initiator identifier (i.e., the node is part of the initiator's subtree) it silently drops the message. Otherwise, it forwards the RREP towards upstream nodes in unicast. Note how forwarding RREQs downstream is useless because the distance to the current group leader increases in that direction. Therefore, if the receiving node is not able to respond, none of its descendants will.

When applied to the situation of Figure 5, the forwarding rule we just introduced yields the scenario in Figure 6. Once the RREQ reaches D, it has already jumped off the tree and is now in a different subtree w.r.t. the one rooted at the initiator. Therefore, it propagates only upstream: the message is unicast only towards B, and E plays no role in this reconnection. The RREP sent by B is safe, as the path it determines contains a single hop off the tree, and therefore does not create loops if activated. In Section IV, we provide quantitative results about the effectiveness of this solution, which allows for faster tree repairs by giving the disconnected node chances to find a replacement link through one of its descendants. In Section VI we apply the same technique in a traditional multicast scenario involving pure message forwarders, showing its effectiveness even in a situation where the path off the tree may include several hops up to the other tree partition.

C. Reply Propagation and Link Activation

Problem. As it is clear from Figure 6, our modified mechanism for propagating RREQs implies that these messages can travel along the links of the initiator's subtree, jump off once (the hop A-D in Figure 6), and then continue along the links of another branch until they reach a node able to respond, if any. In case such a responding node exists, the propagation of the MACT up to it (e.g., the propagation from D to B in Figure 6) is not strictly needed.

Solution. To avoid the useless propagation of the activation message, the node that sets the *onTree* flag (D in Figure 6) rewrites the header of the RREP message by replacing the identifier of the replying node (B in Figure 6) with its own identifier. With this simple trick, the MACT propagation stops as soon as the selected link is activated, avoiding useless network traffic.

D. Group Leader Election with a Lost MACT

Problem. We realized that, in MAODV, if a MACT message is lost the system may end up with a network partition left without a leader. In fact, as far as the initiator is concerned, a new link has been found and the related MACT has been sent. Thus, the initiator considers the repair process as successfully terminated, and does not try to find a new group leader. However, due to the MACT loss, there is no active path connecting a member of this partition to a leader. This is highly undesirable, as the partition merging procedure requires a working group leader in each subtree. A partition without a leader will never be able to merge. We need to address two issues, namely, how to recognize the absence of a group leader and how to elect a new one.

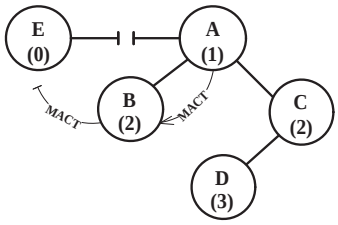
Solution. Our solution to the first issue relies on the special GRPH messages flowing along the tree, described in Section III-A. In the absence of a group leader in a given partition, such messages are never received. Therefore, the absence of a group leader can be easily detected by setting a timeout on their reception.

To address the second problem, we exploit the acyclic topology of the tree by electing as group leader the node whose distance from the former leader is minimal. This node is unique in each disconnected subtree and can be determined through a localized search, requiring each node N to check for the availability of its upstream node. If such node is still reachable, there is another node in the same partition with a smaller distance to the former leader. Otherwise, N is the node with the minimum distance and can therefore safely elect itself as the new leader. An example illustrating the process is shown in Figure 7.

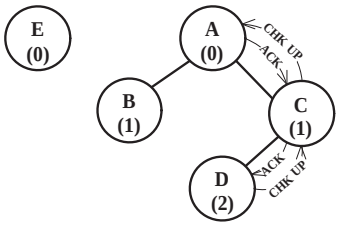
One could argue that checking for availability of the upstream node is superfluous, because the node that is to become the new leader always coincides with the initiator of the most recent repair process. Unfortunately, this does not hold in general: an arbitrary number of disconnections (followed by path activations) involving nodes in the disconnected subtree may take place while the initiator is handling the repair process. Consider Figure 8, where D loses its upstream node C and successfully reconnects to B, while A loses its upstream node E and does not reconnect because the MACT message is lost. If we allow any disconnected node to elect itself as the new leader, then both D and A would become group leaders, leading to an inconsistent state. To address this issue, it is sufficient to note that what differentiates D from A is the fact that D's upstream node is reachable, while A's is not. The check on upstream nodes we described earlier relies on this observation, and guarantees that A safely becomes the new group leader. Note that, as a result, our protocol is able to tolerate an arbitrary number of concurrent repairs.

E. Partition Merging

Problem. As illustrated in Section II, MAODV employs a specific process for merging partitions, conceptually different from the normal repair process. This is no longer required when using our

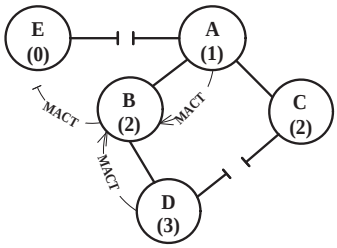


(a) A partition where a MACT is lost. A has the shortest distance from the leader of the disconnected partition containing A, B, C, and D.

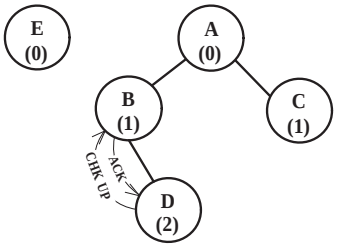


(b) Electing a new leader. CHK UP is the message used for checking if the upstream node is reachable. If so, this replies with an ACK. E eventually rejoins the tree through partition merging.

Fig. 7. Group leader election in case a MACT message is lost.



(a) D detects the absence of the group leader, but it does not try to elect one because its upstream node is reachable.



(b) D is connected to its upstream node, but A is not. A can safely become the new leader.

Fig. 8. A scenario with two concurrent disconnections and a lost MACT.

modified way of propagating RREQ, as partition merging can now be considered a particular case of the standard repair process.

Solution. In COMAN, each GRPH message is effectively treated as if it were a special RREQ. In particular, when a node in a partition P_1 hears a GRPH coming from a different partition P_2, it forwards the message as if it were a RREQ with the *onTree* flag set, i.e., upstream. Due to the condition on the distance from the group leader, the only node able to respond to this special RREQ is the leader in P_1. This replies with a standard RREP that follows the reverse path set up by the GRPH, up to the leader in P_2. The latter reacts by sending a MACT to activate the new link that connects the two partitions.

In essence, this technique changes neither the number of messages needed for merging partitions nor the general behavior of the protocol, which behaves exactly as MAODV to an external observer. Moreover, our modifications do not affect MAODV's ability to join multiple network partitions in subsequent steps. However, the possibility of managing merging as a normal repair process greatly simplifies the implementation.

IV. Evaluation through Simulation

To evaluate the effectiveness of COMAN we measured its performance in both simulated as well as real settings. Therefore, we separate the evaluation in two parts. This section presents the results we obtained in simulated scenarios, with the goal of showing how COMAN is indeed more reliable and better suited to CBR than MAODV's tree maintenance strategy. Instead, Section V reports about our real implementation, demonstrating that our solution can be easily integrated into an existing content-based publish-subscribe middleware, and evaluating the performance obtained in a small-scale, real deployment scenario.

Settings. We implemented our protocol using the NS-2 simulator [3]. Table I summarizes the most significant parameters, along with their default values. While most of them are typical of simulations in MANETs and do not require further discussion, it is worth detailing the strategy we adopted for modeling traffic. Indeed, our goal is to evaluate the ability of COMAN of maintaining the CBR broker tree even in presence of real network traffic beyond that needed for tree maintenance, but *regardless of any specific CBR strategy*. Accordingly, we decided to have each node flooding the entire network with "dummy" packets at a given rate. This traffic generates contention of the wireless medium and, therefore, collisions and message losses that stress COMAN's operations. At the same time, this does not require any specific assumption about the specific CBR strategy adopted.

We run all simulations until a periodic evaluation of the variance of all measures is below 1%, which happens around 980 simulated seconds. It is known [49] that this approach gives more precise results than simply repeating simulations with different seeds. As a mobility model, we employed Random Waypoint [31], as this was the model used in the MAODV papers.

A. Evaluating the Broker Tree

Before evaluating the CBR-specific features of COMAN, it is necessary to assess its ability to keep the tree connected at an

Parameter	Range	Default value
Simulation area (side)	750 m—2,000 m	1,250 m
Network size	50—100 nodes	75 nodes
Node speed	1 m/s—10 m/s	5 m/s
Flooding traffic rate (per node)	no traffic—1 msg/s	0.5 msg/s
Communication range	(fixed)	150 m
MAC layer	(fixed)	IEEE 802.11-2Mb/s
Message Size	(fixed)	256 bytes
Warm-up time	(fixed)	60 s
Mobility model	(fixed)	RandomWaypoint [31] with 0 s pause time

TABLE I
Simulation settings.

acceptable cost. Therefore, we consider the following measures:

- The percentage of *time the tree remains fully connected (TC)*, i.e., with all the nodes connected in a single tree. A link is considered broken when an underlying beaconing mechanism recognizes the absence of a neighbor.
- The average number of *control messages sent per tree repair (MS)*. This includes the RREQ, RREP, MACT, and GRPH messages used to repair a broken link along with the messages not strictly involved in the repair process, e.g., the MACT and GRPH messages needed to update the distance from the leader.
- The average number of *nodes involved in a tree repair (NI)*, i.e., the nodes that sent or forwarded at least one RREQ or RREP message. This measure gives an indirect measure of the amount of processing overhead our protocol imposes on the network.

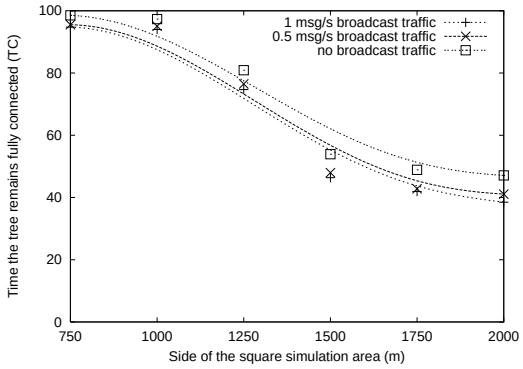
Results. Figure 9 shows the trends w.r.t. the network density and the number of nodes in the system. As the network density decreases, performance obviously degrades. In particular, Figure 9(a) shows that TC rapidly decreases in sparse settings due to the lack of overall connectivity. Similarly, as the network becomes more sparse, more messages are needed for a single tree repair, as shown by the value of MS in Figure 9(b). This is not surprising, as in sparse networks a RREQ usually needs to travel farther before finding a node able to respond, if any. Interestingly, NI shows a small decrease when the side of the simulation area is between 1,250 and 1,500 m. On one hand, more nodes are involved when the network is dense because, being closely located, they are likely to hear the same RREQ message. On the other hand, we already pointed out how messages often travel farther in sparse networks, again involving more nodes. The values between 1,250 m and 1,500 m represent the best trade-off between these two extremes. The network traffic does not seem to affect significantly the performance of our protocol, as the curves for various message rates shown in Figure 9(a), 9(b) and 9(c) are quite close to each other.

Conversely, Figure 9(d) shows how TC varies w.r.t. the number of nodes in the system and their speed. TC initially increases with density until the network becomes so dense that packet collisions start to affect the protocol's ability to carry out the repair processes. The same behavior is exhibited at different node speeds. However, while there is little difference between scenarios with speeds of 1 m/s and 5 m/s, a speed of 10 m/s shows a more marked gap.

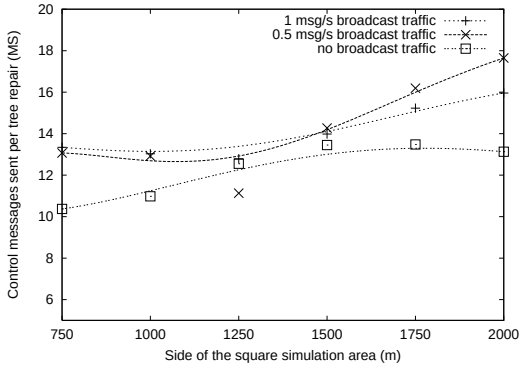
NI and MS are not shown in Figure 9(d), as they turned out to be essentially independent from node speed. This indirectly supports our claims about the limited reconfiguration impact of our solution. Indeed, NI and MS are relative to single repair processes, and speed generally influences only their *number*. Therefore, our measures would vary with speed only in the case of concurrent, overlapping link repair processes. These are more likely to happen when they span larger portions of the system. However, the ability of our protocol to confine reconfigurations to a small portion of the tree makes the probability of concurrent, overlapping link repair processes very low.

B. Evaluating the Benefits to Content-based Routing

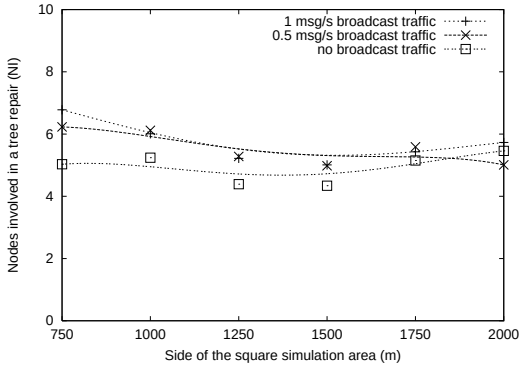
In Section I we pointed out that minimizing the number of nodes on the reconfiguration path is fundamental to achieve



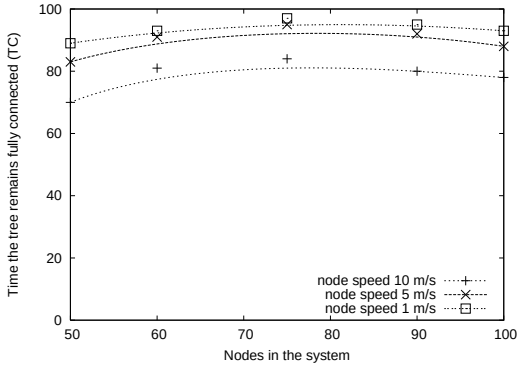
(a) TC vs. side of simulation area.



(b) MS vs. side of simulation area.

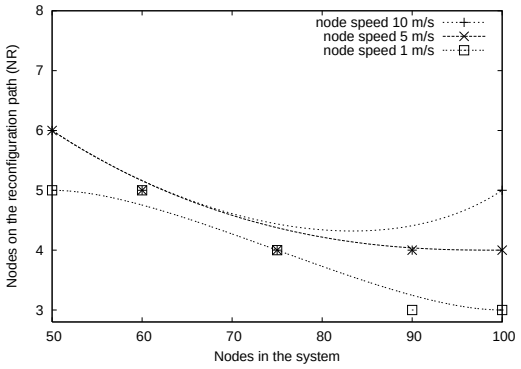


(c) NI vs. side of simulation area.

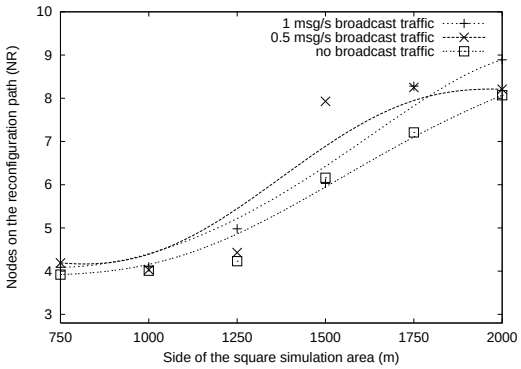


(d) TC vs. nodes in the system and their speed.

Fig. 9. Evaluating the broker tree: average values for time the tree remains connected (TC), messages needed for a successful tree repair (MS), and nodes involved in a repair (NI).



(a) NR vs. number and speed of nodes.



(b) NR vs. simulation area and network traffic.

Fig. 10. Average number of nodes needing routing reconfiguration (NR).

efficient CBR using a tree network. We here show that COMAN addresses this requirement effectively. To this end, we evaluate the average number of *nodes on the reconfiguration path (NR)* per successful repair process. Notice that the minimum number of nodes on the reconfiguration path is three. Indeed, having just two nodes on the reconfiguration path would imply that the initiator reconnected to the same end-point of the broken link. This is clearly a degenerate case that should not happen.

Results. The measures we obtained are reported in Figure 10. These confirm that COMAN is indeed able to limit the number of nodes on the reconfiguration path. This naturally implies that most of the nodes in the system do not change their routing decisions because of a link disruption, and CBR mechanisms can maintain their effectiveness despite topological reconfigurations. Remarkably, NR decreases as the number of nodes in the system increases, as illustrated in Figure 10(a). This can be explained by considering the combined effect of the modifications we introduced in Section III-A and III-B. With more nodes in the system, our changes to RREQ propagation enable more nodes to reply. In addition, our policy for selecting the path to be activated picks the one guaranteeing the minimum reconfiguration path among those available.

Figure 10(a) also shows that, as the node speed increases, NR slightly increases as well. This trend is due to transient effects where nodes in a formerly disconnected subtree do not receive the MACT update regarding their ancestors. This can happen because, as speed increases, a link in this subtree may fail before such a message arrives. In this case, the nodes downstream of the failed link do not have up-to-date information on their ancestors, which is used to compute the length of the reconfiguration paths available. Clearly, this yields sub-optimal selections of RREP messages.

Figure 10(b) shows a trend for NR similar to those observed for NI and MS. As the network is more sparse, the reconfiguration path is longer. Similarly to what observed for NI and MS, network traffic does not seem to influence NR significantly.

C. Evaluating the Improvements over MAODV

To quantitatively evaluate the improvements in a CBR scenario w.r.t. MAODV's tree maintenance, here we describe simulation results comparing the two protocols. We compare them on the same mobility traces, in their ability to keep the CBR tree connected, using the MAODV implementation in [1]. We define $\triangle X = \frac{X_{COMAN} - X_{MAODV}}{X_{MAODV}} \times 100$ as the percentage difference of measure X between our solution and MAODV. In particular, we focus on $\triangle$TC and $\triangle$NR, the latter being computed considering successful reconfigurations only. $\triangle$TC highlights the impact of our mechanisms for request propagation and link activation (Section III-B and III-C) as well as the effectiveness of the leader election procedure (Section III-D). These procedures enable COMAN to find replacement links when MAODV would not. The leader election procedure solves inconsistent states where a network partition is left without a group leader by proactively electing a new one, cutting down the time needed for partition merging. On the other hand, $\triangle$NR assesses the effectiveness of our policy for selecting the replacement link (Section III-A) by highlighting the improvements we obtain in the number of nodes on the reconfiguration path.

Notice that the mechanisms we illustrated in Section III do not have a direct impact on $\triangle$NI and $\triangle$MS, and the simulations confirm this behavior. For this reason, we do not show them here.

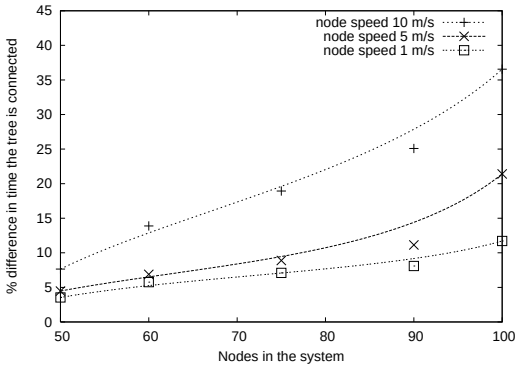
Results. As evidenced by Figure 11(a), for what concerns TC COMAN achieves sensible improvements over MAODV, increasing as the number of nodes in the system grows. The improvement is even more prominent as the number of link breakages increases because of higher speeds. This is not surprising, as our mechanisms affect the performance of single repair processes, and their total impact becomes more and more relevant as the number of link breakages in the system increases.

Furthermore, the number of nodes on the reconfiguration path, NR, is greatly diminished using the mechanism we proposed in Section III-A, as shown in Figure 11(b). The gain w.r.t. MAODV is higher at lower node speed because, with less link breakages, nodes are more likely to have up-to-date information on their ancestors, as discussed earlier. The same trend is exhibited as the number of hosts in the system increases. With more hosts in the system, more replies are received, providing the initiator with more options to minimize NR.

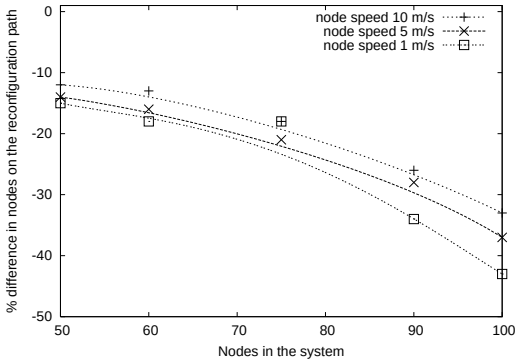
By varying the simulation area, $\triangle$TC and $\triangle$NR exhibit trends similar to the ones in Figure 11(a) and 11(b), with better performance in sparse networks. Moreover, they are basically independent of network traffic.

V. Evaluation through Real-World Experiments

We implemented COMAN in a small (19 Kbyte of jar file) Java software component [2] designed to be fully decoupled w.r.t. the CBR protocol employed. Middleware developers can easily



(a) Percentage difference in the time the tree remains fully connected (△TC).



(b) Percentage difference in the number of nodes required to change routing information (△NR).

Fig. 11. Comparing COMAN against MAODV's tree maintenance.

integrate it into their own CBR systems to support MANET scenarios.

We integrated our component in REDS (REconfigurable Dispatching System) [4], [19], a content-based publish-subscribe middleware we developed that exploits the *deferred unsubscription* technique described in [40] to reconfigure the routing information in face of changes in the broker tree topology. This allowed us to verify the performance of COMAN experimentally. In doing this, we were clearly limited to small scale scenarios. Nevertheless, the experiments we ran provide interesting insights about the trends and values of the critical parameters at stake.

Scenario. In our experiments, eight users participated by carrying a laptop equipped with a 802.11g wireless card configured in ad-hoc mode and without any encryption. The measured communication range was about 35 m. Users moved in a 60×150 m area in a park behind our department. Therefore, the resulting network scenario was rather sparse. Each user was instructed to move according to the Random Waypoint mobility pattern [31], randomly choosing a target point within the test area and walking towards it along a straight line and with constant speed. Upon reaching the target, the user stopped for a random time between 20 s and 60 s, and then repeated the process. We also ran experiments using Random Waypoint extended with "hot spots" as well as the Column mobility model. However, these results are omitted here because they are very similar to those obtained with Random Waypoint.

Traffic was generated by a client installed on each laptop, publishing between 1 and 24 msg/s. These publish rates can be regarded as an overestimate of the real traffic of CBR applications targeted to MANETs [32]. Each message was generated with a 0.1 probability of matching a subscription. Subscriptions were allowed to change dynamically. Notice that, differently from the network traffic we used in simulation, this time we ran actual *content-based* application traffic, with messages delivered only to the nodes that expressed an interest in them. Each test was repeated three times and lasted 11 minutes, with measures starting after the first minute.

Metrics. To provide a measure of the mobility in our scenario, we measured the *disconnection frequency (DF)*, i.e., the average number of link breakages per minute. This information is relevant as the rules our volunteers were given cannot describe precisely the degree of mobility the system experienced. In the setting we described, DF was on average 15.3 link breakages per minute.

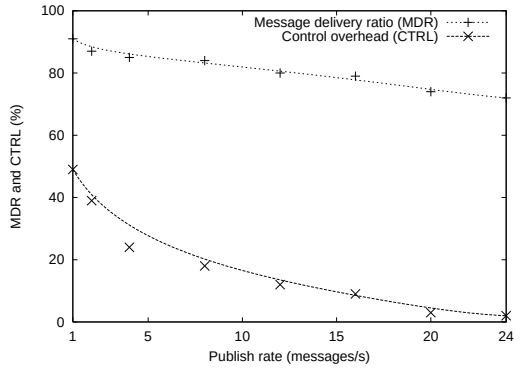
Most importantly, we measured the *message delivery ratio (MDR)*, i.e., the ratio between the published messages actually delivered to a client and the overall number of published messages matching at least one subscription at that client. This figure basically represents, from the application point of view, the effectiveness of the CBR facility as a whole, achieved by combining a given routing strategy with a tree maintenance protocol and a solution to update CBR information when the topology changes. The MDR was measured by relying on a *stable core*, as in [40]. In addition, we also measured the *control overhead (CTRL)*, defined as the ratio between the number of *control messages* used to maintain the tree connected and the number of *published messages* that, matching at least one subscription, actually flow in the network. Basically, CTRL describes the impact of messages for tree maintenance in the presence of real application traffic.

Observe that, differently from the simulation results in Section IV, here we do not consider NR and NI, as the small scale of the system makes these measures pointless.

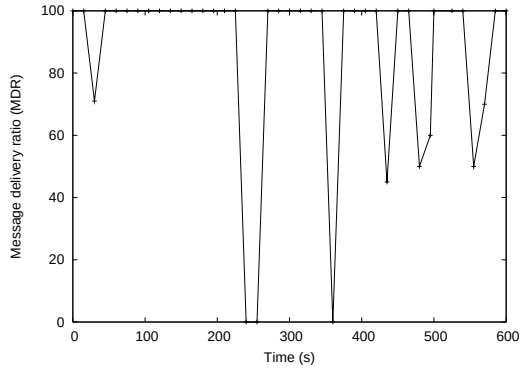
Results. Figure 12(a) shows CTRL against the publish rate. The chart confirms the simulation results: the traffic generated by control messages for tree maintenance is quite independent of the application traffic. Accordingly, CTRL quickly decreases as the publish rate increases.

More importantly, Figure 12(a) also shows the message delivery ratio (MDR) against the publish frequency. Given the high dynamicity of our scenario, where mobility induces frequent network partitions, the values in the chart are good, especially by considering that *no additional measure is taken to recover lost messages*. Indeed, by complementing the base delivery shown with a protocol providing such message recovery (e.g., the one described in [16]), we expect to easily obtain a full delivery of 100%. Again, the figure shows that COMAN is only marginally influenced by the application traffic: an increase from 1 to 24 publish/s brings only a 15% reduction in message delivery, a measure of good scalability.

Figure 12(b) elaborates on this by focusing on the message delivery ratio (MDR) over time. The message delivery ratio drops upon disconnection. By looking at our logs, we determined that MDR reaches the lowest values either when both publisher and subscriber belong to different partitions (e.g., at 250 s and 350 s in Figure 12(b)) or when multiple disconnections and reconfigurations occur concurrently, increasing the time required to restore connectivity. Furthermore, note how the charts show fairly large intervals (e.g., from 50 s to 200 s) where only a few messages



(a) Message delivery ratio (MDR) and control overhead (CTRL) vs. publish rate.



(b) Message delivery ratio (MDR) vs. time for a randomly selected publisher and a core subscriber, publish rate of 8 messages/s.

Fig. 12. Performance of the REDS system running with our topology maintenance protocol.

are lost. Given the aforementioned fairly high disconnection frequency (i.e., DF=15.3 disconnections per minute) observed in our setting, it is likely that several link breakages affected the path between the considered publisher and the subscriber during that time frame. This shows how COMAN is able to repair the tree quickly enough so that only a few messages are lost due to a link breakage.

VI. A Look Back at Multicast

After having verified that COMAN effectively supports CBR in MANETs, an interesting question is whether it performs well also in original MAODV scenario, i.e., multicast communication. In particular, as anticipated in Section III-B, we are interested in investigating whether the technique adopted by COMAN to propagate RREQs is effective even in the presence of pure message forwarders.

Settings. The simulations are carried out using the same settings of Section IV. Instead of using "dummy" traffic, we allow each sender to generate multicast packets addressed to a single multicast group at a rate of 1 msg/s. Each message is 256 byte in size. Sender and receiver nodes are selected randomly among the 75 nodes in the system with the only constraint that a sender cannot be a receiver as well. As we did in Section V, we want to assess the effectiveness of COMAN coupled with a specific routing scheme, which in this case is pure multicast communication. Therefore, we take MDR and CTRL as performance figures. Obviously, these are defined here by considering multicast packets as application traffic.

Results. Figure 13 illustrates some of the results we obtained by varying node speed and network density. In particular, COMAN outperforms MAODV by providing higher delivery ratios with lower control traffic. These results are clearly due to the ability of our solution to keep the tree more connected than MAODV. In turn, this ability comes from the mechanisms we described in Section III-B, III-C, and III-D, whose combined effect is to *i)* find a replacement link where MAODV would end up with a network partition, and *ii)* lower the time to restore connectivity to the tree. Conversely, the policy to select the path to activate discussed in Section III-A (and designed explicitly for CBR) does not affect the protocol's ability to work in a multicast scenario. The mechanism we devised to reduce the number of nodes on the reconfiguration path simply changes the shape of the resulting topology, without affecting the ability of the tree to deliver messages to members of a multicast group.

The trends obtained from these simulations show also increasing gains for COMAN as the network becomes more sparse. As discussed in Section IV-C, this is due to the way RREQ messages are propagated among members of the tree, which enables our protocol to find replacement links where MAODV would not receive any reply. Finally, by comparing the results obtained with nodes moving at 1 m/s, shown in Figure 13(a) and 13(b), with the ones reported in Figure 13(c) and 13(d) gathered with nodes moving at 10 m/s, one can easily appreciate how the improvements obtained are reasonably independent of node speed.

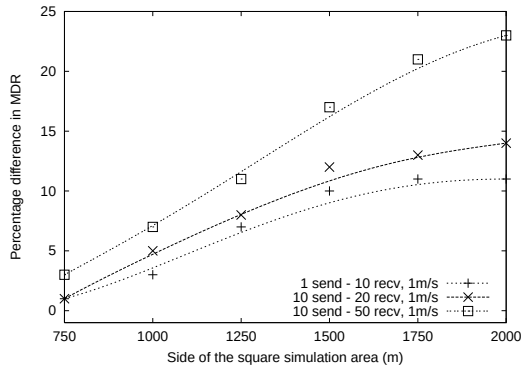
VII. Related Work

In this section we survey related work in CBR, our target domain, as well as in multicast communication for MANETs, the domain that inspired this work.

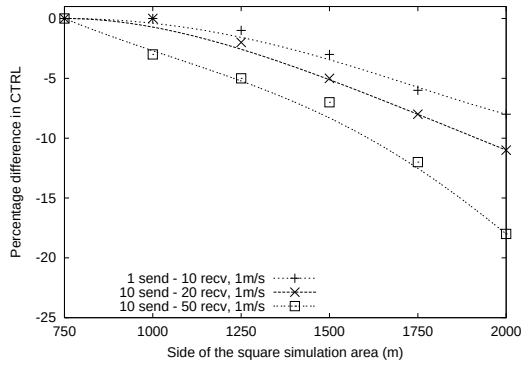
Content-based Routing. CBR in dynamic environments is a challenging issue. The research community has tackled this problem either by trying to adapt solutions designed for existing systems, which mostly use tree-shaped overlay topologies, or by developing dedicated routing mechanisms.

In this work we described and evaluated a protocol to organize the nodes of a MANET in a self-repairing tree enabling the former approach, regardless of the specific CBR scheme in use. In this sense, close to our work is the proposal in [29], which presents a way to implement a content-based publish-subscribe service on a MANET by constructing a hierarchical topology to distribute messages. However, the authors make fairly constraining assumptions, in that they assume knowledge about the placement of publishers (always at the root) and the distribution of messages w.r.t. subscriptions. Also, the evaluation is carried out by simulation in quasi-static scenarios where a node moves only occasionally and then settles down for a period of the order of minutes. For these reasons, their results are not comparable with ours.

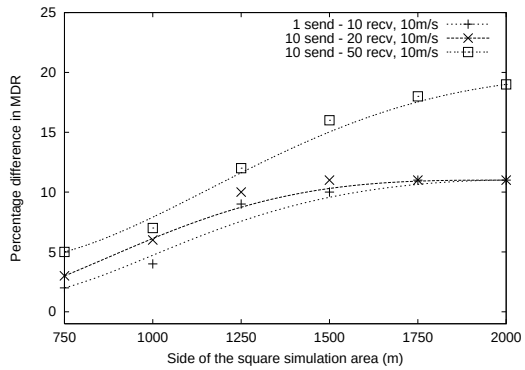
A different form of content-based publish-subscribe is proposed in [13], where the authors describe mechanisms to reconfigure an overlay network according to the changes in the physical topology and to the current brokers' load. Unlike our solution, each broker must be provided with a global view of the system. Moreover, this proposal does not handle partitions and strongly relies on an



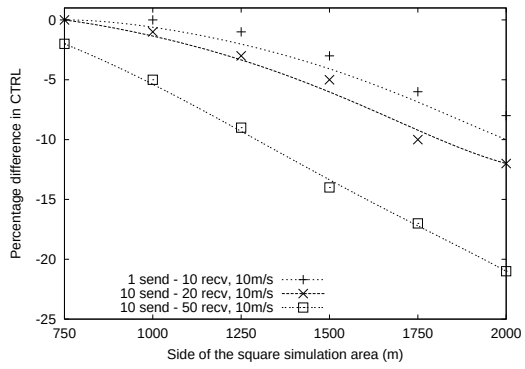
(a) Percentage difference in message delivery ratio (△MDR), node speed of 1 m/s.



(b) Percentage difference in control overhead (△CTRL), node speed of 1 m/s.



(c) Percentage difference in message delivery ratio (△MDR), node speed of 10 m/s.



(d) Percentage difference in control overhead (△CTRL), node speed of 10 m/s.

Fig. 13. COMAN vs. MAODV in multicast scenarios.

underlying unicast protocol. Finally, results are not comparable with ours, as the evaluation concerns a testbed with only 3 nodes.

In [47], [48] the authors propose an extension to the ODMRP [34] multicast protocol to enable CBR on MANETs. The dissemination infrastructure is built using summaries of content-based subscriptions coded as Bloom filters. Again, the results are obtained only in a small-scale simulated scenario of 10 nodes and only with controlled reconfigurations, i.e., insertion or removal of a single link.

The aforementioned works try to achieve efficient CBR by relying on some form of tree. On the contrary, in highly mobile scenarios the overhead to maintain a tree may become unreasonable. Recently, our group addressed this problem in two ways. The work in [17] proposes a semi-probabilistic approach to CBR where subscriptions are forwarded only up to a given number of hops from the subscriber. Where there is no deterministic information, messages are forwarded to a randomly selected subset of the neighbors. Instead, in [5] each node autonomously decides about forwarding messages, based on its estimated distance from the closest node interested in the content of the message. This value is computed by measuring the time since they were most recently able to communicate. In principle, the aforementioned approaches should perform more efficiently than the one described here in highly mobile scenarios. Conversely, the solution described in this work should be more effective in settings with lower mobility, e.g., when node movement can be modeled according to a group mobility pattern [9]. An extensive evaluation of the three approaches is in our immediate research agenda.

A different solution is described in [50], where a form of CBR is proposed to disseminate information coming from sensor-like devices to mobile units within a scope defined by time and space constraints. The scenario and assumptions taken in this case are fairly restrictive. For instance, the authors assume information about the position, speed and direction of mobile units and monitored phenomena. Furthermore, they deal with multi-hop communication by relying on a unicast routing protocol. In this work we considered a much more general scenario and more easily verified assumptions—basically, the availability of local broadcast and unicast.

Additional techniques to address the peculiarities of content-based mobile scenarios are discussed in [28], [36]. The work in [28] relies on replication to overcome the challenges stemming from node mobility. In [36] proximity filters are proposed to define a spatial scope where messages are delivered to the mobile nodes interested in their content. These works do not propose any dedicated routing solution, rather focusing on architectural and design issues. Further investigation is needed towards a possible integration with the solution presented here.

Multicast Communication. The work described here adapted the topology maintenance mechanisms of MAODV to a CBR scenario. The rationale behind the choice of MAODV has already been discussed in Section I. However, here we report about other proposals in the field of MANET multicast that are close to our requirements, i.e., maintaining a flat (i.e., no hierarchies or backbones) acyclic network in the presence of mobility. A comprehensive survey on the subject can be found in [14], [35].

One way of achieving multicast communication in MANETs is to implement it on top of the MAC layer, therefore tackling mobility and link disruptions directly at the network layer. Alternatively, one can rely on some underlying multi-hop unicast mechanism

providing point-to-point communication, and let this deal with mobility and reconfigurations[1]. Notice how the second approach creates a layer of indirection hiding many aspects related to reconfiguration. Instead, we want to retain control of mobility, to tailor the broker tree reconfiguration to our needs. Inevitably, this implies removing any intermediate layer between the topology maintenance mechanism and the network itself.

In AMRIS (Ad-Hoc Multicast Routing protocol utilizing Increasing id-numberS) [45] a bidirectional shared tree is built by exploiting a ranking order among group members. The link repair process is somehow similar to MAODV, with the downstream node trying to reconnect by looking for a new parent node. The modification we illustrated in Section III-B makes this process more general and able to find farther replacement links. CAMP (Core Assisted Mesh Protocol) [23] and ODMRP (On-Demand Multicast Routing Protocol) [34] exploit mesh-like topologies. With respect to the tree-shaped network provided by MAODV, they provide redundant paths at the expense of additional processing for maintaining multiple routes and discarding duplicates. Similarly to MAODV, CAMP and AMRoute require at least one special node for reconnecting lost partitions. ODMRP and MAODV have been extensively compared in [44], showing that the former provides better packet delivery at the expense of higher network traffic, and thus reduced scalability. DCMP [20] is another source-initiated multicast protocol that exploits a mesh topology similar to ODMRP. However, in this case the control overhead is improved by dividing sources into active and passive. Active sources are responsible for creating a shared mesh also on behalf of the passive ones associated to them.

AMRoute (Ad-Hoc Multicast Routing) [7], [46] is a tree-based protocol that exploits unicast tunnels to connect the group members. However, this may result in duplicate traffic when different unicast tunnels exploits the same physical links. The multicast protocol relies completely on the underlying unicast protocol to face mobility. Unicast tunnels are also used in LGT (Location Guided Tree) [11], [12]. However, when a node receives a data packet from its parent, the identities of the children are taken directly from the packet header, instead of being stored locally at the node. PAST-DM (Progressively Adapted Sub-Tree algorithm on Dynamic Mesh) [25] also exploits unicast tunnels. In this case, however, the authors focus on the construction of an optimized virtual topology on top of the unicast facility obtained through a source-based Steiner tree algorithm. Finally, in ALMA (Application Layer Multicast Algorithm) [24] reconfigurations are triggered when application-defined thresholds on link qualities are violated, and overlay maintenance is implemented by having children in the tree looking for new parents. How far the new parent should be searched is decided based on how severe is the link disruption, i.e., how much the measured link quality violated the corresponding threshold.

Finally, let us recall that the problem of building a tree topology goes far beyond network protocols for MANETs. Similar problems have been faced for multicast communication in wired networks [43]. In these scenarios, the Dijkstra algorithm [21] is often used to compute the minimum spanning tree rooted at a source node. However, this assumes global knowledge of the network topology. Furthermore, a large body of research has been devoted to this or similar problems in the field of operational research and graph theory [6]. However, the problem formulation in this case is quite different, as the focus is on determining the optimal spanning tree on top of a general graph according to some predefined metric. Moreover, most algorithms work on a centralized representation of the graph (e.g., [33]), and are therefore not suited to a distributed setting with partial topology knowledge like MANETs.

[1]This approach is also known as overlay multicast [26].

VIII. Conclusions

In this paper we presented and evaluated COMAN, a protocol for maintaining a tree-shaped network interconnecting the brokers of a CBR network in a MANET scenario. COMAN is designed to tolerate the dynamics of the underlying physical network characteristic of MANETs. Moreover, it is also designed to minimize the number of brokers whose routing information are affected by topological changes, therefore improving the efficiency of the CBR network as a whole.

COMAN builds upon the tree maintenance algorithm found in the MAODV multicast protocol for MANETs. We extended this algorithm in a novel way for use in a CBR network, precisely to achieve the aforementioned goals. COMAN was evaluated using simulated, as well as real-world experiments leveraging off our implementation. Results show that the protocol we propose meets the requirements for use in a CBR network, and yields good performance. The latter is significantly better than the original MAODV tree maintenance strategy, therefore showing that our solution does have a strong impact in achieving the desired properties of the broker network.

COMAN is available as open source at [2].

Acknowledgements. We are indebted to Davide Frey and Amy Murphy for their insightful comments and discussions about the topic of this paper. We are also grateful to Antonio Capone, Jon Crowcroft, Renato Lo Cigno, and Mirco Musolesi for their precious comments on an early draft of this paper. The work described here is partially supported by the Italian Ministry of Education, University and Research (MIUR) under the VICOM project, by National Research Council (CNR) under the IS-MANET project, and by the European Union under the IST-004536 RUNES and IST-034963 WASP projects.

References

[1] Carleton University - MAODV extensions for NS-2. `www.sce.carleton.ca/wmc/code.html`.
[2] COMAN Web page. `home.dei.polimi.it/mottola/coman`.
[3] NS-2 Simulator Web Page. `www.isi.edu/nsnam/ns`.
[4] REDS Web page. `zeus.elet.polimi.it/reds`.
[5] R. Baldoni, R. Beraldi, G. Cugola, M. Migliavacca, and L. Querzoni. Structure-less content-based routing in mobile ad hoc networks. In *Proc. of the IEEE Int. Conf. on Pervasive Services*, 2005.
[6] B. Bollobas. *Modern Graph Theory*. Springer, 2002.
[7] E. Bommaiah, M. Liu, A. McAuley, and R. Talpade. AMRoute: Ad-hoc multicast routing protocol. IETF Internet draft, 1998. `www.ietf.org/internet-drafts/draft-talpade-manet-amroute-00.txt`.
[8] A. Bulut, A. K. Singh, and R. Vitenberg. Distributed data streams indexing using content-based routing paradigm. In *Proc. of* 19^{th} *IEEE Int. Parallel and Distributed Processing Symposium (IPDPS)*, 2005.
[9] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing*, 2(5):483–502, 2002.
[10] A. Carzaniga, D.S. Rosenblum, and A.L. Wolf. Design and Evaluation of a Wide-Area Event Notification Service. *ACM Trans. on Computer Systems*, 19(3), August 2001.

[11] K. Chen and K. Nahrstedt. Effective location-guided tree construction algorithms for small group multicast in MANET. In *Proc. of INFOCOM 2002*, volume 3, 2002.

[12] K. Chen and K. Nahrstedt. Effective location-guided overlay multicast in mobile ad hoc networks. *Int. Journal of Wireless and Mobile Computing (IJWMC), Special Issue on Group Communications in Ad Hoc Networks*, 2005.

[13] Y. Chen and K. Schwan. Opportunistic overlays: Efficient content delivery in mobile ad hoc networks. In *Proc. of the* 5^{th} *Int. Middleware Conf.*, 2005.

[14] C. Cordeiro, H. Gossain, and D. Agrawal. Multicast over wireless mobile ad hoc networks: present and future directions. *IEEE Network*, 17(1), 2003.

[15] P. Costa, G. Coulson, R. Gold, M. Lad, C. Mascolo, L. Mottola, G. P. Picco, T. Sivaharan, N. Weerasinghe, and S. Zachariadis. The RUNES middleware for networked embedded systems and its application in a disaster management scenario. In *Proc. of the* 5^{th} *Int. Conf. on Pervasive Communications (PerCom)*, 2007.

[16] P. Costa, M. Migliavacca, G. P. Picco, and G. Cugola. Epidemic algorithms for reliable content-based publish-subscribe: An evaluation. In *Proc. of the* 24^{th} *Int. Conf. on Distributed Computing Systems (ICDCS)*, 2004.

[17] P. Costa and G. P. Picco. Semi-probabilistic content-based publish-subscribe. In *Proc. of the* 25^{th} *Int. Conf. on Distributed Computing Systems (ICDCS)*, 2005.

[18] G. Cugola, D. Frey, A. Murphy, and G. P. Picco. Minimizing the reconfiguration overhead in content-based publish-subscribe. In *Proc. of the* 19^{th} *ACM Symp. on Applied Computing (SAC)*, 2004.

[19] G. Cugola and G.P. Picco. REDS: A reconfigurable dispatching system. In *Proc. of the* 6^{th} *Int. Workshop on Software Engineering and Middleware (SEM)*, 2006.

[20] S.K. Das, B.S. Manoj, and C. Siva Ram Murthy. A dynamic core based multicast routing protocol for ad hoc wireless networks. In *Proc. of the* 3^{rd} *ACM Int. Symp. on Mobile Ad-hoc Networking & Computing*, 2002.

[21] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1, 1959.

[22] P. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 2(35), June 2003.

[23] J.J. Garcia-Luna-Aceves and E.L. Madruga. The core assisted mesh protocol. *IEEE Journal on Selected Areas in Communications*, 17(8), 1999.

[24] M. Ge, S.V. Krishnamurthy, and M. Faloutsos. Application versus network layer multicasting in ad hoc networks: The ALMA routing protocol. *Ad Hoc Networks Journal*, 4, 2006.

[25] C. Gui and P. Mohapatra. Efficient overlay multicast for mobile ad hoc networks. *Wireless Communications and Networking*, 2, 2003.

[26] C. Gui and P. Mohapatra. Scalable multicasting in mobile ad-hoc networks. In *Proc. of INFOCOM 2004*, 2004.

[27] D. Heimbigner. Adapting publish/subscribe middleware to achieve Gnutella-like functionality. In *Proc. of the* 8^{th} *ACM Symposium on Applied Computing*, pages 176–181, 2001.

[28] Y. Huang and H. Garcia-Molina. Publish/subscribe in a mobile enviroment. In *Proc. of the* 2^{nd} *ACM Int. Workshop on Data engineering for Wireless and Mobile access (MOBIDE)*, 2001.

[29] Y. Huang and H. Garcia-Molina. Publish/subscribe tree construction in wireless ad-hoc networks. In *Proc. of the* 4^{th} *Int. Conf. on Mobile Data Management (MDM)*, 2003.

[30] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. Networking*, 11(1), 2003.

[31] D. B. Johnson, D. A. Maltz, and J. Borch. *DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks*. Addison-Wesley, 2001.

[32] S. Kalasapur, K. Senthivel, and M. Kumar. Service oriented pervasive computing for emergency response systems. In *Proc. of the* 4^{th} *IEEE Workshop on Ubiquitous and Pervasive Health Care (UBICARE)*, 2006.

[33] J.B. Kruskal. On the shortest spanning subtree and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7, 1956.

[34] S. J. Lee, W. Su, and M. Gerla. On-demand multicast routing protocol in multihop wireless mobile networks. *ACM/Baltzer Mobile Networking and Applications*, 7(6), 2002.

[35] S. J. Lee, W. Su, J. Hsu, M. Gerla, and R. Bagrodia. A performance comparison study of ad hoc wireless multicast protocols. In *Proc. of INFOCOM 2000*, volume 2, 2000.

[36] R. Meier and V. Cahill. STEAM: Event-Based Middleware for Wireless Ad Hoc Network. In *Proc. of the* 22^{nd} *Int. Conf. on Distributed Computing Systems (ICDCS)*, 2002.

[37] G. Muhl, L. Fiege, and P. Pietzuch. *Distributed Event-Based Systems*. Springer, 2006.

[38] L. Opyrchal et al. Exploiting IP multicast in content-based publish-subscribe systems. In *Proc. of the* 2^{nd} *Int. Middleware Conf.*, 2000.

[39] C. Perkins and E. Royer. Ad hoc on-demand distance vector routing. In *Proc. of* 2^{nd} *IEEE Workshop on Mobile Computing Systems and Applications*, 1999.

[40] G. P. Picco, G. Cugola, and A. Murphy. Efficient content-based event dispatching in the presence of topological reconfigurations. In *Proc. of the* 23^{rd} *Int. Conf. on Distributed Computing Systems (ICDCS)*, 2003.

[41] E. Royer and C. Perkins. Multicast Ad hoc On- Demand Distance Vector (MAODV) Routing. IETF, Internet Draft. Available at `www.cs.ucsb.edu/~ebelding/txt/maodvid.ps`, 2000.

[42] E. M. Royer and C. E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *Proc. of ACM MobiCom*, 1999.

[43] A.S. Tanenbaum. *Computer Networks*. Prentice Hall, 1996.

[44] K. Viswanath, K. Obraczka, and G. Tsudik. Exploring Mesh and Tree-Based Multicast Routing Protocols for MANETs. *IEEE Trans. on Mobile Computing*, 5(1), 2006.

[45] C.W. Wu, Y.C. Tay, and C.-K. Toh. Ad hoc Multicast Routing protocol utilizing Increasing id-NumberS (AMRIS) Functional Specification". IETF, Internet-draft. Available at `www.ietf.org/internet-drafts/draft-talpade-manet-amris-spec-00.txt`, November 1998.

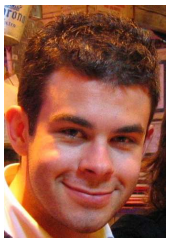
[46] J. Xie, R.R. Talpade, A. McAuley, and M. Liu. AMRoute: ad hoc multicast routing protocol. *ACM/Baltzer Mobile Networking and Applications*, 7(6), 2002.

[47] E. Yoneki and J. Bacon. An adaptive approach to content-based subscription in mobile ad hoc networks. In *Proc. of the* 1^{st} *Int. Workshop on Mobile Peer-to-Peer Computing (MP2P)*, 2004.

[48] E. Yoneki and J. Bacon. Content-based routing with on-demand multicast. In *Proc. of the* 3^{rd} *Int. Workshop on Wireless Ad Hoc Networking (WWAN)*, 2004.

[49] J. Yoon, M. Liu, and B. Noble. Sound mobility models. In *Proc. of ACM MobiCom*, 2003.

[50] H. Zhou and S. Singh. Content-based multicast (CBM) in ad hoc networks. In *Proc. of the* 1^{st} *ACM Int. Symp. on Mobile Ad-hoc Networking & Computing*, 2000.



Luca Mottola Luca Mottola is a Ph.D student at Politecnico di Milano (Italy). He received the Dr.Eng. degree in Computer Engineering from Politecnico di Milano (Italy) in 2004, and the M.Sc. in Computer Science from the University of Illinois at Chicago (USA) in 2005. His research interests include programming abstractions, distributed computing, and routing for wireless sensor networks, and formal verification of distributed software architectures. More information at `http://home.dei.polimi.it/mottola/`.

Gianpaolo Cugola Gianpaolo Cugola received his Dr.Eng. degree in Electronic Engineering from Politecnico di Milano. In 1998 he received the Prize for Engineering and Technology from the Dimitri N. Chorafas Foundation for his Ph.D. thesis on Software Development Environments. He is currently Associate Professor at Politecnico di Milano where he teaches several courses in the area of Computer Science. He has been involved in several projects financed by the EU commission (IST-034963 WASP, IST-511556 POMPEI, IST-11400 MOTION, ESPRIT-34840 PIE, ESSI-21244 MIDAS), and by the Italian governor. He is co-author of tens of scientific papers published in international journals and conference proceedings. His research interests are in the area of Software Engineering and Distributed Systems. In particular, his current research focuses on middleware technology for largely distributed and highly reconfigurable distributed applications with a special attention to the issue of Content Based Routing as the basic mechanism to develop advanced middleware services like publish/subscribe and data sharing.

Gian Pietro Picco Gian Pietro Picco is an Associate Professor in the Dipartimento di Ingegneria e Scienza dell'Informazione (DISI) at University of Trento, Italy. Previously, he has been on the faculty of Washington University in St. Louis, MO, USA (1998-1999) and Politecnico di Milano, Italy (1999-2006). The goal of his current research is to ease the development of modern distributed systems through the design and implementation of appropriate programming abstractions and of communication protocols efficiently supporting them. His work spans the research fields of software engineering, middleware, and networking, and is oriented in particular towards wireless sensor networks, mobile computing, and large-scale distributed systems. More information at `http://disi.unitn.it/~picco`.