SCOMET: A MIDDLEWARE FOR COLLABORATIVE WORKING ENVIRONMENTS IN MANETS

Marc Sánchez-Artigas, Marcel Arrufat, Gerard París, Pedro García López
Departament d'Enginyeria Informàtica i Matemàtiques
Universitat Rovira i Virgili
Avinguda Països Catalans, 26
43007 Tarragona, Spain
{marc.sanchez, marcelgenis.arrufat, gerard.paris, pedro.garcia}@urv.cat

Antonio F. Gómez Skarmeta
Depatarmento de Ingeniería de la Información
y las Comunicaciones
Universidad de Murcia
30001 Murcia, Spain
skarmeta@dif.um.es

Abstract

Nowadays, we are witnessing a growing interest for collaborative working environments (CWEs). The proliferation of portable devices along with the widespread availability of wireless network connectivity is converting mobile ad-hoc networks (MANETs) into a basic scenario where impromptu collaborations take place. Recently, several collaborative middlewares have recently emerged for developing new collaboration tools. However, it is rather difficult to find a middleware for MANETs that enables building reliable applications from scratch, rapidly and easily, while explicitly taking into account the coordination issue, necessary to provide a more modular and standardized way of development.

To cope with this, we present SCOMET, a middleware that permits the provisioning of collaborative services over MANETs. Its major aim is to provide an efficient coordination media layer to allow the coordinable entities to perform complex tasks in a completely decentralized fashion. In order to do this, SCOMET provides efficient group communication and membership services, which are known to be critical for collaborative environments. To address one-to-many communication, SCOMET proposes a new Application Layer Multicast protocol (ALM) called OMCAST. Designed to deal with the specific requirements of CWEs, OMCAST benefits from the broadcast nature of the wireless medium to minimize communication delays and congestion. Evaluation results show that OMCAST can achieve significant improvements.

Keywords: MANET, collaborative middleware, coordination, P2P computing.

1 INTRODUCTION

In the last years, the reduction in price of portable devices has led to an increasing utilization of mobile and portable devices, which in turn has brought chances for new forms of mobile collaboration involving interaction between people who are co-located and organized in an unforeseeable ad-hoc way. However, the provision of such new ways of collaboration over *Mobile Ad-hoc Networks* (MANETs) is still an open issue. Yet, when a group of people wants to work together, it has to agree on a tool set (including general tools such as e-mail and instant messaging) which is rarely prepared for spontaneous collaboration. In MANETs, where participants can appear and disappear in an unpredictable manner and frequently change their access point to collaborative services, standard collaboration tools such as Microsoft Sharepoint Service/Portal [22], BSWC [1], Groove [2] are not suitable for MANETs: they have been designed to run on wired networks, where participants are more stationary and without limitations of battery life, CPU, memory and storage. Conversely, in MANETs, network partitioning, disconnections and the scarce bandwidth make collaboration inherently transient; it continuously varies and it is not subjected to pre-established patterns.

The consequence of this is the lack of an optimal solution that fulfils all the MANET requirements (see [16] for further details). In other words, what is still lacking is a middleware that while utilizing efficiently the scarce resources of portable devices (i.e., memory, bandwidth

and computational power), promotes such spontaneous collaborations. Under these conditions, we believe that the following requirements are critical to the successful implementation and integration of collaborative services over MANETs (note that requirements vary depending on the scenario the middleware is aimed at):

- **Descentralization.** No component is responsible for coordinating other components. Though centralization leads to simple solutions, critical components can restrict the autonomy of the members in a MANET.
- **Self-organization.** The continuous variations of network topology impede any external support to reorganize components in the face of network dynamism (connection, disconnection and mobility) and failures. Therefore, it is very important that a middleware can reorganize its components spontaneously in front of such events.
- **Scalability.** It is vital that the number of members joining the network can grow without significant performance degradation.
- **Group-based.** Groups are the basic unit of organization. So, group membership is clearly necessary: notification of the joining and leaving members is fundamental to guarantee unnecessary service interruptions.
- **Transparent access.** Members should be able to connect from any portable device, with a connection independent view.
- **Efficient communication.** Participants should be capable of forwarding messages to other member efficiently. Since connectivity is only available as far as devices are in range, devices not in wireless range should be able to communicate through intermediate devices, making them feel that they are constantly in range. As a side effect, this causes mobile devices to consume more resources that it was initially expected. Note that they act as message endpoints and forwarders simultaneously.
- **Group availability.** Any group should continue to operate if some components malfunction or become unavailable.
- **Easy deployment and usability.** Beyond the specific requirements of collaboration, a middleware should be easy to deploy, extend and use. Usability and ease of application development transform a middleware into a powerful tool for developing high-level applications. Existing middleware offer some functionalities that can be used to build collaborative applications, but fail to provision the necessary gadgets to develop them in a fast and simple way.
- **Security.** MANETs are vulnerable to attacks due to their features of open medium, changing topology, cooperative algorithms, and lack of a central management point. Without countermeasures, it is possible to track every movement of an individual as well as examine what it is doing. At least, each group should protect the identity of its members and provide a limited access to the content the group decides to share.

Coordination. In addition to the above requirements, handling the specific interactions caused by mobility requires to explicitly handle the coordination interactions of the participating devices. This includes accessing the local resources of the environment and communicating *asynchronously* with the devices, whether belonging to the same group or an external one. For this purpose, it is necessary that the middleware architecture follows a coordination model that allows us to exploit the patterns that appear frequently when coordination among mobile devices is taking place. Some typical examples are the *publish/subscribe*, *blackboard* and *broker* patterns [7].

To support coordination explicitly, our middleware has been designed taking into account the coordination model of Ciancarini et. al. [10]. Their model consists of three components described by the triple $\{E, M, L\}$, where $\{E\}$ represents the *coordinable entities* which ask for coordination, $\{M\}$ refers to the *coordination media* which has to provide appropriate means to address the diverse communication facets needed to coordinate $\{E\}$, and $\{L\}$ stands for the *coordination laws* defining how the interdependencies have to be resolved.

At the present time, SCOMET, the middleware for MANETs we present in this paper, strives to address the communication facets of $\{M\}$. The coordination laws and patterns has been left for future work.

Contributions. In this paper, our major contribution is a middleware solution that permits building complex collaborative applications while meeting all the preceding requirements (excepting security). As just adduced, SCOMET's focus is on communication functionalities as constitute the substrate, i.e., the coordination media $\{M\}$, to which sustain collaborative applications. Note that communication among the members of a MANET requires routing over multi-hop wireless paths. Without a fixed infrastructure, a routing path consists of wireless links whose endpoints are likely to be moving independently of one another. This provokes the frequent failure and activation of links, increasing network congestion, while the system is still reacting to topology changes. It is our responsibility to provide a communication channel that mitigates the instability of those paths. Thusly, our middleware not only supports coordination when the devices are in range, it allows to coordinate entities that are not in direct contact.

Moreover, SCOMET has also to guarantee reliable group communication. The reason for this is to support dynamic growth (scalability requirement) by organizing devices into groups with at least one broker in it, which indexes the information of all services provided by the members in the group, and requests on behalf of them, specific services to the other brokers in the system.

In the past years, several alternatives for group communication over MANETs have been examined, such as *broadcast* [33], *multicast* [16], and even *geocast* [35]. From these candidates, multicast appears to be the best solution, as it makes no assumption about the location of the MANET participants. There are two existing approaches for multicast over MANETs. On the one hand, there are the multicast protocols that operate at the network layer (like MAODV [30] or ODMRP [31]). On the other hand, we can find Application Layer Multicast (ALM) protocols, in which multicast packets are encapsulated in unicast datagrams and delivered to all group members. In ALM, only group members need to keep state information. Futhermore, ALM is easy to deploy and capable of hiding underlying link errors. According to these arguments, we believe that ALM can be more easily adapted to meet the aforementioned collaborative requirements.

As a second contribution, we present a novel Application Layer Multicast protocol called OMCAST. Originally developed in the scope of the EU-funded project POPEYE, OMCAST has been machinated to handle impromptu peer-to-peer collaboration over MANETs. Its main features are the following:

- *A flexible bootstrapping process*, which lets nodes join and leave at any time.
- *Physical broadcasting* of data packets to one-hop neighbors in order to reduce communication overhead.
- *Decentralized membership* information available for higher level layers.

In fact, the OMCAST's performance benefits from the following observation. In many MANET scenarios (e.g., warfront activities, search and rescue, disaster relief operations, etc.), the mobile devices are often organized in groups with different tasks, goals and, correspondingly, different functional and operational characteristics. In particular, the nodes in the same group will have a coordinated motion. For example, attendees of a major conference can be subdivided into teams based on their interests for the purpose of organizing birds of a feather session; various units in a division can be organized into companies and then further partitioned into task forces based on their assignments in the battlefield.

As mentioned above, one of the main challenges of MANET algorithm design is the fact that, unlike in Internet, nodes are moving continuously. In particular, it is difficult to keep track of individual node movements and to route packets to them when the network grows large. However, when devices are organized into groups, the mobility management problem considerably simplifies and allow us to design a a multicast protocol that scales. In fact, it suffices for a source to know the path to one of nodes in the group in order to route a multicast packet to any other destination within that group.

The remainder of the paper is structured as follows: In Section 2, we survey related work. We introduce our middleware solution in Section 3. Section 4 discusses OMCAST in greater detail. In section 5, we show our evaluation results. Section 6 concludes this work.

2 RELATED WORK

In order to be consistent with the introduction, we survey related work in coordination middlewares, collaborative middlewares and multicast over MANETs.

2.1 Coordination

Apart from the model of Ciancarini et. al. [10], the first example of coordination was Linda [14]. In Linda, coordination is carried out by a centralized coordination mechanism while the application that exploits it may be distributed. The participants share information trough a global (persistent) data store, typically implemented as a tuple space. In modern implementations, such as JavaSpaces [25] and TSpaces [34], subparts of the application can coordinate with each other by means of a tuple space maintained at a central location. The main shortcoming of these systems compared to SCOMET is their centralized design, which leads to a potential loss of scalability. SCOMET, however, is a decentralized middleware following the peer-to-peer paradigm, which results in a scalable coordination media for MANETs.

More recently, coordination models have been translated into the MANET setting. LIME [28] proposed the idea of multiple (local) tuple spaces that were transiently shared to form a federated shared dataspace when hosts are in range. This programming abstraction allows tuple sharing between two nodes that are reachable in the device coverage area. In consequence, the users perceive mobility as a sudden change of context. This occurs because the virtual (global) data structure forms a transient shared space, where the accessible tuple set depends on the available connectivity with the other mobile units in the MANET. In our approach, we also adopt the idea of local spaces, but allowing all the participants to define a global space, irrespective of their current connectivity. To do this, SCOMET supports *multi-hop routing* to reliably share data and coordinate mobile devices. In other words, we provide disconnected routing, which allows two devices that are not in direct contact to coordinate with each other.

Other systems based on LIME are LIMONE [12], an enhancement of LIME to tackle highly mobile environ-

ments, and PeerWare [11], a peer-to-peer middleware that provides a global virtual data structure to share documents between peers. Nevertheless, since these architectures do not provide explicit communication mechanisms, we believe that their tuple space-like abstractions are by themselves not enough to meet our goals.

CAST [29] is another coordination model tailored for MANETs that provides Linda-like coordination, but now taking into account as well hosts motions in space and time. Therefore, CAST supports disconnected routing, too. For this purpose, it supposes that devices move according to some locally controlled plan called a motion profile, i.e., for a finite duration of time each device knows where it is heading and will not change it course in short. However, motion profiles are unpractical when the motion of hosts is completely random. Conversely, SCOMET does not require to know a priori the devices motions to work, being practical even when the motion of hosts is completely random.

2.2 Collaborative Middleware

In this section, we review the collaborative middlewares for MANETs which share common features with our proposal. As outlined by [16], it is difficult to establish a general taxonomy of the available middleware solutions. However, we can establish a classification based on their programming model. As such, we classify them as Peer-to-Peer-Based Middlewares (such as JMobiPeer and Peer2Me), Event-Based Middlewares or Message-Oriented Middlewares (such as STEAM and AGAPE).

In general, Event-Based Middlewares are employed to build distributed applications that must react quickly to changes in the environment. Because they do not assume a fixed topology, they are suitable for MANETs. In a similar way, Message-Oriented Middlewares provide asynchronous communication abstractions such as publish/subscribe which are very adequate for pervasive environments. Among this category, we must highlight STEAM [21], an Event-Based Middleware which drops the need of dedicated event servers by exploiting a publish/subscribe service. Consumers subscribe to certain event types and publishers are able to publish particular events. Moreover, STEAM allows different filters to be applied to the published events. The main shortcoming of STEAM is that is limited to the hosts that are in the same radio range. There is no intermediate middleware; instead a publisher will send notifications directly to its subscribers. Contrariwise, as mentioned in the previous subsection, SCOMET supports multi-hop communication, dropping the need of direct host contact to make the publish/subscribe pattern operative.

In EMMA [26], a well-known standard from traditional distributed systems has been adapted to cope with MANET requirements. To be pecific, EMMA is an implementation of the Java Message Service (JMS) that incorporates an epidemic routing mechanism to facilitate message delivery. This middleware provides point-to-point communication as well as a publish-subscribe service. However, it must be taken into account that the epidemic routing protocol does not guarantee the reliability in message delivery like occurs in SCOMET.

Another interesting Message-Oriented Middleware is AGAPE [8]. This middleware provides group membership and message-oriented communication for pervasive environments. It offers context information of colocated group members, such as their attributes and features. AGAPE organizes members in proximity-based clusters, considering two different roles that depend on the specific features of each device: The cluster heads and the managed entities. So, low-resource devices such as mobile phones or PDAs act as managed entities that rely on more powerful devices such as laptops that act as cluster heads. Whereas this static role differentiation could be useful for team operations, like emergency scenarios, we believe that is not suited for scenarios where collaboration between members is highly decentralized.

The next two reviewed solutions belong to the category of Peer-to-Peer-Based Middlewares. Such middlewares use a P2P communication model that involves resource and information sharing in order to a perform common task. P2P systems share many similarities with MANET environments [36]: *decentralization*, *dynamicity*, and *self-adjusting behavior*. Hence, an association of both systems is believed to benefit the global operation of a collaborative application. However, existing P2P systems has been conceived for wired and fixed infrastructures, so adaptations are needed to utilize a P2P architecture for MANETs.

Among the attempts to adapt P2P ideas to MANETs, we highlight two middleware approaches: JMobiPeer and Peer2ME.

JMobiPeer [5] is a JXTA compatible framework designed for J2ME CLDC environments. JXTA, the most mature P2P framework, provides interoperability and it is platform-independent, allowing connections between heterogeneous devices. JMobiPeer benefits from these characteristics and introduces new features like a routing layer, emulation of multicast functionality to adapt JXTA to mobile environments, and the concept of code mobility. However, JXTA may introduce high communication overhead because its architecture does not take into account locality of nodes and relies on the exchange of XML messages.

Peer2Me [32] is another application framework for mobile peer-to-peer applications. Its main feature is that it offers node discovery and messaging services to facilitate the development of collaborative applications. Additionally, it includes several collaborative applications with the framework. It must be noticed that Peer2Me is designed to be deployed on mobile phones under minimal J2ME configuration using Bluetooth devices.

Compared to SCOMET, the greatest shortcoming of the last two frameworks is that they only consider hand-

held devices and therefore, they are not suitable for laptops or notebooks, in which more complex applications could be deployed.

2.3 Application Layer Multicast Protocols

Finally, we review Application Layer Multicast approaches and analyze its main features. Since OMCAST is inspired by PAST-DM, we explain it in greater detail.

AMRoute [19] was the first Application Layer Multicast algorithm exclusively for MANETs. It creates a shared tree for data distribution using only group members as nodes. Built from a virtual mesh with the help of unicast tunnels, the tree tolerates connections between the group members. One of the main drawbacks of AMRoute is the static behavior of the virtual mesh. Since no changes are made in the structure once it has been built, it does not handle network dynamics and leaves all responsibility for the underlying unicast routing protocol.

ALMA [13] creates a tree of logical links between group members. Its major goal 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 to become part of the tree. When the tree performance drops below a defined threshold, the node reconfigures the tree by either switching the parent or freeing children. This mechanism leads to a complex loop avoidance and detection subsystems, as 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.

AOMP [18] relies on reactive routing to construct a delivery tree in a dynamic and decentralized way. This protocol consists of two stages: A first one that connects the joining nodes to the overlay and a second stage that performs the tree construction and maintenance. AOMP takes advantage of the unicast routing protocol, and thus avoids routing overhead and improves scalability. However, such a protocol is limited to use reactive protocols like AODV [27] or DSR [17], and only considers a single source node for the multicast session.

NICE-MAN [6] presents several improvements with respect to the existing Internet NICE algorithm. To be more specific, it exploits the broadcast capability of the medium to reduce network traffic. The basic advantage is the maintenance of a reduced overlay formed only by cluster leaders while the rest of members are located 1-hop away from them. Thus, a node may send a message to various nodes simultaneously by benefiting from the broadcast nature of the medium. Nonetheless, there are several drawbacks like the continuous selection of cluster leaders. Furthermore, non-overlay nodes are loosely connected since they do not send any control messages. This may imply high message loss, as nodes need to recover from the loss of connectivity. Thus, membership information is never available.

PAST-DM [15] (Progressively Adaptive Subtree in Dynamic Mesh) is an overlay multicast protocol based on the construction of a virtual mesh. The mesh is maintained dynamically through the exchange of link state packets, thus adapting it to network topology changes. These packets provide link state table information, i.e., a partial view of the network. All nodes need to start the multicast session simultaneously, and afterwards initiate the bootstrapping process by sending TTL-bounded broadcast messages. With the topology information extracted from the mesh, nodes compute a source-based Steiner tree to propagate information to all members in the multicast group. Logical and physical hop distances are used as heuristics to compute the Steiner tree. The source node takes its logical (virtual) neighbors as children in the tree. The rest of the nodes are packed into subgroups, which form a subtree where the root of this tree is one of the logical neighbors. Thus, 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 has computed its corresponding subtree.

3 MIDDLEWARE DESIGN

Most middlewares that has been utilized so far in the development of distributed applications has appeared inadequate in supporting coordination activities in mobile and wireless scenarios. In this respect, Zambonelli [20] identified three basic techniques of communication: (i) *direct communication* (unicast), (ii) *shared data spaces* (e.g. tuple spaces), and (iii) *event-based models* relying on publish/subscribe. In order to address coordination, communication in SCOMET has been mainly modelled using unicast and publish/subscribe functionalities, with the publish/subscribe being sustained by OMCAST. To support inter-device interactions through a shared data structure, SCOMET includes a naming service intended to store the minimal but the critical information required to coordinate devices in this way.

With the three communication components, we view SCOMET as a ready-to-deploy solution for developing collaborative applications for MANETs. Since we have not constrained our communication abstractions to be of a particular class (as specified in the above paragraph), developers can use all set of functionalities to coordinate devices.

In summary, SCOMET provides the following features:

- *Group management services* to dynamically handle the addition, deletion, and modification of the

information relative to group membership.

- *Communication services.* In general, developing collaborative applications requires two important communications abstractions: *unicast*, that is, the sending of messages to a single destination, and *multicast*. Further, we provide a publish/suscribe and a naming service to support coordination.

3.1 Communication Services

The main component is the communication channel, which allows participating devices to send messages to one or to all the members of a given group. Also, named communication channels are available, so applications built atop do not have to filter the messages from other applications. Messages destined for all group members are sent through ALM, avoiding using multiple unicast messages. Moreover, the channel provides mechanisms for synchronous and asynchronous reception. On top of this channel, we have settled the other two basic pillars that facilitate the coordination of devices' activities: The publish/subscribe and naming services.

The implementation of the publish/subscribe service supports a set of the JMS [23] (Java Message Service) interface. As it follows from the JMS specification, its behavior is practically identical to the one defined in the standard. More precisely, the service furnishes a topic-based publish/subscribe model with persistent and non-persistent subscriptions. It is important to note here that SCOMET does not set up a central JMS server to handle subscriptions. Subscriptions are partially maintained by each member in the group. In this way, when a device rejoins a group due to a temporary disconnection, all required to do is to ask one member in the group to obtain all the previous messages published on a given topic.

The naming service also follows a Java standard. It implements a subset of the JNDI [24] (Java Naming Directory Interface) interface. Its aim is to be used to store lightweight data such as resource discovery, group and coordination information. The mechanism to store this data is simple but effective: whenever changes occur in the naming service, the changes are sent via multicast to all group members. As we know that too many updates might be resource-expensive for the system, the naming service is supposed to maintain only minimal but critical data. In constrast, what we gain is performance. Stored information can be rapidly made available to all current group members.

3.1.1 Routing protocols: Ordering and Reliability

As SCOMET has been designed for collaboration, routing protocols should also consider the particular requirements of spontaneous collaboration over mobile ad-hoc networks.

Typical scenarios such as meetings or scientific conferences are characterized by having a moderate number of nodes located in a small area. Normally, they are located in a room or in a large hall, where some of them may remain static for long periods of time. Some may change its location from time to time in order to interact with other existing groups. However, the major part of communication is performed in well-defined areas, where nodes are located at most two or three hops from the most distant hop.

Consequently, multicast communication represents a good option to provide reliable group communication. Owing to this fact, we have devised OMCAST, an ALM protocol specially designed for CWEs. We will describe OMCAST carefully in Section 4. By the moment, only to say that, with the use of OMCAST, we have reduced network traffic. In order to do this, OMCAST forwards a multicast message just once by broadcasting it to the devices located at one physical hop from the sender. In this way, multiple unicast transmissions are avoided. It must be pointed out that, like almost all ALM protocols, OMCAST does not guarantee neither reliable message delivery nor message ordering. In what follows, we explain how OMCAST addresses these issues.

Ordering and Reliability. TCP has been discouraged for MANETs. For this reason, we focus on bringing ordering and reliability to the UDP protocol. As outlined in the introduction, these two characteristics are critical when designing a middleware for CWEs over MANETs. To provide reliability and ordering in a transparent way, we advocated for the toolkit JGroups [4] as the foundation of our communication media layer. JGroups, based on Java technology, offers reliable and efficient group communication through a flexible protocol stack, where each protocol provides a specific functionality. The list of functionalities includes for instance lost message retransmission, ordering and encryption.

As JGroups main characteristic, we find reliability in multicast communication, which is a deployment issue and does not have to be implemented by the developer.

However, the most interisting feature of JGroups is its extensibility: A new protocol can be easily added to the protocol stack, thereby extending its capabilities. In order to achieve this, JGroups operation is based on *the responsibility chain's pattern*, where each protocol only handles its corresponding part of the message when it is passes through the protocol stack.

A protocol may modify, reorder, simply pass, drop a message, or add a new header to it. Further, a fragmentation layer can break up a message into several smaller messages, adding a header with an id to each fragment, and re-assembling the fragments later on the receiver's side.

The composition of the protocol stack is determined by the creator of the channel: An XML archive specifies the layers to be employed (and the parameters for each layer).

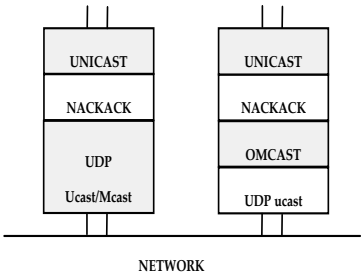

Figure 1: JGroups protocol stack.

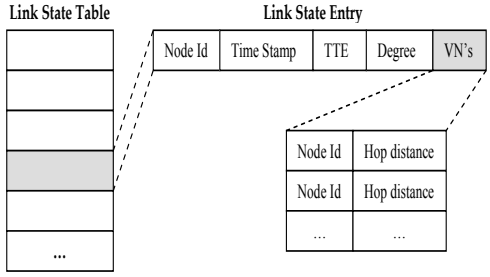

Figure 2: Link State Table structure.

In SCOMET, we have exploited the reliability and ordering protocols provided natively by JGroups. Since we do not need network layer multicast, we replaced it by OMCAST, our application layer multicast, as shown in Figure 1.

UNICAST protocol is in charge of ensuring FIFO ordering and reliability for unicast packets. NACKACK behaves in a similar way for multicast packets. UDP allows sending unicast and multicast messages in the original JGroups structure.

In this way, we ensure that all application messages will be delivered to the corresponding receivers. Also, setup parameters like the maximum number of retries and retransmission timeouts can be more easily tuned to get the desired behavior.

4 OMCAST: COLLABORATIVE ALM PROTOCOL

Next, we describe OMCAST in greater detail. OMCAST is based on PAST-DM and hence, it relies on link state information exchanging and Steiner trees. As a result, a virtual mesh connects all participants in the multicast group, which is built dynamically by periodically exchanging link state information. However, OMCAST supplies a more flexible bootstrap method than PAST-DM and takes advantage of the broadcast nature of wireless communication.

4.1 Basic Operation

Thanks to its flexible bootstrap method, OMCAST allows multicast members to join at any time. Any node wishing to enter the multicast session only needs to send a $JOIN$ message. This join message consists of a local broadcast message, which is successively forwarded by non-member nodes. When the message contacts a group member, a unicast reply is sent to the source. This reply message contains local link state information as well as the number of hops the node is located from the replier. Thus, the bootstrapping node automatically receives information about the current neighbors in the group, and it is ready to send multicast messages to any node.

The mesh maintenance in OMCAST is similar to the one used by PAST-DM, but some changes are necessary to improve the whole operation. Each device maintains one Link State Table (LST) (refer to Figure 2) for each multicast group it belongs. The most recent entries are periodically exchanged with the node virtual neighbors, i.e., the members of the multicast group located at one logical hop from the local node. As a result, each LST keeps a Link State Entry (LSE) for each member of the current multicast group. Each LSE stores information about the link state of the corresponding member: Node degree, virtual neighbors; together with the node identifier and the Time To Expire (TTE). Lower values on the expiration time indicate that the information kept is less accurate than the entries with higher values. When TTE drops below a certain threshold, the entry is marked as invalid. When more recent versions of an entry arrives, TTE is set to the highest value and the entry is marked again as valid.

Thanks to Link State information, devices can set up source-based Steiner trees and deliver data to all members. Besides, membership information can be provided to upper layers, which is actually very useful for CWEs. Providing this information only implies offering a list of node identifiers retrieved from each LSE.

When a node wants to send a multicast message to all members, it computes the source-based Steiner tree, and sends a copy of the multicast message to each virtual neighbor (1-level children). Each virtual neighbor is responsible for delivering the message to a certain subgroup of nodes as determined by the source-created tree. Along with the message content, each copy encloses a header with the information about the subgroup of nodes that has not received the message yet, but should receive it. Upon receiving its copy, each virtual neighbor sets up its subtree on basis of the information kept in the header. Next, it sends a copy of the message to its children with the header content updated. This process repeats until the subgroup which must receive the message is empty.

4.2 Local broadcast delivery

In protocols like PAST-DM, when a node wants to forward a message through a generated tree, it resorts to its state information generating as many copies as members wish to receive the message. One different unicast packet is then sent to each member.

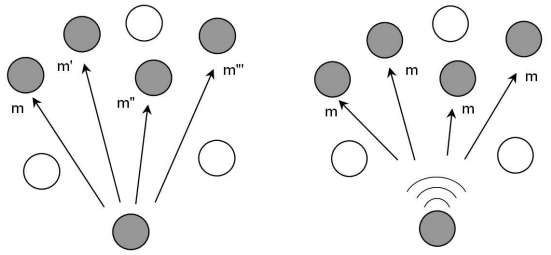

Figure 3: Standard vs. local broadcast delivery.

In OMCAST, we use a bandwidth-saving technique that consists in transmiting the message via local broadcast. If there exist enough neighbors at one physical hop willing to receive the message, the message will be sent just once as a broadcast message, saving bandwidth. In Figure 3, we show the standard mechanism to send a multicast message to each receiver in comparison to our broadcast-based method. In the first method, the packet m is sent 4 times through the network while in our approach is sent just once to the medium, as nodes in range can hear the packet.

It must be noted that, even taking as a basis CWEs, devices may change its location unexpectedly. Hence, it is likely that some of the packets locally broadcasted do not reach their final destination due to mobility issues (e.g., virtual neighbor information might be temporally inaccurate). To minimize packet loss, broadcast delivery will be performed only under certain conditions:

- The number of virtual neighbors that are ready to receive the message must be greater than a threshold, say MIN_BC_NEIGH.
- A virtual neighbor is considered to be ready to receive a broadcast message only if the sender has received one link state message from it in the last BC_PERIOD seconds. This constraint ensures that a broadcast message is sent only to the nodes truly located at one physical hop. The assumption is that within a close time period, a 1-hop neighbor will vary its location negligibly.

Another problem occurs when a node is supposed to forward a message to a receiver that does not appear in its tree. This can happen because there is only one node responsible of determining the first subgroups of nodes: the source. However, OMCAST solves this problem by forcing forwarder nodes to unicast the message to those receivers that did not appear in the original source-based Steiner tree.

5 PERFORMANCE EVALUATION

We performed extensive tests on our middleware to validate its functionalities. We focused on general tests. First, we examined OMCAST performance. Recall that multicast is critical for group communication in CWEs. For this reason, we measured the network traffic and the end-to-end delay incurred by OMCAST. Finally, we examined SCOMET in a real setting in order to verify that SCOMET communication media layer works correctly.

5.1 OMCAST

To evaluate OMCAST, we performed numerous emulations mimicking realistic CWEs. We designed a scenario of 30 devices with multicast groups oscillating between 5 to 20 members. To approximate emulations to reality, we set up an emulation environment consisting of VMware images along with an enhanced version of MobiEmu [37]. APE Mackill [3] was used in conjuction with MobiEmu to emulate wireless connectivity. Also, DYMOUM, the most recent implementation of DYMO [9] protocol, was the MANET routing protocol chosen to support the multihop connections of participating devices.

DYMO protocol enables reactive, multihop unicast routing between DYMO devices. The basic operations of DYMO are route discovery and route management. During route discovery, the source DYMO device starts the dissemination of a Route Request (RREQ) throughout the network to find a route to the target DYMO node. During this hop-by-hop dissemination process, each intermediate DYMO router records a route to the originator. When the target DYMO router receives the RREQ, it responds with a Route Reply (RREP) sent hop-by-hop towards the originator. Each intermediate DYMO router that receives the RREP records its path to the target, and forwards the RREP to the next hop towards the originator. When the originator receives the RREP, paths have been definetly established between the source router and the target router in both directions.

DYMO utilizes sequence numbers to guarantee loop freedom. Sequence numbers enable DYMO routers to preserve the order of route discovery messages (RREQ), thus minimizing stale routing information.

Since we wanted to measure OMCAST performance in a real CWE, we could not take the random waypoint model as mobility model. So, we had to design specific scenarios to do so. More specifically, we designed a set of scenarios in which the devices were located in 3 non-connected areas. In each scenario, there was one or two nodes linking the first with the second area, and the second with the third area, so that reaching all members in the MANET was initially possible.

We spent 140 seconds per simulation, enough time for letting some of the devices switch from one area to another, remaining static for a pause time of 30 seconds. Members of the multicast group transmitted packets at a constant bit rate of 4 Kb/s.

5.1.1 Congestion analysis

In our first test, we evaluate the potential benefit one can obtain from using broadcast delivery rather than multiple unicast transmissions. The simulation compares the

multicast traffic injected into the MANET by unicast vs. broadcast delivery. With this test, we want to prove the efficiency of broadcast delivery in areas with co-located participants.

We designed a scenario with 20 devices that injected multicast messages into the MANET at different rates. The value of MIN_BC_NEIGH was varied to restrict the power of broadcast delivery in each simulation. Recall that broadcast messages are only sent if there are at least MIN_BC_NEIGH virtual neighbors ready to receive the multicast packet. To simplify the exposition, we use the notation $OMCAST-2$, $OMCAST-3$, $OMCAST-5$ to indicate that the value of parameter MIN_BC_NEIGH was set to 2, 3 and 5, resp. In addition, $OMCAST-NOBC$ signals the non-use of broadcast delivery, which means that all messages were sent via unicast.

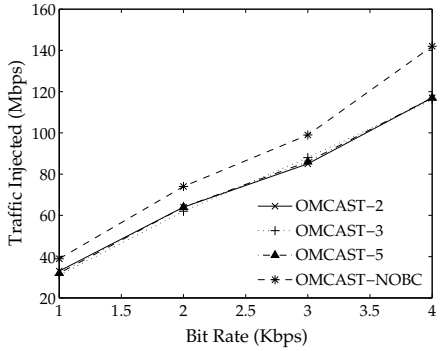

Figure 4: Overall network traffic.

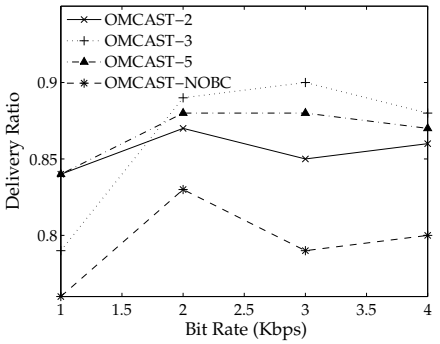

Figure 5: Packet delivery ratio.

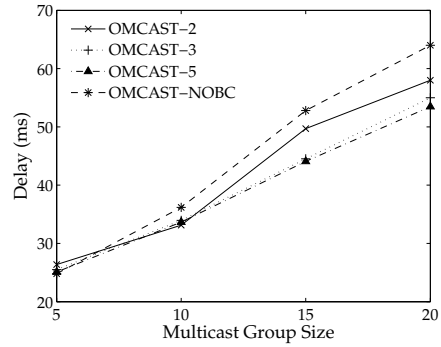

Figure 6: Round-trip-time delay (ms).

As shown in Figure 4, broadcast delivery managed to reduce about 20% the multicast traffic. This fact can be explained as follows. Since each multicast packet is consumed by multiple receivers simultaneously, fewer packets are injected into the network. Further, the extra traffic due to unicast-based multicast increases end-to-end delays and the packet loss rate, as will be shown in the next section.

It is worth noting that the three variants of broadcast delivery performed similarly. This is because it is quite unlikely that a group member has more than 5 neighbors in wireless range in a real scenario.

5.1.2 Impact of broadcast delivery

The second evaluation focuses on measuring the influence of broadcast transmissions on packet delivery ratio. At this point, it is interesting to note that a unicast packet is addressed to a specific node, while a broadcast packet is addressed to all nodes located physically at one hop, contacting only those devices in wireless range with the source. Consequently a node might not receive a broadcast packet due to inconsistencies in its LST. However, since multicast traffic reduces with broadcast (see Figure 4), we can expect that congestion diminishes, too, in benefit of a higher packet delivery ratio.

Assuming the same parameters of the previous simulation, Figure 5 shows that broadcast-based OMCAST is approximately 15% better than the non-broadcast version.

5.1.3 Round-trip-time delay

The last test explores the effect of broadcast delivery on end-to-end delay. As just observed in the preceding two evaluations, traffic reduces when multicast messages are broadcasted. Therefore, it is not strange to suspect that network delay will experience a certain reduction when broadcast transmission is active. To prove this claim, it is necessary to show if latency diminishes as function of the group size.

End-to-end delay was measured by forwarding multiple multicast requests with timestamp. Upon receiving a multicast request, each group member replied with a unicast ACK message to the source which enclosed the original timestamp. As a result, sources could measure the round-trip-time with respect to all nodes in its group.

The average network delay is shown in Figure 6. Non-surprisingly, network delay was higher when non-broadcast delivery was used. Broadcast incurred lower delays, which can be explained by the traffic reduction in the MANET as result of the exploitation of the broadcast nature of wireless connections.

In conclusion, OMCAST provides a high packet delivery ratio without renouncing to low mean delays (in comparison to unicast-based multicasting).

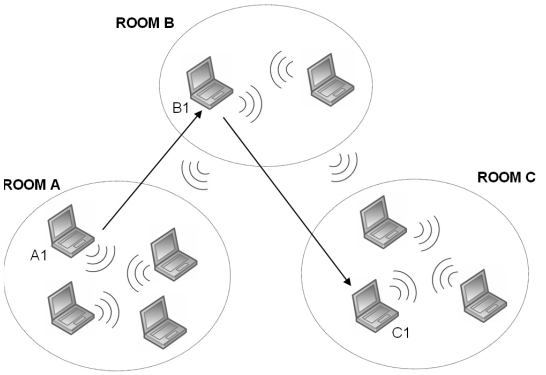

Figure 7: Real Test Scenario.

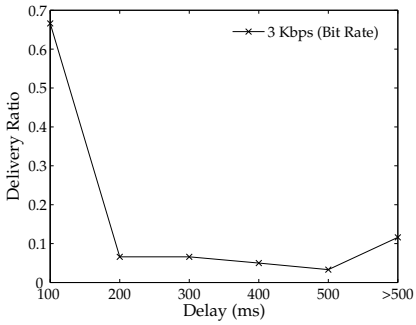

Figure 8: Delivery Ratio.

5.2 SCOMET middleware

To evaluate our middleware, we built a simple chat application on top of SCOMET. This proof was a sanity check to verify whether SCOMET can react quickly to sudden membership changes (devices joining or leaving groups), and whether unicast and multicast communication was reliable, too.

From a coordination viewpoint, the chat serves as a first approximation to the problem of coordinating equal entities with the same rights that need to cooperate in a certain task with no information about each other.

To increase realism, we recreated a typical CWE. As shown in Figure 7, three rooms were set up with different mobile devices within each one. Devices in room A and C could only communicate with nodes in room B. The two nodes located in room B could communicate directly with the rest of the nodes in the MANET. In this configuration, when a node, say A1, needed to communicate with a node C1, the message had to be forwarded via a node in room B. In this way, we configured a 2-hop scenario, which became a 3-hop scenario when mobile devices started to move. In particular, two nodes from Room A moved to Room B and Room C during the test.

For this particular test, all the laptops ran Windows XP service pack 2 and Andreas Tonnesen's OLSR implementation. The test measured the end-to-end delay of multicast packets. To do it, initially, one node was asked to send a message REQ to all members of its group. The rest of nodes, when receiving this REQ message, replied with an ACK unicast message to the source. ACKs held the timestamp value of the original REQ message. Once the source collected all the ACKs from the other group members, the average round-trip-time was computed. A total amount of 50 REQ messages were sent via multicast.

In Figure 8, we depict the mean round-trip-time for both multicast and unicast packets. In the figure, a measure of 100 ms means that delay is between 0 and 100 ms; a measure of 200 indicates values between 100 and 200 ... We can observe that nearly a 70% of packets are delivered in less than 100 ms.

It is important to note here that SCOMET guarantees reliability and ordering of messages. More specifically, retransmitted packets are also delivered in order. Thus, we can conclude that SCOMET performs well in a real scenario.

5.2.1 Application development with SCOMET

SCOMET characterizes for exhibiting a fast application development and a reliable and fast communication (as a result of the complete set of services it provides). Some settings that present mobile entities as devices with the same privileges that need to cooperate in a common task will highly benefit from SCOMET multicast capability. More specifically, with SCOMET, developers will have a clean manner to free entities from the brake of explicitly holding the addresses of the other entities involved in coordination.

Other applications that may face coordination issues will also benefit from SCOMET's rich set of tools. For illustration purposes, consider an application requiring a meeting point [7] to coordinate mobile entities. To gain access to the meeting point, mobile entities could resort to our naming service to resolve its location. Once the meeting point was made visible to all participants, they could start to communicate through the unicast channel and therefore, registering, being notified about changes or calling for meetings.

In fact, some collaborative applications have already taken profit of SCOMET. At the time present, SCOMET provides the basic middleware services of the European project POPEYE. In such terms, SCOMET has been the pillar to implement some applications (plug-ins) which include a file-sharing, a presentation viewer, and chat-like communication utilities. The complexity on the development of these applications was remarkably diminished thanks to the communication services provided by SCOMET: the unicast/multicast channel, and the publish/suscribe and naming services.

6 CONCLUSIONS

In this paper, we have introduced SCOMET, a middleware for developing spontaneous collaborative tools.

Since most of the current middlewares cannot handle all requirements found in mobile collaboration, SCOMET proposes a novel group-based middleware that provides group management and communication functionalities. To do so, SCOMET integrates toolkit JGroups into its design to offer reliable multicast and other services to aid coordination. To provide one-to-many communication, SCOMET includes a propietary Application Layer Multicast protocol named OMCAST.

Targeted to collaborative scenarios, OMCAST benefits from broadcast-based transmission to reduce traffic and minimize packet loss. It also reduces end-to-end delay. Evaluation showed that OMCAST performs well in collaboratve scenarios in which nodes are grouped into differentiated areas interconnected by one or two nodes.

Currently, we are performing several real tests with a 15-node testbed, in which devices (*located in different rooms*) periodically switch their corresponding regions. The motivation behind these tests is to study the stability of SCOMET under mobility situations. By the moment, a chat application has been tested, performing satisfactorily.

For short-term future work, we want to implement some coordination patterns on top of SCOMET, our coordination media layer, in order to verify how it behaves under typical coordination scenarios. With the patterns, our next step will be to define a new coordination model through which developers can build applications with a certain quality of service.

Acknowledgements

This research is partially supported through project POPEYE, FP6-2006-IST-034241, under the Information Society Technologies programme, European Commission.

REFERENCES

[1] Basic Support for Cooperative Work (BSCW). http://www.bscw.de.

[2] Groove virtual office web site (Microsoft Corporation). http://www.groove.net.

[3] The Ad hoc Protocol Evaluation (APE) testbed. http://apetestbed.sourceforge.net/.

[4] The JGroups Project. http://www.jgroups.org.

[5] M. Bisignano, G. D. Modica, and O. Tomarchio. A JXTA Compliant Framework for Mobile Handheld Devices in Ad-Hoc Networks. In *Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC 2005)*, Murcia, Cartagena, Spain, June 2005.

[6] S. Blodt. Efficient End System Multicast for Mobile Ad-Hoc Networks. In *Proceedings of the 2nd IEEE Annual Conf. on Pervasive Computing and Communication Workshops (PERCOMV'04)*, Orlando, Florida, USA, March 2004.

[7] M. Bortenschlager, S. Reich, and G. Kotsis. A Generic Coordination Architecture as an Enabler for Mobile Collaborative Applications. In *WETICE '06: Proceedings of the 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 125–130, Washington, DC, USA, 2006.

[8] D. Bottazzi, A. Corradi, and R. Montanari. AGAPE: a Location-aware Group Membership Middleware for Pervasive Computing Environments. In *Proceedings of the ISCC'03*, Kemer-Antalya, Turkey, July 2003.

[9] I. D. Chakeres and C. E. Perkins. Dynamic MANET On-demand (DYMO) Routing. Internet Draft, draft-ietf-manet-dymo-06.txt (Work in progress), October 2006.

[10] P. Ciancarini, A. Omicini, and F. Zambonelli. Coordination technologies for Internet agents. *Nordic Journal of Computing*, 6(3):215–240, Fall 1999.

[11] G. Cugola and G. P. Picco. PeerWare: Core Middleware Support for Peer-to-Peer and Mobile Systems. Technical report, Politecnico di Milano, 2001.

[12] C. Fok, G. Roman, and G. Hackman. A Lightweight Coordination Middleware for Mobile Computing. In *Proceedings of the 6th International Conference on Coordination Models and Languages (Coordination 2004)*, Pisa, Italy, February 2004.

[13] M. Ge, S. V. Krishnamurthy, and M. Faloutsos. Application versus Network Layer Multicasting in Ad-Hoc Networks: the ALMA Routing Protocol. *Ad Hoc Networks*, (Vol. 4, No. 2):283–300, March 2006.

[14] D. Gelernter. *ACM Computing Surveys*, (Vol. 7, no. 1), Janurary 1985.

[15] C. Gui and P. Mohapatra. Efficient Overlay Multicast for Mobile Ad Hoc Networks. *IEEE Wireless Comm. and Networking Conf., IEEE Press*, (Vol. 2):1118–1123, 2003.

[16] S. Hadim, J. Al-Jaroodi, and N. Mohamed. Trends in Middleware for Mobile Ad Hoc Networks. *Journal of Communications*, (Vol. 1, No. 4):11–21, July 2006.

[17] D. Johnson, D. Maltz, and J. Broch. *DSR The Dynamic Source Routing Protocol for Multihop Wireless Ad-Hoc Networks*.

[18] M. A. Kaafar, C. Mrabet, and T. Turletti. A Topology-Aware Overlay Multicast Approach for Mobile Ad-Hoc Networks. In *Proceedings of 2006 Asian Internet Engineering Conference (AINTEC)*, Bangkok, Thailand, November 2006.

[19] M. Liu, R. Talpade, and A. McAuley. AMRoute: Ad-Hoc Multicast Routing protocol. *ACM/Baltzer Mobile Networks and Applications, special issue on Multipoint Communications in Wireless Mobile Networks*, (Vol. 7 no. 6), 2002.

[20] M. Mamei and F. Zambonelli. Field-based Approaches to Adaptive Motion Coordination in Pervasive Computing Scenarios. *Handbook of Algorithms for Mobile and Wireless Networking and Computing CRC Handbook, 2004*.

[21] R. Meier and V. Cahill. Exploiting Proximity in Event-Based Middleware for Collaborative Mobile Applications. In *Proceedings of the 1st Workshop on Middleware for Network Eccentric and Mobile Applications (MiNEMA)*, Dublin, Ireland, 2004.

[22] Microsoft. Microsoft Sharepoint Services-Portal. http://www.microsoft.com/sharepoint/.

[23] S. Microsystems. Java Message System (JMS). http://java.sun.com/products/jms/.

[24] S. Microsystems. Java Naming and Directory Interface (JNDI). http://java.sun.com/products/jndi/.

[25] S. Microsystems. JavaSpaces(TM) Service Specification. http://java.sun.com/products/jini/.

[26] M. Musolesi, C. Mascolo, and S. Hailes. EMMA: Epidemic Messaging Middleware for Ad hoc networks. *Journal of Personal and Ubiquitous Computing, Springer*, (Vol. 10, No. 1):28–36, February 2006.

[27] C. Perkins. Ad-Hoc On-Demand Distance Vector (AODV) Routing. IETF, Internet Draft, draft-ietf-manet-aodv-00.txt, November 1997.

[28] G. Picco, A. Murphy, and G. Roman. LIME: Linda Meets Mobility. In *Proceedings of the 21st International Conference on Software Engineering (ICSE)*, Los Angeles, USA, May 1999.

[29] G.-C. Roman, R. Handorean, and R. Sen. Tuple Space Coordination Across Space and Time. In *8th International Conference, COORDINATION 2006, Bologna, Italy, June 14-16, 2006*, volume 4038 of *Lecture Notes in Computer Science*, pages 266–280, 2006.

[30] E. Royer and C. Perkins. Multicast Ad-Hoc On-Demand Distance Vector (MAODV) Routing. IETF, Internet Draft: draft-ietf-manet-maodv-00.txt, March 2000.

[31] W. S. S. Ho Bae, S.J. Lee and M. Gerla. *IEEE Network Magazine*, (Vol. 14, no. 1), Janurary 2000.

[32] A. I. Wang, T. Bjornsgard, and K. Saxlund. Peer2Me - Rapid Application Framework for Mobile Peer-to-Peer Applications. In *Proceedings of International Symposium on Collaborative Technologies and Systems (CTS'07)*, Orlando, Florida, USA, May 2007.

[33] B. Williams and T. Camp. Comparison of Broadcasting Techniques for Mobile Ad Hoc Networks. In *Proceedings of the 3rd ACM International Symposium on Mobile Ad-Hoc Networking and Computing (MobiHoc'02)*, Lausanne, Switzerland, June 2002.

[34] P. Wyckoff, S. W. McLaughry, T. J. Lehman, and D. A. Ford. Tspaces. *IBM Syst. J.*, 37(3):454–474, 1998.

[35] P. Ya, E. Krohne, and T. Camp. Performance Comparison of Geocast Routing Protocols for a MANET. In *Proceedings of the 13th IEEE International Conference on Computer Communications and Networks (IC3N)*, pages 213–220, 2004.

[36] L. Yan, K. Sere, X. Zhou, and J. Pang. Towards an Integrated Architecture for Peer-to-Peer and Ad-Hoc Overlay Network Applications. In *Proceedings of the 10th Workshop on Future Trends in Distributed Computing Systems (FTDCS'04)*, 2004.

[37] Y. Zhang and W. Li. An Integrated Environment for Testing Mobile Ad-Hoc Networks. In *Proceedings of the 3rd ACM International Symposium on Mobile Ad-Hoc Networking and Computing (MobiHoc'02)*, Lausanne, Switzerland, June 2002.