Group management for mobile Ad Hoc networks: design, implementation and experiment

Jinshan Liu, Daniele Sacchetti, Françoise Sailhan, Valérie Issarny

Group Management for Mobile Ad Hoc Networks: Design, Implementation and Experiment

Jinshan Liu
jinshan.liu@inria.fr

Françoise Sailhan
francoise.sailhan@inria.fr

Daniele Sacchetti
Daniele.Sacchetti@inria.fr

Valérie Issarny
valerie.issarny@inria.fr

INRIA - Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105
78153 Le Chesnay Cedex, France

ABSTRACT

Mobile ad hoc networks (MANET) offer a convenient basis toward realization of pervasive computing, due to its ease of deployment and inherent support for anytime, anywhere network access for mobile users. However, the development of applications over such networks is faced by the challenge of network dynamics attributed to node mobility and the scalability issue. Group management poses as a promising paradigm to ease the development of distributed applications for dynamic, mobile networks. Specifically, group management makes transparent the failures due to node mobility and assembles mobile nodes to meet target functional and non-functional properties. Various network-level grouping schemes over MANET have been investigated over the last couple of years. In this paper, we introduce the design and implementation of a generic group service for MANET, defined with respect to the various attributes of relevance. Generic group management is further demonstrated with its support of scalable service discovery in MANET.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Communications Applications; C.2 [**Computer-Communication networks**]: Distributed Systems

General Terms

Design, Performance

Keywords

Group management, mobile ad hoc networks, middleware, mobile computing

MDM 2005 05 Ayia Napa, Cyprus

1. INTRODUCTION

The development of applications over wireless ad hoc networks is confronted by the challenge of frequent changing of environments, e.g., network topology. One approach to master this complexity lies in the grouping (clustering) of nodes over the network, i.e., applications execute on top of group services that manage the execution context dynamics, including node mobility. Another advantage of grouping in mobile ad hoc networks is to achieve system scalability, with group leaders responsible for inter-group communications. There has been extensive research on group management and related group communication services in the context of fixed networks, with special emphasis on providing availability properties [22, 10]. However, proposed solutions cannot be applied directly to mobile wireless networks due to their highly dynamic topology [3]. This necessities the adaption of group membership to the specifics of mobile networks.

Group membership is primarily defined according to the functional property to be collectively achieved by the group, e.g., collaborative editing, sharing computational load, providing fault tolerance [22]. In general, a member may leave a group because it fails, explicitly requests to leave, or is expelled by other members. Similarly, a member may join a group because it explicitly files a request or recovers from failure. A group management protocol must handle such dynamics in a coherent way, i.e., all members of the group must have a consistent view of the group's membership despite failures [10]. The highly dynamic topology of wireless ad hoc networks introduces additional complexity in the management of group membership because connections can be transient and partitioned networks may never be rejoined together.

It can be argued that group membership in MANET may be defined as in fixed networks, assuming that the network is able to restore lost connections with the underlying routing protocol and it is sufficient to adapt the group communication service to the dynamic topology of the network [18].

Such an approach does not adapt the system's functions to the network specifics, but rather modify the implementation of underlying distributed system functions. It is, however, advantageous to revise the definition of group membership to integrate the network dimensions in addition to the functional one. This enables the group to enact quality of service (QoS) requirements by bounding communication

latency [14] and/or supporting location-aware applications. For example, group constraints may be set according to the network's scale such as group size (e.g., [23, 8]) or the respective geographical positions of the group's members (e.g., [20]). The definition of group membership may also be extended with integrity constraints (e.g., security constraints, size) [21].

In general, group management in mobile ad hoc networks embeds a number of design dimensions related to group member attributes and the non-functional properties that are offered to applications. In this paper, we provide a characterization of the attributes of group membership in MANET (Sec. 2), and then introduce the design of a group management service that is generic with respect to the elicited attributes (Sec. 3). The assessment is based on its implementation in a middleware aimed at mobile computing (Sec. 4), and its usage for enabling scalable service discovery in MANET (Sec. 5). Finally, we conclude with a summary of our contribution and future work (Sec. 6).

2. ATTRIBUTES OF GROUP MEMBERSHIP

Many of the application scenarios where group membership is considered are location-dependent, i.e., ad hoc groups are formed according to the presence of nodes in a given geographical region, in addition to the functional properties supported by the group. The geographical region can be defined with respect to either a fixed geographical point or the position of a mobile node. In the former case, presence in the geographic region enacts a location-dependent application in scenarios such as business-to-consumer interactions in malls, airports and trade fair communities [21].

In the latter case, communities are formed by proximate nodes, which can be dedicated to resource sharing with QoS properties, such as enhanced data availability [6] or real-time processing [14, 25]. Group membership can also depend upon trustworthiness of group members, e.g., it is possible to restrict membership to authorized nodes to ensure secure interactions among members. Constraints on group membership may be considered at the level of the overall group (e.g., group size as in [21]) and/or at the level of group members (e.g., storage capacities of nodes). Ideally, a group management service for wireless ad hoc networks must be configurable with respect to the above attributes to support various applications. In the following, we provide a characterization of these attributes, by first defining the underlying network model that we assume.

Groups are first defined with respect to a functionality, denoted as *f*, which can characterize various features supported by nodes, e.g., resources and services. We use *support(x, f)* to denote the fact that node x offers function *f*, and G^f to denote a group with function f, i.e., $G^f = \{x|\ x \in N$ and $support(x, f)\}$.

2.1 Network Model

We consider a network consisting of a set of N nodes, and assume that every node x of $\mathcal{N}$ has a unique identifier $Id(x)$. The following notations are used to reason about the attributes of nodes.

- $Proximity(x, p)$ returns the geographical distance in meters between the location of node x and geographical position p.
- $Distance(x, y)$ returns the geographical distance in meters between x and $y \in \mathcal{N}$.
- $Hops(x, y)$ returns hop count between x and y for any y reachable from x.

2.2 Location

We now define group membership with respect to constraints on the location of member nodes.

Location-unaware groups (e.g., in [18, 19]), are defined only with respect to the functions offered by the group. Hence, $LocationUnaware(G^f)$ always holds.

Proximity-based groups (e.g., in [20, 21]), refine group members to be within a given geographical area, with reference location either be fixed *a priori* or the position of other group members. Let *pos* denote a referenced geographical position and *dist* denotes the maximal geographical distance that is allowed, we have:

$$Geographical\_Proximity(G^f, pos, dist) \Leftrightarrow \forall x \in G^f : Proximity(x, pos) < dist$$
$$Relative\_Proximity(G^f, dist) \Leftrightarrow \forall x, y \in G^f : Distance(x, y) < dist$$

2.3 Scale

Bounded groups (e.g., in [1, 23, 8]), are defined based on the maximal number of hops ($\mathcal{H}$) between nodes and a specific node (usually group leader, notated as node l):

$$Bounded(G^f, \mathcal{H}) \Leftrightarrow \forall x \in G^f : Hops(x, l) \leq \mathcal{H}$$

Group size (e.g., in [9]) defines the maximal number of members in a group. And we have (assuming the maximum group size is N).

$$Size(G^f, N) \Leftrightarrow |G^f| \leq N$$

2.4 Trustworthiness

Group membership can be restricted to nodes which trust each other, meaning that each member has a decent reputation towards every other based on past experiences and peer recommendations [16]. The group membership is thus subject to a node's reputation (*Rep*) among its peers in the group, which is at least δ to be considered trustworthy:

$$Trusted(G^f, \delta) \Leftrightarrow \forall x, y \in G^f : Rep(x, y) \geq \delta$$

Higher trustworthiness can be achieved with signed certificates (SC), which is managed by a trusted third party, e.g., some certificate authority (CA). Newcomers which lack previous interactions with other peers can also start with a SC. Secure group communication can be enforced through the implementation of group key agreement (GKA) within the group [4]. Thus we have:

$$SC(G^f, CA) \Leftrightarrow \forall x \in G^f : x \in Authorized(CA)$$

2.5 QoS Awareness

Groups can also be defined by certain QoS attributes, e.g., the following that appear to be the most dominant in the context of mobile wireless networks [17]: *reliability*, *performance* and *transaction* for service level attributes, and

Category	Location			Scale		Trustworthiness		QoS-Awareness
	Location Unaware	Proximity		Bounded	Size	Reputation	Certificate	
		Geographical	Relative					
Local	√	√		√			√	√
Group			√		√	√		

Table 1: Categorization of attributes

CPU load, *memory*, *bandwidth* and *battery* for resource level attributes.

In addition, group membership can be restricted in terms of node mobility to limit the probability of a node leaving a group. For instance, the mobility of nodes has been suggested as a criterion for integration within a group [23], which possibly assumes a group-based mobility model [13].

In general, disconnection of a node from a group can result from the node's mobility and/or the node's resource scarcity. The former can be predicted based on the node's movement pattern, and the latter can be anticipated based on information about resource level QoS values of the node.

2.6 Attributes Categorization

The above listed attributes can be categorized by their ranges of application (see Table. 1): (1) *local constraints*, which apply to a single node, e.g., *Geographical_proximity* and *SC*. (2) *group constraints*, which apply to the whole group, e.g., *Relative_Proximity* and *Size*.

The difference between these two categories is that for the former, group membership can be confirmed locally (e.g., by calculating the distance from a fixed point), while for the latter one, group membership has to be confirmed in a group range (e.g., by calculating the total number of nodes), which has to be carried out during the group formation. Note that the above attributes are not exclusive of each other and may be combined for the definition of a group.

After introducing the relevant group attributes, we present the design of a group management service that is generic with respect to the aforementioned attributes.

3. GROUP SERVICE DESIGN

Due to the challenges posed by mobility and resource limitation of nodes, we consider the following properties necessary for group management design:

1. **Resource-saving**. The overhead of group management should be affordable for resource-constrained devices.
2. **Distributed**. Group management cannot accommodate a centralized solution where a single node is always responsible for managing the group. This is due to node mobility and necessity of load balancing.
3. **Dynamic**. The group management service must accommodate the highly dynamic topology of the network.

We distinguish three functions in group management:

- Group member discovery, i.e., discovering mobile nodes that are eligible for membership according to relevant membership attributes.
- Group initialization, i.e., exchanging meta-data relevant to the group's functionality and enforcing global membership constraints.
- Group dynamics management, i.e., updating group membership according to the dynamics of the network's topology.

In our work, we assume that each node in the network is able to (1) know its 1-hop neighbors and (2) detect neighbors leaving and coming by underlying routing table. By *neighbors*, we refer to nodes reachable in 1 hop, while *peers* refer to the nodes reachable in 1 or multiple hops. The two assumptions can be achieved with help of proactive routing protocols (e.g., OLSR[11]), or having each node periodic sending beacons with reactive routing protocols. Before presenting group member discovery, we first introduce 2-hop bordercasting we use to broadcast within $\mathcal{H}$ hops.

3.1 Group broadcasting

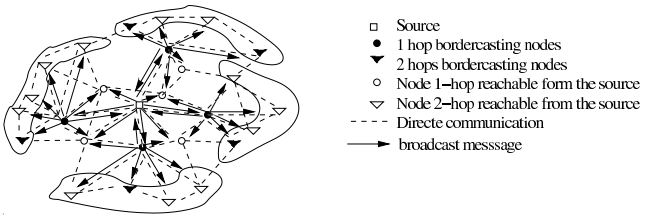

Figure 1: 2-hop bordercasting

For broadcasting within $\mathcal{H}$ hops, we utilize *2-hop bordercasting* (Fig. 1) to reduce the traffic overhead. Succinctly speaking, the broadcast sender, by utilizing two-hop topology information gathered from periodic beacons, selects among (1-hop) neighbors the minimal number of nodes (called *bordercast nodes*) that can cover all the nodes within 2 hops. Only the selected bordercast nodes repeat the process until $\mathcal{H}$ hops are reached. Looping is prevented in bordercasting by embedding a unique message number.

3.2 Group Member Discovery

When a node intends to create some group G^f, it broadcasts via 2-hop bordercasting a *disc* message to its neighbors. This node, who initializes the group member discovery, is named *group initializer*. The *disc* message from a *group initializer* includes (1) the functional and non-functional attributes of the group it intends to create; (2) a unique group id. For example, to create a group with all nodes close to a fixed location, the initializer includes that reference location in the *disc* message.

On receiving a *disc* message, the receiver node carries out conformance checking of local constraints, if it is interested

in joining in. In the meanwhile, if it is a bordercast node assigned by the sender, it needs to determine whether to rebroadcast the *disc* message based on the following rule: it does **not** rebroadcast only if (1) it has no neighbors to rebroadcast (i.e., its only neighbor is the sender of the received *disc*), or (2) it can ascertain that peers further discovered by its rebroadcasting will not meet the group membership constraints (e.g., bounded group).

Contrary to non-bordercast nodes who only need to reply with their own willingness to join the group, a bordercast node needs to wait for the arrival of the responses from all its neighbors before it can respond to its preceder node (either the group initializer or a bordercast node) with a *join* message. The *join* messages include the peers intending to join the group (including itself if it intends to join), along with peer information (e.g., geographical location). Note that even if the bordercast node has received no *join* message indicating willingness to join, it sends back a "negative" *join* message to indicate so.

In the case that there are two nodes initializing group member discovery simultaneously with the same functionality and group constraints, it can be merged by selecting a new initializer from the two (e.g., by id), with the ancient initializer forwarding all *join* messages to the newly selected initializer. If the initializer fails during the initialization phase, the discovery process is restarted.

3.3 Group Initialization

Each group has a node named *group leader* responsible for (1) managing the group dynamics while enforcing group constraints (2) inter-group communication. However, before the group is formed, it is always the *group initializer* who does the group initialization, i.e., acting as the temporary group leader. The group initialization does not start until the initializer has received responses from all its neighbors. One vital part of group initialization is to enforce global constraints. As listed in Table. 1, there are following possible global constraints that need to enforce:

- *Relative_proximity*, which enforces that every two nodes have to be within certain distance. This can be accomplished by calculating the distance from the geographical locations of group members included in *join* messages.

- *Reputation*, which enforces that every two nodes have good reputation towards each other. It is difficult to enforce since reputation towards a peer is considered private in most cases and it is unlikely to provide all reputation values to the leader. However, it can be accomplished by the constraint that the group leader is trustworthy enough to coordinate the interactions between group members, or recommendation from the leader is strong enough to fortify a member's reputation for others. This constraint can be locally enforced during the group member discovery.

- *Size*, which enforces that the total number of group members cannot exceed a certain threshold number. This can be easily achieved by calculating the number of nodes intending to join by *join* messages.

Note that global constraints on group membership may lead to exclusion of peers from the group, even if they have sent *join* messages. For instance, as the consequence of the group size constraint, some nodes can be excluded from the group if the leader has already admitted maximal nodes. For those excluded nodes, they may create a group by starting group discovery process as described above.

In the case where the group is based on signed certificate, the initialization process is complemented with a group key agreement (GKA) protocol for establishing a shared key among the group's members. The key will then be used to encrypt any message subsequently exchanged within the group [4].

To discover leaders of other groups, the leader periodically broadcasts (e.g., to the range of twice the group size) a message to announce its existence to potential nearby leaders of other groups with same functionality and group constraints.

3.4 Group Dynamics Management

Since leaders might drift away, leader might need to be rotated to other members, also to distribute the load of group management among members. We start by illustrating how to select the leader.

Leader Selection. Several criteria have been proposed for leader selection in ad hoc networks, including: (i) *highest degree*, where nodes with the maximum number of neighbors are selected as group leader. This approach has low rate of group leader change, but suffers from low input since the throughput is shared among group members [9]; (ii) *extrema-id* (e.g., [7]), where the node with the smallest or greatest id is assigned as the group leader. The shortcoming of this approach is its bias against nodes with extrema ids; (iii) *node weight* (e.g., [2, 9, 26]), which is an integrated metric for evaluating the suitability of a node as a group leader. It can depend on various factors such as the resource richness and the neighbor count.

Above criteria are essentially based on group attributes as illustrated in Sec. 2. Therefore, it is natural to integrate them to evaluate a node's suitability for acting as a leader, by tuning different weights on different metrics for different scenarios. In particular, in this paper, we consider the following two metrics, which are locally available on each node for election of group leader:

1. Resource richness. Powerful nodes with richer resources are considered more suitable as group leaders. This is because group leader consumes more resources such as battery and computing than other group members. It also improves group performance, because weak group leaders tend to be the bottleneck, e.g., group leader is responsible for forwarding inter-group communication. In particular, we consider the following resource-related attributes: (a) CPU load; (b) memory; (c) battery and (d) bandwidth.

2. Time until now since last time being a leader. This factor contributes to load balance among peers.

The overall weight can be calculated as follows, with relative importance (e.g., w_1, w_{11}) specifically defined for different applications (e.g., by group initializer):

$$
\begin{aligned}
weight &= Resource * w_1 + Elapsed\_time * w_2 \quad (1)\\
w_1 + w_2 &= 1\\
Resource * w1 &= CPU\_Load * w_{11} + memory * w_{12} + \\
&\quad battery * w_{13} + bandwidth * w_{14} \quad (2)\\
w_1 &= w_{11} + w_{12} + w_{13} + w_{14}
\end{aligned}
$$

Note that it can be easily extended to cover other metrics, and metrics need normalization since they are in different units and are not comparable. In case of tie of overall weights, the leader is selected randomly. Leader rotation is started if some node has an overall weight higher than the current leader. This may result from multiple reasons: the current leader's weight keeps falling with longer time of being leader, or a member's resources get richer (e.g., gets power plugged-in). Change of leader requires the transfer between the old and new leaders, the information[1] that is only kept in the leader (e.g., the leaders of other neighbor groups). And the new leader needs to inform the whole group the changes of leader, along with the group information including group member list.

As stated, group management over MANET requires managing mobility-induced changes in group membership, in a manner that is transparent to applications. Since a new node is detected by the network layer, an event-based mechanism can be installed on the leader to invite the new comer to join the group, on the condition that the group is able to accommodate more members. The leaving of a node can be similarly handled by the leader's sending an event-invoked group-wide announcement of updated member list.

4. IMPLEMENTATION

To ease the development of ambient intelligence applications, we have introduced the WSAMI (Web Services for AMbient Intelligence) middleware that supports seamless access to mobile services for the mobile user [15]. WSAMI allows for the abstract specification of Ambient Intelligence applications in the form of composite services, together with their dynamic composition logic according to the environment. WSAMI builds on the Web services architecture[2], and dynamic service composition is carried out by integrating services deployed on mobile, wireless terminals and on the Internet.

4.1 WSAMI Middleware for Mobile Web Services

The WSAMI core middleware can be divided into [15]: (i) a core broker, including the CSOAP-based SOAP container for wireless, resource-constrained devices, and (ii) the Naming&Discovery (ND) service for the dynamic discovery of (possibly mobile) services that are available in the local and wide area, according to network connectivity and available resources.

We have carried out a number of experiments to investigate the performance of our lightweight Web services platform, and in particular compare it against existing Web services platforms, e.g., traditional middleware platforms being considered for the mobile environment [15]. Experiments show that the overhead related to the functions handling user mobility (i.e., dynamic service composition relying on service discovery and connector customization for enforcing QoS) is comparable to the cost of base Web services access.

Our prototype is being used for the implementation of demonstrator applications in the field of ambient intelligence, as part of the Ozone IST project[3]. It is in particular being extended with a number of value-added middleware services for mobility management as well as for enabling intelligence-aware interfaces.

4.2 Group Service in WSAMI

The ND service discovers instances of Web services implementing a given WSAMI interface within one-hop ad hoc networks in our current prototype. The Group service then implements *DiscPeer* on top of the ND service, which takes as input the URI of the WSAMI document defining the function of the group, and other group constraints. *DiscPeer* utilizes the ND service to multicast *Disc* messages and to receive *Join* messages for realizing group service as detailed in Sec. 3.

Note that the functional specification of any group not only includes the group's specific functional operations, but also operations for handling membership constraints (e.g., conformance checking) and acting as leader, since both types of operations must be supported by the group members. The Group service is deployed on any node taking part in group management, and offers a *RegGroup* (resp. *UnRegGroup*) function for registering (resp. canceling) participation to a given group.

Note that our group service is different from WS-ServiceGroup [12], which is a recent proposal to standardize the terminology, concepts, message exchanges, WSDL and XML needed to express the aggregation of Web services and resources, in the following two most important aspects: (1) Group service of WSAMI aims to facilitate grouped nodes to share services, instead of to define the aggregate interface of services; (2) Group service of WSAMI defines membership via various constraints as illustrated in Sec. 2, instead of service interfaces.

4.3 Group Service Assessment

In order to evaluate our group management, we conduct experiments of group initialization of from 2 to 7 terminals, which are either laptops or PDAs. PDAs are Compaq iPaq H3970 with Intel XScale 400MHz processor and 64MB built-in RAM, running on familiar v0.7[4] and J2ME Personal Profile. Laptops are equipped with 500 MHz PIII CPU and 192 MB RAM, running on RedHat 9.0 and J2SE 1.4.

We measure *group CPU time*, which is the dedicated CPU time during the group member discovery and initialization. We compare the CPU consumption with and without the group leader calculating a group key with group key agreement[5] (GKA). We thus measure the average CPU time for (i) a laptop when it acts as a leader or a nonleader (Fig. 2); (ii) a PDA when it acts as a leader or a nonleader (Fig. 3).

From the figures, it can be observed that (1) group key agreement (GKA) is expensive by inducing almost twice the CPU time; (2) a laptop consumes considerably less CPU

[1] In the case of abrupt leaving of leader, the information need to be gathered again.

[2] http://www.w3.org/2002/ws/arch

[3] http://www-rocq.inria.fr/arles/download/ozone/

[4] http://familiar.handhelds.org/

[5] Readers are referred to [4] for details.

time, which validates our choice of leader based on resource richness; (3) leaders indeed consume much more CPU time than ordinary nodes, which favors rotation of leaders among members. The rotation assures fairness among group members, especially when all members are resource-lite nodes.

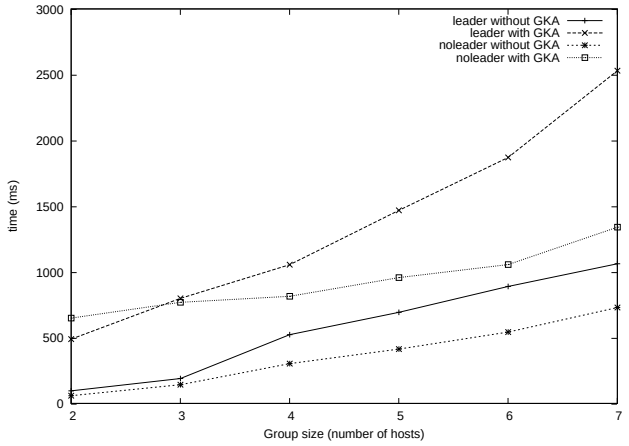

Figure 2: *CPU_time of a laptop*

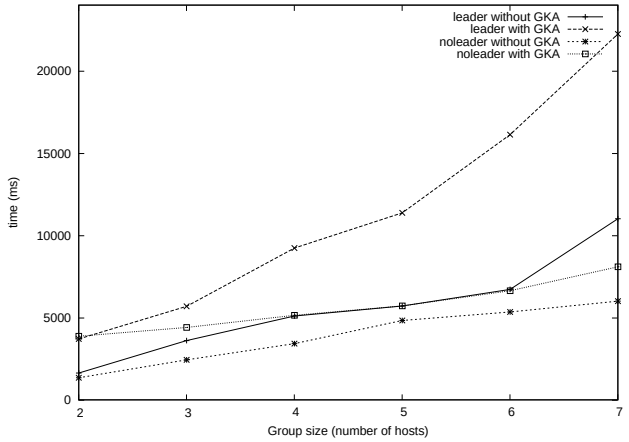

Figure 3: *CPU_time of a PDA*

5. CASE STUDY

This section discusses the implementation of one specific group management over MANET supporting scalable and efficient service discovery.

5.1 Scalable Service Discovery

To support scalable service discovery, a group called $G^{ServDisc}$ is defined with an associated membership constraint – $Bounded(G^f, \mathcal{H})$, i.e., all member nodes are reachable from the leader node within $\mathcal{H}$ hops. A service discovery process can be divided into two stages: local and global service discovery.

5.1.1 Local Service Discovery

Leader of a $G^{ServDisc}$ group has one specific functionality for service discovery: caching the description of the services available in the group. A service seeker thus always demands the group leader for a service query. When a leader receives a service query (including the service's WSDL-based functional interface and QoS parameters), it checks its cache content and sends back a hit message embedding the matching service description(s) if the group has the requested service. In order to provide users with better quality of service and balance the load among service providers, leaders select among multiple service provider candidates, if there exist, the provider that: (i) provides the service that best matches the query in terms of both functional interface and provided QoS, and (ii) is the less loaded. If the leader does not cache the description of the requested service, it initiates a global discovery, detailed as follows.

5.1.2 Global Service Discovery

In order to locate services available in the overall MANET in an efficient manner in terms of both response time and generated traffic, leaders need to interact with the other leaders that most likely cache the requested service description. This is achieved through (1) group profile, i.e., summary of available services in the group (2) cooperation among leaders.

We use Bloom filters [5] to summarize the available services in the group, i.e., the set of cached WSDL-based service descriptions. Bloom filters are an efficient way of describing sets because of its compactness and independence of any predefined keywords characterizing services. Succinctly speaking, with Bloom filter, we are able to know with high possibility that if a service is provided within a group by simply checking if certain bits of the group service summary (i.e., a vector of certain length) are set to 1[6].

As in our group management service, a leader advertises its group profile to other leaders by periodically broadcasts over $\mathcal{N} = 2 \times \mathcal{H}$ hops (i.e., 2 times the group size), using 2-hop bordercasting. For a newly elected leader, after it receives a beacon from a nearby leader, it demands from that leader the list of nearby group leaders, along with their profiles (e.g., Bloom filter and host capacity). In case of no beacon from any other leader, the leader broadcasts a network-wide advertisement using 2-hop bordercasting. By this means, every leader is able to obtain a global view of leaders and available services.

For less traffic overhead, our solution does not update the group profiles upon every change of available services. Instead, a request for updated profile of a leader is reactively demanded by a peer leader when the false misses (i.e., returns a *Hit* message while there is actually no matching service) percentage reaches a threshold value. When a leader receives a profile request, it sends either the complete bit vector or the list of bits that changed, depending on their respective storage size.

Therefore, as shown in Fig. 4, when a group member A seeks a service, it sends a query message to its leader B (message 1). B looks up its cache and sends back a hit message to A if it stores the description of the requested service. Otherwise, B looks up the Bloom filters of other nearby leaders, and send a query message to the M ($M < Max$) other leaders that are likely to cache the description of the target service (message 2 and 3). C or D sends back a

[6]For more details, please refer to [24]

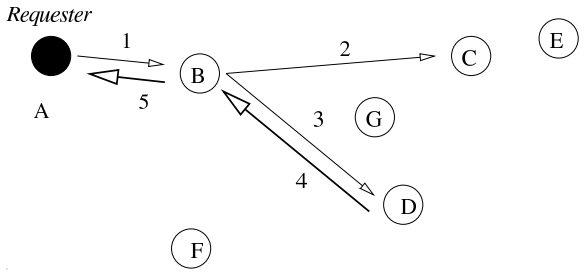

Figure 4: Global discovery

hit message (message 4) if the requested message is stored. And if B does not receive any *Hit* message, it sends back to A a *Miss* message (message 5).

5.1.3 Performance

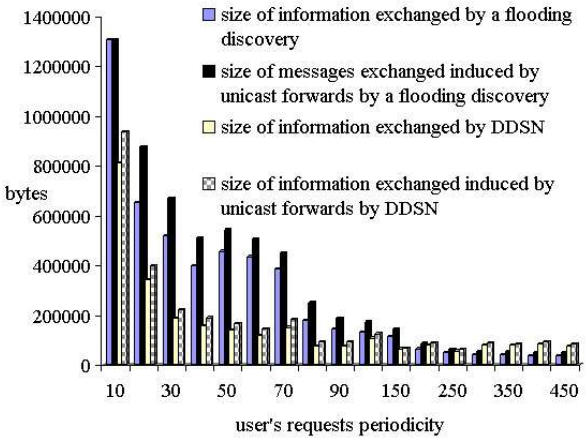

Figure 5: Traffic at initial sender, compared with decentralized pull-based discovery

The NS-2 simulator[7] is used to evaluate the performance of our service discovery protocol in terms of the cost of leader deployment at initialization time. Simulations are run for a duration of 500s. Each mobile terminal has a transmission range of 200m, and assumes a random waypoint mobility model with an average speed of 1m/s. OLSR [11] is used as the underlying multi-hop routing protocol on top of the IEEE 802.11 MAC protocol. We consider a MANET of N nodes over a surface 600m $\times$ 600m. A group is bounded by $\mathcal{H} = 1$ hop. In the simulation, 10% of the mobile nodes are service providers that offer $E = 2$ services, and each client node periodically seeks a service among the $E \times N \times 10\%$ services that are provided in the network.

To evaluate the performance, we measure (i) the traffic generated by the service seeker, and (ii) the traffic resulting from message forwarding. We compare our protocol with a decentralized, pull-based service discovery protocol that floods the network on demand, i.e., service query at a node is broadcasted using 2-hops bordercasting over the network. As shown in Fig. 5 and Fig. 6, Our service discovery protocol

[7]http://www.isi.edu/nsnam/vint

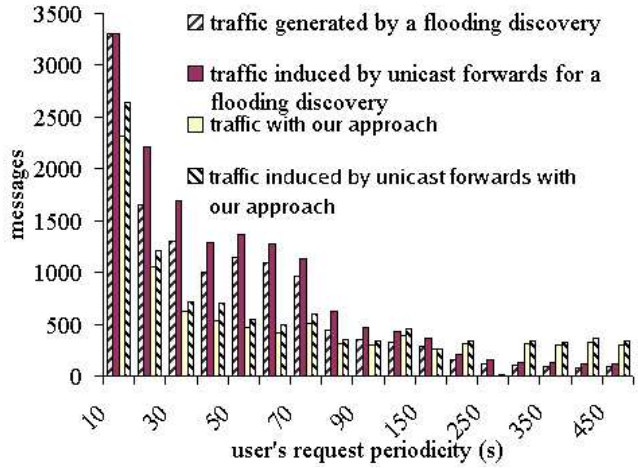

Figure 6: Traffic due to message forwarding, compared with decentralized pull-based discovery

introduces a fixed cost corresponding to the deployment and updates of directories. This explains why our protocol is outperformed when there are very sporadic requests (i.e., as shown in the figures, one request every more than 200 seconds). Our protocol surpasses the decentralized, pull-based one when the rate of user request reaches a certain threshold. Note that with our protocol, the node density has low impact on the generated traffic due to the deployment of groups, since it does not affect the number of groups. By contrast, the performance of the flooding protocol suffers proportionally with the increase of node density.

6. CONCLUSION

Group management poses as a key middleware functionality for assisting the development of applications over mobile wireless ad hoc networks. Group management manages a dynamic network on top of which the application implements certain functions and achieve certain non-functional properties. Group management over wireless ad hoc networks has actually given rise to various studies concentrating on specific applications. However, a distinctive set of key attributes may be identified for groups, which can be exploited to design a generic group service that is customizable by applications.

This paper presents the design of such a generic group management service. At first, we introduce key attributes for group management over MANET, in particular based on applications in published literature. Those attributes amount to membership constraints related to location, group scale, trustworthiness and supported QoS of group members. We then present a group service that is generic with respect to membership constraints, and realizes three basic functions: group member discovery, group initialization and group dynamics management. Implementation of the generic group service is concretized in the context of the WSAMI middleware aimed at mobile distributed computing, which is based on the Web services Architecture. Finally, we present an example of group management built on our generic group service, which supports scalable service discovery in MANET. Our future work includes implement-

ing group applications based on current middleware group service, such as collaborative editing session and group conferences.

7. REFERENCES

[1] A. D. Amis, R. Prakash, T. H. P. Vuong, and D. T. Huynh. Max-Min D-cluster formation in wireless ad hoc networks. In *Proc. of IEEE INFOCOM*, 2000.

[2] S. Basagni. Distributed clustering for ad hoc networks. In *Proc. of Int'l Symp. Parallel Architectures, Algorithms, and Networks*, 1999.

[3] C. Basile, M.-O. Killijian, and D. Powell. A survey of dependability issues in mobile wireless networks. Technical report, LAAS CNRS, France, February 2003.

[4] R. Bhaskar. Group key agreement in ad hoc networks. Technical Report 4832, INRIA-Rocquencourt, France, 2003.

[5] B. Bloom. Space/time trade-offs in hash coding with allowable errors. *CACM*, 13(7), 1970.

[6] M. Boulkenafed and V. Issarny. Middleware service for mobile ad hoc data sharing, enhancing data availability. In *Proc. of the 4th ACM/IFIP/USENIX Int'l Middleware Conf.*, June 2003.

[7] M. Boulkenafed, D. Sacchetti, and V. Issarny. Using group management to tame mobile ad hoc networks. In *Proc. of the IFIP TC8 Working Conf. on Mobile Info. Systems*, 2004.

[8] L. Briesemeister and G. Hommel. Localized group membership service for ad hoc networks. In *Proc. of the Int'l Workshop on Ad Hoc Networking*, 2002.

[9] M. Chatterjee and D. Das, S. K.and Turgut. WCA: A weighted clustering algorithm for mobile ad hoc networks. *Cluster Computing*, 5(2), 2002.

[10] G. Chockler, I. Keidar, and R. Vitenberg. Group communication specifications: A comprehensive study. *ACM Computing Surveys*, 33(4), December 2001.

[11] T. Clausen and P. Jacquet. Optimized link state routing protocol. IETF RFC 3626, 2003.

[12] S. Graham, T. Maguire, J. Frey, and N. e. a. Nagaratnam. Web services service group - specification. Version 1.0, 2004.

[13] X. Hong, M. Gerla, G. Pei, and C. Chiang. A group mobility model for ad hoc wireless networks. In *Proc. of the ACM MSWiM*, 1999.

[14] B. Hugues and V. Cahill. Towards real-time event-based communication in mobile ad hoc wireless networks. In *Proc. of the 2nd Int'l Workshop on Real-time LANs in the Internet Age*, July 2003.

[15] V. Issarny, D. Sacchetti, F. Tartanoglu, F. Sailhan, R. Chibout, N. Levy, and A. Talamona. Developing Ambient Intelligence Systems: A solution based on Web services. *Journal of Automated Software Engineering*, 12(1), 2005.

[16] J. Liu and V. Issarny. Enhanced reputatation mechanism for mobile ad hoc networks. In *Proc. of 2nd Int'l Conf. on Trust Management*, 2004.

[17] J. Liu and V. Issarny. QoS-aware service location in mobile ad hoc networks. In *Proc. of the 5th IEEE Int'l Conf. on Mobile Data Management*, 2004.

[18] J. Luo, P. Eugster, and J.-P. Hubaux. PILOT: Probabilistic lightweight group communication system for mobile ad hoc networks. *IEEE transactions on mobile computing*, 3(2), 2004.

[19] J. Luo, J.-P. Hubaux, and P. Eugster. PAN: Providing reliable storage in mobile ad hoc networks with probabilistic quorum systems. In *Proc. of the 4th ACM MobiHoc*, 2003.

[20] R. Meier, M.-O. Killijian, R. Cunningham, and V. Cahill. Towards proximity group communication. In *Proc. of the 1st Workshop on Middleware for Mobile Computing*, 2001.

[21] A. Meissner and S. B. Musunoori. Group integrity management support for mobile ad hoc communities. In *Proc. of the 1st Workshop on Middleware for Pervasive and Ad Hoc Computing*, 2003.

[22] D. Powell. Group communication. *CACM*, 39(4), 1996.

[23] G.-C. Roman, Q. Huang, and A. Hazemi. Consistent group membership in ad hoc networks. In *Proc. of ICSE*, 2001.

[24] F. Sailhan and F. Issarny. Scalable service discovery in MANET. In *Proc. of IEEE Int'l Conf. on Pervasive Computing (PerCom)*, 2005.

[25] S. Sha, K. Chen, and K. Nahrstedt. Dynamic bandwidth management for single-hop ad hoc wireless networks. In *Proc. of IEEE Int'l Conf. on Pervasive Computing (PerCom)*, 2003.

[26] S. Vasudevan, J. Kurose, and D. Towsley. Design and analysis of a leader election algorithm for mobile ad hoc networks. In *Proc. of 12th IEEE ICNP*, 2004.