REDMAN: REplication in Dense MANET

Paolo Bellavista, Antonio Corradi, Eugenio Magistretti
Dip. Elettronica, Informatica e Sistemistica - Università di Bologna
Viale Risorgimento, 2 - 40136 Bologna - ITALY
Phone: +39-051-2093001; Fax: +39-051-2093073
{pbellavista, acorradi, emagistretti}@deis.unibo.it

Abstract

The increasing diffusion of wireless-enabled portable devices is pushing towards service provisioning over dense Mobile Ad-hoc NETworks (MANETs), i.e., limited spatial regions, such as shopping malls, railway stations and airports, where a high number of mobile wireless peers autonomously cooperate, without the need for statically deployed network infrastructures. Dense MANET deployment scenarios can take advantage of high node population to replicate common-interest resources to increase their availability, by overcoming the unpredictable node exit from the dense region. The paper proposes a lightweight middleware, called REDMAN, to manage, retrieve and disseminate replicas of data/service components made available by cooperating nodes in a dense MANET. The paper presents the REDMAN novel solutions to determine the nodes belonging to dense MANETs and to dynamically elect a suitable replica manager node. In addition, original proposals for dissemination of replica placement information and innovative dual strategies of replica retrieval are discussed. Experimental results show that REDMAN solutions are lightweight and effective in dense MANET scenarios with almost constant node density and even high node mobility.

1. Introduction

The mass market of wireless devices suggests novel service deployment scenarios where there are no constraints on device mobility and distributed applications are the result of impromptu collaborations among wireless peers. In these scenarios, not only wireless nodes should have location-aware access to distributed resources without requiring static knowledge about their execution environment, but also resources should be permanently available, not depending on node movement, disconnection, and battery shortage [1].

Some research activities are investigating MANET-specific solutions to bind/rebind to newly discovered distributed resources, thus enabling wireless clients to automatically redirect requests

to service components also by considering their mutual location [2, 3]. On the contrary, the idea of increasing the availability of MANET applications by replicating data and service components close to their clients is still at its very beginning. Only few state-of-the-art proposals have started to face the very challenging issue of resource replication in mobile environments, with the goal of increasing access probability and effectiveness [4]. Most investigations have recently addressed the issue of information availability in cellular and infrastructure-mode IEEE 802.11 networks. However, in that context, it is possible to exploit the fixed part of the network infrastructure, e.g., to store and replicate personal data at highly available servers on wired stable links. The most critical issue in infrastructure-based scenarios is to properly manage user disconnections during update operations on shared data, possibly by automatically handling the reconciliation of multiple modified copies [5, 6].

Due to the complete lack of a static support infrastructure, the effective replication of data and service components in MANET dynamic environments is a hard challenge, which requires rethinking and significantly modifying traditional approaches to replication. So far, the research has mainly focused on replication to ensure data availability in case of network partitions. Many proposals assume that wireless nodes are aware of their physical position, e.g., by imposing the constraint of hosting Global Positioning System hardware at any participant [7]. Other research activities aim at answering strict requirements about replica synchronization and consistency, and therefore impose a heavy overhead, in terms of both network traffic and requested time to ensure the consistency of all replicas [8].

We claim that the management of data/service component replicas is a very hard task to perform in an effective and lightweight way when dealing with general-purpose MANETs and with strict consistency requirements. Therefore, we focus on a specific deployment scenario of increasing relevance for the service provisioning market, called dense MANET in the following. We use the term *dense MANET* to indicate a MANET that:

- includes a large number of wireless devices located in a relatively small area at the same time, e.g., as it will probably happen in the near future in shopping malls, airport waiting rooms, and university campuses;
- has a node density (the average number of wireless nodes at single-hop distance from any dense MANET participant) that is almost invariant during long time intervals.

The primary idea behind our approach is to provide a lightweight middleware solution to instantiate, disseminate, and manage replicas of common-interest resources (shared data and middleware/service component code) among wireless nodes in the dense MANET, so that any client could access at least one replica in its vicinity at any moment and anywhere while in the dense region. Resource replicas should be effectively placed in order to enable efficient retrieval solutions, while imposing limited communication overhead both for replica placement and retrieval. Resource accessibility should be maintained notwithstanding the unpredictable movements of wireless devices (with their hosted resource replicas) inside/outside the dense MANET and their temporary loss of network connectivity.

We claim that resource replica management is a very hard task to perform in an effective and lightweight way when dealing with general-purpose infrastructure-free wireless networks and when dealing with strict consistency requirements over wide-scale provisioning environments [1]. For this reason, we have decided to specifically focus on dense MANETs, where the assumption of relatively high and constant node density (nodes can unpredictably move in/out the dense region, even with high frequency, but their overall number in the dense MANET does not change significantly over large time intervals) permits to exclude the possibility of network partitioning and sub-network merging at provision time. However, limiting the investigation to dense MANETs is not too restrictive: in fact, it is valid in most provisioning environments where there is commercial interest in offering distributed services, e.g., for entertainment, public utility, and advertisement.

Given the above considerations, we have worked to design and implement a middleware, called REDMAN (**RE**plication in **D**ense **MAN**ETs), that transparently disseminates, manages, and retrieves resource replicas among cooperating nodes in dense MANETs. Transparently from the point of view of application developers, the REDMAN middleware works to maintain, with a lightweight and lazily consistent approach, the desired resource replication degree within the dense MANET, independently of possible exits of replica-hosting nodes from the dense region. REDMAN addresses the primary dense MANET challenging issues, i.e., the determination of nodes belonging to the dense region and the sensing of nodes entering/exiting the dense region. Moreover, it proposes novel strategies for the dissemination of common-interest resources and their dynamic and prompt retrieval, in a highly decentralized way via original protocols specifi-

cally designed for dense MANETs. Let us observe that REDMAN does not impose the availability of any Global Positioning System (GPS) equipment installed on participating devices, thus enabling also the cooperation of terminals with very strict constraints on local resources, especially on battery power consumption.

At the moment, REDMAN addresses replica management for read-only resources (files with still/moving images, audio tracks, hypertexts) or, anyway, with no consistency requirements in the case of replica modification. Let us observe that this kind of replica dissemination is suitable not only to inject entertainment/advertisement-related data in dense MANETs, but also to dynamically distribute the code of needed support/application components, e.g., driver updates, format-specific multimedia players, and game clients.

REDMAN operates at the application level because several replica management decisions, such as the suitable replication degree depending on differentiated resource criticality, are typically at this abstraction layer [9]. Working at the application level also simplifies portability over heterogeneous connectivity technologies and routing protocols. In addition, REDMAN performs replica management transparently from the point of view of service developers/administrators, who only have to indicate the criticality of the shared resources involved. Moreover, REDMAN has been specifically designed for battery/memory-constrained devices (PDAs, smart phones, …), which typically cannot host positioning hardware and cannot store all needed data and service components in their local memory permanently; REDMAN simplifies the mutual interactions of limited portable devices to enable collaborative service provisioning in dense MANETs.

The rest of the paper is organized as follows. Section 2 overviews basic REDMAN concepts and facilities. Each following section focuses on a REDMAN facility: Section 3 describes the protocols for dense MANET identification and manager election; Sections 4 and 5 deeply investigate replica dissemination and retrieval strategies; Section 6 details the behavior of the RDM facility. Then, Section 7 reports extensive simulation results about REDMAN performance, which show the effectiveness and the limited overhead of the proposed solutions, by confirming the suitability of the approach even in conditions of high node mobility. Related work, on-going research, and conclusions end the technical report.

2. The REDMAN Middleware

REDMAN addresses the issue of disseminating replicas of resources of common interest in dense MANETs, independently of unexpected node exit from the dense region, e.g., due to node mobility and battery shortage. More formally, a dense MANET is defined as the set of MANET nodes DM(n) = $\{d_0, \ldots, d_{N-1}\}$, where i) $\forall j \in [0, N-1]$ d_j has at least *n* neighbors at single-hop distance, and ii) the spatial node density in the area where DM(n) nodes are is almost constant with regards to time. Given a resource with a desired replication degree *r*, REDMAN is in charge of instantiating and distributing *r* replicas of it, and of maintaining the *r* replication degree notwithstanding the changes in the composition of the DM(n) set.

In particular, to suit resource-limited nodes, REDMAN proposes dense MANET-specific lightweight solutions. REDMAN decides not to guarantee the strict any-time consistency of the replication degree of shared resources. Instead, it employs reactive strategies to counteract the reduction of replicas in DM(n) when resource delegates either fail or leave the network. In addition, to reduce the overhead and the complexity of distributed replica management, REDMAN currently manages the replication of read-only resources, thus permitting to exclude heavy and expensive operations for possible reconciliation of concurrently updated resource replicas. Dealing with read-only resources is sufficient for guaranteeing the availability of a large class of services of primary interest in MANETs, as pointed out in the introduction. General-purpose protocols for replica reconciliation in traditional wired systems are not suitable, even in adapted forms, for dense MANETs, due to their relevant overhead and connectivity requirements [10].

In addition, we claim the suitability of providing REDMAN facilities at the application level to improve flexibility, configurability, and portability over different MANET communication solutions, and to hide lower layer implementation details from application developers. Developers of services for dense MANET should only provide each service component with metadata to describe the shared resource and to suggest the suitable replication degree depending on application-specific resource criticality. Clients in the dense MANET transparently perform discovery and retrieval operations, i.e., resource replication does not affect at all their application logic.

Figure 1 depicts the REDMAN middleware architecture organized in four facilities: Dense MANET Configuration (DMC), Replica Distribution (RD), Replication Degree Maintenance

(RDM), and Resource Retrieval (RR). In the following, we will describe each facility and accurately detail the original REDMAN algorithms.

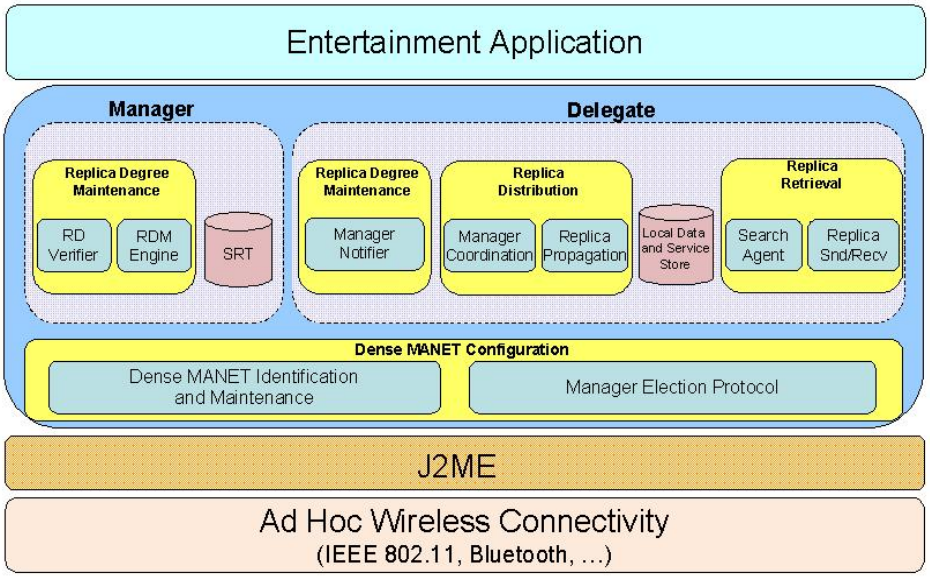

Figure 1. The REDMAN middleware architecture of facilities.

3. Dense MANET Configuration in REDMAN

DMC is the REDMAN facility in charge of determining which nodes belong to the dense MANET and play the role of replica managers. These functions are crucial for the realization of all other REDMAN facilities and require original solutions that fit the specific characteristics of the dense MANET deployment scenario.

3.1. Dense MANET Identification

REDMAN proposes an original solution to dynamically determine the nodes that currently participate to a dense MANET. The primary idea is not to maintain a centralized, global, and always up-to-date vision of all the nodes and of their network topology, but to design a simple, lightweight, and decentralized protocol where any node autonomously determines whether it belongs to the dense MANET. One node is in the dense MANET DM(n) only if the number of its neighbors, i.e., the nodes at single-hop distance, is greater than *n*. Each node autonomously dis-

covers the number of its neighbors by exploiting simple single-hop broadcast discovery messages.

In more detail, at any time one REDMAN node can start the process of dense MANET identification/update; in the following, we will call that node the *initiator*. The initiator starts the protocol by broadcasting a discovery message that includes the number of neighbors required to belong to the dense region. When receiving this message, each node willing to participate replies by forwarding the message to its single-hop neighbors, if it has not already sent that message. After a specified time interval, any node autonomously checks whether the number of received discovery messages is greater than the specified number of neighbors to belong to the dense region, and autonomously decides whether it belongs to the dense MANET. Let us observe that discovery broadcasts could provoke packet collisions (broadcast storm issue [12]): to avoid this problem, any REDMAN node defers node broadcasts of a random time interval. Notwithstanding this introduced random delay, the time needed to complete the dense MANET identification protocol is limited and largely acceptable; in fact, it shows a linear dependence on the dense region diameter, not on the number of its participants.

Let us consider the network example in Figure 2; the sequence diagram of the dense MANET identification protocol triggered by node I is reported in Figure 3. The first discovery message (including the number of neighbors required to belong to the dense MANET - 2) reaches nodes A, B, C, and NP1, as illustrated in Figure 3 phase (a). These nodes, after a random interval, rebroadcast the message unchanged (first NP1, then A, B, and C – phase (b) in the figure) and set a timeout to determine whether they belong to the dense region. Since I, A, B, and C receive 4, 6, 3, and 4 messages (phase (c) in the figure) before timeout expiration, they realize they either belong or not to the dense MANET, as shown in phase (d) in the sequence diagram.

Since dense MANET nodes can move after the identification process, the proposed algorithm includes a lightweight lazy-consistent maintenance phase. Nodes periodically exchange Hello packets; each node receiving a Hello message records its source in a table entry, with an associated timeout; next Hellos received from the same source restart a new timeout. Dense MANET nodes periodically check whether their table entries are still valid; if an entry has expired, the node removes it from the table, and verifies whether the condition for dense MANET belonging still holds.

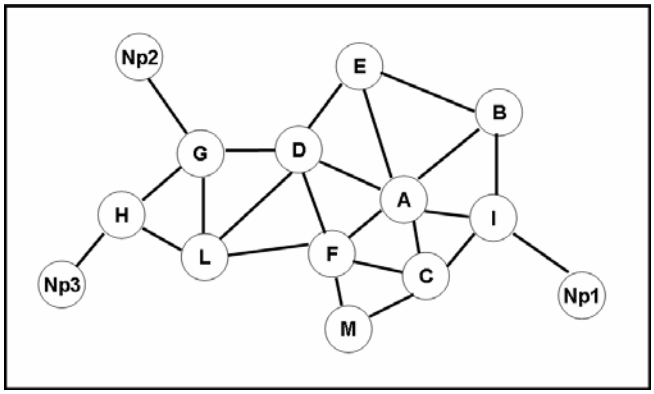

Figure 2. An example of node placement and connectivity.

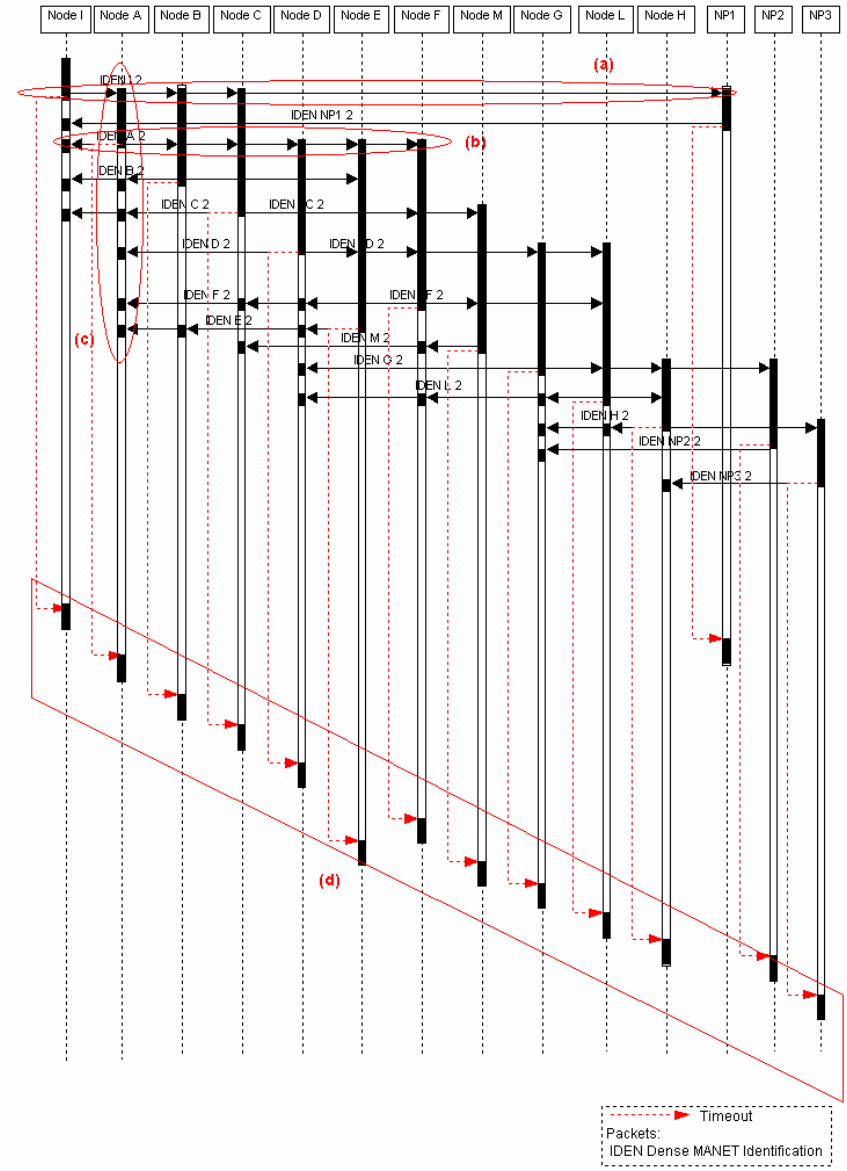

Figure 3. Sequence diagram of the dense MANET identification protocol.

3.2. Replica Manager Election

REDMAN proposes an original, lightweight, and decentralized protocol to elect the replica manager. To reduce the communication overhead (both in terms of dissipated energy and elapsed time) for the manager to get in touch with all dense MANET participants, it could be useful to choose a node located in a topologically central position. More precisely, REDMAN aims at electing one node that minimizes the number of hops required to reach its farthest nodes belonging to the dense MANET. In addition, the proposed protocol for replica manager election has not the goal of finding the optimal solution with regards to the above minimization criterion: REDMAN exploits some heuristics to relevantly limit the election overhead while achieving a good quality manager designation. The proposed solution explores, as manager candidates, only a subset of nodes in the dense MANET, called Investigated Nodes (INs). To avoid the overhead of exhaustive search, REDMAN adopts an exploration strategy that limits the IN number, by only choosing successive INs to get closer to the dense MANET topology center at each exploration step, thus decreasing the distance of each successive IN from farthest participants.

Therefore, a primary issue is how INs can autonomously determine the direction towards the dense MANET topology center by exclusively exploiting information about MANET farthest nodes. To this purpose, the adopted guideline is to explore the nodes located along the direction of farthest nodes from previously considered INs. In fact, by moving toward that direction, each protocol step considers an IN that is placed one-hop closer to the previously identified farthest nodes; therefore, the IN distance from farthest nodes tends to decrease and to converge close to the best solution. Figure 4 shows an example of application of that guideline. The first step of the protocol considers node I: its farthest node is H, located at 4-hop distance; so, I is tagged with the value of that distance (I^4 in the figure). Then, the REDMAN manager election protocol considers A because it is the first node along the path from I to H: A's farthest node is H, at 3-hop distance (A^3). At the next iteration, the protocol explores node D and elects it as the replica manager; in fact, D can reach any other node in the dense MANET with a maximum of two hops.

After having informally introduced the above main guidelines of the protocol, let us now better detail how the manager election works. The protocol considers the initiator as the first IN; then, it re-iterates the farthest node determination process (see Section 3.2.2) to evaluate other promising nodes, until it reaches a satisfying solution. Each IN executes three operations: i) it

determines the number of hops of the shortest paths connecting it to any farthest node in the dense MANET (the maximum of those hop numbers is called *INvalue*); ii) it identifies its neighbors located in the direction of its farthest nodes (*forwarding neighbors*); iii) it autonomously chooses the next IN among all the unexplored forwarding neighbors of already explored minimum-valued INs. To take devices heterogeneity into account, the algorithm promotes the exploration only of nodes suitable to carry manager tasks, e.g., with sufficient memory and expected battery life. The manager election protocol ends when either the REDMAN heuristic criterion presented in Section 3.2.1 determines there are no more promising nodes, or the *current INvalue = MinInt((worst explored INvalue)/2)*, where *MinInt(x)* returns the least integer greater than *x*. Since REDMAN considers bi-directional links among MANET nodes, when the above equation is verified, it is easy to demonstrate that REDMAN has reached the optimal solution for the manager election.

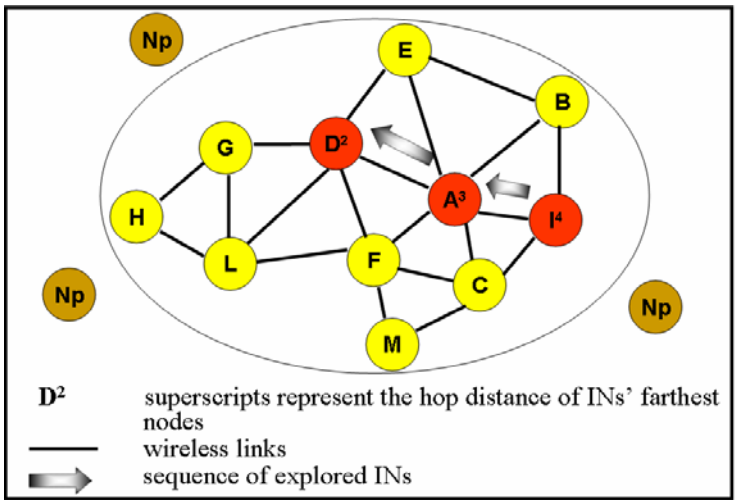

Figure 4. REDMAN exploring the sequence of INs I→A→D.

Figure 5 shows the sequence diagram of the election process example illustrated in Figure 4. For the sake of clarity, the farthest nodes identification protocol is not represented. The figure only shows Dist messages, incoming from neighbors placed in the direction of farthest nodes, e.g., node I receives two Dist messages from A, claiming the 2-hop distance of node E (a) and the 4-hop distance of node H (c), and one from C, claiming the 2-hop distance of node M (b). Let us point out that D stops the election process because it verifies the condition discussed above

(d), i.e., its *INvalue* (2) is equal to the half of the *worst explored INvalue* (in fact, I's *INvalue* is 4).

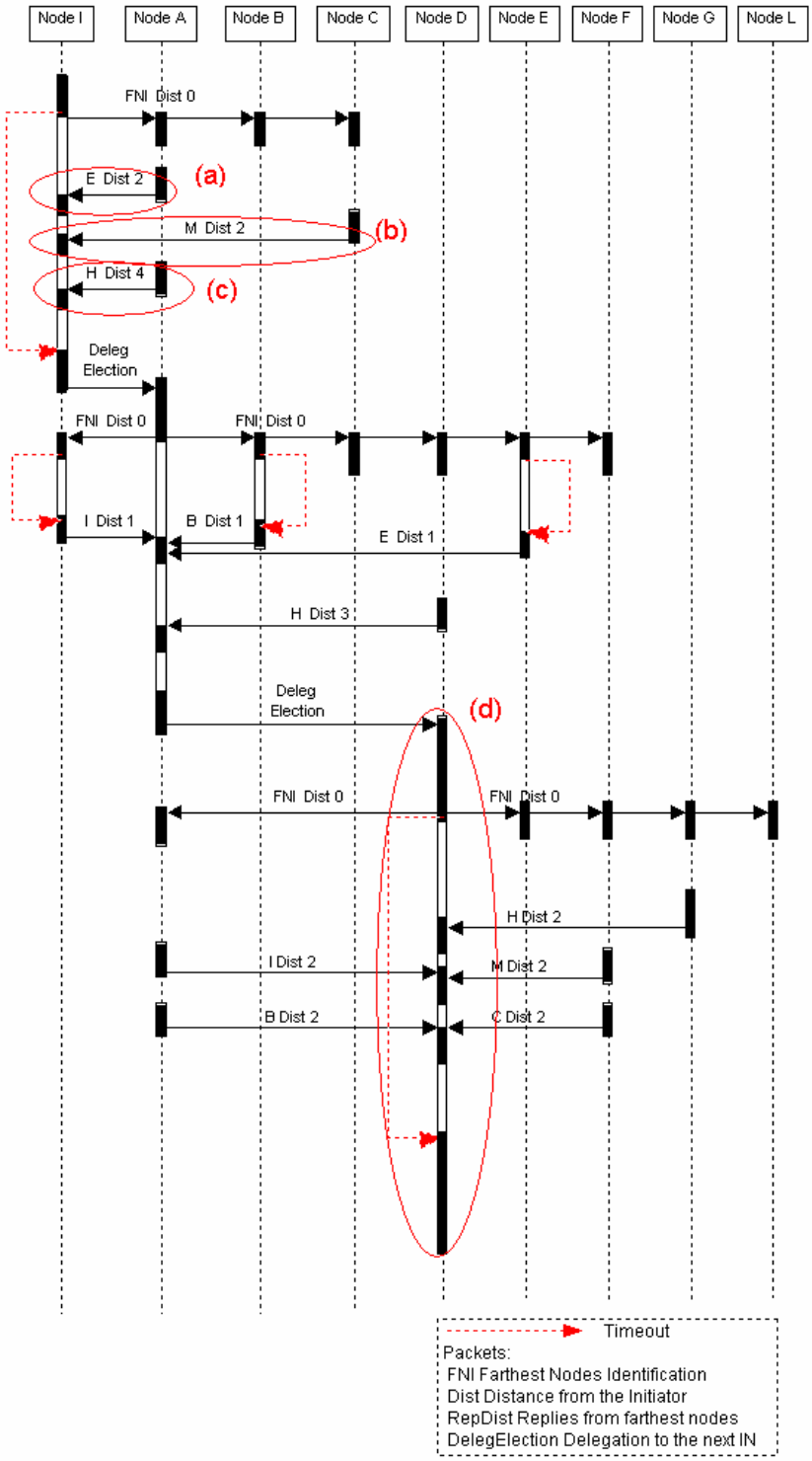

Figure 5. Sequence diagram of the election protocol.

The following subsections detail the adopted heuristic to limit the number of explored INs and the exploited solution to determine, given a node, its farthest nodes in the dense MANET.

3.2.1. Heuristic-based Overhead Reduction

To reduce the overhead due to a large number of re-iterations of the manager election protocol, REDMAN exploits a heuristic approach that has experimentally demonstrated to reach high quality solutions, close to the optimal choice of the manager (see Section 7.1.2).

To improve the flexibility and adaptability of the REDMAN election strategy to the peculiar characteristics of the dense MANET where it is deployed, REDMAN provides two tuning parameters that enable dense MANET administrators to trade between the quality of the manager election protocol and its performance. The first parameter, *DesiredAccuracy*, permits the initiator to tune the approximation considered acceptable for the election solution already found (see Figure 6).

```
exploredList = ø; forwarderList = ø;
bestNode = ø; bestValue = MAX; worstValue = 0;
unexploredList = Initiator;
while (unexploredList != ø) {
    IN = Head(unexploredList);
    INValue = DistanceFromFarthest(IN);
    exploredList = exploredList U IN;
    unexploredList = unexploredList – IN;
    forwarderList = GetPromisingNeighbors(IN);
    forwarderList = forwarderList – exploredList;
    if ((INValue == MinInt(worstValue/2) || (INValue <=
    worstValue * desired_accuracy)) exit;
    if (INValue < bestValue) {
        bestNode = IN; bestValue = INValue;
        consecutiveEqualSolutions = 0;
        unexploredList = forwarderList ; }
    if (INValue > worstValue) { worstValue = INValue;
        if ((bestValue == MinInt(worstValue/2) || bestValue
        <= worstValue * desired_accuracy)) exit; }
    if (INValue == bestValue) {
        consecutiveEqualSolutions++ ;
        if (conecutiveEqualSolutions == max_consecutive_
        equal_solutions) exit;
        unexploredList = unexploredList U forwarderList ; }
    } Print(bestNode)
```

Figure 6. Pseudo-code of the REDMAN manager election protocol.

The second parameter, *MaxConsecutiveEqualSolutions*, is introduced by observing that, when the REDMAN election protocol approaches the optimal solution, it often explores other candidate nodes without improving the current best IN value. For each explored solution equal to the

current best, REDMAN increases a counter; the counter resets when REDMAN finds a new solution outperforming the old best. The adopted heuristic stops the iterations when the counter reaches *MaxConsecutiveEqualSolutions*. Figure 6 shows the pseudo-code of the election protocol, also represented by the flow chart in Figure 7.

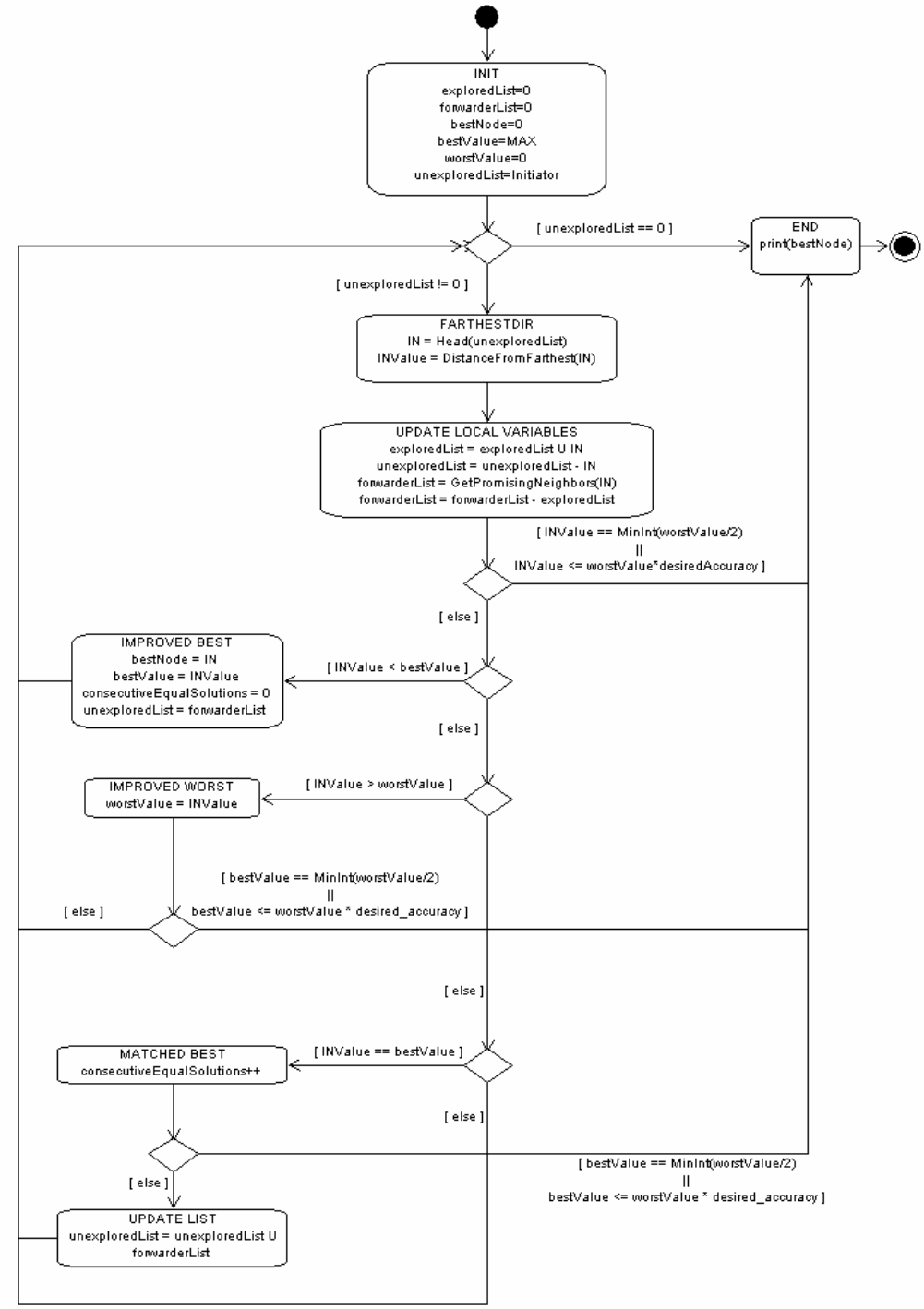

Figure 7. Flow chart of the REDMAN manager election protocol.

3.2.2. Determination of Farthest Nodes

REDMAN proposes a simple broadcast-based strategy to detect the lengths of the shortest paths connecting the current IN to the farthest participants in the dense MANET. The current IN starts the protocol by broadcasting a farthest node determination message including a counter initialized to 0. Every node receiving that message and belonging to the dense MANET increases the counter and forwards the message, without resending an already sent message, similarly to the case of dense MANET identification. Figure 8.a shows the message propagation from node I (the current IN) to all the dense MANET nodes. Each node is marked with the value of its counter, i.e., its distance from I in number of hops. For the sake of simplicity, the figure does not show all broadcast messages exchanged, but only those from closer nodes to farther ones with regard to I.

To limit bandwidth consumption, each node replies to the IN by communicating its distance if and only if it cannot detect any node farther than itself at single-hop distance. In fact, when a node receives a farthest node determination request, it starts a timeout; at timeout expiration, it replies if it has not received any other broadcast from a node farther than itself (with a greater counter). Let us rapidly observe that the choice of that timeout is simple because it represents the time for single-hop neighbors to re-broadcast the message and does not depend on the number of participants and on the dense MANET diameter [13].

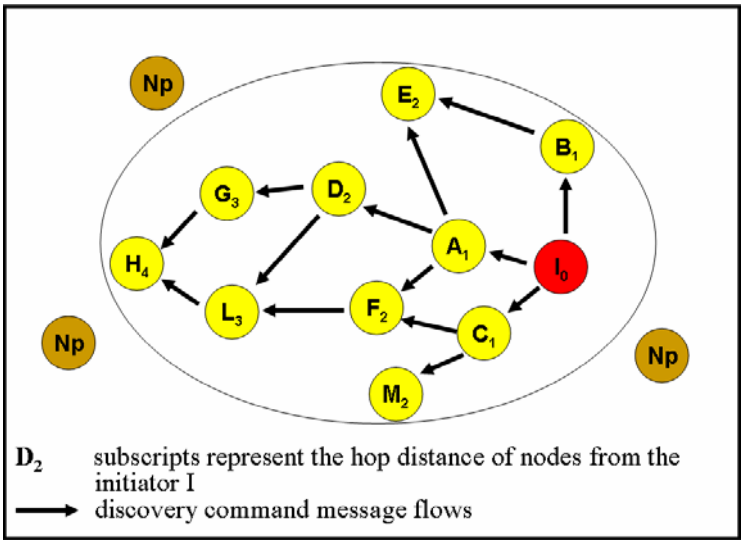

Figure 8.a. The current IN (I) broadcasts exploratory messages for manager election.

The nodes replying back to the farthest node determination message include not only the actual farthest nodes for the current IN, but also other nodes situated at the dense region boundaries. Figure 8.b shows that not only H (the only farthest node) replies to I, but also E and M, which are at the boundaries of the dense MANET. All other nodes, e.g., node A, do not reply because they are prevented by single-hop neighbors placed at greater distance, e.g., nodes E, F, and D.

Every time the IN receives a reply, it records the message source identity, its distance, and the incoming direction, i.e., the neighbor that last forwarded the message. Finally, the IN determines the identity of the farthest nodes, by excluding non-farthest nodes. The IN assumes the distance of the determined farthest node(s) as its INvalue. Figure 9 reports the sequence diagram of the protocol execution.

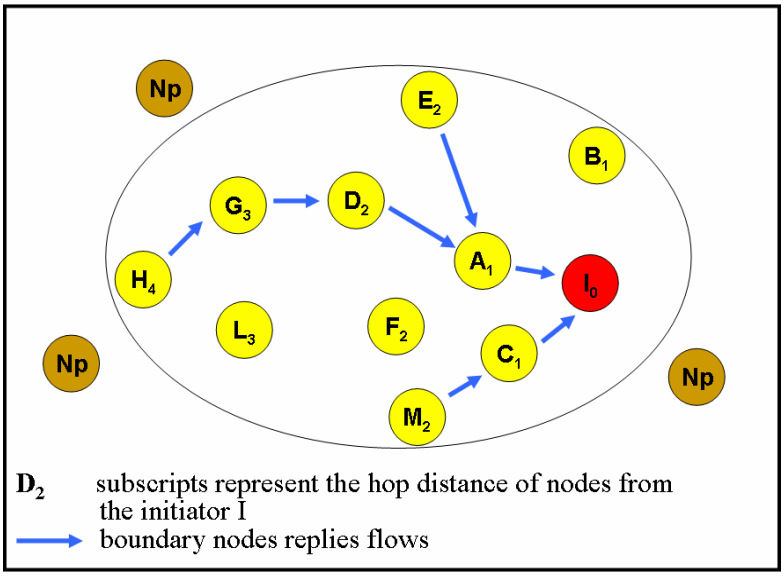

Figure 8.b. Only nodes at dense MANET boundaries (E, M, H) reply to the current IN.

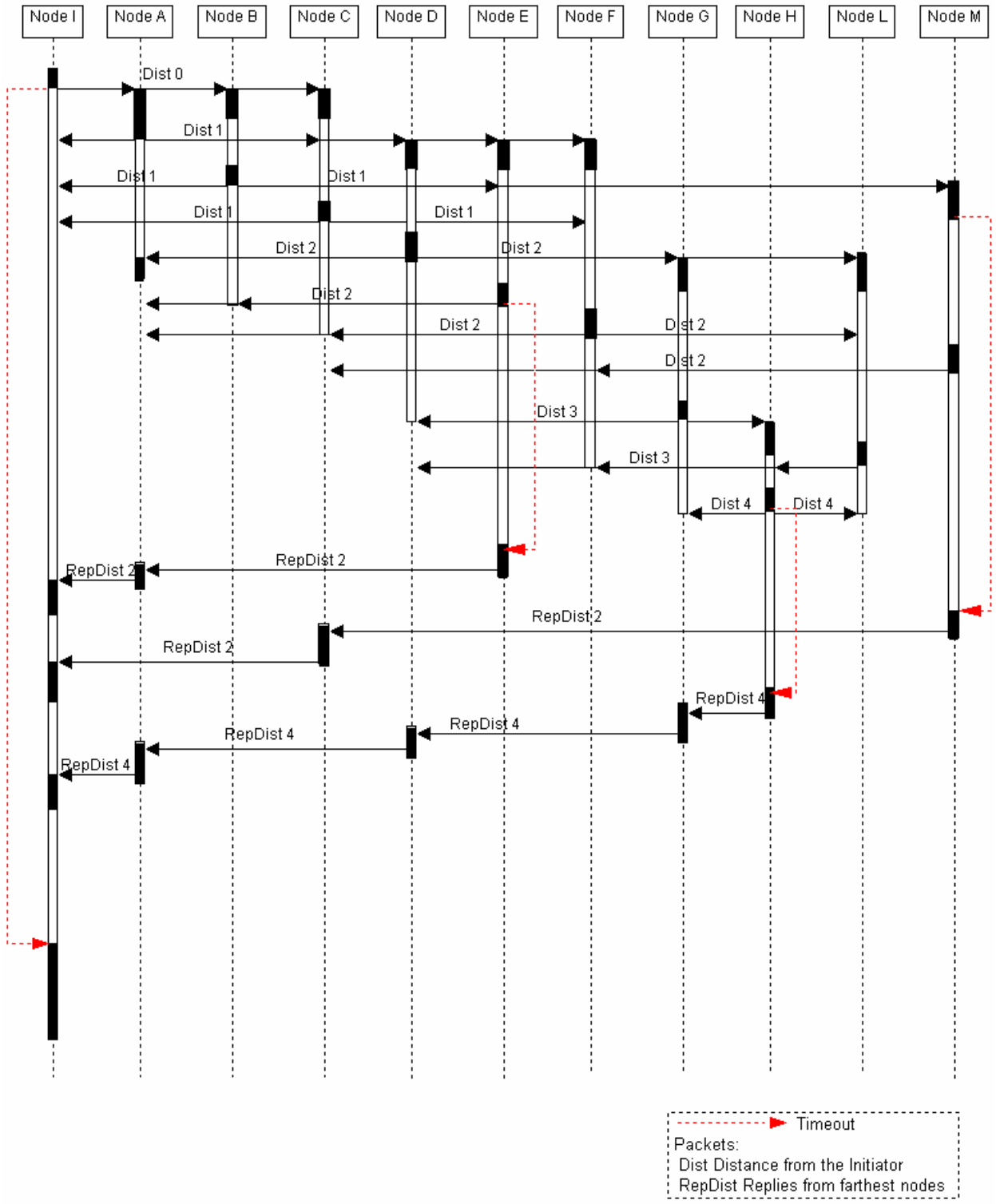

Figure 9. Sequence diagram of the farthest nodes identification protocol.

4. Replica Distribution (RD)

RD is responsible for the lightweight dissemination of replicas on MANET nodes, also with the goal to enable the effective lightweight resource retrieval at provision time (see Section 5.3). When a delegate enters the dense MANET, it communicates the RDF description of its resources to the manager that decides the replication degree (deg) of each resource on the basis of the provided descriptor and of other external indicators such as the estimated number of nodes in the dense region. The manager maintains a table with one entry for each managed resource: any entry contains the replication degree to be enforced and weakly consistent information about replica placement. After having updated its table, the manager delegates the resource owner for the actual implementation of the replica distribution process.

The main guideline of the RD protocol is to disseminate resource replicas on nodes at r-hop distance along an approximately constant direction. When a delegate is commanded to replicate one of its resources, it sends a replication packet specifying the number of replicas still required and the desired r-hop distance between replicas. The replication packet is propagated on nodes placed along an approximately straight line with a fixed direction. Let us observe that REDMAN does not require dense MANET participants to be GPS-equipped and exploits lightweight heuristic-based estimations, specific for dense MANETs, to determine constant directions.

Roughly speaking, the solution guideline is that a node determines its successor by choosing, among its neighbors (the nodes at single-hop distance from it), the one sharing fewer neighbors with its predecessor. To this purpose, RD locally broadcasts the neighbor list of its predecessor to all neighbors; only the neighbors sharing with the predecessor a number of neighbors lower than a threshold reply. This determines a roughly constant direction if the node density is almost uniform in the dense MANET. When a replication packet reaches a node at r-hop distance from a replica, that node becomes a delegate; the new delegate commands the manager to update the information about replica placement and reiterates the process by decreasing the number of requested replicas.

For the sake of clarity, additional details and sequence diagrams about the REDMAN RD solution are in the following, after the presentation of the dual REDMAN RR strategy (for instance, see Figure 11 in Section 5.3).

5. Replica Retrieval (RR)

RR aims at enabling clients to effectively find their requested resources at provision time on the basis of the resource RDF descriptions, i.e., to dynamically determine the IP address of one node hosting a requested matching resource and the unique name of the resource on that node. Resource retrieval is a hard task in MANETs, where a static infrastructure is not continuously available, thus preventing the usage of a fixed centralized lookup service known by all participants [13]. The usage of a single centralized repository with replica placement information is not viable: i) a large number of requests could overwhelm the single point of centralization; ii) the repository would represent a single point of failure; iii) the repository should be updated with strict consistency requirements not to hinder resource accessibility.

In several cases, it makes sense to improve the RR performance by paying the overhead of disseminating Information about Replica Placement (IRP) to a suitably chosen subset of nodes belonging to the dense MANET. In the following, the section first analyzes main advantages and drawbacks of some common retrieval solutions; then, it presents and evaluates the original REDMAN RR strategy, called SID, designed and implemented to effectively fit the dual REDMAN replica distribution solution.

5.1. An Overview of Possible RR Strategies

We propose to consider three main factors to determine the effectiveness of RR strategies:

- the overhead imposed in terms of both memory required to maintain IRPs and messages exchanged to retrieve the requested resources at runtime;
- the scalability when applied to large deployment environments;
- the accuracy, i.e., the found/searched resource ratio in the finite time interval spent for the retrieval of the requested resource.

Different RR strategies can decide different trade-offs among these factors. In particular, the memory and network overhead imposed by IRP dissemination is often traded against the overhead required at provision time for resource discovery (the growth of IRP diffusion costs usually corresponds to a decrease of runtime retrieval costs). Therefore, the choice of the optimal RR strategy depends on the characteristics of supported applications and of the deployment scenario, e.g., on the expected ratio between the number of searches and of replica instantiations, the fre-

quency with which replicas leave/enter the dense MANET, the time requirements for discovering the requested resources, and the size of retrieval messages and IRPs.

The most intuitive and simple RR strategies are flooding-based. A first possible solution, which we will call IRP Flooding (IF) in the following, could establish that delegates disseminate IRPs about all their hosted resources to all nodes in the dense MANET (*flooding of IRP messages*). However, "the cost of making sure that everyone knows about everything is prohibitive" in MANET environments, mainly for scalability reasons [15]. In fact, to maintain an eager-consistent up-to-date view of IRPs, any delegate with a changing set of hosted resources should overburden the network with continuous IRP update message flooding; moreover, also the memory required to locally maintain IRPs of all replicas in the dense region could be unsustainable.

A second possible flooding-based solution (Query Flooding - QF) could specify that delegates do not have to diffuse any IRP; message flooding is necessary in the search phase where all nodes are exhaustively explored to look for requested resources (*flooding of search messages*). QF makes sense only if the number of RR searches is very limited and it is not worth paying the overhead for diffusing IRPs. Moreover, if the frequency of node entrances/exits in/out the dense MANET is very high, IRPs tend to become stale very soon, and the advantages of IF is significantly reduced.

5.2. k-hop Distance IRP Dissemination

Our research activity has focused on investigating novel RR strategies to avoid message flooding during the search phase by distributing IRPs only to a subset of nodes in the dense MANET. An original solution investigated, called *k*-hop Distance IRP Dissemination (*k*-DID), specifies to place IRPs only on nodes positioned at fixed *k*-hop distance the ones from the others. In other words, i) the IRP related to a specified resource should be placed at exactly *k* hops from at least another copy of the same IRP; ii) there should not be a path shorter than *k* hops between two copies of the same IRP; and iii) IRPs should be distributed over the whole network so that each node is at most at *k* hops from the IRP of any replicated resource. When adopting *k*-DID, IRP retrieval (and consequently replica discovery) only requires to explore the nodes situated, on average, at (*k*/2)-hop distance from the searcher. Let us observe that *k*-DID improves the solution in [16], where placement information similar to REDMAN IRPs is duplicated on all nodes lo-

cated at fixed distance from resource-hosting nodes; in [16] IRP nodes could also be at single-hop distance the one from the other.

We have carefully investigated how *k*-DID could be effectively implemented in a lightweight way. Let us preliminary note that an IRP distribution strategy based on a single network flooding is infeasible, since it cannot determine which nodes at *k*-hop distance from an IRP-owning node are also at *k*-hop distance the one from the other. To practically present our *k*-DID implementation, suppose a delegate is willing to diffuse IRPs of its resources at *k*-hop distance. *k*-DID executes the following steps:

1. the delegate prevents its neighbors distant less than *k* hops to host an IRP copy by flooding a denial message with *TTL=k*. Nodes receiving those messages with *TTL>0* change their state to unavailable to reflect their unavailability for IRP hosting;
2. the delegate sends an IRP copy and assigns its initial role of IRP distributor to one of the nodes, identified at step 1, that are placed exactly at *k*-hop distance and whose state is free;
3. steps 1 and 2 are reiterated until the IRP distributor cannot identify any free node at *k*-hop distance. If there are no free nodes, the backtracking phase at step 4 starts;
4. during backtracking, the IRP distributor commands its distributor predecessor to repeat step 4. If a free potential IRP distributor is found, then IRP dissemination goes on from step 1. Otherwise, step 4 is repeated until the initial resource delegate is reached, thus ending the protocol.

Let us say that *N* is the number of nodes belonging to the network and *F(k)* the average number of nodes belonging to a *k*-hop-diameter sphere included in the dense MANET. *k*-DID imposes a relevant overhead in terms of messages sent for distributing IRPs, mainly depending on *N* and *k*. However, the number of nodes with IRPs is low (about one node every *F(k/2)* ones) and the search message overhead is very low. In fact, each client expects to find a replica of the requested resource by querying all nodes within its surrounding *k/2* hops. However, *k*-DID hardly scales in presence of node mobility since it is difficult to preserve the validity of the three above constraints without imposing heavy communication-intensive maintenance protocols for IRP distribution.

5.3. REDMAN Replica Retrieval

We have deeply investigated the performance of the *k*-DID RR strategy (see Section 7.2) and found it does not well fit the addressed dense MANET deployment scenario, mainly due to its excessive cost in both IRP diffusion and IRP updates in mobile environments. Therefore, we have decided to design, thoroughly evaluate, and then integrate in REDMAN an alternative original RR solution, called Straight IRP Dissemination (SID) and described in the following. SID exploits an IRP dissemination strategy strictly integrated with the REDMAN RD one and has demonstrated to impose lower overhead than the other investigated RR solutions in most common usage scenarios.

Let us briefly recall that REDMAN RD disseminates replicas on nodes at fixed distance along an approximately constant direction. The SID IRP diffusion consists in propagating IRPs, for any replicated resource and at the time of replica distribution, on all nodes located along the almost constant direction used during resource replication, that is along the approximately rectilinear path between disseminated resource replicas. In other words, differently from k-DID, SID does not aim at spreading IRPs over the whole dense MANET but only along a single direction. In addition, SID respects the first two *k*-DID constraints in stationary conditions (non-mobile nodes), but not the third one. Consequently, SID permits both to store IRPs on a limited number of nodes and, at the same time, to limit IRP message overhead during IRP dissemination and resource retrieval. In fact, a REDMAN client looking for a resource exploits search messages that also propagate along approximately constant directions: the probability to rapidly determine an intersection between the direction of retrieval and that of IRP placement is high for most usual deployment scenarios, thus enabling efficient replica searches [17]. In Figure 10, delegate A disseminates a resource replica to node E; the result of replica distribution is also that B, C, and D store IRPs with the information of resource availability at node A and E. When node G looks for that resource, the search message propagates until it reaches node D that owns the needed IRP. In more details, during replica dissemination all nodes forwarding replication packets maintain a reference to their sending delegate. During retrieval it is sufficient that searchers reach one of the nodes along the replica placement line to discover where a delegate is, with no need to contact the replica manager. Since search messages are locally broadcasted to all neighbors to determine suitable successors, anyone owning the IRP of the searched resources can notify the client. In the

case of search failure after a timeout, REDMAN clients start exploring a different direction for retrieval exploration.

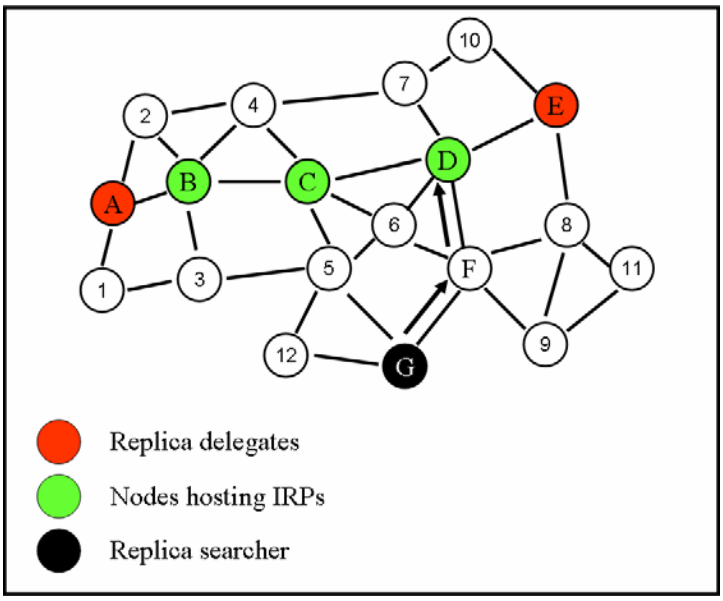

Figure 10. The distribution of replicas/IRPs and replica retrieval exploration exploit approximately constant directions in REDMAN.

Figure 11 details the RD protocol execution by referring to the network example in Figure 10. Node A starts the RD protocol - phase (a) - by randomly choosing one of its neighbors, i.e., B, to diffuse IRPs and a 4-hop distant replica of its resource (r=4). Node B identifies a straight direction, by investigating which neighbor shares less neighbors with A – phase (b). Since C shares only one neighbor with A, B delegates the execution of the RD protocol to it – phase (c). C repeats the steps above and, since both D and Node6 share only one neighbor with B, it randomly chooses to delegate the operations to D – phase (d). Finally, D delegates E – phase (e) - that realizes to be 4-hop away from A and thus locally replicates the resources. Let us point out that intermediaries (B, C, and D) know that A hosts one copy of the resource.

Figure 12 shows the RR operations when G starts searching for a needed resource. First, it randomly delegates the search to F – phase (a). F executes the SID protocol to determine a straight direction: it broadcasts the list of G neighbors to its neighbors Node6, D, G, Node8 and Node9 – phase (b). Since D knows where the needed resource is placed, it immediately replies to F – phase (c). As G receives the reply, it downloads the resource from A – phase (d).

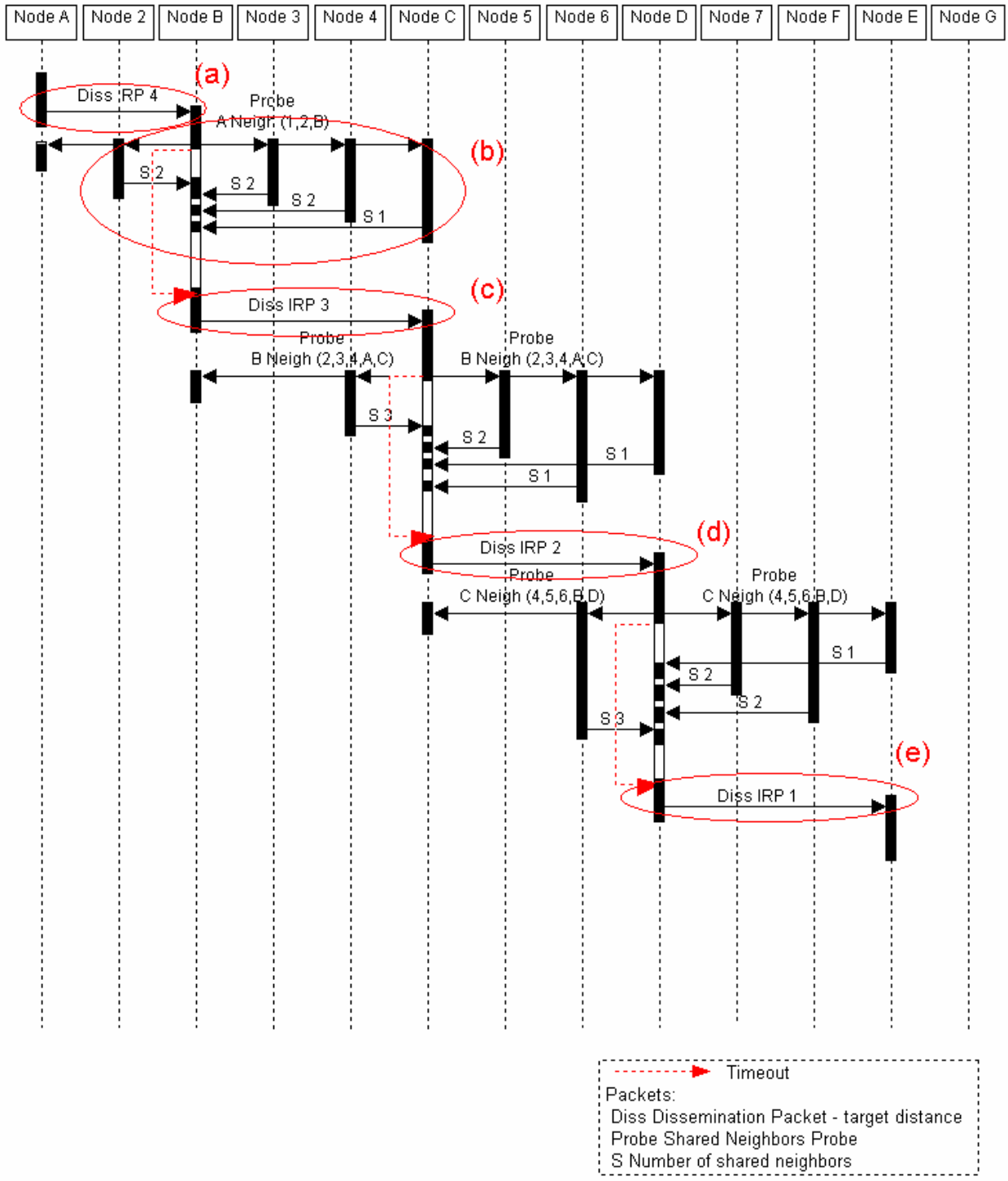

Figure 11. Sequence diagram for the REDMAN RD protocol.

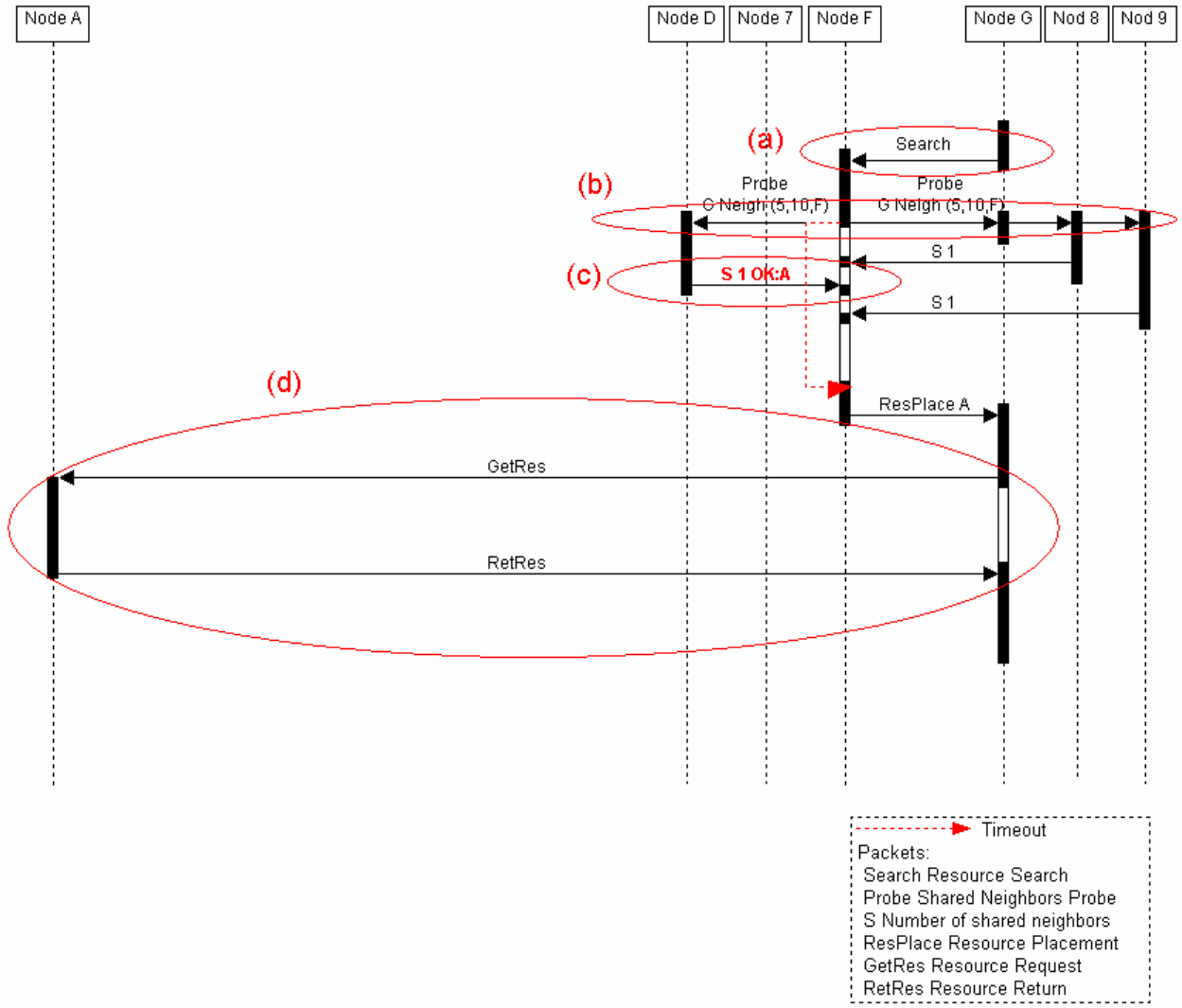

Figure 12. Sequence diagram for the REDMAN RR protocol.

The mobility of REDMAN devices that host resource replicas and/or IRPs could potentially affect resource availability (see Section 7.2.3). For this reason, we have extended SID with a decentralized and completely local maintenance protocol (SID reconstruction) in charge of loosely understanding that some IRP-hosting nodes have moved and of re-distributing IRPs to suitable neighbors. When a node, notified by the DMC facility, loosely detects the leaving of its predecessor/successor along the straight replication path, it locally broadcasts a reconstruction message. All nodes receiving that message from both a predecessor and a successor are eligible to replace the moved node. These nodes reply to the predecessor, and it designates one of them. Only in the case the predecessor does not receive any reply (there is no suitable node or more

consecutive successors have simultaneously moved), after a relatively large time interval, it re-starts a new IRP distribution. Let us point out that SID reconstruction does not aim at maintaining the strict any-time consistency of IRPs but only at lazily re-establishing IRP alignment. In addition, the implementation of SID reconstruction is optimized to enable single-message cumulative adjustments for IRPs of different resources.

6. Replica Degree Maintenance (RDM)

Proactive RDM solutions, similar to [7], do not fit the REDMAN addressed provisioning environments. In fact, proactive strategies usually require GPS-equipped wireless nodes that continuously monitor their mutual positions to foresee network partitions, thus producing non-negligible network/computing overhead. REDMAN RDM, instead, implements a reactive solution with very low communication overhead by relaxing the constraint of anytime perfect consistency in the number of available replicas. In other words, in REDMAN it is possible to have time intervals when the desired replication degree differs from the actual number of replicas in the dense region.

After the initial replica distribution, RDM aims at maintaining unchanged the replication degree only by reacting to node movements/failures. When a delegate realizes it is going to exit the dense MANET, it autonomously offloads its shared resources on its neighbors still within the dense region, which communicate the occurred change to the replica manager. On the contrary, if a delegate does not succeed in foreseeing its exit from the dense region and realizes to be already out of it, it tries to notify the replica manager by specifying its hosted resources. Once the manager receives the notification, it commands other delegates for those resources in the dense MANET to distribute new replicas. Finally, when a delegate either fails or leaves the network by abruptly interrupting all its connections, to achieve scalability and to limit overhead, REDMAN accepts a temporary inconsistency in the replication degree. Only at large time intervals delegates are required to confirm their presence to their associated manager, by sending an updated list of their shared resources. In that way, the manager can lazily re-establish the replica degree consistency by commanding still alive delegates to distribute new replicas only when some delegate update messages are missing.

7. Experimental Results

To evaluate the effectiveness of our approach, we have both measured the performance of the REDMAN prototype for Wi-Fi-enabled PDAs with Java2 Micro Edition and extensively simulated the behavior of REDMAN protocols over wide-scale deployment scenarios. In fact, to validate the scalability of the proposal, it is necessary to evaluate REDMAN in challenging wide-scale provisioning environments with several hundreds of mobile nodes, which we have simulated on top of the widespread ns-2 network simulator [18].

In the following, the paper will show the results we obtained respectively for DMC and RD/RR facilities. The code of the REDMAN protocols, the exhaustive description of all the ns-2 simulation parameters used, and additional performance figures about REDMAN are available at the REDMAN Website `http://lia.deis.uniboT.it/Research/REDMAN/`.

7.1 The DMC Facility

Our simulation deployment scenario consists of randomly positioned nodes in a square area. The area is split in two zones, a smaller Internal Square (IS) at the center of the area, and the remaining area around the IS, called External Square (ES). Nodes are distributed so that the dense MANET almost coincides with the IS area (the IS node density is relevantly higher than the ES one). If not differently specified, we have used default ns-2 values for simulation parameters, e.g., constant 250m circular transmission ranges, bi-directional connectivity, and IEEE 802.11 link layer protocol.

The simulations of the DMC facility have three main goals: i) to determine the network overhead of the proposed protocols, in terms of the number of packets exchanged among dense MANET nodes; ii) to evaluate the accuracy of the manager election protocol notwithstanding the heuristic-based overhead reduction; iii) to evaluate the robustness of the dense MANET identification protocol while varying node mobility.

7.1.1. Network Overhead

First, we have carefully evaluated the network overhead of the REDMAN protocols for dense MANET identification and manager election, to verify that the proposals are lightweight enough for the addressed deployment scenario. We have tested the two protocols in different simulation

environments, with a number of nodes ranging from 50 to 550 (increasing of 20 nodes each step). The size of ES/IS areas have been changed with the changing number of nodes involved, to maintain the same ES/IS node densities in all simulations. For both protocols we have measured the average number of packets sent by each participant, over a set of more than 1,000 simulations.

The results reported in Figure 13 are normalized to the number of nodes actively participating in the related protocol. The dense MANET identification protocol is designed to determine participant nodes by requiring only one local broadcast from each node reachable from the initiator. With regard to the replica manager election protocol, also in this case the number of sent packets is very limited and grows very slightly with the number of participants. In fact, sent packets tend to be proportional not to the total number of nodes, but to the number of iterations required to identify an acceptable solution. The number of iterations is roughly proportional to the dense MANET diameter (approximately 3 hops for the 50-node case and 12 hops for the 550-node case) and grows less than the number of participants.

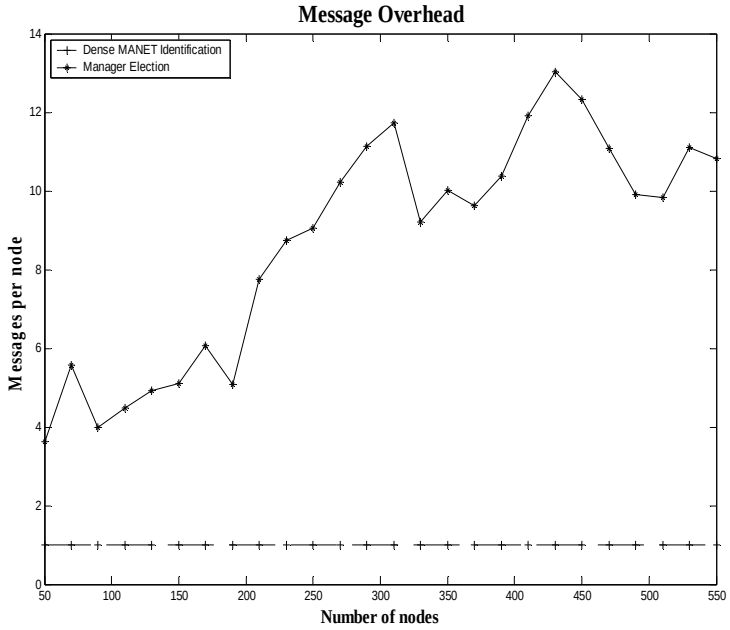

Figure 13. Messages sent/received in dense MANET identification and manager election.

Figure 14 reports how the number of iterations grows in function of the growing of network participants. The results are average values on a set of simulations where the role of initiator is assigned to a different node at each simulation; the results have shown a very limited dependence on the choice of the initiator node. In summary, the experimental results about REDMAN overhead demonstrate the feasibility and the good scalability of the proposed solutions (the network overhead slowly increases when the number of participants grows).

Figure 14. Number of iterations needed for the REDMAN manager election protocol.

7.1.2. Manager Election Accuracy

To quantitatively assess the effectiveness of the REDMAN election protocol, we have measured its accuracy in assigning the manager role to a node close to the actual topology center of the dense MANET. We have run over 200 simulations in the most populated scenario of 550 nodes and, for any simulation, we have measured the hop distance between the manager chosen by the REDMAN protocol and the actual optimal solution. The results in Figure 15 are obtained by starting each election from a different initiator node. In more than 90% of the runs, the REDMAN protocol has identified either optimal solutions or quasi-optimal solutions at 1-hop distance from the actual optimum. The average inaccuracy is only 0.385 hops, which represents a largely acceptable value for the addressed application scenario.

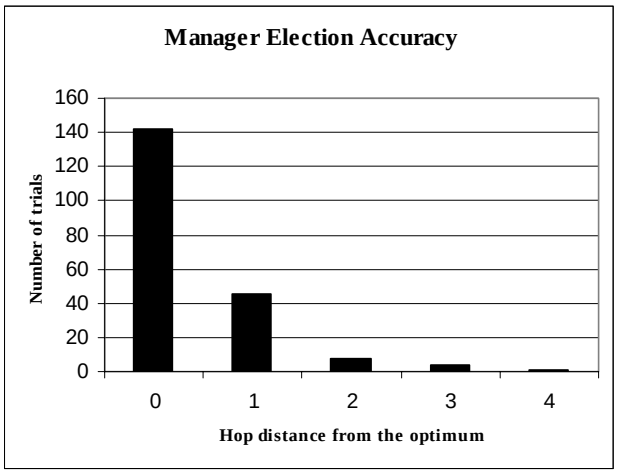

Figure 15. Accuracy of the REDMAN manager election protocol.

7.1.3. Robustness in Function of Node Mobility

To test the REDMAN dynamic behavior, we have evaluated the dense MANET identification protocol by varying the mobility characteristics of network nodes. Let us rapidly note that the manager election protocol executes for very limited time periods and re-starts its execution only after a long time interval; to a certain extent, it is assumable that it operates under static conditions. On the contrary, the dense MANET identification protocol should continuously work to maintain an almost consistent and updated view of the dense MANET participants, crucial for the effective working of REDMAN solutions. For this reason, we have focused on REDMAN identification behavior in function of node mobility.

We have considered the same 110-node scenario of the tests in Section 7.1.1. and 7.1.2. For any node (randomly chosen among the ones close to the IS boundary) that exits the dense region, a new node enters the IS, to keep unchanged the spatial node density according to the dense MANET definition. Any pair of random movements of randomly chosen nodes occurs every *M* seconds, with *M* that varies from 10 to 60. Any other node movement not producing entrances/exits in/from the dense MANET does not affect at all the behavior of the REDMAN identification solution.

Figure 16 reports the dense MANET identification inaccuracy, defined as the difference between the number of dense MANET participants determined by the REDMAN protocol and its actual value. The inaccuracy is reported as a function of the mobility period *M* and for different

values of the time period used for Hello packets. Each point in the figure represents an average value obtained by capturing the state of the network in 20 different runs. The figure shows that the average inaccuracy is very limited and always within a range that is definitely acceptable for lazy consistent resource replication in dense MANETs. As expected, the inaccuracy grows when node mobility grows, for fixed values of the Hello message period. However, even for relatively high values of the Hello period, the REDMAN identification inaccuracy is negligible for the addressed application scenario (on the average, always less than 1.7), for the whole range of node mobility frequencies that can be of interest for dense MANETs. Let us observe that this permits to set relatively high periods for Hello packets, while obtaining a low inaccuracy for the dense MANET identification, thus significantly reducing the message exchange overhead.

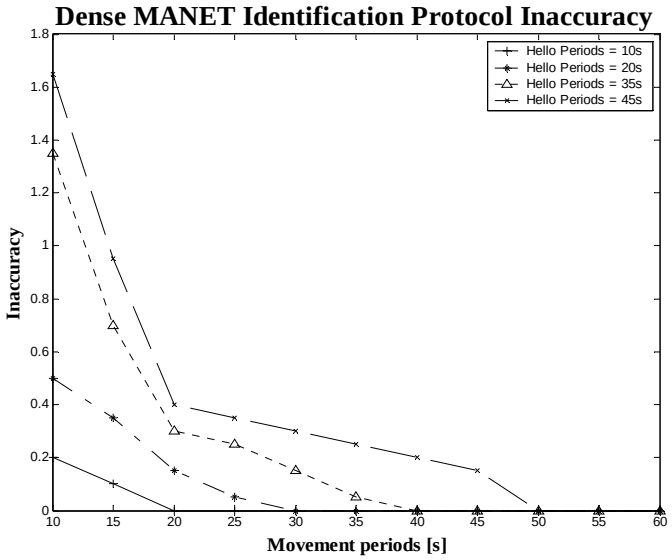

Figure 16. Inaccuracy of the REDMAN dense MANET identification protocol.

7.2. The RD and RR facilities

To evaluate the effectiveness and the performance of RD and RR, we have used a square deployment area (side=2.7km) with 400 randomly positioned nodes. The dense MANET diameter, i.e., the longest minimum path between two nodes belonging to the dense region, is in most cases equal to 12.

The simulation-based experimental results presented in the following have three main goals: i) to evaluate the accuracy of the SID RR strategy depending on different parameter tuning; ii) to determine SID network overhead and compare it with the other RR solutions discussed; iii) to assess the SID robustness in the case of node mobility.

7.2.1. Accuracy

The first performance indicator considered for SID evaluation is accuracy, defined as the ratio between the number of successful resource searches and the total number of searches in stationary scenarios with fixed nodes. No re-iterations of the SID protocol are taken into account; searches are considered successful only if they find the needed IRPs by exploring the first retrieval direction. Mainly two tunable parameters have demonstrated to affect the SID accuracy value in stationary scenarios: the number *i* of nodes hosting IRPs (that is the number of nodes along the straight path where replicas are positioned) and the maximum number *s* of hops explored in the retrieval phase.

The tuning of *i* and *s* permits to trade the SID accuracy against its imposed message overhead. Figure 17 shows the results obtained over more than 1,000 simulations with *i* and *s* independently varying from 2 to 15 hops (in the case distribution/retrieval paths reach network boundaries before arriving at their maximum number of allowed hops, REDMAN automatically makes paths continuing with a new random direction back in the dense MANET). Each plotted value represents the average of 20 simulations where delegates, dynamically determined by REDMAN RD, distribute *i* IRPs, and randomly chosen clients look for a resource replica by exploiting an *s*-hop-limited query. The reported results show that, when *i* and *s* are greater than the diameter of the considered dense MANET, the accuracy overcomes 85%. In all simulations done, we have experienced that choosing *i* and *s* values approximately equal to the dense MANET diameter permits to achieve sufficient accuracy, with limited message overhead in both phases of SID-based IRP dissemination and RR.

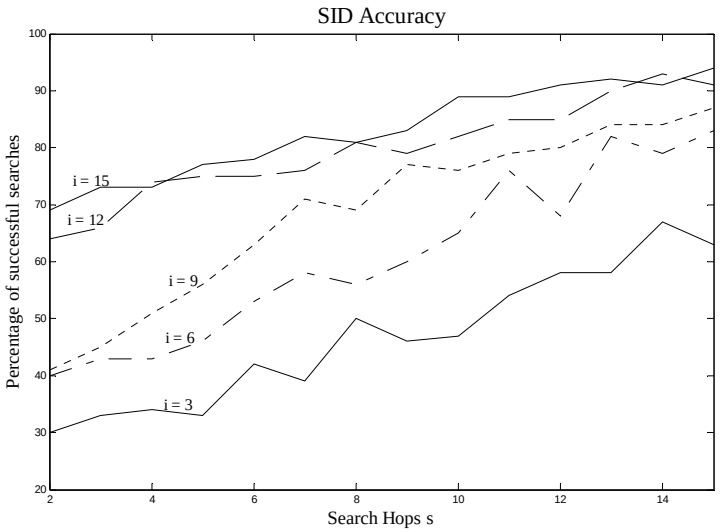

Figure 17. SID accuracy while varying *i* and *s*.

7.2.2. Overhead

To quantitatively compare the SID message overhead with the other presented RR solutions, we have measured the overall number of messages required to distribute and find replicas in the same challenging simulation scenario of Section 7.2. Figure 18 reports the experimental results obtained while increasing the number of searches. Each point represents the average of 20 simulations. SID parameters i and s are set to 12, according to what observed in the previous section. k-DID exploits the same number (12) of disseminated IRPs, so to realize an actual deployment scenario where it makes sense to perform a comparison between message overheads.

Figure 18 shows how QF produces a rapidly growing amount of message exchanges also for a limited number of searches. As expected, k-DID imposes a high overhead for IRP placement (about 4 messages per node in the dense MANET), while its search phase demonstrates to be very effective. SID exhibits linear growth in overall message overhead, by imposing a lower number of IRP distribution messages than QF and k-DID and only a slightly greater number of RR messages than k-DID. Both k-DID and SID require low memory occupation on IRP-hosting nodes, since they diffuse IRPs only on a limited node subset (see the concise comparison among RR solutions in Table 1). IF represents a lower bound in terms of communication overhead, but

it is a practically unviable solution because it requires all nodes in the dense MANET to store IRPs about all replicas of all available resources.

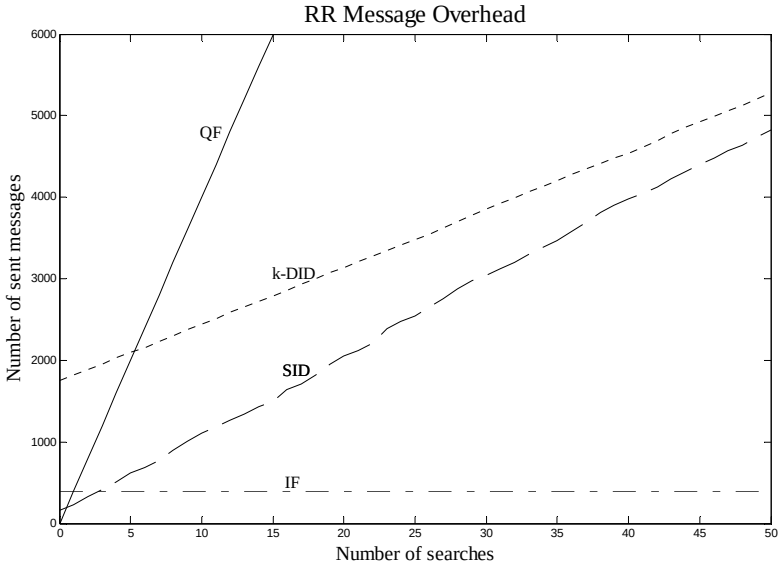

Figure 18. Message overhead for the four presented RR solutions.

7.2.3. Accuracy in Non-Stationary Scenarios with Mobile Nodes

To evaluate the dynamic behavior of SID in non-stationary deployment scenarios with mobile MANET nodes, we have adopted a mobility model establishing that, after a randomly chosen time interval, any node in the dense region starts moving along a rectilinear path, with random speed direction and speed module randomly chosen in the range [1, 5] m/s.

Figure 19 presents the temporal evolution of SID accuracy by comparing it with a SID version without the local IRP maintenance reconstruction protocol. Plotted results are average values over 20 simulated scenarios where only 10% of IRP owners remain fixed. As for the overhead evaluation in Section 7.2.1., *i* and *s* are set to 12. The figure shows that, after a sufficient number of nodes has moved and left the dense MANET, the reconstruction-enabled SID version outperforms the other. Reconstruction-enabled SID has demonstrated to maintain good accuracy notwithstanding the challenging mobility model adopted with limited message overhead.

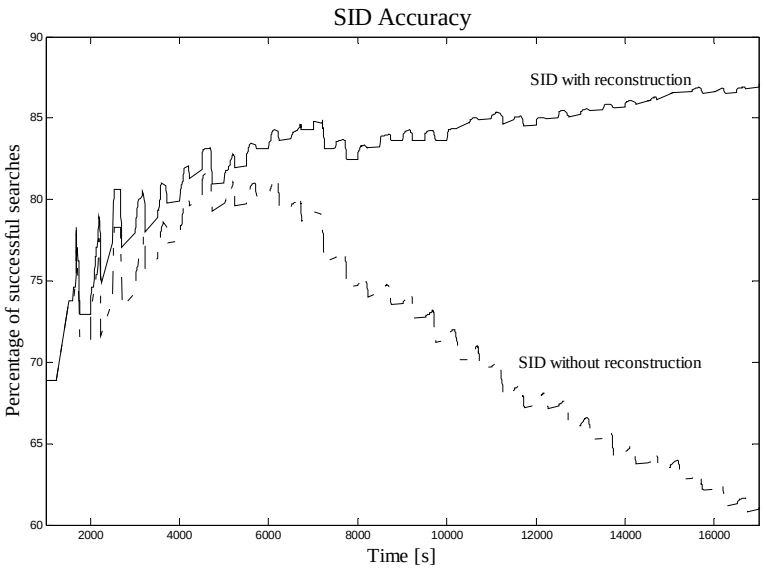

Figure 19. SID accuracy with/without reconstruction in non-stationary scenarios.

RR Strategy	IF	QF	*k*-DID	SID
Global IRP memory occupation	N * size(IRP)	size(IRP)	O(N/F(k/2)) * size(IRP)	O(i) * size(IRP)
Dissemination message overhead	N	0	O(N)	depends on i, density
Retrieval message overhead	0	N	O(F(k/2))	depends on s, density
Scalability	Suits deployment scenarios with few searches	Unviable for large scenarios	Fits stationary scenarios with a very high number of searches	Fits also mobile scenarios with limited number of searches

Table 1. Concise comparison of IF, QF, *k*-DID, and SID RR strategies.

8. Related Work

The idea of increasing the availability of MANET applications by replicating data/service components is still at its very beginning. So far, the research has mainly focused on replication to ensure data availability in the case of MANET partitioning. Most proposals assume that wireless nodes are aware of their physical position, e.g., by imposing all participants to be GPS-equipped [7].

Other activities mainly aim at respecting strict requirements about replica synchronization/consistency and, therefore, impose a non-negligible network overhead [8, 19]. In particular, [19] presents a secure middleware that aims at enhancing data availability by exploiting an adaptive replication protocol that also permits replica updates; however, that proposal only addresses scenarios with nodes in direct single-hop visibility and assumes that any node has complete

knowledge of position of all maintained replicas and of memory/battery/mobility state of other peers. [20], instead, proposes a novel solution for read-only replicas based on the guideline to counteract network partitioning via proactive replication depending on the frequencies of resource access; the approach imposes coordination and synchronization among participants and does not scale well with growing numbers of nodes and replicated resources.

In addition, infrastructure-free and completely decentralized MANETs pose novel challenges for resource retrieval. Some activities have investigated MANET-specific broadcast-based peer-to-peer RR solutions, with limited scalability and excessive overhead for resource-constrained clients [21]. Recent research proposals aim at limiting broadcast communications by exploiting quorum-based solutions [17, 22, 15]: the primary idea is to disseminate resource placement information on a node subset determinable without message flooding. [17] and [22] specifically address ad-hoc sensor networks where nodes are fixed. On the one hand, [22] assumes GPS-equipped nodes to spread advertisement/search packets along orthogonal directions; on the other hand, similarly to REDMAN, [17] considers provisioning environments where geographic routing cannot apply and proposes to diffuse placement data along paths with approximately constant directions, determined by choosing, as the next hop, a node that is not the neighbor of any node belonging to the already built sub-path. [15] extends the GPS-based solution in [22] for replication in mobile environments: each node only stores placement data about its nearest replica. A different approach is presented in [16]: any resource is associated with its origin place, which is in charge of distributing all replicas according to an original protocol similar to k-DID, as stated in Section 5.2.; in that proposal, replica retrieval exploits geographical routing.

9. Conclusions and On-going Research

The challenge of supporting distributed services over dense MANETs can significantly exploit the assumption of high node population to enable lazy consistent forms of resource replication. This can increase the availability of common interest resources notwithstanding the unpredictable node exit from the dense MANET. The REDMAN project demonstrates how it is possible to provide middlewares for lightweight and effective replica management also in dense MANETs. In particular, REDMAN proposes original and completely decentralized protocols, specifically designed for dense MANET, to dynamically determine dense region participants, to

elect replica managers by minimizing the distance from farthest nodes and to distribute and retrieve resource replicas. These protocols impose a limited overhead in terms of exchanged messages and are sufficiently accurate even in very large-scale deployment scenarios with high node mobility. To the best of our knowledge, there are no other research activities that have already proposed and evaluated solutions for dense MANET identification and dynamic determination of the MANET topology center.

The experimental results obtained are encouraging further REDMAN-related research activities. First, we are exploring decentralized security mechanisms to enable the dense MANET participation (and the access to shared resources) only to authorized nodes. Secondly, we are working on testing and measuring the in-the-field performance of the REDMAN prototype, by deploying the middleware in actual dense MANETs consisting of about one hundred devices with IEEE 802.11b connectivity in ad-hoc mode. Finally, we are developing application prototypes that exploit our replication middleware, also to obtain additional feedback on REDMAN usability.

Acknowledgements

Work supported by the MIUR FIRB WEB-MINDS Project "Wide-scale Broad-band Middleware for Network Distributed Services" and by the CNR Strategic IS-MANET Project "Middleware Support for Mobile Ad-hoc Networks and their Application".

References

[1] I. Chlamtac, M. Conti, J. J.-N. Liu, “Mobile Ad Hoc Networking: Imperatives and Challenges”, *Elsevier Ad Hoc Networks*, Jul. 2003.

[2] Y. Yuan; W. Arbaugh, “A Secure Service Discovery Protocol for MANET”, *14th IEEE Conf. on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Sep. 2003.

[3] A. Helmy, S. Garg, P. Pamu, N. Nahata, “Contact-based Architecture for Resource Discovery (CARD) in Large Scale MANETs”, *17th IEEE Int. Parallel and Distributed Processing Symp. (IPDPS)*, Apr. 2003.

[4] T. Qing, D.C. Cox, “Optimal Replication Algorithms for Hierarchical Mobility Management in PCS Networks”, *IEEE Wireless Communications and Networking Conf. (WCNC)*, Mar. 2002.

[5] J. J. Kistler, M. Satyanarayanan, “Disconnected Operations in the Coda File System”, *ACM Transactions on Computer Systems*, Feb. 1992.

[6] A. J. Demers, K. Petersen, M. J. Spreitzer, D. B. Terry, M. M. Theimer, B. B. Welch, “The Bayou Architecture: Support for Data Sharing among Mobile Users”, *1st IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, Dec. 1994.

[7] K. Chen, K. Nahrstedt, “An Integrated Data Lookup and Replication Scheme in Mobile Ad Hoc Networks”, *SPIE Int. Symp. on the Convergence of Information Technologies and Communications (ITCom 2001)*, Aug. 2001.

[8] K. Rothermel, C. Becker, J. Hahner, “Consistent Update Diffusion in Mobile Ad Hoc Networks”, *Technical Report 2002/2004*, CS Dep., Univ. of Stuttgart.

[9] S. Garg, Y. Huang, C.M.R. Kintala, K.S. Trivedi, S. Yajnik, “Performance and Reliability Evaluation of Passive Replication Schemes in Application-level Fault Tolerance”, *29th IEEE Int. Symp. on Fault-Tolerant Computing*, June 1999.

[11] L.P. Cox, B.D. Noble, “Fast Reconciliations in Fluid Replication”, *21st Int. Conf. on Distributed Computing Systems (ICDCS)*, Apr. 2001.

[12] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, J.-P. Sheu, “The Broadcast Storm Problem in Mobile Ad Hoc Networks”, *5th ACM Int. Conf. on Mobile Computing and Networking (MOBICOM)*, Aug. 1999.

[13] S. Nesargi, R. Prakash, “MANETconf: Configuration of Hosts in a Mobile Ad Hoc Networks”, *21st Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM)*, Jun. 2002.

[14] http://wwws.sun.com/software/jinni

[15] J. Tchakarov, N. Vaidya, “Efficient Content Location in Mobile Ad hoc Networks”, IEEE Int. Conf. Mobile Data Management, Jan. 2004.

[16] M. Tamori, S. Ishihara, T. Watanabe, T. Mizuno, “A Replica Distribution Method with Consideration of the Positions of Mobile Hosts on Wireless Ad-hoc Networks”, *22nd IEEE ICDCS Workshop*, Jul. 2002.

[17] D. Braginsky, D. Estrin, “Rumor Routing Algorithm for Sensor Networks”, *1st ACM Int. Workshop Wireless Sensor Networks and Applications*, Sep. 2002.

[18] http://www.isi.edu/nsnam/ns

[19] M. Boulkenafed, V. Issarny, “A Middleware Service for Mobile Ad Hoc Data Sharing, Enhancing Data Availability”, *ACM/IFIP/USENIX International Middleware Conference*, Jun. 2003.

[20] T. Hara, “Effective Replica Allocation in Ad Hoc Networks for Improving Data Accessibility”, *20th IEEE INFOCOM*, Apr. 2001.

[21] A. Helal, N. Desai, V. Verma, L. Choonhwa, “Konark - a Service Discovery and Delivery Protocol for Ad-Hoc Networks”, *3rd IEEE Conf. Wireless Communications Networks*, Mar. 2003.

[22] I. Aydin, C.-C. Shen, “Facilitating Match-Making Service in Ad Hoc and Sensor Networks using Pseudo Quorum”, *11th IEEE Int. Conf. Computer Communications and Networks*, Oct. 2002.