ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA
DEIS – DIPARTIMENTO DI ELETTRONICA, INFORMATICA E SISTEMISTICA

Dynamic Services in Mobile Ad Hoc Networks

Settore Disciplinare: ING-INF05
Ciclo XIX

Candidate:
Eugenio Magistretti

Advisors:
Chiar.mo Prof. Ing. **Maurelio Boari**

Char.mo Prof. Ing. **Antonio Corradi**

Ph.D. School Chair:
Chiar.mo Prof. Ing. **Paolo Bassi**

Esame Finale 2007

ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA
DEIS – DIPARTIMENTO DI ELETTRONICA, INFORMATICA E SISTEMISTICA

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "**Dynamic Services in Mobile Ad Hoc Networks**" by **Eugenio Magistretti** in partial fulfillment of the requirements for the degree of **Dottore di ricerca**.

Dated: March 2007

Advisors:	Chiar.mo Prof. Ing. Maurelio Boari
	Chiar.mo Prof. Ing. Antonio Corradi
Co-advisors:	Chiar.mo Prof. Ing. Paolo Bellavista
	Chiar.ma Prof. Ing. Rebecca Montanari
Ph.D. School Chair:	Chiar.mo Prof. Ing. Paolo Bassi
Examining Committee:	Chiar.ma Prof. Ing. Anna Ciampolini
	Chiar.ma Prof. Ing. Letizia Leonardi
	Chiar.mo Prof. Ing. Cesare Stefanelli

Abstract

The increasing diffusion of wireless-enabled portable devices is pushing toward the design of novel service scenarios, promoting temporary and opportunistic interactions in infrastructure-less environments. Mobile Ad Hoc Networks (MANET) are the general model of these higly dynamic networks that can be specialized, depending on application cases, in more specific and refined models such as Vehicular Ad Hoc Networks and Wireless Sensor Networks. Two interesting deployment cases are of increasing relevance: resource diffusion among users equipped with portable devices, such as laptops, smart phones or PDAs in crowded areas (termed dense MANET) and dissemination/indexing of monitoring information collected in Vehicular Sensor Networks. The extreme dynamicity of these scenarios calls for novel distributed protocols and services facilitating application development. To this aim we have designed middleware solutions supporting these challenging tasks. REDMAN manages, retrieves, and disseminates replicas of software resources in dense MANET; it implements novel lightweight protocols to maintain a desired replication degree despite participants mobility, and efficiently perform resource retrieval. REDMAN exploits the high-density assumption to achieve scalability and limited network overhead. Sensed data gathering and distributed indexing in Vehicular Networks raise similar issues: we propose a specific middleware support, called MobEyes, exploiting node mobility to opportunistically diffuse data summaries among neighbor vehicles. MobEyes creates a low-cost opportunistic distributed index to query the distributed storage and to determine the location of needed information. Extensive validation and testing of REDMAN and MobEyes prove the effectiveness of our original solutions in limiting communication overhead while maintaining the required accuracy of replication degree and indexing completeness, and demonstrates the feasibility of the middleware approach.

To my parents

Contents

Abstract **v**

Acknowledgements **xi**

Introduction **1**

1 Background and Related Work **7**
1.1 Background Research 8
1.1.1 Mobile Ad Hoc Networks 8
1.1.2 Vehicular Ad Hoc Networks 13
1.1.3 Wireless Sensor Networks 16
1.2 Related Work 20
1.2.1 Content Distribution 20
1.2.2 Resource Retrieval 22
1.2.3 Vehicular Sensor Networks 23
1.2.4 Opportunistic Networking 24

2 The REDMAN middleware for resource replication in dense MANET **29**
2.1 Practical case studies 32
2.1.1 REDMAN at work: game playing at the railway station 34
2.2 The REDMAN middleware 38
2.3 Dense MANET Configuration in REDMAN 41
2.3.1 Dense MANET Identification 42
2.3.2 Replica Manager Election 44
2.4 Replica Distribution (RD) 53
2.5 Replica Retrieval (RR) 54
2.5.1 An Overview of Possible RR Strategies 55
2.5.2 k-hop Distance IRP Dissemination 56
2.5.3 REDMAN Replica Retrieval 58
2.6 Replica Degree Maintenance (RDM) 60

3 The MobEyes middleware for opportunistic dissemination in Vehicular Sensor Networks **63**
3.1 Practical case studies 64

3.1.1 MobEyes at work: criminal tracking . . . 66
3.2 MobEyes middleware . . . 66
3.3 MDHP Protocols . . . 68
3.3.1 Summary Diffusion . . . 69
3.3.2 Summary Harvesting . . . 71
3.4 Analysis of MDHP Protocols . . . 75
3.4.1 Summary Harvesting Delay . . . 75
3.4.2 Efficiency of Bloom Filters in Summary Harvesting . . . 80
3.4.3 Scalability . . . 82

4 REDMAN Validation **85**
4.1 Simulation Setup . . . 86
4.2 Dense MANET Configuration . . . 87
4.2.1 Manager Election Inaccuracy . . . 88
4.2.2 Impact of REDMAN Heuristics on Manager Election Accuracy . . . 88
4.2.3 DMC Network Overhead . . . 90
4.2.4 Impact of Node Mobility on the Accuracy of the REDMAN Dense MANET Identification Protocol . . . 92
4.3 Replica Dissemination and Retrieval . . . 95
4.3.1 Accuracy . . . 95
4.3.2 Overhead . . . 96
4.3.3 Accuracy in Non-Stationary Scenarios with Mobile Nodes . . . 98
4.4 Replica Degree Maintenance . . . 99
4.4.1 RDM Accuracy . . . 101
4.4.2 RDM Overhead . . . 101

5 MobEyes Validation **105**
5.1 Simulation Setup . . . 107
5.2 Analysis Validation . . . 108
5.3 Effect of k-hop Relay and Multiple Agents . . . 111
5.4 Summary Diffusion Overhead . . . 112
5.5 Stability Check . . . 114
5.6 Tracking Application . . . 116
5.7 Border Effects and Turn Over . . . 121

6 REDMAN Implementation **125**
6.1 Design . . . 126
6.1.1 DMC . . . 127
6.1.2 Delegate . . . 128
6.1.3 Manager . . . 130
6.2 Implementation . . . 131
6.2.1 J2ME . . . 132
6.2.2 Implementation Experience . . . 133
6.3 Test . . . 135
6.4 Security . . . 137

7 MobEyes Implementation **141**
7.1 Design and Implementation 142
7.1.1 MobEyes Sensor Interface 142
7.1.2 MobEyes Data Harvesting Processor 148
7.2 Test 150
7.3 Security 152
7.3.1 PKI Model 154
7.3.2 Attack Model 155
7.3.3 Location Tracking Attack 156
7.3.4 Denial of Service 157
7.3.5 False Data Injection 157
7.3.6 Query Confidentiality 158

Conclusions **161**

Bibliography **164**

Acknowledgements

First and foremost, I would like to thank my advisors, Maurelio Boari and Antonio Corradi, and my co-advisors Paolo Bellavista and Rebecca Montanari for their precious guide during these years. They have followed my work with constant attention, providing me with continuous encouragement, advice and (both academic and human) support. A grateful thank is also due to Mario Gerla, who guided me during the research period I spent at the University of California, Los Angeles.

I would also like to thank many friends and colleagues that I met during these years, Luca, Federico, Alessio, Dario, Carlo, Alessandra, Alessandro, Daniela, Silvia, Gianluca, Fabrizio, Michele, Marco M., Marco G., Luca, Fabio, and Giovanni for always encouraging me with enthusiasm along the way. Many thanks are also due to all the people of the Advanced Computer Science Laboratory (LIA) of the University of Bologna, who gave me support and advises.

Further thanks are due to the other external researchers and academic people I have been collaborating with during these years, especially all valuable researchers of the Network Research Laboratory at UCLA. I would like to address many thanks to my dear friends Uichin, Gustavo, Claudio, Giovanni, Jiejun, and also Daniela, Flavio, Tony, Rosario, Riccardo, Massimiliano, Dzmitry, Paolo, Enzo, Franco, Melike, Eugenio F., Eugenio G., Alexandro, Alessandro, Maria.

I would like to express my gratitude to all undergraduate students who have contributed to the development of the REDMAN and MobEyes middlewares.

Let me additionally thank all people I met at conferences who showed interest for my projects and gave me helpful suggestions and feedback.

Finally, a loving thank is to all my family (in particular, my dear niece Lucilla) and my friends, who invaluably supported me during all these years.

Introduction

The recent development of wireless communications gave birth to novel scenarios of ubiquitous and pervasive computing where users enjoy full-fledged services "anytime, anywhere." Cellular networks allow users to freely move during provisioning of traditional services, e.g., Internet browsing and media streaming. However, these scenarios call for the deployment of a base-station infrastructure that, given the short and unpredictable coverage of radio waves in the air medium, should be carefully designed. This represents a major hindrance and limitation to the development of more dynamic applications, where users want to discover and possibly share resources, independently on their location. This calls for real dynamic interactions, characterized by being non-mediate, timely, and temporary: in one word, *opportunistic*.

Mobile Ad Hoc Networks (MANET) propose a dynamic model to support this paradigm: devices setup temporary connections as they want to communicate, without the the need of any previously deployed infrastructure. This critical lack calls for cooperative behavior of all devices to allow communication beyond the wireless coverage of a single node. In fact, in case the receiver is located outside the radio range of the sender, intermediate relay nodes are needed to route messages (multi-hop communications). Thus participant devices, even if with different capabilities and widely heterogeneous as for the set of locally available resources, are functionally equivalent, i.e., they are required to carry analogous tasks such

as packet routing. The traditional distinction between routers and end-points faints, since all nodes undertake both tasks. Finally, in common cases, MANET nodes are carried by humans or embedded in moving objects, e.g., vehicles. On the one hand, mobility increases model dynamicity; on the other hand, it aggravates communication issues, since recipient mobility may complicate message delivery. Generally speaking, due to the symmetry of all MANET participants with regards to reachability, solutions based on fixed location servers are not feasible. Hence, novel distributed routing protocols effective in mobile scenarios have been devised [110]. MANET model especially benefits applications where the deployment of a fixed infrastructure is difficult, e.g., emergency relief in impervious environments and disaster recovery, or where temporary interactions make it economically unfavorable, e.g., opportunistic entertainment networking or resource exchange. Finally, MANET are preferable for applications targeting a wide area of interest, such as urban monitoring, that would require a massive "ad hoc" deployment of base stations.

Many of these applications benefit from the access to remote resources, i.e., data or software components carried by other devices. Given reachability issues of MANET nodes, this task is particularly crucial. First, mobile nodes can leave network area without any notice, by suddenly disrupting the availability of local resources. Even worst, generally speaking mobility can lead to potential network partitions, due to the disconnection of traffic forwarders, where all nodes belonging to one network segment cannot communicate with nodes in the other. Second, resources need to be discovered and located with as low communication overhead as possible; in fact, trivial solutions based on network-wide flooding, either of resource advertisements or of resource requests, do not scale as network size increases. Node symmetry and mobility arguments prevent once again the usage of fixed servers and call for highly distributed strategies. Moreover, it is necessary to consider

that, if the network is sparse, it is not always possible to find a connected path from a user to needed remote resources. This definitely rules out flooding-based solutions and calls for the design of novel opportunistic protocols where nodes interact and exchange data only with single-hop neighbors. This paradigm enables "delayed" multi-hop interactions, where information spreads as an epidemy via several single-hop exchanges, as nodes move into direct transmission ranges. However, these controlled interactions definitely exacerbate discovery issues, demanding original resource indexing strategies.

This thesis tackles these issues and proposes middleware solutions for resource dissemination and retrieval. First of all, access to remote resources and related issues acknowledgedly affect many applications in MANET and deserve to be jointly faced. Common middleware supports significantly benefit the implementation of the application logic, by reducing development complexity and design costs. In addition, proposed middlewares operate at the application level mainly because several dissemination and retrieval decisions are possible only at this abstraction layer. Thus, application developers only need to set service parameters, while the middleware is transparently in charge of the actual operations. Finally, working at the application level also simplifies portability over heterogeneous communication technologies and routing protocols, by hiding low-layer implementation details from application developers. This particularly benefits MANET scenarios, where no acknowledged networking standard has been developed.

The primary idea behind our approach is to improve resource access by disseminating replicas. We claim that replica management is very hard to perform in an effective and lightweight way over wide-scale environments when dealing with general-purpose MANET. Therefore, we focus on a specific deployment scenario of increasing relevance

for the entertainment service market, called dense MANET. These are limited spatial regions, such as university campuses, airports and shopping malls, where many mobile wireless peers autonomously cooperate, and maintain a node density almost invariant during long time intervals. The first part of this thesis proposes REDMAN (REplication in Dense MANet), a middleware solution to manage, retrieve and disseminate replicas of data/service components among nodes participating in a dense MANET. REDMAN has the main goal of improving the resource availability, by exploiting lightweight solutions specifically suitable for the characteristics of dense MANETs. REDMAN aims at maintaining a fixed replication degree for the needed resources, notwithstanding the unpredictable movements of wireless devices (with hosted resource replicas) inside/outside the dense area, possible network disconnections, and device power shortage. To suit resource-limited nodes, REDMAN decides not to guarantee the strict any-time consistency of the replication degree of shared resources, and employs reactive strategies to counteract the reduction of replicas. In addition, to reduce the overhead and the complexity of distributed replica management, REDMAN currently manages the replication of read-only resources. This is sufficient for guaranteeing the availability of a large class of services of primary interest in MANETs (e.g., multimedia data sharing or support/application components code distribution).

In case the network is sparse, i.e., a connected sender-receiver path is not always available, its size does not allow effective network-wide communications, or nodes are too dynamic to effectively maintain network replicas, original diffusion strategies are needed. Vehicular Ad Hoc Networks (VANET), and more specifically, Vehicular Sensor Networks (VSN) represent typical cases where these conditions hold. Many car manufacturers are planning to install wireless connectivity in their vehicles to enable communications, for

the purposes of safety, driving assistance, and entertainment. VANET differentiate from MANET for high node speed (up to 30m/s) and mobility patterns relatively easy to predict, due to constraints imposed by roads, speed limits, and commuting habits. VSN can be built on top of VANET by equipping vehicles with onboard sensing devices. Unlike traditional sensor networks, VSN nodes are not subject to major memory, processing, storage, and energy limitations. The second part of this work proposes MobEyes, a solution supporting data diffusion and indexing in challenging VSN environments. Since it is usually unfeasible to directly report the sheer amount of sensed data to a centralized collector, e.g., police authority, MobEyes proposes that sensed data stay with mobile nodes. Vehicle-local processing extracts features of interest, and generates data summaries. Then, MobEyes collectors, move and opportunistically harvest summaries from neighbor vehicles. Collectors use summaries to identify, and then pump out, only the sensed data of interest from the carrying vehicles. The original MobEyes protocols for summary diffusion/harvesting take advantage of vehicle mobility and only exploit single-hop communications.

The dissertation is structured as follows. Chapter 1 provides a wide overview of MANET and, in particular, focuses on state-of-the-art research most related to our contribution. Chapters 2 and 3 introduce design guidelines and protocols of middleware services for resource replication (REDMAN) and sensed information diffusion (MobEyes). Chapters 4 and 5 provide simulative evidences of protocol effectiveness and overhead reduction, while Chapters 6 and 7 describe the design, implementation, and testing of middleware prototypes. Finally, we conclude the dissertation by summarizing the technical contribution of this work and by discussing possible future investigations.

Chapter 1

Background and Related Work

The increasing diffusion 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. These scenarios have motivated the study of MANET dynamic network model. In this distributed model, it is important to enable the access to remote resources from distant mobile devices. This raises two important issues in MANET: how to distribute contents, by maintaining reachability despite possible node movements and connection disruptions; and how to retrieve remote resources, given the absence of centralized servers. As anticipated in the Introduction these issues are exacerbated in large-scale networks, such as Vehicular Ad Hoc Networks (VANET) composed by cars moving on the roads. Due to the extremely wide deployment area, in these networks it is not always possible to find a connected sender-receiver path with decent bandwidth. This particularly undermines the feasibility of applications of high importance based on sensed data exchange, e.g., for urban monitoring. A possible solution to this issue is trading communication resources for increased delay. Single-hop opportunistic communications pave the way to leveraging node mobility for effective packet delivery.

Initially, this chapter provides an overview of background research areas, that can be

identified in the main field of Mobile Ad Hoc Networks, and its subfields of Vehicular Ad Hoc Networks, and Wireless Sensor Networks. In the following, we organize the state-of-the-art research most tightly related to our work along four primary perspectives. First, we focus on i) content distribution, and ii) distributed resource retrieval issues in MANET. Given that the second part of our work specifically deals with iii) Vehicular Sensor Networks, we propose a short review of that field. Finally, we conclude the chapter by presenting the research in iv) opportunistic networking, with a specific focus on sensed data diffusion, which will be a strong motivating case for our work.

1.1 Background Research

1.1.1 Mobile Ad Hoc Networks

Ubiquitous and pervasive computing envisions timely and temporary communications among wireless devices in environments without infrastructure support. This calls for a novel model, extending traditional single-hop cellular wireless networks. In MANET, node communications are not assumed to be mediated by base stations, but they take place on a peer-to-peer basis. Nodes are allowed to directly exchange messages with all participants within their radio transmission range (these direct communications are termed single-hop). However, in many useful cases, nodes are interested in reaching participants beyond their single-hop capability. To support these interactions, MANET propose multi-hop communications, where intermediate nodes are in charge of forwarding remote traffic. In traditional networks, these routing operations are carried by specialized powerful devices; MANET instead ideally distribute this task to all participants. This consideration can be further extended to other management functions, e.g., location services, by suggesting equality principles stating that, under a functional point of view, all nodes are equivalent and that

global operation effectiveness is possible only if co-operation takes place.

In most common scenarios, MANET are composed of portable devices, e.g., PDAs or laptops carried by humans, with limited resources, such as battery, CPU-power or storage. This highly limits their possibility to contribute to network management by, at the same time, challenging the co-operation principle on which MANET thrive. Energy issues particularly undermine MANET functionality; thus, a large amount of research addressed the design of effective trade-offs between device lifetime extension and network operations volunteering. If excepting specific scenarios, e.g., wireless sensor networks, MANET nodes are mobile. On the one hand, this increases model dynamicity, on the other hand, it exacerbates communication issues, raising the problem not only of locating the communication partner, but also of continuously renewing location information. This challenge, notably influencing routing operations, has pushed MANET research to devise protocols based on novel principles to cope with all these issues.

In the following, we shortly survey most important trends in MANET literature. The main focus of our report will be on routing area, since it has been the most actively researched: this is due both to routing protocol importance in enabling communications, and their criticality in addressing unprecedented challenges. Then, we review major problems reducing the effectiveness of transport protocol implementations, and proposed solutions. We conclude the section by discussing some issues, in particular energy saving, that are not related to any specific protcol layer, but crosswisely affect MANET design.

Given the absence of specialized router devices, MANET nodes need to establish multi-hop routing paths before any unicast communication is possible. Paths are composed of a chain of multiple one-hop links; thus, their availability critically depends on the stability of any involved link: if one link fails, the whole path is invalidated. Differently from wired

networks, link disruptions are common in MANET, mainly due to node mobility; thus, frequent maintenance operations are required.

Traditionally, MANET routing protocols can be roughly classified as topology-based (either proactive or reactive) and geographic-based. Topology-based proactive protocols, e.g., DSDV [103] and OLSR [63], derive from distance-vector and link-state [] employed on the wired network. These periodically exchange topological information, with the goal of maintaining a consistent view of the network, in spite of node movements. Distance-vector solutions, e.g., DSDV, maintain a local routing table including, for each possible destination, the one-hop neighbor nearest to it. Locally computed tables are periodically delivered to single-hop neighbors, which update their own tables by considering received information. Thus, topological changes indirectly spread one hop per step, as reflected on the newly computed tables. Routing is performed by forwarding packets to the one-hop neighbor indicated in the table as the nearest to the destination. On the other hand, link-state protocols, e.g., OLSR, locally maintain a view of the whole network. Topological changes are propagated with single network-wide packets, allowing all nodes to consequently update local view. Routing is performed by locally computing the shortest path toward the destination, and sending the packet to the neighbor on the path.

Topology-based reactive protocols, e.g., DSR [71] and AODV [105], have been originally conceived for MANET. Proactive protocols impose a high communication overhead due to the continuous table exchange needed to maintain local paths toward any possible destination. Reactive routing is based on the consideration that nodes do not generally need to communicate with all other participants, but only with few of them. Thus, they propose to setup paths only when needed. Since nodes do not have any information about the location of the destination, most of these protocols rely on a preventive flooding to discover

network paths: this obviously imposes a time delay with respect to proactive solutions. As soon as a path is setup, communication can take place, until any link breaks. In that case, path maintenance operations are performed usually based on route error messages delivered to the source, or on local repairing strategies, transparent to the source.

Previous protocols do not assume nodes are equipped with any positioning devices. Instead, geographic-based solutions take advantage of GPS to enroute messages toward the target location (i.e., of the destination). With respect to topology-based protocols, these are generally deemed stateless, since nodes do not need to setup paths and maintain routing tables, but forward packets by considering only local and target positions. Different forwarding strategies have been proposed based either on the relaying of single or multiple copies of each packet. In single-copy based, packets are generally forwarded toward the single-hop neighbor minimizing a function depending on target location (e.g., absolute, or projected distances). In multiple-copy based, delivery ratio probability is increased by forwarding packets to all single-hop neighbors minimizing (or satisfying) location functions (e.g., the angular distance from an ideal sender-receiver straight line). A drawback of these approaches is that packet source needs to know the location of the receiver: this is possible only if a location service is suitably deployed. Let us rapidly mention that these protocol families do not exahust MANET research. Recent approaches propose hybrid protocols, by combining on different scales reactive and proactive solutions, or geographic and topology-based.

Routing protocol design is not the only critical enabler to MANET networking. Experimental results showed that connection-oriented transport protocols, such as TCP, are not effective in MANET environments. In fact, TCP assumes that packet loss are only due to

congestions; as a consequence, it reacts to losses by reducing transmission rates via congestion window adjustments. This assumption is reasonable in wired network; however, wireless network medium is far less realiable than wired counterpart, and causes severe packet loss. This is well-known also from cellular wireless networks literature: solutions have been provided based on a smoother modification of the congestion window.

In MANET however, a number of additional effects further exacerbate this phenomenon [33]. Route failures, due to node mobility, produces high packet loss and abnormal delays (while the protocol tries to reconstruct the path). TCP again reacts by enacting congestion control mechanisms. Also the congestion window size is source of inefficiencies. [47] proves that TCP operates with a window size larger than the one achieving optimal throughput, increasing low-level contention. Physical and MAC-layer interferences add other inefficiencies: nodes undergo interferences of nodes within the carrier sense range, or suffer from well-known hidden and exposed terminal effects [107]. Finally, the interaction with MAC-layer protocols can lead to severe unfairness or channel capture by few flows. Several solutions have been proposed to solve these issues. [26] introduces explicit signaling from intermediate nodes in case of route failure; similarly, [57] proposes to notify intermediate link failures to the sender with Explicit Link Failure Notifications. Other approaches [47] focus on MAC-layer and TCP interactions: they observe that it is possible to determine a congestion window achieving maximum channel reuse, that is tipically smaller than the one TCP uses. [47] proposes a solution based on Random Early Detection [126] and longer backoff intervals.

Some research issues are not peculiar of any layer but should influence the design of every MANET solution. First and foremost, we recall energy saving: experimental results prove that communication is the most power-hungry activity, and that sleep mode

enabling is the only way to reduce this consumption. Thus, several MAC layer strategies [29, 136, 117] propose to turn on only the minimum number of nodes needed to keep the network connected [29, 136]. Topology control solutions [44, 94] address the optimal tuning of nodes transmission ranges, by observing that smaller ranges require more packet relays, while larger increase energy consumption as well as interference. Finally, power-aware routing protocols have been designed, enrouting messages either on paths leading to minimum energy consumption [81], or maximizing network lifetime [28]. Security is another cross-layer issue: threats span from physical layer (e.g., denial of service via radio jamming), to data-link (e.g., packet sniffing [90]), to network (e.g., dissemination of fake information to concentrate paths on a single node), to application (e.g., cooperation refusal). Specific solutions to most of these threats have been proposed [111, 20, 101].

1.1.2 Vehicular Ad Hoc Networks

The deployment of MANET nodes on vehicles represent a relevant and challenging novel scenario. Many car manufacturers are planning to install wireless connectivity equipment in their vehicles to enable opportunistic communications between vehicles. One distinct feature is that vehicles are highly mobile, with speed up to $30m/s$, though their mobility patterns are more predictable than those of nodes in MANET due to the constraints imposed by road, speed limits, and commuting habits. Therefore, these networks require specific solutions and identify a novel research area within the MANET field, i.e., Vehicular Ad-hoc Networks (VANET).

Recent research is envisioning a large number of applications specifically designed for VANETs, ranging from (i) safe cooperative driving where emergency information is diffused to close vehicles and real-time response systems automatically manoeuvr to avoid

accidents [135]; to (ii) entertainment support, e.g., aiming to enable file sharing [92], diffusion of commercial advertisements [93] and peer-to-peer marketing [78]; to (iii) distributed data collection, e.g., with the goal of providing drivers with information about available parking lots [22] or traffic jams en route [121]. So far, most VANET research has addressed routing issues. Three different and complementary routing strategies have been proposed and evaluated: unicast [96, 112], broadcast [130, 135, 75], and carry-and-forward [32, 121, 18, 134, 139]. Unicast strategies are usually applicable only when it is possible to assume the availability of an uninterrupted path between source and destination; broadcast solutions enhance message delivery reliability in more general deployment environments; carry-and-forward permits to have delivery reliability similar to broadcast with minor overhead in delay-tolerant application scenarios, even with scarce node density. Due to intrinsic difficulties in deploying large size VANETs, most protocols have been validated and evaluated via simulations, which exploit novel mobility models based on both real maps and realistic vehicle behaviors in order to accurately match the conditions of target environments [112, 96]. In particular, research efforts to identify suitable unicast routing protocols for VANET stem from the adaptation of traditional MANET protocols, often considering on-demand and geographic-based solutions given the high mobility of VANET nodes. In [96], authors evaluate AODV [105] and GPSR [73] with accurate traffic micromobility models: they find that both protocols suffer from poor packet delivery ratios when applied to VANET. Therefore, they devise improving schemes based on the identification of best forwarders, i.e., nodes placed neither too far from the sender (leading to a higher probability of link breakage) nor too close (increasing the number of routing hops).

Several VANET applications, e.g., related to safety or traffic/commercial advertising,

call for the delivery of messages to all nodes located close to the sender, with high delivery rate, and short delay. [130] evaluates how effective priority-based broadcast solutions, e.g., those included in 802.11e [1] specifications, are in enhancing the probability of reception of important messages. In high-traffic conditions, even priority is not decisive to guarantee broadcast success, since it is not able to counteract the hidden terminal problem [48]. Recent resarches addressed this issue by proposing original broadcast strategies. [135] proposes simple original solutions; results identify beneficial elements to design a reliable broadcast protocol, such as carrier sensing, fixed repetition schemes, i.e., where nodes repeat messages for a fixed number of times, and time slotting, with global clock synchronization. However, single-hop broadcast does not provide a full support to advertising applications; thus, effective multi-hop dissemination solutions are investigated in [75]. In addition to common issues affecting single-hop broadcast, multi-hop dissemination also suffers from the broadcast storm issue [98]. [75] proposes Urban Multi-hop Broadcast (UMB), where the farthest receiver is the only node responsible of acknowledging and relaying the broadcast message.

Packet delivery issues in areas with sparse vehicles have encouraged several recent research contributions to investigate carry-and-forward strategies. In [32], authors simulate a straight highway scenario to compare two ideal strategies: pessimistic (i.e., synchronous), where sources send packets to destinations only as soon as a multi-hop path is available, and optimistic (i.e., carry-and-forward), where intermediate nodes hold packets until a neighbor closer to the destination is detected. Under the implicit assumptions of i) unbounded message buffers and bandwidth, and ii) of easily predictable mobility patterns as for vehicles on a highway, the latter has demonstrated to achieve a lower delivery delay. However, in more realistic situations carry-and-forward protocols need to be carefully deigned and

tuned. MaxProp [18], part of the UMass DieselNet project [38], is a ranking strategy to determine the packet delivery order when occasional node encounters occur, as well as dropping priorities in the case of full buffers. Precedence is given to packets destined to the other party, then to routing information, to acknowledgements, to packets with small hop-counts, and finally to packets with a high probability of being delivered through the other party. VADD [139] hangs on the consideration that most node encounters will take place in intersection areas. Since many forwarding options will be typically available at intersections, effective decision strategies are proposed, highly reducing packet delivery failures and delay. Distributed data collection applications in VANET call for geographic dissemination strategies that deliver packets to all nodes belonging to target remote areas, despite possibly interrupted paths [134, 121]. MDDV [134] exploits geographic forwarding to the destination region, favoring paths where vehicle density is higher. In MDDV, messages are carried by *head* vehicles, i.e., best positioned toward the destination with respect to their neighbors. Instead, [121] proposes several strategies based on virtual potential fields generated by propagation functions: any node estimates its position in the field and retransmits packets until nodes placed in locations with lower potential values are found; this procedure is repeated until minima target zones are detected.

1.1.3 Wireless Sensor Networks

A completely different MANET model with respect to VANET, is represented by stationary wireless networks set up among small sensing devices. Recent advances in miniaturization and communication technologies have enabled the creation of Wireless Sensor Networks (WSN) consisting of a large number of low-cost digital devices, which integrate sensors, processors, and radios on a single few centimeter-wide board. The WSN primary goal is to promptly deliver final users with monitoring information of interest about the physical

world, by achieving high accuracy while meeting the specific limitations of WSN nodes. Notably, WSN generally operate in harsh environments preventing the deloyment of any wireless communication support infrastructure.

To form an adequate and reliable sensing base, WSN generally include a large number of devices. In fact, theoretical results in the literature prove that the coordinated behavior of hundreds/thousands of WSN nodes spread near the interesting phenomenon to monitor can highly improve the sensing accuracy if compared with centralized macro-sensors [106]. However, the exploitation of a large number of devices definitely precludes the possibility for human operators to manually configure and manage WSN: there is the need for self-configuration and self-organization solutions (WSN should have the ultimate goal of being pervasive and disappearing), also capable of adapting the WSN behavior in the case of variations of the monitored environment and of the monitoring requirements.

Also because of their deployment in remote and hostile terrains, WSN devices are usually severely limited in their on-board resources. First of all, they are outfitted with batteries that, once depleted, cannot be easily replaced. Therefore, the reduction of energy consumption is crucial in WSN solutions and applications. As in the MANET case, recent experimental evaluations have shown that, with current technologies, communication is the most power-consuming task: to extend WSN lifetime, it is necessary to carefully reduce both the number of transmissions and the amount of transmitted data. To this purpose, lightweight node coordination is emerging as a crucial issue, e.g., via collaborative in-network processing to enable the automated fusion of sensed data and the transmission of only concise and aggregated monitoring indicators, only when needed depending on the sensing goals. In addition, energy limitations and reduced sensor size impose further constraints, such as limited CPU power and memory.

WSN solutions can apply to several different application domains. Environmental monitoring offers a rich collection of possible sensing scenarios: well-recognized application areas include air/water/soil chemical analysis and the support/validation/feedback of the development of complex ecosystem models. The deployment of hidden sensors fits surveillance and military tasks, such as detecting objects overstepping a fixed perimeter or tracking mobile vehicles. In addition, home automation and smart houses can significantly benefit from WSN-based control of lighting and heating systems. Moreover, envisioned critical applications, e.g., patient health monitoring, pose hard concerns about WSN additional properties, such as reliability and security, thus suggesting further on-going research activities and investigations.

WSN research has addressed issues at every protocol layer: we rapidly sketch several interesting results. MAC layer WSN protocols aim at creating link-layer topologies among nodes in direct visibility and at supporting the access to the communication channel. The SMACS protocol [120] carries on at the same time link-layer topology discovery and creation, and channel access organization. It is essentially based on a TDMA algorithm, and exploits a distributed, local scheme to schedule nodes channel accesses. Topology control solutions aim at identifying redundant nodes that can be powered off without affecting the system capability of matching user requirements, e.g., neighbor nodes alternatively supporting the same routing paths [114]. Similarly to the MANET case, the routing layer has catalyzed most research efforts: the design of dissemination protocols for sensed monitoring information has generated very interesting proposals. Again, we can roughly classify approaches, with regard to the type of task they support, in proactive, reactive, and hybrid. Proactive protocols [55, 83] aims at periodically returning sensed attribute values, by keeping platforms off all remaining time to save energy. By properly tuning the report interval

rate it is possible to trade between a prompt solution (short report intervals) and a more conservative one (long report intervals). The former has the advantage of transmitting critical data with low delays. Unfortunately, it reveals energy-inefficient because it probably reports also irrelevant data. The latter solution conserves energy but requires large delays in critical data delivery. Reactive protocols [86] deliver data only if a sensed attribute exceeds a threshold value specified by the user. On the one hand, this improves delivery latency; on the other hand, if the threshold is never reached, the user cannot determine neither if the network is still active nor all nodes have died. Hybrid protocols [87, 62] represent a more well-balanced approach. In fact, users can thoroughly control networks behavior by tuning a report interval and a threshold value: the network can emulate a proactive one by increasing the report rate; by decreasing the same parameter, it can emulate a reactive one. By employing this solution, heavy traffic burdens, typical of proactive networks with short report intervals, do not burden the system. At the same time, the network is quick to react to most important environmental changes because nodes transmit new data whenever an attribute exceeds its threshold. A hybrid approach deserving to be mentioned is Directed Diffusion [62], aiming at establishing paths between data sources and sinks by exploiting only localized interactions. In Directed Diffusion, interests (specifying data threshold values as well as periodic report timelines) diffuse all over the network, establishing back-path routes on intermediate nodes called gradients. These are exploited to return relevant data to the interest originator. At the application layer, novel WSN scenarios open brand new perspectives and technical challenges; the most investigated application areas relate to environmental monitoring, e.g., ARGO project [3].

1.2 Related Work

In the first part of this chapter, we provided a wide overview of the challenging MANET field, and its VANET and WSN subfields. In the reminder, we will focus our attention on specific issues tightly related to our contribution, i.e., distribution and retrieval of contents with regards to the resource replication service for dense MANET, and vehicular sensor networks and opportunistic networking with regards to the sensed data dissemination in VANET.

1.2.1 Content Distribution

The continuous growth of Internet traffic has promoted the idea of increasing availability by replicating service contents depending on popularity [45, 13]. Resource replication is even more crucial in MANETs where continuous node availability is unfeasible: replication is essential both to maintain resource availability in the case of MANET partitioning and to reduce access latency and battery consumption by placing replicas close to requesters. However, due to the novelty of MANET service provisioning, a very few approaches have already emerged.

A couple of proposals address latency reduction for resources on the fixed Internet and accessed by MANET nodes. [113] describes a cooperative strategy for caching Web contents in wireless localities composed by mobile terminals with also cellular-phone capabilities. A similar goal is addressed in [46], which tends to decrease access latency of Web Services and to optimize energy consumption at the same time: each participant maintains a local cache with recently accessed Web Services components; when searched components are not in cache, the requester first explores the ad hoc network and, as the last chance, exploits long-range telecom connectivity.

In very dynamic MANET environments it is unfeasible to assume that long-range base stations, e.g., connecting to GPRS/UMTS core networks, are always available. In any case, multi-hop packet forwarding exacerbates MANET networking issues, especially when exploited to download multimedia flows and large files [127]. SPAWN proposes a cooperative strategy for content delivery in Vehicular Ad Hoc Networks (see 1.1.2) where gateways are only intermittently available [92]. Client nodes start downloading files from gateways installed on freeway service stations while the client is within the gateway coverage area. If not yet terminated before losing the gateway connectivity, the download can continue with a peer-to-peer strategy: SPAWN leverages gossip-based control message exchanges to discover cars that own missing file chunks; strategies based on chunk rareness and replica distance are used to select which file pieces to download first.

By considering pure MANETs, Chen et al. propose a well-known solution to counteract network partitioning [7]. They place resource replicas on the basis of node positions/movements, by assuming that all nodes are GPS-equipped and aware of their physical positions. To the best of our knowledge, [14], [54], and [24] are the research proposals more related to our from the point of view of content distribution. [14] enhances data availability via an adaptive replication protocol suitable only for deployment scenarios with nodes in direct single-hop visibility. In addition, it assumes that any node knows the position of all replicas and the memory/battery/mobility state of all other nodes. [54], instead, proposes proactive replication depending on resource access frequency; however, its coordination and synchronization solutions do not scale well in wide deployment scenarios. Finally, Cao et al. propose collaborative caching of nodes along client-to-server paths during resource forwarding with the goal of reducing resource access latency [24]. Due to the number of alternative paths connecting any node pair, the solution is not effective when

applied to unstructured, non-hierarchical, and wide-scale MANETs such as dense ones.

1.2.2 Resource Retrieval

Infrastructure-free and completely decentralized MANETs pose innovative challenging issues also for resource retrieval. A very few research activities have investigated MANET-specific broadcast-based peer-to-peer solutions for resource retrieval, with limited scalability and excessive overhead for resource-constrained clients [56]. Some recent proposals aim at limiting broadcast communications by exploiting quorum-based solutions [5, 15, 124]: the primary idea is to disseminate resource placement information on a node subset that can be determined without message flooding.

Some interesting research results on resource retrieval have been achieved in the partially similar deployment scenario of stationary wireless sensor networks. [5] assumes GPS-equipped nodes to spread advertisement/search packets along orthogonal directions. On the contrary, [15] considers provisioning environments where geographic routing cannot apply and proposes the dissemination of 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.

Some of the above work is suggesting solution guidelines that can also apply to MANET environments. [128] extends the GPS-based solution in [5] for replication in mobility-enabled deployment scenarios: each node is asked for storing placement data about its nearest replica. A different approach is presented in [124]: any resource associates with its origin place, which is in charge of distributing all replicas to nodes placed at k-hop distance from it. In that proposal, replica retrieval exploits geographical routing to forward queries toward the resource origin place.

1.2.3 Vehicular Sensor Networks

Vehicular Sensor Networks (VSNs) can be built on top of VANET by equipping vehicles with onboard sensing devices as shown in Figure 1.1. VSN are emerging as a new network paradigm for effectively monitoring the physical world, especially in urban areas where a high population of vehicles, working as mobile sensors, is expected to be always present [76]. Vehicles are typically not affected by strict energy constraints and can be easily equipped with powerful processing units, wireless transmitters, and sensing devices even of some complexity, cost, and weight (chemical spill detectors, still/video cameras, ...). Let us note that VSN represent a significantly novel and challenging deployment scenario, relevantly different from more traditional wireless sensor network environments, thus requiring innovative specific solutions. In fact, differently from wireless sensor nodes, vehicles usually exhibit constrained mobility patterns due to street layouts, junctions, and speed limitations. In addition, they usually have no strict limits on processing power and storage capabilities. Most important, they can host sensors that may generate sheer amounts of data, such as multimedia streaming recorded by cameras, thus making inapplicable data reporting solutions already known in the wireless sensor network literature.

The typical scale of a VSN over wide geographic areas (e.g., thousands of nodes), the volume of generated data (e.g., streaming data), and mobility of vehicles make it infeasible to adopt traditional sensor network solutions where sensed data tends to be systematically delivered to sinks using data-centric protocols [62]. To the best of our knowledge, the only research work dealing with these issues is MIT's CarTel project [25]. In CarTel [61] users submit their queries about sensed data on a *portal* hosted on the wired Internet. Then, an intermittently connected database (*ICEDB*) is in charge of installing queries on vehicles and of receiving replies via opportunistic communications that take place as vehicles move

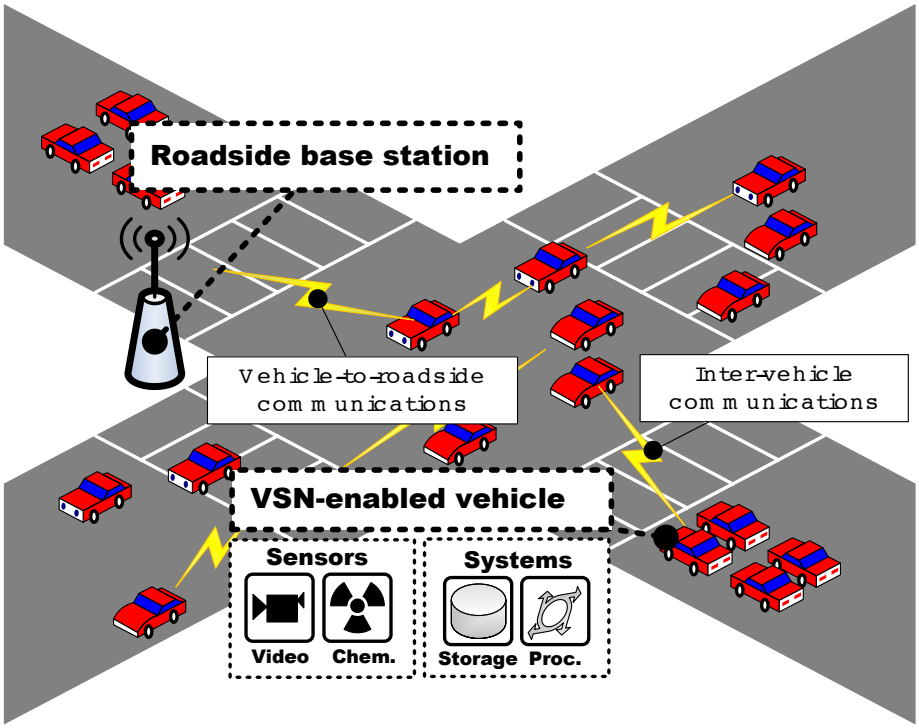


Figure 1.1: Vehicular Sensor Networks

in the proximity of open access points, i.e., Internet-connected wireless gateways with unrestricted access for passer-by users.

1.2.4 Opportunistic Networking

Opportunistic data dissemination is primarily based on data diffusion via continuous single-hop broadcast advertisements [74, 133]. As nodes encounter, they mutually exchange memory contents, thus favoring data spreading over a large portion of network participants. Node mobility and churn catalyze data diffusion process to as many nodes as possible, with the final goal of reaching intended destinations; in particular, this enables communications between disconnected nodes in sparse networks. In Opportunistic Report Dissemination (ORD) [118], a mobile node spreads reports in its local storage to encountered nodes and

obtains new reports in exchange. For each resource type, a node keeps the top-k relevant reports based on an exponential decay relevance function, taking into account report longevity and generation place. Similarly, in Autonomous Gossiping (AG) [36], a node can set a profile by choosing resource categories of interest; then, information is selectively disseminated with profile matching strategies. Unlike previous schemes, MAID [6] and MPR [91] restrict dissemination such that only the source node can disseminate its data to local neighbors. This choice leverages the observation that mobility increases throughput, even if at the price of increased delivery delay [52].

Recent research projects addressed sensed data diffusion by implementing opportunistic networking in the application field of naturalistic environment monitoring, namely ZebraNet [72] and SWIM [119]. ZebraNet addresses remote wildlife tracking, e.g., zebras in Mpala Research Center in Kenia, by equipping animals with collars that embed wireless communications devices, GPS, and biometric sensors. As the animals drift within the park, their collars opportunistically exchange sensed data, with the goal of pumping it toward base stations. ZebraNet proposes two communication protocols: a flooding-based approach where nodes exchange all the data within their buffers (either locally generated or received from other animals) with neighbors, and a history-based protocol where data is uploaded only to nodes with high probability of encountering base stations. Shared Wireless Infostation Model (SWIM) [119] addresses sparse mobile sensor networks with fixed Infostations. Sensed data is epidemically disseminated via 1-hop flooding to encountered nodes and offloaded when Infostations are in reach.

In the field of urban monitoring, opportunistic networking principles are applied in Dartmouth's MetroSense [89, 42]. [42] describes a three-tier architecture for MetroSense:

servers in the wired Internet are in charge of storing/processing sensed data; Internet-connected stationary Sensor Access Points (SAPs) act as gateways between servers and mobile sensors (MS); MS move in the field opportunistically delegating tasks to each other, and support gathering by "muling" [115, 82] data to the SAP. We remark that, differently from previous solutions, MetroSense requires an infrastructure support, consisting of Internet-connected servers and remotely deployed SAPs.

Let us observe that, broadly speaking, opportunistic networking principles do not restrain to single-hop epidemic relay. It is possible to classify as opportunistic also those approaches that "serendipitously" take advantage of resources as they become available [109, 39, 19]. These can be classified as opportunistic approaches because of the exploitation of already present sensing devices, such as cameras in mobile phones, symbiotically attached to mobile entities. Application-level protocols for the resolution of queries about sensed data have been proposed in [109, 39]. [109] describes three middleware solutions: Contory abstracts the network as a database, and resolves declarative queries; Spatial Programming hides remote resources, such as nodes, under local variables, thus enabling transparent access; finally, Migratory Services are components that react to changing context, e.g., the target moving out of range, by migrating to other nodes. [39] presents VITP, a query-response protocol to obtain traffic-related information from remote areas. The source injects a query in the environment, specifying an area where the query should be moved for resolution. Nodes belonging to the target area form a sequential Virtual Ad Hoc Server: they check if they can reply to the query and relay to neighbors. Query routing is out of the scope of both [109] and [39]. Among recent research projects, let us also mention Intel IrisNet [51] and Microsoft SenseWeb [95] which, even if partially related to MobEyes, still address opportunistic sensing issues. Both projects investigate the

integration of heterogeneous sensing platforms in the Internet via a common publishing architecture. Finally, we point out CENS' Urban Sensing project [132, 19]: this is a recently started multi-disciplinary project addressing "participatory" sensing, where applications receive data from mobile sensors operated by people.

Chapter 2

The REDMAN middleware for resource replication in dense MANET

In MANET scenarios, where users are allowed to freely move, resource availability and accessibility is hardly challenged. In fact, if users carrying important information and services leave the network, their resources become immediately unavailable for all MANET participants. REDMAN aims at providing the dissemination of resource replicas among nodes, so that any client could access at least one replica in its vicinity at any moment and anywhere in the network site. Resource availability should be maintained notwithstanding unpredictable movements of wireless devices (with hosted resource replicas) inside/outside the area, possible network disconnections, and device power shortage.

We claim that replica management is very hard to perform in an effective and lightweight way over wide-scale environments when dealing with general-purpose Mobile Ad hoc NETworks (MANETs) [33]. Therefore, we focus on a specific deployment scenario of increasing relevance for the entertainment service market, called dense MANET and defined as 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,

airports, and university campuses;

- has a node density, i.e., the average number of wireless nodes at single-hop distance from any dense MANET participant, almost invariant during long time intervals.

The assumption of relatively high and constant node density (nodes can unpredictably move in/out the dense region, even with high frequency, but the number of nodes in the dense MANET does not relevantly change) permits to exclude network partitioning and subnetwork merging at provision time. This assumption is not too restrictive: it is valid in most entertainment service provisioning environments of commercial interest.

In addition, we are interested in disseminating replicas of read-only resources (still/moving images, audio streams, HTML web pages recently downloaded from the Internet) or, anyway, with no consistency requirements in the case of replica modification. This kind of replica dissemination is suitable not only to inject data but also to distribute the code of support/application components, e.g., driver updates, format-specific players/renderers, and game clients.

We claim the need for distributed middleware solutions to disseminate resource replicas among wireless cooperating nodes with no impact on the implementation of the application logic of dense MANET services. Replica management in dense MANETs would significantly increase the complexity and the costs of designing, developing, and deploying applications that directly deal with replica distribution and maintenance, thus slowing down their wide-spread usage. Middleware solutions can significantly facilitate the work of entertainment service developers who should only describe the involved resources and inject a single copy of them, while the middleware is transparently in charge of replica management. In addition, middlewares should operate at the application level because several replica management decisions, such as the proper replication degree depending on resource criticality,

are typically at this abstraction layer. Working at the application level also simplifies portability over heterogeneous communication technologies and routing protocols [50].

Novel middlewares for replica management in dense MANETs should primarily answer three crucial requirements:

- producing a very limited network/computing overhead,
- well scaling in large deployment environments, and
- being accurate enough even if adopting lazy-consistent heuristics.

The addressed deployment scenarios may involve several hundreds of collaborating wireless nodes. To achieve scalability, it is essential that middlewares adopt novel and completely decentralized solutions for replica dissemination, maintenance and retrieval, where participants perform replica management operations as much as possible in an autonomous way. In other words, middleware solutions should act locally whenever possible, by involving only a limited number of participants in their proximity.

The goals of limited overhead and wide scalability suggest the adoption of heuristic-based non-optimal solutions for replica placement, retrieval, and degree maintenance in presence of mobility. However, novel middlewares should be accurate enough to make all entertainment resources available and efficiently retrievable in terms of both search time and traffic. A suitable trade-off between accuracy and overhead is crucial in dense MANET deployment scenarios.

The clients involved in collaborative replica dissemination are usually battery/memory-constrained devices, which typically cannot host positioning hardware and cannot permanently store all needed data/service components in their local memory. Middleware solutions should be lightweight under several perspectives: they should impose very limited

network traffic and local processing, so to preserve network bandwidth and node battery; in addition, they should be modular to allow the dynamic installation of only the middleware/service components actually required at each node.

2.1 Practical case studies

Let us start by presenting actual deployment scenarios for entertainment services to practically show the motivations of our proposal. Large sport events, such as Formula One Grand Prix or Winter Olympic Games, will have several thousands attendees and official delegates distributed in a limited area, many of them carrying wireless portable devices, from Wi-Fi personal digital assistants to Bluetooth-enabled phones. The organizers could be interested in providing that large number of people with entertainment services while they are waiting or during the event, to enable them to access previous results, pictures, and short multimedia streams, e.g., about significant episodes just occurred in a remote part of the circuit or in other Olympic areas. Another interesting scenario is represented by travelers with Personal Digital Assistants (PDAs) who are interested in playing a strategy game while spending their time in a waiting room of a railway station. Waiting rooms are usually crowded places, where travelers come and go, and the number of people co-located there is almost constant notwithstanding continuous arrivals and departures. Several travelers could be keen on spending their waiting time by playing videogames.

All these scenarios are suitable to represent dense MANETs, while each of them emphasizes distinctive issues:

- stadium/stand-like environments, with a huge number of possibly mobile users co-located in a relatively small area, where it is possible to deploy Wi-Fi/Bluetooth-enabled Internet access points;

- alpine ski run-like environments, where several mobile users are distributed in a quite large spatial region, typically with no possibility to cover the whole area with wireless access points;

- station/airport waiting room-like environments, where travelers independently come and go, interacting within short time frames.

In the first case, entertainment service provisioning represents a technical challenge mainly because of scalability issues. It is not viable to exploit multimedia streaming/adaptation servers running in fixed Internet hosts and wireless access points to provide spectators with Internet connectivity. Only to mention a basic scalability issue, some research activities have shown that, to achieve usable performance for multimedia distribution, the number of concurrent clients for a single access point should be largely less than one hundred for IEEE 802.11b and less than 6 for Bluetooth [21]. In Formula One stands (with tenth thousands seats), this would require installing several hundreds/thousands of Wi-Fi/Bluetooth access points, no longer useful after the race. In the second case, in addition to scalability concerns, the large and mountain-type area makes definitely unpractical the installation of even a few access points, which require power lines and wired Internet connections. The waiting room case likely reduces scalability constraints but further exacerbates network dynamicity issues (however affecting also the previous scenarios).

Our proposal is to provide the lightweight REDMAN middleware to support entertainment services in the above challenging environments, with no need of any statically deployed network infrastructure. In all depicted scenarios, attendees could be interested in requesting different types of “official” information provided by the organizing committee (competition results, moving/still images related to preferred athletes, ...) and also in sharing “unofficial” data or software components directly collected on the field by other

onlookers (digital pictures, but also videogame playing sceneries). On the one hand, the organizers could either distribute devices with REDMAN already installed or upload it on the portable terminals of attendees who are willing to cooperate in resource sharing during the event. Then, they could distribute still/moving images of just-ended events at any time by injecting one copy and by specifying the desired replication degree; spectators' devices could efficiently retrieve and access resources among their peers. On the other hand, travelers PDAs cannot have all the desired game components statically pre-installed, also because of their usual memory limitations. Nonetheless travelers could be interested in playing new videogame sceneries dynamically discovered in the community of cooperative travelers and downloaded to their devices, if allowed; moreover, some videogames have a distributed implementation and require the continuous availability of one or more servers. REDMAN can help discovering and obtaining remote components, but also maintaining needed running servers in the dense MANET, independently of unpredictable movements of the subset of nodes that host server components.

2.1.1 REDMAN at work: game playing at the railway station

To practically introduce the REDMAN middleware functions, let us consider travelers with REDMAN-enabled Personal Digital Assistants (PDAs) interested in playing a strategy game in a waiting room. In this scenario, REDMAN enables the lightweight and transparent distribution of replicas of game components (with the desired resource-associated replication degree) to a subset of devices in the waiting room. In addition, REDMAN supports the dynamic retrieval of needed resources, thus providing travelers with a set of locally available games that dynamically grows depending on the new resources that visiting travelers bring into the waiting room and decide to share with other dense MANET

cooperating nodes. For instance, consider the well-known single-player Civilization strategy game [116] and let us illustrate how REDMAN can distribute replicas of Civilization game components, to support game availability in the dense MANET with no need of explicit and static installations, even if mobile nodes continuously enter/exit the waiting room. Consider the example of deployment scenario depicted in Figure 2.1. Suppose that a device enters the dense MANET by carrying new game components (the Civilization server engine, the Civilization lightweight client, and some playing sceneries). If the game is considered of interest, REDMAN transparently works to guarantee, via replication, the availability of the different Civilization components in the dense MANET, independently of the unpredictable mobility of nodes hosting the component replicas.

Before discussing a practical sample situation, let us observe that REDMAN classifies node responsibilities according to two main roles:

- *resource delegates* are dense MANET nodes that host resource replicas, reply to retrieval requests for hosted replicas, and participate in resource dissemination;
- *replica managers* are dynamically elected dense MANET nodes in charge of determining and maintaining the proper replication degree for the associated resources.

When the delegate D with the Civilization server engine enters the waiting room, the REDMAN middleware component running on D's device sends the descriptions of D's resources to the REDMAN replica manager M, running in another node of the dense MANET (message 1). The replica manager is in charge of enforcing the needed replication degree and of maintaining information about shared resources replicated in the dense region (Shared Resource Table - SRT). For any shared resource, the corresponding SRT entry includes the associated target replication degree and weakly consistent information about the nodes where the resource is currently replicated. If M decides that D's resources should

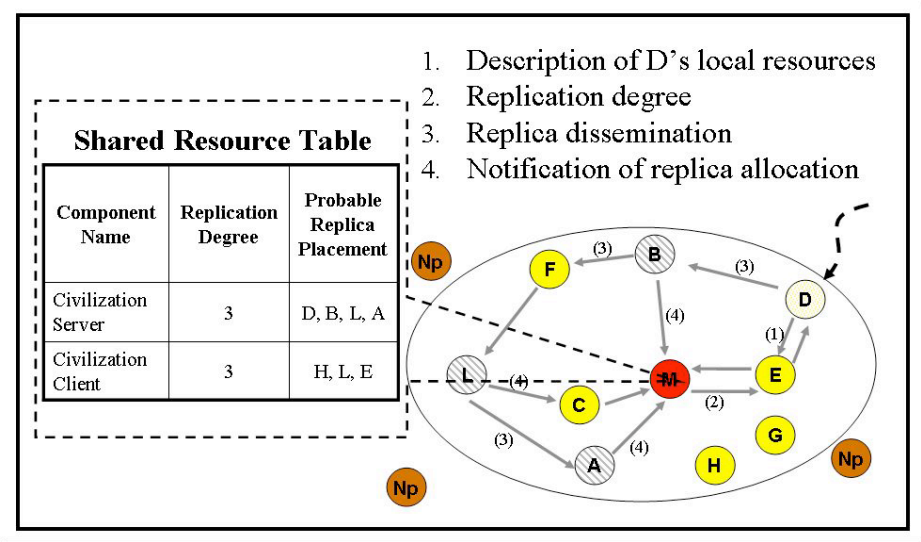


Figure 2.1: REDMAN-based replication of the Civilization server

be replicated, it commands D to start the replication operations (message 2). D reacts by forwarding a resource replica to a randomly chosen neighbor (message 3 to B), according to a possible replica distribution protocol detailed in the following 2.4, by including the still requested number of resource replicas in the message. If the neighbor accepts to locally store the resource, it makes a local copy and recursively forwards the message, by decreasing the number of replicas yet to be instantiated. Otherwise, it only forwards the message. In the figure, node F refuses to host a server engine replica, while nodes B, L, and A accept to cooperate and notify their decisions to M (message 4). Let us rapidly observe that, for sake of clarity, in this example we have chosen to spare details of the distribution strategy REDMAN provides2.4.

Beyond resource replication and discovery/retrieval issues, the sketched case study already points out two additional technical challenges for replication in dense MANETs: how to dynamically identify the nodes belonging to the dense region, given that the nodes continuously move, and how to suitably choose the replica manager node, e.g., by taking

into account its position in the dense MANET topology. These two issues are at the basis of all other REDMAN middleware features, which directly relate to the actual distribution and retrieval of resource replicas. The addressed deployment scenarios require considering highly decentralized and lightweight solutions for dense MANET configuration and resource replication, capable of scaling well in execution environments with several hundreds of mobile wireless peers. Therefore, it is necessary to design and implement original mechanisms specialized for the peculiar characteristics of dense MANETs.

In addition, to maintain the replication degree unchanged after first replica distribution, the REDMAN replica manager should react to possible exits of Civilization delegates from the dense MANET: it should be notified of such kind of events, should understand the replicas of which resources are leaving, should identify other delegates for those leaving resources still available in the dense region (via the corresponding SRT entries), and should command new replicas. Note that there is not the need to guarantee that, for any replicated resource, at any moment, the desired replication degree is exactly enforced; it is sufficient to try to maintain the replication degree in a lazy consistent way, by avoiding that all the replicas of a resource leave the dense region before performing a further distribution of resource copies.

When a new user enters the waiting hall, she can decide to play Civilization even if she has not yet installed the needed game components on her PDA. REDMAN supports her distributed discovery to understand which games are currently available in the dense MANET and which nodes host the desired resource replicas. Then, REDMAN is in charge of downloading both the Civilization client and the playing scenery to her device. Once downloaded them locally and once identified via REDMAN a suitable node with a Civilization server engine replica, the client can interact with the remote engine by adopting

usual MANET communication protocols, by bypassing the REDMAN middleware intermediation. The assumption of dense MANET guarantees that network partitioning will not occur during service provisioning. In the case that the used engine delegate leaves the waiting hall, the client can ask the REDMAN middleware for retrieving another engine replica in the dense MANET; then, the client can restart her gaming session by connecting to the newly retrieved replica.

Let us remark that the above scenario only represents a possible REDMAN use case. Any other service where resources to replicate are read-only data (the list of train departures/arrivals, of movies on in town, of utility phone numbers, and of pictures taken at a rock concert that fans would like to share with other people among the public) is a simplification of the sketched case study, with no need of dynamic composition and distributed coordination of service components.

2.2 The REDMAN middleware

According to the above guidelines, we have designed and implemented the REDMAN application-level middleware [10, 8]. REDMAN disseminates replicas of common interest resources and maintains their desired replication degree, independently of unexpected node exits from the dense MANET. 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 k, REDMAN is in charge of instantiating and distributing k replicas of it, and of maintaining the k 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 protocols that achieve approximated non-optimal solutions, e.g., they do not guarantee the strict any-time consistency of replication degree. In addition, to reduce the overhead and the complexity of distributed replica management, REDMAN 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 already pointed out in the previous sections. 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 [34].

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 low-layer implementation details from application developers. Entertainment service developers can simply benefit from REDMAN replica management facilities via very little modifications on their application code: they only need to include Resource Description Framework-based [37] descriptors for resources to share, to specify the suggested replication degree, and to invoke methods of the REDMAN API to command service-transparent resource replication and retrieval.

Figure 2.2 depicts the REDMAN middleware architecture organized in two layers of facilities: Dense MANET Configuration (DMC) includes mechanisms for participant identification and manager election; on top of DMC, REDMAN provides facilities for Replica Distribution (RD), Replica Retrieval (RR), and Replica Degree Maintenance (RDM).

The **DMC** facility is in charge of determining the participants of the dense MANET

$DM(n)$, i.e., the subset of nodes that have more than n nodes at single-hop distance. In addition, DMC optimistically identifies when nodes enter/exit the dense MANET and triggers the dynamic election of replica managers by trying to minimize the number of hops required for their messages to reach any dense MANET participant.

The **RD** facility operates to transparently distribute resource replicas in the dense MANET. When a delegate enters a dense region, it communicates the metadata of its shared resources to the replica manager, which decides the replication degree to enforce, creates new SRT entries for newly arrived resources, and commands the delegate to start the replication operations. REDMAN distributes resource replicas to distant nodes located at the endpoints of straight paths (as described in Section 2.4).

The **RR** facility has the goal of effectively retrieving resource replicas, by exploiting lightweight distributed search protocols. RR provides simple retrieval solutions, whose effectiveness mainly depends on the chosen diffusion strategy. The REDMAN guideline is to exploit the process of replica distribution itself to disseminate also placement data in a lightweight way. In the RR phase, when a dense MANET node receives a search message, it directly replies to the searcher if it owns either a copy of the needed resource, or the associated placement information. Otherwise, it forwards the request according to the SID strategy described in Section 2.5.3.

The **RDM** facility works to maintain unchanged the replication degree enforced for each shared resource, without strictly guaranteeing consistency: it is possible to have time intervals when the requested replication degree differs from the actual number of replicas in the dense MANET. RDM mainly works by reacting to resource delegates leaving the dense region. To face also delegate abrupt and non-detected failures/exits, at large time intervals RDM checks the number of available replicas and possibly commands the manager to

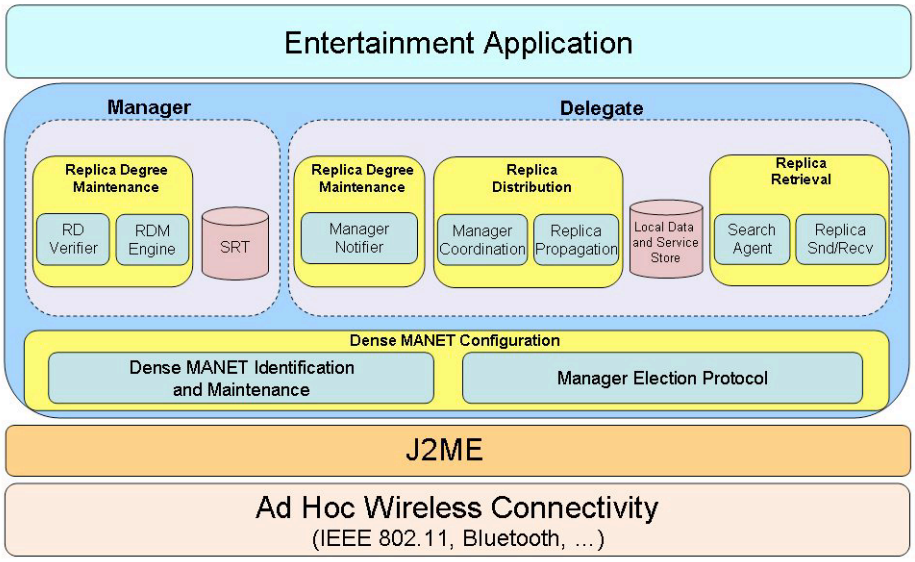


Figure 2.2: REDMAN modular architecture.

coordinate the distribution of additional copies.

In the following, Sections 2.3, 2.4, 2.5, 2.6 delve into the detailed technical description of the design and implementation choices followed by REDMAN original solutions for, respectively, the DMC facility layer and the RD, RR and RDM ones.

2.3 Dense MANET Configuration in REDMAN

The REDMAN low-layer DMC [11, 9] facilities are in charge of determining which nodes belong to the dense MANET and which ones among them have to play the role of replica managers. These low-layer mechanisms 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.

2.3.1 Dense MANET Identification

REDMAN proposes an original solution to dynamically determine the nodes that are currently willing to 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 or not. 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 discovers the number of its neighbors by exploiting simple single-hop broadcast discovery messages.

By delving into finer details, 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 original REDMAN protocol for dense MANET identification by broadcasting a discovery message that includes the number of neighbors (NoN) required to belong to the dense region and the identity of the sender. That number can be autonomously decided depending on the desired degree of connectivity redundancy: typically, a value between 10 and 20 ensures a sufficiently large set of alternative connectivity links. When receiving the discovery message, each node willing to participate replies by forwarding the message to its single-hop neighbors, if it has not already sent that message, and by updating a local list with IP addresses of detected neighbors. The current DMC implementation assumes that each participant node adopts a zero configuration solution, such as [104, 97]; guaranteeing IP address uniqueness is out of the scope of our research. After a specified time interval, any node autonomously checks whether its list contains more than n nodes, and autonomously decides whether it belongs or not to the dense MANET.

Let us observe that discovery broadcasts could provoke packet collisions; the problem

is well-known in the literature and usually identified as the ”broadcast storm” issue [98]. In order to avoid the problem, any REDMAN node defers node broadcasts of a random and limited time interval. Techniques for collision and overhead reduction such as the ones presented in [98, 63] do not fit dense MANET identification, since local density evaluation is based on the assumption that each node receives messages from all its neighbors. The dense MANET identification protocol has demonstrated to scale well also in large deployment scenarios, with a completion time linearly dependent on the dense MANET diameter, i.e., on the maximum hop distance among any couple of nodes belonging to the dense region, and an overall number of exchanged messages equal to the number of dense MANET participants, as illustrated in Section 4.

Given that dense MANET nodes can move during and after the identification process, the proposed algorithm achieves an approximated solution and requires including a lightweight lazily-consistent maintenance phase. Nodes periodically exchange Hello packets with their neighbors; each node receiving a Hello message records its source in a table entry, with an associated timeout; next Hello packets 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.

Let us rapidly observe that dense MANETs are virtual and dynamic group organizations of MANET nodes. Therefore, there is also the possibility to have more than one coexisting dense MANET, each one with a different NoN, involving the same MANET peers, e.g., in the case that several initiators trigger the dense MANET identification process with different NoN values. However, since the addressed deployment scenarios generally exhibit high density gradients close to boundaries, several different NoN values usually determine

the same $DM(NoN)$ sets. Given the limited applicability and the additional overhead of maintaining more than one dense MANET over the same set of physical nodes, the current REDMAN implementation does not support multiple overlapping dense MANETs with different NoN: the NoN value for a dense area cannot be changed for a long time interval after the first identification procedure.

2.3.2 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, the REDMAN middleware works to assign the manager role to 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. The proposed protocol for replica manager election has not the goal of finding the optimal solution: the idea is to exploit 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 that goes from the considered IN to its farthest node. 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 2.3 shows a practical 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 (I4 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 (A3). At the next iteration, the protocol explores node D, which is chosen as replica manager by respecting the termination criteria described in the following of the section. Node D can reach any other node in the depicted dense MANET with a maximum path of two hops. Let us observe that REDMAN provides a simple way to react also to manager exits from the dense MANET. If the manager realizes it is going to exit, e.g., because its battery power is lower than a specified threshold, it delegates its role to the first neighbor node found with suitable battery charge and local memory. In the case the manager abruptly fails, any resource delegate that senses the manager unavailability can trigger a new election procedure.

After having informally introduced the main guidelines of the protocol, let us now precisely specify how the manager election works. The protocol considers the initiator as the first IN. Then, it re-iterates the farthest node determination process (see 2.3.2) to evaluate other promising nodes until it reaches a solution which is considered satisfying according to the adopted heuristics. 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

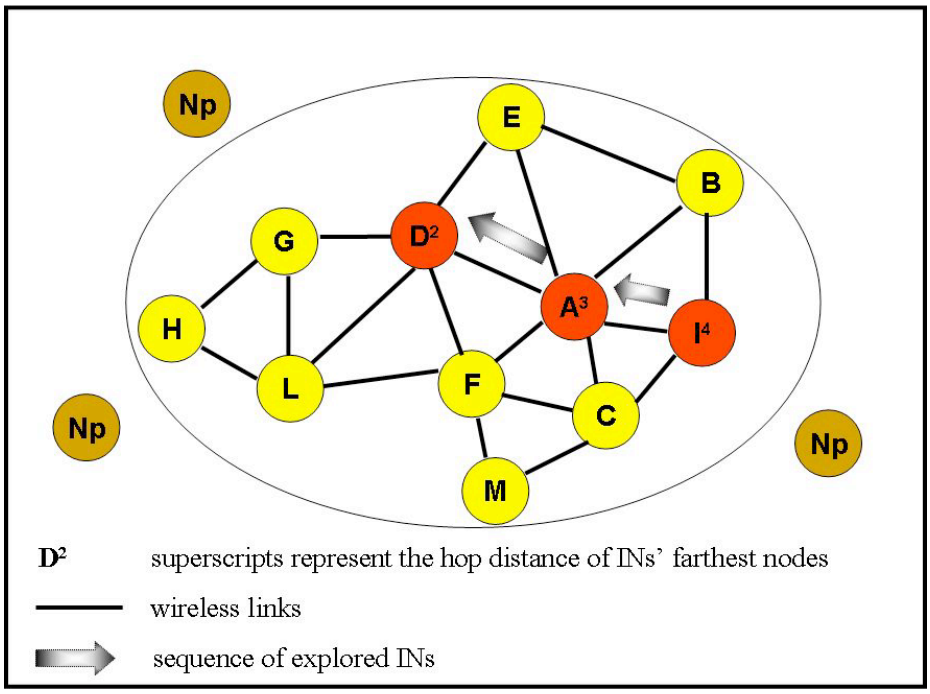


Figure 2.3: REDMAN exploring the sequence of INs $I \longrightarrow A \longrightarrow D$.

the direction of its farthest nodes (forwarding neighbors); and iii) it autonomously chooses the next IN among all the unexplored forwarding neighbors of already explored INs with lowest associated values.

To take possible device heterogeneity into account, REDMAN promotes the exploration only of INs suitable to play the role of replica manager once elected. For instance, if a potential IN device has insufficient memory and too low battery life (if compared with configurable REDMAN thresholds), it is excluded from the manager election protocol. The protocol ends when either the REDMAN heuristic criterion of Section 2.3.2 determines there are no more promising nodes, or the current $INvalue = MinInt((worstexploredINvalue)/2)$, 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.

The frequency of node entering/exiting dense regions, as well as their speed, are generally low in the addressed dense MANET deployment scenarios if compared with the time interval to complete the election protocol operations. This consideration has led us, at first, to adopt the simplifying assumption of fixed nodes, by approximating the actual deployment environment with a stationary ad hoc wireless network. However, also the REDMAN simple protocol for manager election can tolerate node mobility to some extent. First, let us observe that the value of each explored IN is determined exclusively by anyone of its farthest nodes. Thus, within each exploration round, it is necessary that only the current IN and one of its farthest nodes do not change their distance in the network topology. Moreover, since our middleware does not assume that participant nodes support MANET-specific routing protocols to cooperate in REDMAN identification and manager election, it is necessary that all nodes placed along the path from the current IN to farthest nodes remain within mutual communication range at least for the duration of a single protocol round. The number of nodes involved in that condition is very low with regards to the overall number of dense MANET participants. All these assumptions are largely acceptable because, even in very large dense MANET deployment scenarios with several hundreds of nodes, each round of the election protocol lasts for less than a few seconds.

After the end of the election process, REDMAN works to ensure that node mobility does not degrade too much the central position of the elected manager. With the goal of limiting the protocol overhead, REDMAN proposes a lazy strategy that allows short time intervals when the manager could have slightly moved from the dense MANET topologic center. In particular, REDMAN combines two different strategies against manager assignment degradation, one proactive and one reactive, which operate, respectively, over large

and medium time periods (Tp and Tr with $Tp \gg Tr$). The proactive maintenance strategy establishes that the current manager always triggers a new manager election after Tp seconds. In addition to probabilistically improving the centrality of the manager position, the periodical re-execution of the election process contributes to distribute the burden of the role among different nodes, thus avoiding to deplete the energy resources of a single participant. Moreover, let us rapidly observe that only the nodes located in the proximity of the dense MANET topology center have high probability to assume the manager role. Therefore, a target $INvalue$ can be easily determined equal to the $INvalue$ of the current manager, thus speeding up manager election and reducing the protocol overhead. In addition to the above proactive degradation counteraction, REDMAN exploits a reactive strategy that consists in repeating the farthest node determination at regular Tr periods, with the goal of understanding the current manager distance from the optimal placement. If the distance of manager farthest nodes, i.e., its $newINvalue$, has increased if compared with the distance estimated at the moment of its election, i.e., its $INvalue$, REDMAN realizes that the manager has moved from the topologic center in a significant way; in that case, the manager itself triggers a new election process. The following subsections detail the adopted heuristics to limit the number of explored INs and the exploited solution to determine, given a node, its farthest nodes in the dense MANET.

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-based approach that has experimentally demonstrated to reach high quality solutions, close to the optimal choice of the manager (see Section 4). 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 (see Figure 2.4). 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 $INvalue$. 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 2.4 shows the pseudo-code of the REDMAN election protocol.

Determination of Farthest Nodes

Let us formally define farthest nodes, with respect to a node d_k in the dense MANET, the set $FN(d_k) = d_{f0}, \ldots, d_{fF}$, where:

- $\forall i \in [0, F], d_{fi} \in DM(n)$, and
- $\forall d_j \in DM(n), sp(d_k, d_{fi}) \geq sp(d_k, d_j)$

where $sp(d_x, d_y)$ is the length of the shortest path that connects the nodes d_x and d_y. REDMAN proposes a simple broadcast-based strategy to detect the length of 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 the dense MANET identification protocol. Figure 2.5 shows the message propagation from node I (the current IN) to all other nodes in the

```
exploredList = 0; forwarderList = 0;
bestNode = 0; bestValue = MAX; worstValue = 0;
unexploredList = Initiator;
while (unexploredList != 0) {
    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) {
        conecutiveEqualSolutions++;
        if (consecutiveEqualSolutions == max_consecutive_
            equal_solutions) exit;
    unexploredList = unexploredList U forwarderList; }
    } Print(bestNode)
```

Figure 2.4: Pseudo-code of the REDMAN manager election protocol

dense MANET. 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 regards to I. Let us observe that the farthest node identification protocol works as if all dense MANET nodes were fixed during the determination of $FN(d_k)$ and, consequently, usually reaches an approximated non-optimal solution. However, given the limited time interval needed to complete the $FN(d_k)$ set determination, that stationary assumption has demonstrated to be

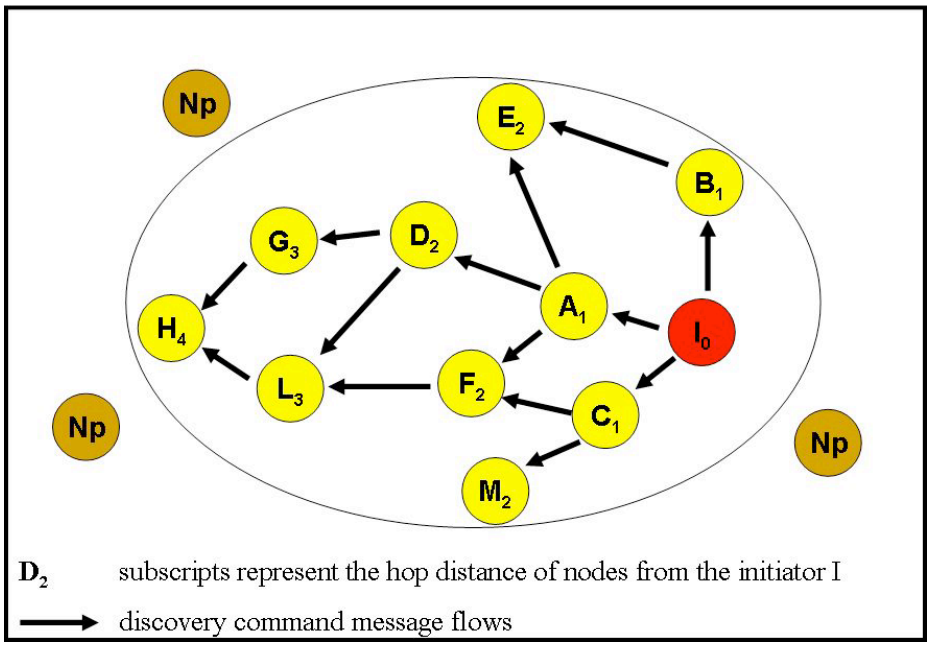


Figure 2.5: The current IN (I) broadcasts exploratory messages for manager election.

acceptable and to permit the set identification with adequate accuracy for the lightweight and lazily-consistent REDMAN form of replication.

Given that the protocol is flooding-based, it could incur in the broadcast storm issue, as explained in Section 2.3.1. However, state-of-the-art approaches to avoid broadcast storm, such as gossip-based strategies, are not applicable to the REDMAN farthest node identification protocol because in REDMAN there is the need to determine shortest paths of maximum length between investigated farthest nodes and the current IN [98]. Consequently, it is mandatory to adopt solutions that permit the identification of shortest paths: it is easy to demonstrate that traditional flooding complies with this property, while gossip-based flooding does not. At the moment, similarly to what described for dense MANET identification, REDMAN simply adopts a trivial technique based on delaying message broadcasts of a

random time interval. We are currently working to experimentally evaluate the effectiveness of storm-prevention flooding techniques similar to [63] when applied to REDMAN farthest node identification and to possibly integrate them in the next release of our middleware prototype.

By delving into finer technical details about the REDMAN protocol solution, 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 a message and does not depend on the number of participants and on the dense MANET diameter [97]. 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 2.6 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 do not reply because they are prevented by single-hop neighbors placed at greater distance, e.g., nodes E, F, and D in the case of farthest node determination for node A. 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 ones. The IN assumes the distance of the determined farthest node(s) as its INvalue.

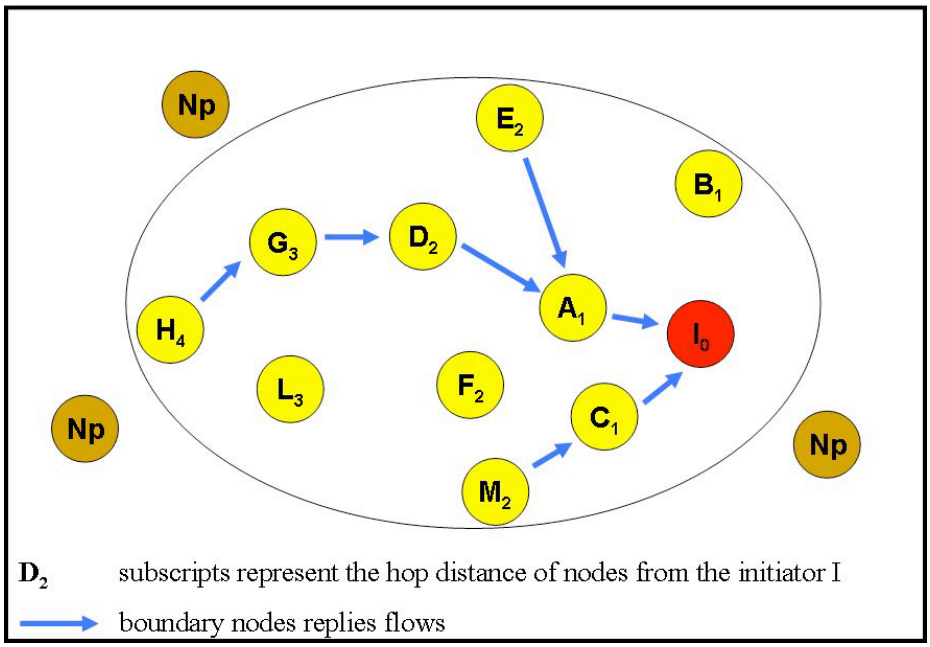


Figure 2.6: Only nodes at dense MANET boundaries (E, F, H) reply to the current IN.

2.4 Replica Distribution (RD)

RD [7] 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 2.5.3). 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. Then, it 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 a constant direction. When a delegate has 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 reiterates the process by decreasing the number of requested replicas.

2.5 Replica Retrieval (RR)

RR [7] aims at enabling clients to effectively find their requested resources at provision time on the basis of 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 [37]. 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.

2.5.1 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 ex-changed to retrieve the requested resources;
- the scalability when applied to large deployment environments;
- the accuracy, i.e., the found/searched resource ratio in the finite time interval spent for retrieval.

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 deployment scenarios, e.g., on the expected ratio between searches and replica instantiations, the frequency with which replicas leave/enter the dense MANET, the time requirements for resource discovery, 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 [128]. 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.

2.5.2 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

(this value is a rough estimate from simple geometric considerations). Let us observe that k-DID improves the solution in [124], where placement information similar to REDMAN IRPs is duplicated on all nodes located at fixed distance from resource-hosting nodes; in [124] 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; in that case, a backtracking step is done, returning control to the previous IRP distributor;

4. if a free potential IRP is found, then IRP goes on from step 1; otherwise, a further backtracking step (3) is done, 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 circle 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.

2.5.3 REDMAN Replica Retrieval

We have deeply investigated the performance of the k-DID RR strategy (see Section 4) 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 [15]. In Figure 2.7, 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 nodes 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 broadcast 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.

In many profitable scenarios, devices are carried by users almost fixed during most time of service delivery, e.g., spectators seated or slowly moving on terraces during a sport event. However, in general, the movements of REDMAN devices hosting replicas and/or IRPs could affect resource availability. 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 anytime 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.

2.6 Replica Degree Maintenance (RDM)

Proactive RDM solutions available in the literature, such as [30], do not fit the provisioning environments addressed by REDMAN. In fact, proactive RDM approaches usually require GPS-equipped wireless nodes that continuously monitor their mutual positions to foresee network exits, thus producing non-negligible network/computing overhead. REDMAN RDM [8], instead, implements a reactive solution with very low communication overhead by relaxing the constraint of anytime perfect consistency in the number of available replicas.

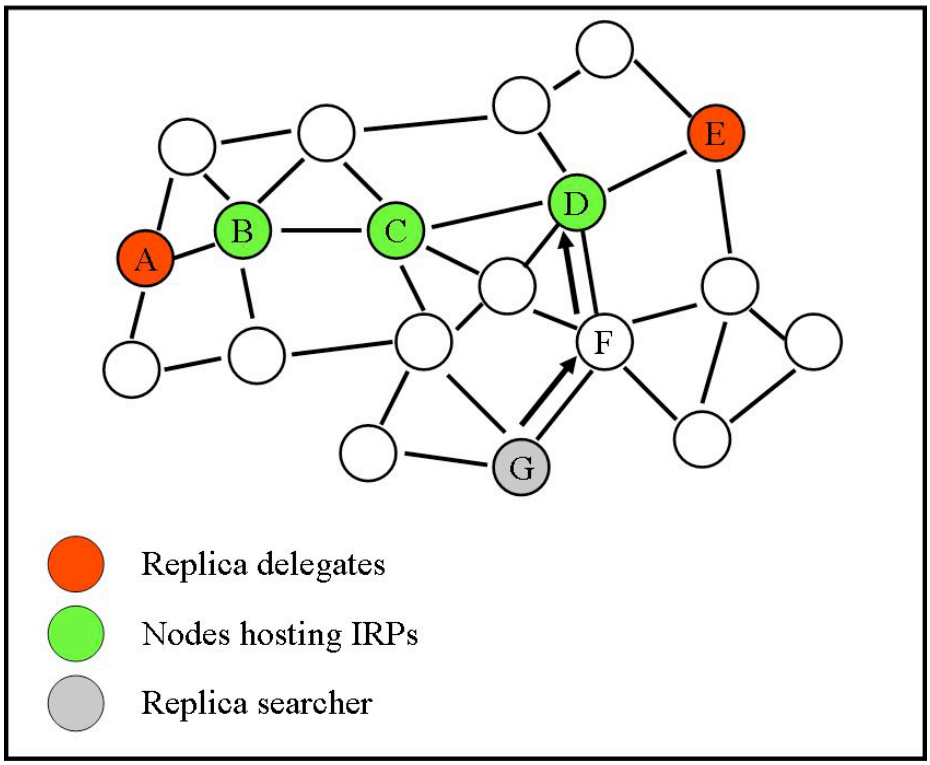


Figure 2.7: The distribution of replicas/IRPs and RR exploit approximately con-stant directions in REDMAN.

After the initial replica distribution phase, the lightweight RDM facility works to maintain unchanged the replication degree decided for each shared resource, without guaranteeing strict consistency, i.e., it is possible to have time intervals when the requested replication degree differs from the actual number of replicas in the dense MANET. RDM aims at maintaining unchanged the replication degree only by reacting to resource delegate movements/failures.

When a delegate realizes it is going to exit the dense MANET, it autonomously offloads its shared resources on neighbors that are still within the dense region; in their turn, these new delegates 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.

Chapter 3

The MobEyes middleware for opportunistic dissemination in Vehicular Sensor Networks

Recent advances in vehicular communications pave the way for novel urban monitoring applications where vehicles continuously sense events, process data and route messages toward harvesting agents. This challenging environment proposes specific issues: on-board sensors likely generate sheer amount of data, and vehicles are highly mobile. MobEyes is an effective middleware solution specifically designed for proactive data dissemination and harvesting in Vehicular Sensor Networks (VSN). MobEyes exploits mobility to opportunistically diffuse sensed data summaries among neighbor vehicles and to create a low-cost opportunistic index supporting queries on a distributed sensed data storage.

Vehicular sensor networks are emerging as a new network paradigm of primary relevance, especially for proactively gathering monitoring information in urban environments (proactive urban monitoring). Each vehicle, which typically has no strict constraints on processing power and storage capabilities, can sense one or more events (e.g., imaging from streets and detecting toxic chemicals), processing sensed data (e.g., recognizing license plates), and routing messages to other vehicles (e.g., diffusing relevant notification

to drivers or police agents). This is a novel and challenging mobile environment: the communication paradigm differentiates from one-to-one typical of MANET scenarios, and resembles sensor networks many-to-one (i.e., where many sensors convey gathered information toward one sink). Nonetheless, differently from traditional sensor networks, in Vehicular Sensor Networks nodes are mobile and generate a sheer amount of data. These characteristics invalidate known data reporting techniques [62].

MobEyes exploits wireless-enabled VSN vehicles equipped with video cameras and a variety of sensors to perform event sensing, processing/classification of sensed data, and inter-vehicle ad hoc message routing. Since it is impossible to directly report the sheer amount of sensed data to the authority, MobEyes keeps the sensed data in the mobile node storage; on board processing capabilities are used to extract features of interest, e.g., license plates; mobile nodes periodically generate data summaries with extracted features and context information such as timestamps and positioning coordinates; mobile agents, e.g., police patrolling cars, move and opportunistically harvest summaries as needed from neighbor vehicles. MobEyes adopts VSN custom designed protocols for summary diffusion/harvesting that exploit intrinsic vehicle mobility and single-hop inter-vehicle communications. In that way, MobEyes harvesting agents can create a low-cost opportunistic index to query the distributed sensed data storage, thus enabling to answer questions such as: which vehicles were in a given place at a given time?; which route did a certain vehicle take in a given time interval?, and which vehicle collected and stored the data of interest?

3.1 Practical case studies

An interesting application case for VSN is proactive urban monitoring, which could be usefully applied to post-facto crime scene investigation. Reflecting on tragedies such as

9/11 and London bombing, VSN could have actually helped loss recovery and a posteriori investigation. In London bombing police agents were able to track some of the suspects in the subway using closed-circuit TV cameras, but they had a hard time finding helpful evidence from the double-decker bus; this has motivated the installation of more cameras in fixed locations along London streets. VSN could be an excellent complement to the deployment of fixed cameras/sensors. The completely distributed and opportunistic cooperation among sensor-equipped vehicles has the additional positive side effect of making it harder for potential attackers to disable surveillance. Another less sensational but relevant example is the need to track the movements of a car, used for a bank robbery, in order to identify thieves, say. It is highly probable that some vehicles have spotted the thieves' car in the hours before the robbery, but is extremely difficult for the police to extract that information by identifying and collecting all the related multimedia streams recorded by fixed cameras.

As shown by the above examples, the reconstruction of a crime and, more generally, the forensic investigation of an event monitored by VSN require the collection, storage, and retrieval of massive amounts of sensed data. This is a major departure from conventional sensor network operations where data is dispatched to "sinks" under predefined conditions such as alarm thresholds. Obviously, it is impossible to deliver all the streaming data collected by video sensors to a police authority sink because of sheer volume. Moreover, input filtering is not possible because a priori nobody knows which data will be of any use for future investigations. The problem becomes one of searching for sensed data in a massive, mobile, opportunistically collected, and completely decentralized storage. The challenge is to find a completely decentralized VSN solution, with low interference to other services, good scalability (up to thousands of nodes), and tolerance of disruption caused by

mobility and attacks.

3.1.1 MobEyes at work: criminal tracking

For the sake of clarity, let us present the MobEyes solution using one of its possible practical application scenarios: collecting information from MobEyes-enabled vehicles about criminals that spread poisonous chemicals in a particular section of the city (say, subway station). We suspect the criminals used vehicles for the attack. Thus, MobEyes will help detect the vehicles and permit tracking and capture. Here, we assume that vehicles are equipped with cameras and chemical detection sensors. Vehicles continuously generate a huge amount of sensed data, store it locally, and periodically produce short *summary chunks* obtained by processing sensed data, e.g., license plate numbers or aggregated chemical readings. Summary chunks are aggregated in *summaries* that are opportunistically disseminated to neighbor vehicles, thus enabling metadata harvesting by the police in order to create a distributed metadata index, useful for forensic purposes such as crime scene reconstruction and criminal tracking.

3.2 MobEyes middleware

To support all the above tasks, we have developed MobEyes according to the component-based architecture depicted in Figure 3.1. The key MobEyes component is the MobEyes Diffusion/Harvesting Processor (MDHP) which will be discussed in detail in the next section. MDHP works by opportunistically disseminating/harvesting summaries produced by the MobEyes Data Processor (MDP), which accesses sensor data via the MobEyes Sensor Interface (MSI). Since vehicles are not strictly resource-constrained, our MobEyes prototype is built on top of the Java Standard Edition (J2SE) virtual machine. MDP is in charge

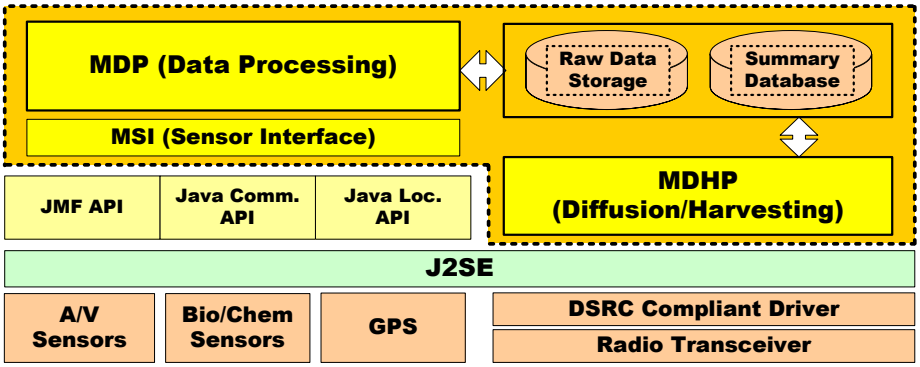


Figure 3.1: MobEyes sensor node architecture.

of reading raw sensed data (via MSI), processing it, and generating chunks. Chunks include metadata (vehicle position, timestamp, vehicle ID number and possible additional context such as simultaneous sensor alerts) and features of interest extracted by local filters (See Figure 3.2). For instance, in our scenario, MDP includes a filter determining license plate numbers from multimedia flows taken by cameras [40]. Finally, MDP commands the storage of both raw data and chunks in two local databases. MDHP disseminates/harvests summaries by packing a set of chunks into a single packet for the sake of effectiveness. Therefore, chunk/summary generation rate and chunk/summary size are relevant to MobEyes performance.

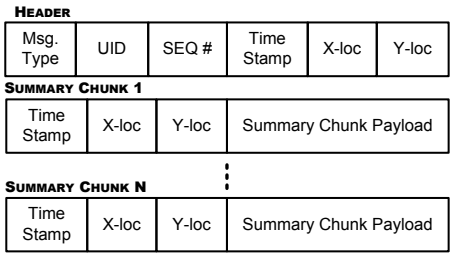


Figure 3.2: Packet format: a single summary packet contains multiple *summary chunks*.

Developers of MobEyes-based applications can specify the desired generation rate as a function of vehicle speed and expected vehicle density. The chunk size mainly depends

on application-specific requirements: in the considered scenario, each recognized license plate is represented with 6B, sensed data with 10B (e.g., concentrations of potential toxic agents), timestamp with 2B, and vehicle location with 5B. Then, in our scenario, MDP can pack 65 chunks in a single 1500B summary, even without exploiting any data aggregation or encoding technique. In usual deployment environments chunks are generated every [2-10] seconds and, thus, a single summary can include all the chunks about a [2-10] minute interval. MSI permits MDHP to access raw sensed data independently of actual sensor implementation, thus simplifying the integration with many different types of sensors. MSI currently implements methods to access camera streaming outputs, serial port I/O streams, and GPS information, by only specifying a high-level name for the target sensor. To interface with sensor implementations, MSI exploits well known standard specifications to achieve high portability and openness: the Java Media Framework (JMF) API, the Sun Communication API, and the JSR179 Location API.

3.3 MDHP Protocols

The section first details our original summary diffusion protocol where private vehicles (regular nodes) opportunistically and autonomously spread summaries of sensed data by exploiting their mobility. Then, it describes our novel summary harvesting protocol used by police agents (authority nodes) to proactively build a low-cost distributed index of the mobile storage of sensed data. The main goal of the MDHP process is the creation of a highly distributed and scalable index that allows police agents to place queries to the huge urban monitoring database without ever trying to combine this index in a centralized location.

3.3.1 Summary Diffusion

Any regular node periodically advertises a new packet with generated summaries to its current neighbors in order to increase the opportunities for agents to harvest summaries.Clearly, excessive advertising will introduce too much overhead, while no advertising at all will require considerable more time, as agents will need to contact each individual car to complete the harvesting process. Thus, MobEyes tries to identify the optimal tradeoff. As depicted in Figure 3.2, a packet header includes a packet type, generator ID, locally unique sequence number, packet generation timestamp, and generator's current position. Each packet is uniquely identified by the generator ID and its sequence number pair, and contains a set of summaries locally generated during a fixed time interval.[1]

Neighbor nodes receiving a packet store it in their local summary databases. Therefore, depending on the mobility and the encounters of regular nodes, packets are opportunistically diffused into the network (*passive* diffusion). MobEyes can be configured to perform either single-hop passive diffusion (only the source advertises its packet to current single-hop neighbors) or k-hop *passive* diffusion (the packet travels up to k-hop as it is forwarded by j-hop neighbors with $j < k$). Other diffusion strategies could be easily included in MobEyes, for instance single-hop active diffusion where any node periodically advertises all packets (generated and received) in its local summary databases, at the expense of a greater traffic overhead. As detailed in the experimental evaluation section, in a usual urban VANET (node mobility restricted by roads), it is sufficient for MobEyes to exploit the lightweight k-hop passive diffusion strategy with very small k values to achieve the desired diffusion levels.

[1]The optimal interval can be determined from the harvesting time distribution with average (μ) and standard deviation (ρ). Then, Chebyshev inequality, $P(|x - \mu| \geq k\rho) \leq \frac{1}{k^2}$ allows us to choose k such that we guarantee *harvesting latency* and thus we can determine the period as $\mu + k\rho$. Readers can find details in Section 3.4.

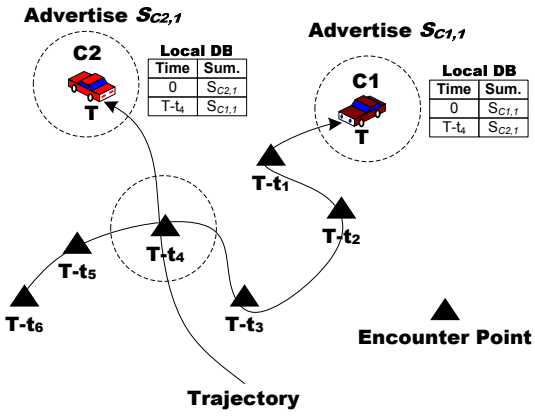
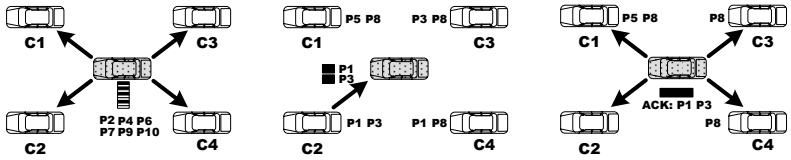


Figure 3.3: MobEyes single-hop passive diffusion

Figure 3.3 depicts the case of two sensor nodes, C1 and C2, that encounter with other sensor nodes while moving (the radio range is represented as a dotted circle). For ease of explanation, we assume that there is only a single encounter, but in reality any nodes within dotted circle are considered encounters. In the figure, a black triangle with timestamp represents an encounter. According to the MobEyes summary diffusion protocol, C1 and C2 periodically advertise a new summary packet $S_{C1,1}$ and $S_{C2,1}$ respectively where the subscript denotes $\langle ID,\ Seq.\# \rangle$. At time $T - t_4$, C2 encounters C1, and thus they exchange those packets. As a result, C1 carries $S_{C2,1}$ and C2 carries $S_{C1,1}$. Summary diffusion is time and location sensitive (spatial-temporal information diffusion). In fact, regular nodes keep track of freshness of summary packets by using a sliding window with the maximum window size of fixed expiration time. In addition, since a single summary packet contains multiple summary chunks, it is possible to define packet sensing location as the average position of all summaries in the packet. When a packet expires or the packet originator moves away more than a threshold distance from packet sensing location, the packet is automatically disposed. The expiration time and the maximum distance are system parameters that should be configured depending on urban monitoring application requirements. Let



(a) Broadcast a harvest request (b) C2 first returns missing packets (c) Broadcast acknowledgement

Figure 3.4: MobEyes proactive summary harvesting

us also briefly note that summaries always include, of course, the time and location where the sample was taken. Upon receiving an advertisement, neighbor nodes keeps the encounter information (the advertiser's current position and current timestamp). This allows MobEyes nodes to exploit spatial-temporal routing techniques [53] and a geo-reference service when accessing actual raw data. That is obtained as a simple byproduct of summary dissemination, without additional costs.

3.3.2 Summary Harvesting

In parallel with diffusion, MobEyes summary harvesting takes place. There are two possible modes of harvesting the "diffused" information, namely *on demand* and *proactive*. In the on demand mode, the police agents react to an emergency call, for example, the earlier mentioned poisonous gas incident. Police agents will converge to the outskirts of the area (keeping a safe distance of course) and will query vehicles for summaries that correspond to a given time interval and area (i.e., time-space window). Suppose 1000 such summaries exist. The police agents as a team will collect as many summaries as they can, up to 1000. They will collectively examine the summaries and decide to inspect in more detail the video files collected by 100 vehicles, say. The vehicles can be contacted based on the vehicle ID number stored in each summary. A message is sent to each vehicle requesting it to upload the file at the nearest police access point. The request message is generally routed using

georouting, either exploiting the Geo Location Service that maps vehicle ID to the current vehicle location, or using the "Last Encounter Routing" technique [80, 53]. The latter technique is particularly convenient here because at the time the summary exchange takes place, nodes memorize the time and place of the encounter.

Naturally, the on demand harvesting incurs the problem of latency in dispatching the vehicles to the location and in collecting the summaries. To overcome this latency, we are proposing also a "proactive" version of the index construction. Namely, in each area there are agent vehicles that collect all the summaries as a background process and create a distributed index. In this case, there is no time-space window concern during collection. The only requirement is to collect all the summaries in a particular area. Now, if the poisonous gas emergency occurs, the query is directed to the proactively created distributed index. The time-space window concept is applied to the index to find the vehicles in a particular place and time and then pursue the hot leads.

It is apparent that the two processes become identical after the desired summaries have been identified on board of the agent vehicles. The proactive approach is much more powerful in that it can speed up the search considerably. For instance, if the inspection of the information collected in the crime area indicates a possible escape direction of the terrorists, one can immediately search again the proactively created index for a new time-space window, without having to do another time consuming collection of summaries from vehicles. On the negative side, maintaining that index is costly, as agent resources must be dedicated to the task.

In the sequel we will assume proactive index construction. Thus, the agents collect all summaries indiscriminately. There is no loss of generality, however, since the procedure will also allow on demand index construction for a specific time-space request. In fact, the

only difference between the two harvesting schemes is the size of the set being harvested. In the on demand scheme, the target set is a specific space-time window set. In the proactive scheme, the target set is the entire geographic area within agent responsibility; there is no limit on harvesting time, though old records are timed out.

By considering the proactive harvesting model, the MobEyes police agent collects summaries from regular nodes by periodically querying its neighbors. The goal is to collect all the summaries generated in the specified area. Obviously, a police node is interested in harvesting only summary packets it has not collected so far: to focus only on missing packets, a MobEyes authority node compares its list of summary packets with that of each neighbor (set difference problem), by exploiting a space-efficient data structure for membership checking, i.e., a Bloom filter. A Bloom filter for representing a set of ω elements, $S = \{s_1, s_2, \cdots, s_\omega\}$, consists of m bits, which are initially set to 0. The filter uses ℓ independent random hash functions $h_1, \cdots, h_\ell$ within m bits. By applying these hash functions, the filter records the presence of each element into the m bits by setting ℓ corresponding bits; to check the membership of the element x, it is sufficient to verify whether all $h_i(x)$ are set to 1.

A MobEyes police agent uses a Bloom filter to represent its set of already harvested and still valid summary packets. Since each summary has a unique node ID and sequence number pair, we use this as input for the hash functions. The MobEyes harvesting procedure consists of the following steps:

1. The agent broadcasts a “harvest” request message with its Bloom filter.

2. Each neighbor prepares a list of “missing” packets from the received Bloom filter.

3. One of the neighbors returns missing packets to the agent.

4. The agent sends back an acknowledgment with a piggybacked list of returned packets and, upon listening to or overhearing this, neighbors update their lists of missing packets.

5. Steps 3 and 4 are repeated until there is no remaining packet.

An example of summary harvesting is shown in Figure 3.4. The agent first broadcasts its Bloom filter related to collected packets so far (P2, P4, P6, P7, P9 and P10) as in Figure 3.4(a). Each neighbor receives the filter and creates a list of missing packets. For example, C3 has P3 and P8 to return, while C4 has P1 and P8. In Figure 3.4(b), C2 is the first node to return missing packets (P1, P3) and the agent sends back an acknowledgement piggybacked with the list of received packets. Neighbor nodes overhear the message and update their lists: C3 and C4 both remove P1 from their lists, as depicted in Figure 3.4(c). Note that membership checking in a Bloom filter is probabilistic and false positives are possible. In Figure 3.4(b), for example, a false positive on P1 makes C2 return only P3. Then, the other neighbors will attempt to deliver P1. Harvesting fails only if none of the neighbors can send P1. Even with failure, by repeating the harvesting procedure over a period of time, the agent can gather missing packets. In the following section, we prove that harvesting is guaranteed with high probability notwithstanding the possibility of Bloom filter false positives.

For the sake of simplicity, thus far we assumed that there is a single agent working to harvest summaries. Actually, MobEyes can handle concurrent harvesting by multiple agents that can cooperate by exchanging their Bloom filters among multi-hop routing paths; thus, this creates a distributed and partially replicated index of the sensed data storage. In particular, whenever an agent harvests a set of j new summary packets, it broadcasts its Bloom filter to other agents, with the benefits in terms of latency and accuracy shown in

the following section. Note that strategically controlling the trajectory of police agents, properly scheduling Bloom filter updates, and efficiently accessing the partitioned and partially replicated index are part of our future work. In the following section, instead, we focus on the primary goal of identifying the tradeoffs between dissemination and harvesting in a single geographic area, and the dependence of MobEyes performance on various parameters. We also analyze the traffic overhead created by diffusion/harvesting and show that it can scale well to very large node numbers.

3.4 Analysis of MDHP Protocols

To evaluate and validate the effectiveness of the original MobEyes protocols, here we present an analytic investigation about summary harvesting, efficiency of Bloom filter adoption, and scalability.

3.4.1 Summary Harvesting Delay

In MobEyes regular nodes receive summaries from their neighbors (passive harvesting) and these summaries will be harvested by the police agents (active harvesting). Obviously, the effectiveness of active harvesting depends also on passive harvesting. Therefore, we model the progress of passive harvesting, from which we formulate the progress of active harvesting. Finally, we extend the model to analyze k-hop relay scope.

We assume that there are N nodes in the network and each node advertises a single summary packet (total N summary packets). We basically assume that nodes are uniformly distributed within a square area with length L. The node density is given as $\rho = N/L^2$. When we consider non-uniform node distribution under different mobility models, the

node density is simply given as $\rho = \delta N/L^2$ where δ is the constant compensation factor for a given mobility model. For ease of analysis, we assume that nodes move towards random directions (chosen out of [0,2π]) at a speed of v on average (random direction mobility model). Let v^* denote the average relative speed of nodes. As shown in [122], $v^* = \frac{v}{2\pi}\int_0^{2\pi}\sqrt{(1+\cos\theta)^2+\sin^2\theta}d\theta = 1.27v$. For non-uniform mobility models, we simply assume that the average relative speed can be represented with constant multiplication of the average speed: $v^* = cv$ where c is a constant. Let R denote the communication range of a node.

By extending [6], we now decided to develop a deterministic, discrete time model. Let us first reason on how many summaries a node can receive for a given time slot. For ease of exposition, we assume that all nodes are static except one regular node. This node randomly moves and collects summaries by passively listening to advertisements from encountered nodes. In this case, the node (or the passive harvester) behaves just as a data mule in traditional sensor networks [115]. During the time slot Δt, a regular node travels a distance $r = v\Delta t$ and covers an area of $v\Delta t2R$. The expected number of encountered nodes in this area is simply $\alpha = \rho v\Delta t2R$. Since each of these nodes will advertise its summaries, the regular node will receive α summaries. The dual scenario is when all nodes are mobile but the passive harvesting node is static. Without loss of generality, if all nodes are mobile, we can simply replace the average speed with the average relative speed: thus, $\alpha = \rho v^*\Delta t2R$ where v^* is the average relative speed.[2]

Given α, we can estimate the progress of passive harvesting as follows. Let E_t denote

[2]We can think of this as follows. Let us say that in front of a freeway (where everybody is driving in one direction at a constant speed v), we count the number of vehicles passing by. During Δt, it will be $\rho v\Delta t$. Now, let us assume that an observer is moving also. If it moves on the same direction, i.e., the relative speed is 0, it always observes the same vehicles. On the contrary, if it moves on the opposite direction, the relative speed is $2v$ and it will see $\rho 2v\Delta t$ vehicles.

the number of distinct summaries collected by a regular node by time slot t. As described above, at time slot t a regular node will receive α summaries. Since the node has E_t summaries, the probability of acquiring a new summary is simply $1 - E_t/N$. Thus, the expected number of new summaries out of α is given as $\alpha(1 - E_t/N)$. It is obvious that non-uniform movement patterns (e.g., two nodes moving together along the same path) will affect the *effective* number of neighbors. Since we are interested in the average behavior, we can model this by simply multiplying α with a constant compensation factor η. Therefore, we have the following relationship:

$$E_t - E_{t-1} = \alpha\eta\left(1 - \frac{E_{t-1}}{N}\right) \tag{3.1}$$

Equation 3.1 is a standard difference equation with solution:

$$E_t = N - (N - \alpha\eta)\left(1 - \frac{\alpha\eta}{N}\right)^t \tag{3.2}$$

Equation 3.2 tells us that the distinct number of collected summaries is exponentially increasing and thus, as time tends to infinity, $E_t = N$. Let us define a random variable T to denote the time for a regular node to encounter any random node, thus receiving a summary from it. The cumulative distribution of random variable T can be derived by dividing Equation 3.2 by N.

$$F_T(t) = 1 - \left(1 - \frac{\alpha\eta}{N}\right)^{t+1} \tag{3.3}$$

From this, we can derive the Probability Mass Function $f_T(t)$ as follows

$$f_T(t) = \frac{\alpha\eta}{N}\left(1 - \frac{\alpha\eta}{N}\right)^t \tag{3.4}$$

Equation 3.4 is a *modified* geometric distribution with success probability $p = \frac{\alpha\eta}{N}$. The average is given as $\mathbb{E}[T] = \frac{1}{p} - 1 = \frac{N}{\alpha\eta} - 1$. Since $\alpha = \rho v^* \Delta t 2R$, by replacing $\rho = \delta N/L^2$ we have $\alpha = \delta N/L^2 v^* \Delta t 2R$. Thus, we have:

$$\mathbb{E}[T] = \frac{N}{\alpha\eta} - 1 = \frac{L^2}{\delta v^* \Delta t 2R\eta} - 1 \tag{3.5}$$

As shown in Equation 3.5, given a square area of L^2, the average time for a regular node to collect a summary is independent of node density. In fact, it is a function of average relative speed and communication range. Intuitively, as node density increases (N increases), a node can collect more summaries during a given time slot, but this also means that it has to collect a higher number of summaries.

Unlike regular nodes, the agent actively harvests summaries from its neighbors. Since every node moves randomly and it starts passive harvesting at time 0, it is expected that every node has the same number of summaries collected by time t (E_t). Therefore, the probability that a neighbor node does not have a random summary is given as $1 - \frac{E_t}{N}$.

The probability that none of $\alpha\eta$ neighbors has a summary is simply $(1 - \frac{E_t}{N})^{\alpha\eta}$. The probability that at least one of neighbors has a random summary is $1 - (1 - \frac{E_t}{N})^{\alpha\eta}$. The expected number of distinct summaries the agent receives from its neighbors at time step t can be expressed by simply multiplying that probability by N:

$$N\left(1 - \left(1 - \frac{E_{t-1}}{N}\right)^{\alpha\eta}\right) \tag{3.6}$$

Let E_t^* denote the expected number of distinct summaries harvested by the agent till time step t. Since the agent has E_t^* summaries, the probability of acquiring a new summary is $1 - E_t^*/N$. Hence, multiplying this probability by the expected number of summaries harvested from neighbors (Equation 3.6) gives us the number of new summaries harvested during time step t as follows:

$$E_t^* - E_{t-1}^* = \gamma N\left(1 - \left(1 - \frac{E_{t-1}}{N}\right)^{\alpha\eta}\right)\left(1 - \frac{E_{t-1}^*}{N}\right) \tag{3.7}$$

where the constant compensation factor γ adjusts the expected number of summaries received from neighbors to consider non-uniform mobility. Note that as described before non-uniform mobility reduces the rate of new encounters (adjusted by η). Such mobility also exerts baleful influence on the rate of active harvesting since neighbors tend to carry overlapping summaries (adjusted by γ).

From Equation 3.7, we see that E_t^* grows much faster than E_t. During a time slot, the number of collected summaries in Equation 3.2 is constant (α), whereas in Equation 3.7 it is a function of time. Moreover, Equation 3.6 is a function of the number of neighbors, i.e., related to node density. As N (or node density) increases, we can see that the harvesting delay also decreases.

The growing rates of E_t and E_t^* depend on mobility models. The above equations are based on the random motion model, but for restricted mobility models such as the Manhattan model, the rate will be smaller than for the others (as shown in Section 5). In this case, we use k-hop relay scope where a summary is flooded up to k hops (as long as connectivity is available). As stated before, we are assuming a rectangular area, $\Delta t 2R$. Increasing k-hop relay scope is the same as multiplying the area by k times. Let E^k denote the number of summaries collected by time slot t with k-hop relay scope. Thus we have:

$$E_t^k - E_{t-1}^k = k\alpha\eta\left(1 - \frac{E_{t-1}^k}{N}\right) \tag{3.8}$$

This tells us that even though E_t grows rather slowly due to the mobility model, by increasing the hop count we can increase the E_t rate (from α to $k * \alpha$). Let E^{k*} denote the number of summaries harvested by the agent by time step t with k-hop relay scope. Then, we have:

$$E_t^{k*} - E_{t-1}^{k*} = \gamma N\left(1 - \left(1 - \frac{E_{t-1}^k}{N}\right)^{k\alpha\eta}\right)\left(1 - \frac{E_{t-1}^{k*}}{N}\right) \tag{3.9}$$

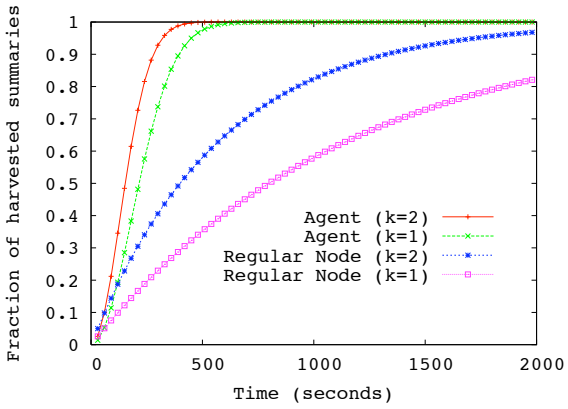


Figure 3.5: Fraction of harvested summaries with $k = 1, 2$

For illustration, let us assume that we have total $N = 200$ nodes within an area of $2400m \times 2400m$. The transmission range is $R = 250m$, and node relative speed is $10m/s$ on average. For system parameters, we used $\eta = 1$, $\gamma = 0.2$ and $\Delta t = 1s$. The iterative solutions of both E_t and E_t^* are presented in Figure 3.5. The figure shows that the agent can harvest summaries much faster than a regular node. The figure also shows that k-hop relay relevantly decreases the overall delay.

3.4.2 Efficiency of Bloom Filters in Summary Harvesting

Let us assume that a Bloom filter for representing a set of ω elements, i.e., $S = \{s_1, s_2, \cdots, s_\omega\}$, uses m bit table and ℓ hash functions (i.e., $h_1, \cdots, h_\ell$), each of which is uniformly random over the m bits and sets one bit to 1. As stated before, membership checking in a Bloom filter may result in false positives. But in this section, we analytically show that the false positives have negligible impacts on the overall harvesting process.

A false positive happens when an element s is not in S, but its bits $h_i(s)$ are collectively marked to 1 by other elements in S. After all ω elements are inserted (by marking the hash

table), the probability that a bit is 0 is $(1 - \frac{1}{m})^{\ell\omega} \simeq e^{-\frac{\ell\omega}{m}}$. To have a false positive, for all i $h_i(s)$ has to be set as 1. There are ℓ hash functions, and thus, the probability of a false positive is $p_f = (1 - (1 - \frac{1}{m})^{\ell\omega})^{\ell} \simeq (1 - e^{-\frac{\ell\omega}{m}})^{\ell}$

As previously described, at time step t an agent will meet $\eta\alpha$ neighbors on average. Let us calculate the probability that all $\eta\alpha$ neighbors fail to deliver a specific summary. This happens if all the neighbor nodes with that summary experience false positives. The probability that a node has a random summary packet at time t is $\frac{E_t}{N}$ where E_t is the number of packets that a node has collected till time t. The probability for a node to successfully send the packet to the agent is $(1-p_f)\frac{E_t}{N}$. Then the probability that the agent fails to receive the packet from any of $\eta\alpha$ neighbors is given as $(1-(1-p_f)\frac{E_t}{N})^{\eta\alpha}$. Assuming that each time step is independent of each other, after repeating β time steps, the agent fails to receive the packet with probability:

$$\prod_{\tau=t}^{t+\beta} \left(1 - (1 - p_f)\frac{E_\tau}{N}\right)^{\eta\alpha} \tag{3.10}$$

Since the failure probability is monotonically decreasing with exponential rate (Equation 3.10), it converges almost surely to zero. For example, given that we use m/ω bits to represent ω summaries, the probability of a false positive is roughly 0.15. In fact, given the ratio m/ω, the optimal number of hash functions ℓ that minimizes the probability of a false positive is given as $\lceil(\ln 2)m/\omega\rceil$ (see [43] for details). If m/ω=4, the optimal value is $\ell = \lceil(\ln 2)4\rceil = 3$, and the probability of a false positive is roughly $p_f = 0.15$. Let us assume that the average number of neighbors is given as $\alpha = 10$ and a node has collected half of the summaries. Then, the probability that the agent fails to receive a specific packet from neighbors is $(1 - 0.85 \times 0.5)^{10} = 0.004$. If we repeat this process for 5 time steps, the probability that the agent can collect the packet is roughly given as $1 - 0.004^5 \simeq 1$.

From this we conclude that since it is highly probable that other nodes have the packets as time passes, and the harvesting procedure is repeated as the agent moves, the agent always obtains missing packets with high probability.

3.4.3 Scalability

The feasibility of MobEyes strictly depends on its scalability over wide VSN, in terms of both the network traffic due to passive diffusion when the number of regular nodes grows and to the number of regular nodes that a single harvesting agent can handle with a reasonable latency.

About passive diffusion network traffic, it is possible to analytically estimate the MobEyes radio channel utilization. In the diffusion process, nodes periodically advertise their packets, without any synchronization among them. Therefore, we can model the process by considering a packet randomly sent within $[iT_a, (i+1)T_a)$ time slot for all i, where T_a is the advertisement period. So, the number of packets received by a node is bounded by the number of its neighbors while it is traveling for T_a, thus depending on node density but not on the overall number of nodes. In contrast, any "flooding"-based diffusion protocol is not scalable because a node could potentially receive a number of packets proportional to the network size.

To give a rough idea of the traffic generated by MobEyes diffusion, let us simply use $T_a = 2R/v^*$ (the time for a mobile node to traverse the diameter of its coverage area) where R is the transmission range and v^* denotes the relative speed of two nodes. In fact, for a given speed, the T_a interval should be neither too short nor too long compared to the average connection duration among nodes: if it is too short, then we are unnecessary sending out more packets to the same set of nodes, thus increasing link bandwidth utilization; on the contrary, if it is too long, a node misses chances to send packets to encountered nodes,

thus slowing down dissemination. In our target deployment environment, $v^* = 20m/s$, $R = 250m$, the advertisement period $T_a = 12.5s$, and the fixed packet size $S = 1500B$. Consequently, the transmission time for one packet is about $T_x = 1ms$. While traveling for T_a, a regular node will be exposed to advertisements from an area of $\mathcal{A} = \pi R^2 + 4R^2$. In the worst case, all nodes within this area are distinct and potentially send their generated packets to the considered node (potential senders $n = \mathcal{A}\rho$). Therefore, the worst case link utilization could be estimated as nT_x/T_a where T_x is the transmission time of a packet: for instance, given a relatively high populated area with $N = 2,000$, the number of potential senders is $n \simeq 179$, and the MobEyes protocol has a worst case link utilization of 0.014.

Similarly, we can give an approximated idea of the scalability of the harvesting process via a simple queuing model. Consider the usual situation of a police agent that harvests only fresh summaries, i.e., generated in the last T_{exp} seconds. Let us assume that the summary arrival rate is Poisson with rate $\lambda = N\lambda'$ and the harvesting rate is deterministic with rate μ. Given that the harvesting rate is limited by the channel utilization ϑ, the maximum μ is simply $\frac{\vartheta T_{exp}}{T_x}$. As a result, the system can be modeled using an M/D/1 queue. The stability condition, $N\lambda' < \frac{\vartheta T_{exp}}{T_x}$, gives us the upper bound $N < \frac{\vartheta T_{exp}}{\lambda' T_x}$. Therefore, it is possible to conclude that for a given T_{exp} and arrival rate, there exists a limit in the number of regular nodes that a single harvesting agent can handle. For instance, in the considered scenario ($\vartheta = 0.01$, $\lambda' = 2$, and $T_{exp} = 250s$), that number is $N < \frac{0.01 \times 250}{2 \times 0.001} = 1,250$. As a consequence, in the case of node numbers equal or not far from 1,250, there is the need to deploy more than one harvesting agent to maintain the system stable, i.e., to be able to harvest summaries more rapidly than regular nodes generate them.

Chapter 4

REDMAN Validation

To evaluate the effectiveness of our approach, we have extensively simulated the behavior of REDMAN solutions, in the NS2 simulator [99]. Large-scale protocol testing results infeasible to carry on real prototype networks, given the high number of participants required. Thus, simulations provide a means to verify the effectiveness of our solutions, at the same time reliably measuring performance trends. In addition, these artificial environments allow us to finely control and change the scenario conditions, e.g., node positions, much more precisely than real-world implementations. Obviously, simulations suffer from the impossibility to test solutions in a real wireless medium, but model it according to statistical formulas [107]. As simulative tool, we chose NS2 [99], a well-known network simulator, particularly used in the MANET community. Differently from commercial alternatives, NS2 development involves a user community, which continuously improves the product and integrates novel supports.

We simulated most interesting protocols presented in chapter 2, to verify their suitability to be employed in dense MANET scenarios. Generally speaking, these simulations have three main goals:

- to evaluate the accuracy of REDMAN protocols notwithstanding the heuristic-based

overhead reduction;

- to determine the network overhead of the proposed protocols in terms of the number of messages exchanged among MANET nodes;

- to evaluate the robustness of replica retrieval and degree maintenance while increasing node mobility.

In particular, we validated DMC, by verifying the dense MANET identification, and manager election protocols. Then, we evaluated the effectiveness of the SID replica and IRP dissemination and retrieval solutions, and compared it with the discussed alternatives (see Section 2.5), namely IRP-flooding, Query-flooding, and k-hop Distance IRP Dissemination. In both cases, i.e., DMC and RD/RR, simulations were performed in stationary as well as mobile scenarios (with users abandoning the dense area). Finally, we investigated RDM accuracy in maintaining the desired replication degree, in different size scenarios.

4.1 Simulation Setup

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). The sides of IS and ES are in the range respectively [$500m$, $1700m$], [$1000m$, $3400m$]. Nodes are distributed so that the dense MANET almost coincides with the IS area. In particular, the IS node density is relevantly higher than the ES one (generally about 8 times). The number of nodes spans from 50 to 700, and nodes move according to a random direction mobility model. Node speed is randomly chosen in the range $[1, 5]m/s$ to represent the typical velocity of human beings. Furhter details about the mobility models will be included in the respective

subsections.

Finally, if not differently specified, we have used default NS2 values for communications parameters, e.g., bi-directional connectivity, IEEE 802.11 link layer protocol, transmission band 2.4GHz, bandwidth 11Mbps, constant 250m circular transmission ranges, and Two-ray Ground propagation model [107]. For the sake of simplicity, the reported results refer to deployment scenarios where all nodes are equal from the point of view of locally available resources, such as battery power and storage space.

The code of the REDMAN prototype, the full algorithmic description of REDMAN protocols, the exhaustive list of all NS2 simulation parameters used, and a wide set of additional performance figures are available at the REDMAN Web site http://www.lia.deis.unibo.it/Research/REDMAN/

4.2 Dense MANET Configuration

First, we have carefully evaluated i) the accuracy of DMC manager election protocol, i.e., its effectiveness in choosing a manager located close to the center of the topology, ii) the network overhead, in terms of number of sent messages, of dense MANET identification and manager election, and iii) the impact of node mobility, to verify that DMC provides accurate solutions despite limiting the netork overhead. We have tested DMC 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. Given that IS node density is 8 times higher than ES density, 72% of the nodes are located within IS, and the average density is respectively about 30 in IS, and about 4 in ES.

REDMAN parameters were set as following: 11 Neighbor Limit (i.e., number of neighbors required to beloging to the dense MANET): this value experimentally proved to make the dense MANET bounds almost coincide with IS bounds; and $5s$ Rebroadcast Delay (to avoid broadcast storm issues [98]).

4.2.1 Manager Election Inaccuracy

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 election inaccuracy defined as the hop distance between the manager chosen by the REDMAN protocol and the actual optimal solution.

The results in Figure 4.1 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.

4.2.2 Impact of REDMAN Heuristics on Manager Election Accuracy

A decisive parameter affecting the results achieved by the REDMAN manager election protocol has demonstrated to be $MaxConsecutiveEqualSolutions$. Let us recall that this parameter influences the protocol termination, by permitting to stop the iterations when the specified number of best solutions has already been explored. However, the $INvalue$ of the elected IN ($electedINvalue$) could not be still the optimum achievable in that network ($optimalINvalue$).

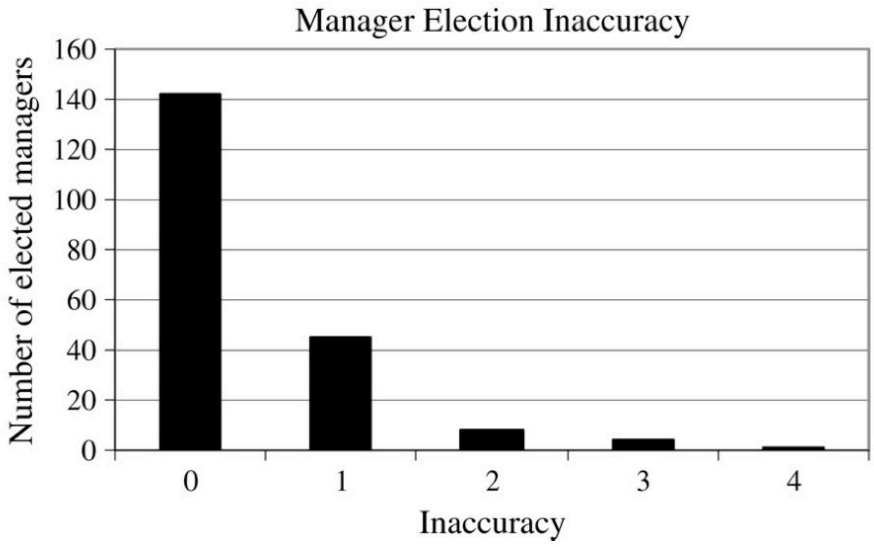


Figure 4.1: Inaccuracy of the REDMAN manager election protocol.

To determine whether the $electedINvalue$ obtained with low values of $MaxConsecutiveEqualSolutions$ is acceptable, we have run a large set of simulations in the deployment scenarios described above. Figure 4.2 plots the average $electedINvalues$ obtained over 30 different runs for each scenario, for values of $MaxConsecutiveEqualSolutions$ ranging from 2 to 4. The results are compared with two reference traces: the lower represents the $optimalINvalue$ in each scenario; the higher the $optimalINvalue$ multiplied by 1.25, thus depicting a tolerance strip of 25% from $optimalINvalue$.

As expected, the figure shows that the difference between $electedINvalues$ and $optimalINvalues$ grows as $MaxConsecutiveEqualSolutions$ decreases. However, even for the lowest tested value ($MaxConsecutiveEqualSolutions = 2$), most points lie below the higher reference trace. This means that REDMAN achieves solutions not too far from the $optimalINvalue$ also for very low values of $MaxConsecutiveEqualSolutions$. For that reason, all the other experimental results in the section refer to simulation runs where

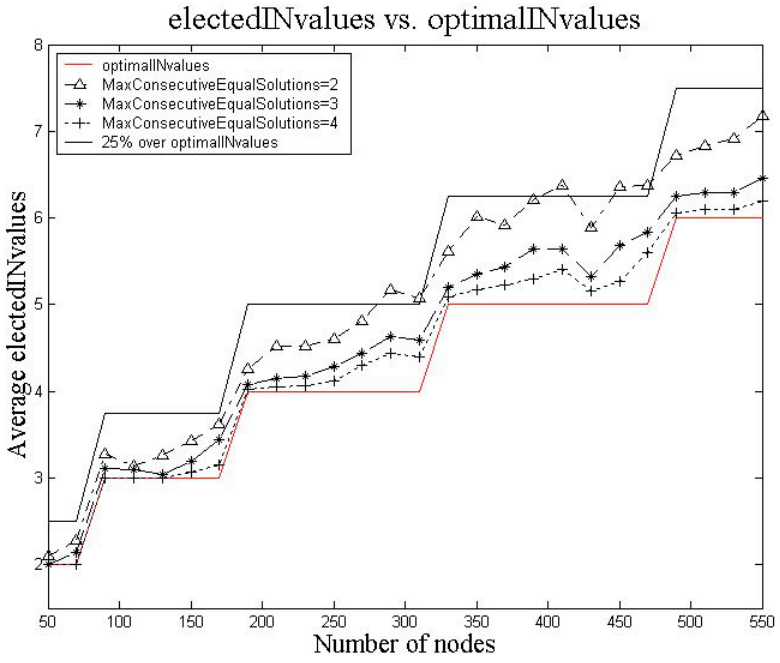


Figure 4.2: Accuracy of the manager election protocol as a function of MaxConsecutiveEqualSolutions.

$MaxConsecutiveEqualSolutions$ has been set to 3.

Let us briefly observe that manager election accuracy is also influenced by $DesiredAccuracy$, the other configurable parameter in the REDMAN election heuristic. For the sake of briefness, we have decided not to report also experimental results interest about $DesiredAccuracy$, which are of minor interest: simulations have confirmed that election accuracy linearly depends on that parameter, as intuitively expected. In all experimental results in the section, we have used $DesiredAccuracy = 100\%$.

4.2.3 DMC Network Overhead

We have carefully evaluated the network overhead of the REDMAN protocols for dense MANET identification and manager election, to verify that the DMC proposals are

lightweight enough for the addressed deployment scenario. For both protocols we have measured the average number of messages sent by each participant, over a set of more than 1,000 simulations.

The results reported in Figure 4.3 are normalized to the number of nodes actively participating in the 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 REDMAN solution for manager election, also in this case the number of sent messages is very limited and grows very slightly with the number of participants. In fact, sent messages 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.

Given the dependence of manager election network overhead on the number of iterations, we have carefully investigated the convergence of the REDMAN protocol while varying the number of dense MANET nodes (see Figure 4.4). 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 experimental results about the number of iterations have demonstrated the almost linear dependence on dense MANET diameter and have shown to be negligibly affected by the choice of the initiator node.

In summary, the experimental results about REDMAN DMC overhead demonstrate the feasibility and the good scalability of the proposed solutions: the network overhead slowly increases when the number of participants grows. Let us briefly observe that the non-monotonic growth of the overhead trace in Figure 4.3 is due to different factors. First,

the dense MANET diameter only increases in correspondence with some threshold values of the number of participants, as discussed and experimentally shown in Section 4.2.2, thus affecting the number of optimal solutions in the network. Moreover, the adopted heuristic parameters, such as $MaxConsecutiveEqualSolutions$, influence the number of iterations of the manager election protocol (see additional performance results and comments at the REDMAN Web site).

About the latency imposed by the REDMAN manager election protocol, let us briefly observe that it mainly depends on three factors: obviously, one is the number of dense MANET participants; the others are two REDMAN configurable timers that establish, respectively, the message delay for preventing broadcast storm in farthest node determination and the time interval waited by the current IN before delegating its role. All the presented results have been obtained by setting the former to $2.5s$ and the latter to $5s$ per hop of the diameter. These values guarantee that the current IN passes node exploration responsibility only after having received most replies from farthest nodes, thus achieving its correct INvalue.

4.2.4 Impact of Node Mobility on the Accuracy of the REDMAN Dense MANET Identification Protocol

To test the robustness of REDMAN solutions, we have evaluated the accuracy of 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

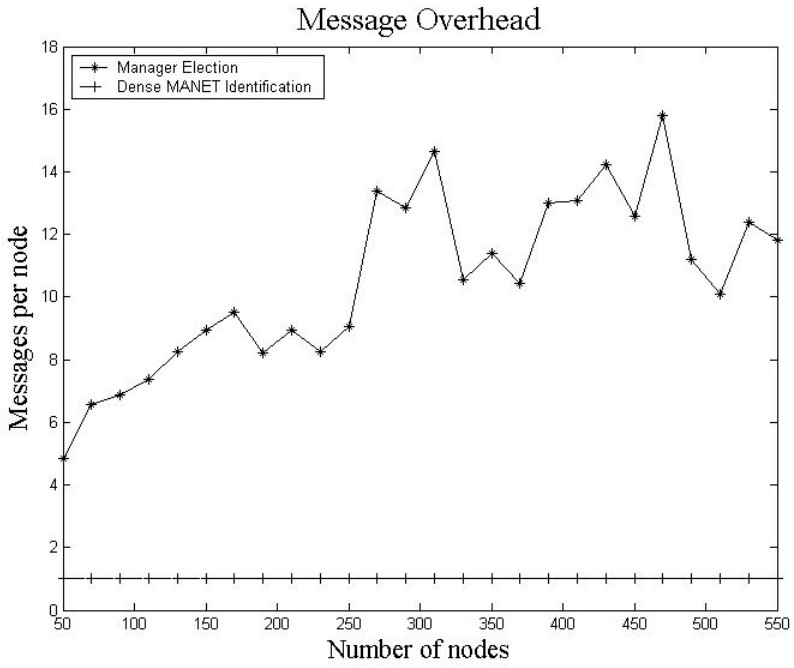


Figure 4.3: Messages sent per node in dense MANET identification and manager election.

solutions. For this reason, the sub-section focuses on the behavior of our dense MANET identification protocol as a function of node mobility.

We have considered the 110-node scenario described above. 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 arrival/departure in/out the dense MANET does not affect at all the behavior of the REDMAN identification solution.

Figure 4.5 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

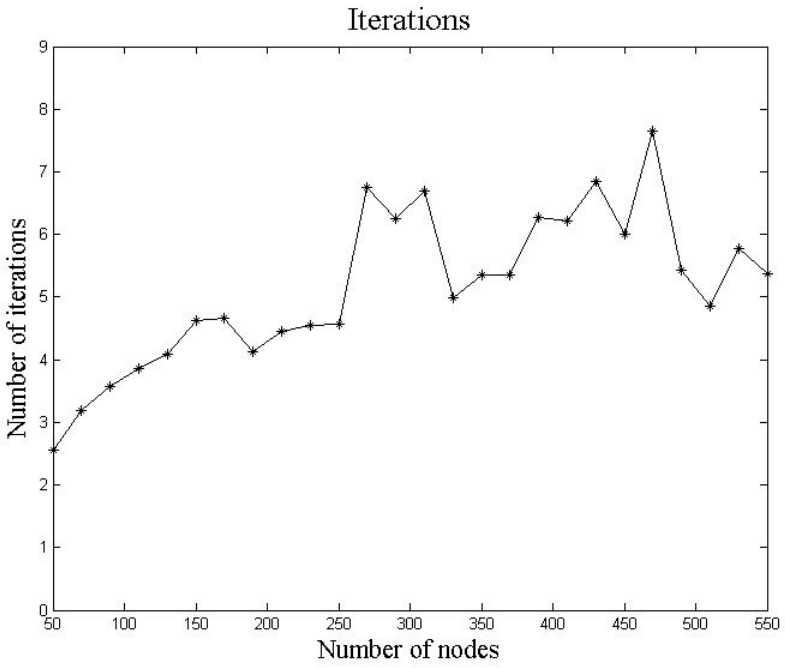


Figure 4.4: Number of iterations needed for the REDMAN manager election protocol.

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 over 30 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 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.

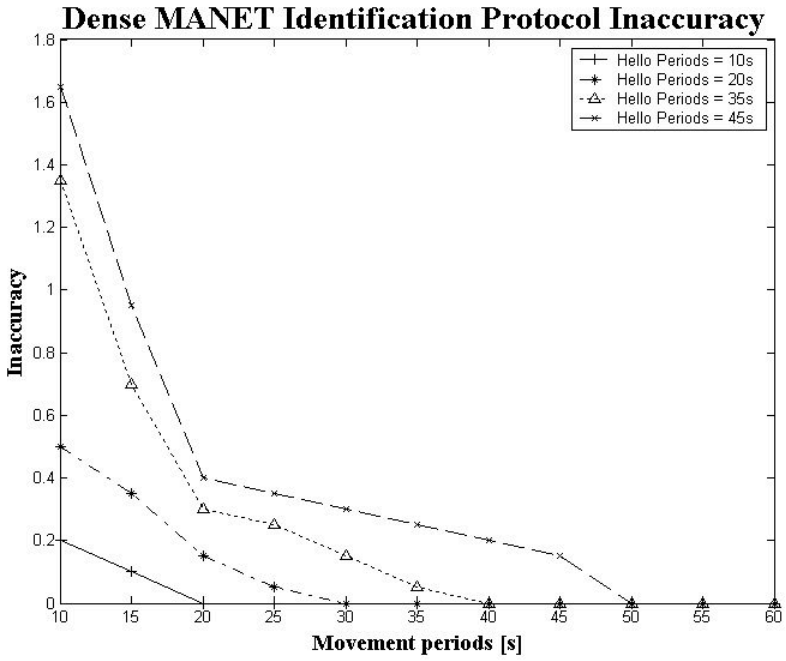


Figure 4.5: Accuracy of the REDMAN dense MANET identification protocol as a function of node mobility.

4.3 Replica Dissemination and Retrieval

In this set of simulations we test the different Replica Dissemination and Retrieval solutions presented in Section 2.5 in the most populated scenario, i.e., IS $side \backsimeq 1.7km$ with 400 randomly positioned nodes. We do not consider nodes within the ES but outside the IS, since they do not participate in the protocol. 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.

4.3.1 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 reiterations 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 4.6 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.

4.3.2 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 4.3.1. Figure 4.7 reports

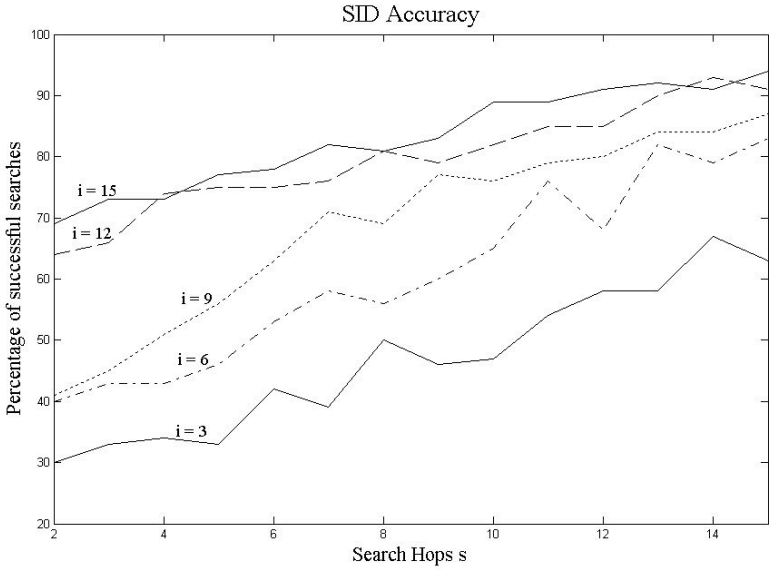


Figure 4.6: SID accuracy while varying i and s.

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 4.7 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 4.1). IF represents a

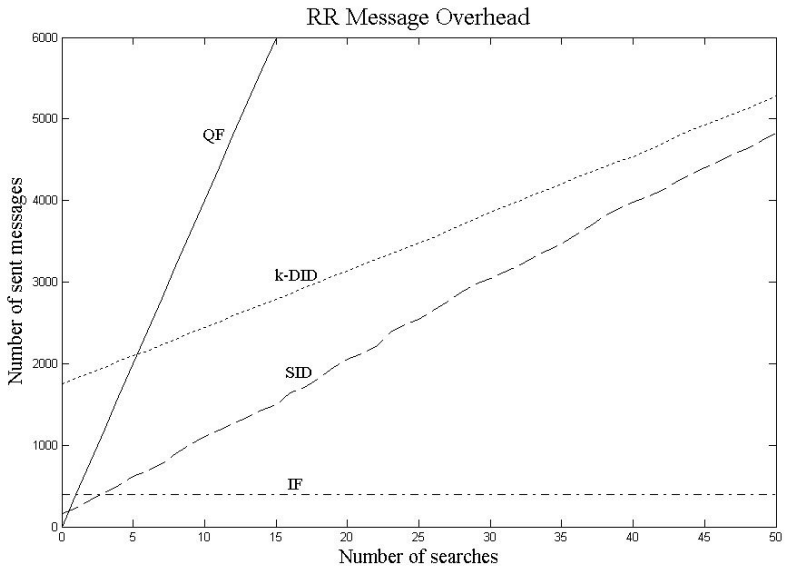


Figure 4.7: Accuracy of the manager election protocol as a function of MaxConsecutiveEqualSolutions.

lower bound for 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.

4.3.3 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 randomly chosen speed direction and speed mod-ule randomly chosen in the range $[1, 5]m/s$.

Figure 4.8 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

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	$O(N)$	dependes on i, density
Retrieval message overhead	0	N	$O(F(k/2))$	dependes 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 4.1: Concison comparison of IF, QF, k-DID, and SID RR strategies

owners remain fixed. As for the overhead evaluation in Section 4.3.2 , 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 out-performs the other. Reconstruction-enabled SID has demonstrated to maintain good accuracy notwithstanding the challenging mobility model adopted with limited message overhead.

4.4 Replica Degree Maintenance

In this section, we report average performance figures about the accuracy and overhead of the crucial RDM facility, which exploits most lower-layer REDMAN mechanisms. To validate the scalability of the proposed RDM solution, we set up two scenarios in the IS $side \backsimeq 1.7km$ area. In the first, we consider a very high population of about 700 nodes (in the whole ES area); in the second, a lower but still considerable population of about 400 nodes.

All nodes are potentially mobile: after a randomly chosen time interval (uniformly distributed in the whole simulation $duration = 7200s$), they start moving along a rectilinear

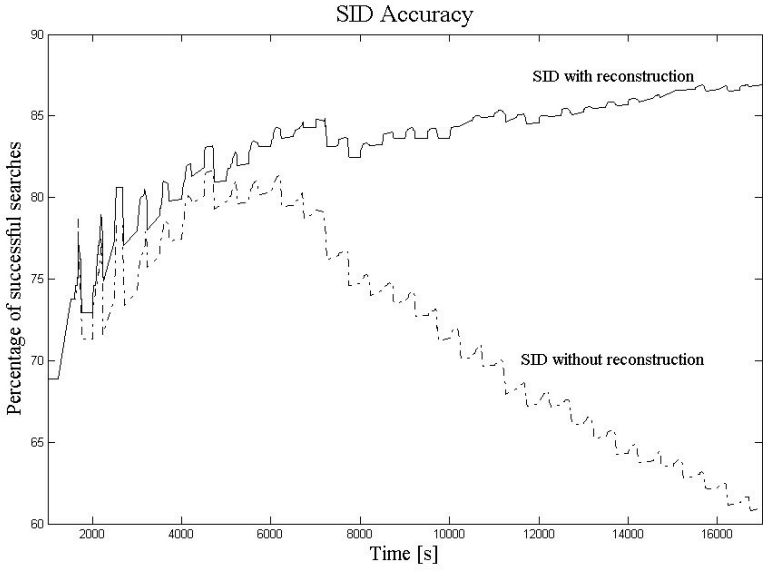


Figure 4.8: SID accuracy with/without reconstruction in mobile scenarios.

path, with randomly chosen speed direction and fixed speed module, ranging from $1m/s$ to $4m/s$. REDMAN disseminates 20 resource replicas in the dense MANET (4 different resources, each one with target replication $degree = 5$). With the above node mobility model, there are on average 51 delegate nodes that exit from the dense MANET during one simulation run (almost all the 20 initial delegates, but also most nodes replacing the exited delegates, and most nodes replacing the substitutes). These are significant worst-case scenarios, since they combine the high number of attendees of Grand Prix stands/stadiums events, with the dynamicity of railway stations/airports waiting rooms. Since in RDM communications are not always local, we considered the ad-hoc on-demand distance vector routing [105] for messages between non-neighbors.

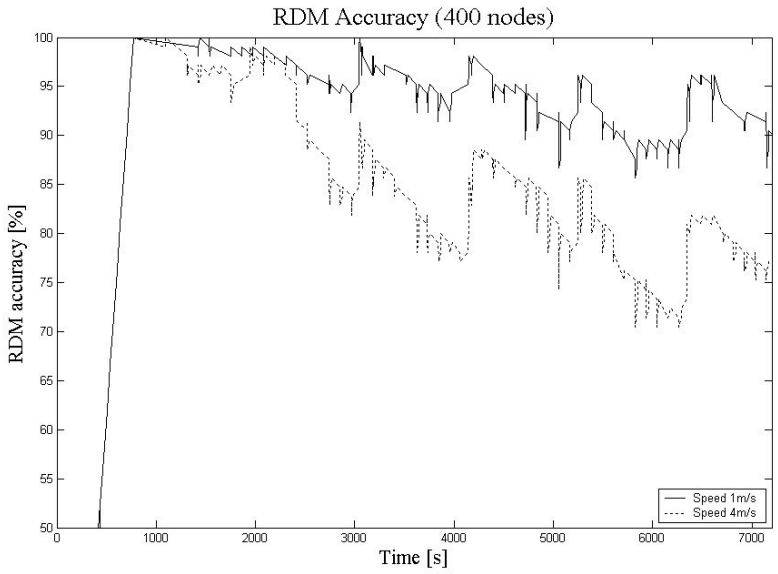


Figure 4.9: RDM accuracy in the 400 nodes scenario.

4.4.1 RDM Accuracy

Figures 4.9 and 4.10 report RDM accuracy, i.e., the ratio between the number of available replicas within the dense MANET and the target replication degree, and its evolution over the simulation time. The different traces refer to the two deployment scenarios and to different speed modules. Notwithstanding the lazily consistent RDM strategy, REDMAN maintains a sufficiently high number of replicas in the dense MANET (almost always over 70%), with a distance from the goal replication degree that slightly increases as the node speed grows.

4.4.2 RDM Overhead

Figure 4.11 provides a quantitative evaluation of the REDMAN network overhead while main-taining 20 replicas of small-sized snapshots (4 snapshots of 60KB each, with 5 replicas per snapshot). In particular, the figure reports the overall traffic due to REDMAN, i.e.,

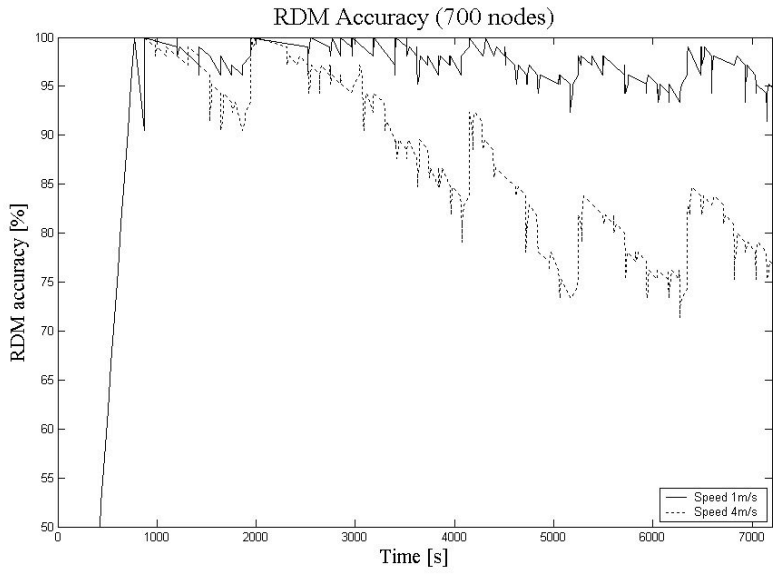


Figure 4.10: RDM accuracy in the 700 nodes scenario.

the grand total of all bytes exchanged between any couple of dense MANET nodes during the whole 2-hours simulation. The results do not include the payload of messages with application-specific content (the snapshot files) exchanged when disseminating new replicas to delegates; only the payload of those messages is excluded, not their headers. This is motivated by the fact that application payload only depends on resource size and do not provide any useful information about the REDMAN overhead.

The REDMAN traffic overhead is very limited, on average about $1.02/0.96KB$ per node for the 400/700-node scenario during the whole 2-hours interval. In addition, it exhibits a very slight growth while increasing the node speed; most important, it almost linearly increases when passing from the 400 node scenario to the 700 one, thus validating the scalability of our proposal. Moreover, most traffic (from 83% to 89%) is due to DMC Hello packets that nodes exchange to maintain an updated view of the network topology.

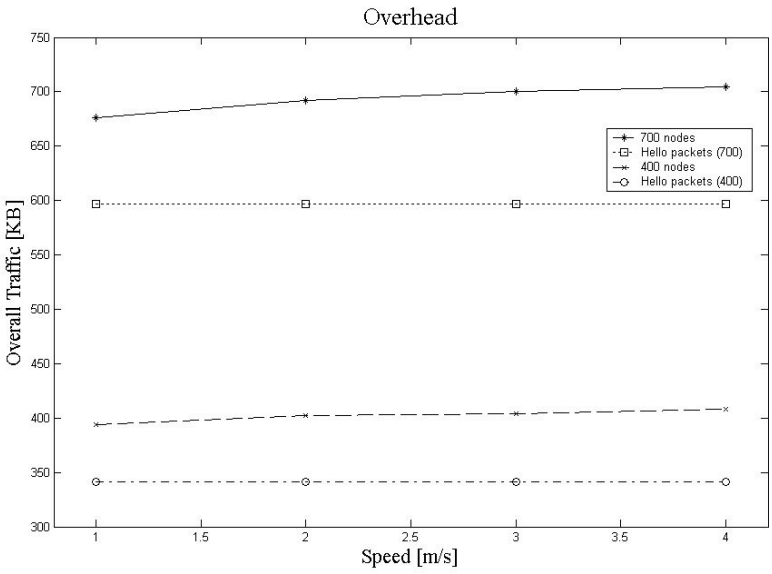


Figure 4.11: RDM network traffic during the two-hour simulation.

The Hello period is configur-able depending on the desired REDMAN promptness to detect node exits and to re-establish the desired replication degree: in all simulations, we have used a $100s$ Hello period, which has demonstrated to be small enough for the $4m/s$ worst-case scenario. With lower node speeds, the Hello period can be increased, thus relevantly reducing the REDMAN overhead: for in-stance, with node $speed = 2m/s$, it is sufficient to have Hello $period = 200s$, which produces a 44% reduction of the overall REDMAN traffic in the stadium-like environment.

Chapter 5

MobEyes Validation

Similarly to the case of REDMAN protocols, we evaluted the MobEyes solution via extensive NS2 [99] simulations. In this case, NS2 reveals even more crucial because of the nature of VSN and MobEyes. First of all, current state-of-the-art research has only recently focused on VANET testbed setup. Most deployed networks only consider a very limited number of vehicles [60]. In addition, MobEyes protocol communication redundancy apparently makes our solution more resilient to packet collision, hence reducing the influence of the medium modeling.

This section shows the most important results, with the goal of investigating MobEyes performance from the following perspectives:

- *Analysis Validation*. We simulate MobEyes protocols for summary collection on regular nodes as well as for agent harvesting and show that they confirm our main analytic results;

- *Effect of k-hop Relay and Multiple Agents*. We examine how MobEyes effectiveness can be increased by leveraging k-hop passive diffusion and the deployment of multiple agents;

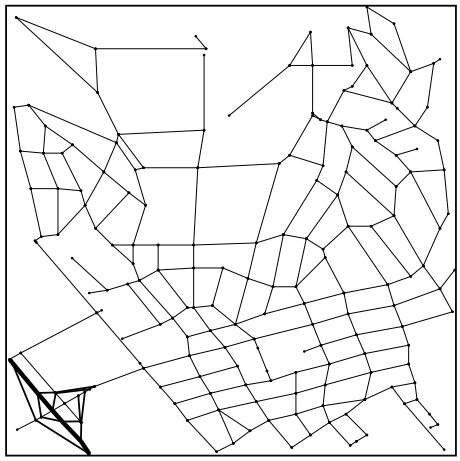


Figure 5.1: Map of Westwood area in vicinity of UCLA campus

- *Summary Diffusion Overhead.* We investigate the tradeoff between harvesting delay and the load imposed on the communication channel;

- *Stability Check.* We verify that the system is stable, even in the worst case of a single harvesting agent and of highest summary generation rate reported in Section 3.4;

- *Tracking Application.* We prove MobEyes effectiveness in supporting a challenging tracking application, where trajectories of regular nodes are locally reconstructed by a police agent based on harvested summaries;

- *Border Effects and Turn Over.* We show that MobEyes performance does not dramatically change in case of more dynamic mobility models, where nodes are allowed to enter/exit from the simulated area.

Additional experimental results are available at `http://www.lia.deis.unibo.it/Research/MobEyes/`.

5.1 Simulation Setup

We consider vehicles moving in a fixed region of size $2400m \times 2400m$. The default mobility model is Real-Track (RT), introduced by our colleagues in [140]. RT permits to model realistic vehicle motion in urban environments. In RT nodes move following virtual tracks, representing real accessible streets on an arbitrary loaded roadmap. For this set of experiments, we used a map of the $Westwood$ area in the vicinity of the UCLA campus, as obtained by the US Census Bureau data for street-level maps [129] (Figure 5.1). At any intersection, each node randomly selects the next track it will run through; speed is periodically allowed to change (increase or decrease) of a quantity uniformly distributed in the interval [0,$\pm\Delta s$]. To evaluate the impact of the mobility model on MobEyes performance, we tested two additional well-known models, namely Manhattan (MAN) [6] and Random WayPoint (RWP) [16]. Similarly to RT, MAN builds node trajectories following urban roads; however, in MAN roads are deployed according to a regular grid, thus allowing a more uniform node deployment. In our simulation, we adopted a 10×10 grid. RWP instead does not constrain node positions to follow actual road tracks, but moves nodes toward randomly selected destinations with random speeds. When a node reaches its destination, it remains still for a fixed period (which we set equal to 0 by homogeneity with the other models), and then selects a new destination. Surprisingly, RWP is considered "a good approximation for simulating the motion of vehicles on a road" [24], generally producing limited distortion on protocol performance. Let us remark that MobEyes agents do not exploit any special trajectory or controlled mobility pattern, but move conforming with regular nodes.

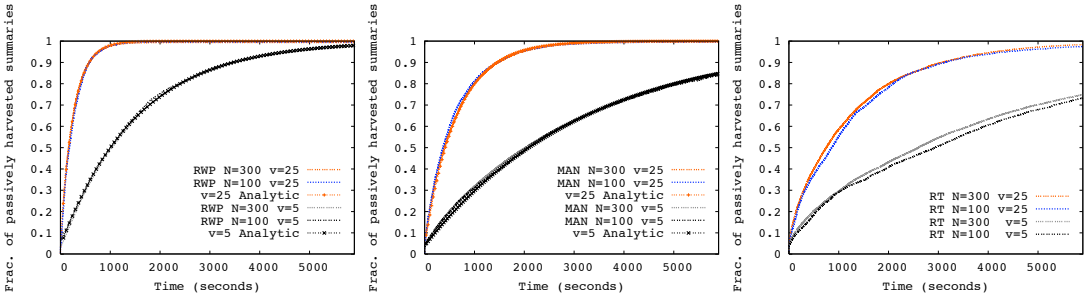
Our simulations consider number of nodes $N = 100, 200, 300$. Vehicles move with average speed $v = 5, 15, 25$; to obtain these values, we carefully tuned maximum (v_M)

and minimum (v_m) speeds depending on the mobility model. The summary advertisement period of regular nodes and the harvesting request period are kept constant and equal to 3s through all the simulations. We note that if the value of this parameter is too large, MobEyes effectiveness is reduced since it is possible that two nodes do not exchange messages, even if they occasionally enter in each other transmission range; this effect is magnified, as node speed v increases. The chosen value has been experimentally determined to balance the effectiveness of our protocol and the message overhead, even in the worst case, i.e., $v = 25$. A deeper and more formal investigation of the optimal value of the advertisement period is object of future work.

Finally, we modeled communications as follows: MAC protocol IEEE 802.11, transmission band 2.4 GHz, bandwidth 11Mbps, nominal radio range equal to 250m, and Two-ray Ground propagation model [107]. The values of these parameters have been chosen similar to other work in the field [139] [32]. Where not differently stated, reported results are average values out of 35 repetitions. Other MobEyes configuration parameters will be introduced in the following sections, when discussing the related aspects of MobEyes performance.

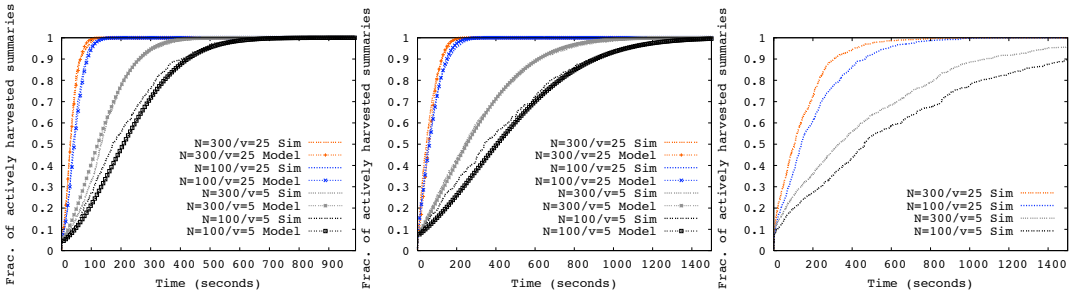
5.2 Analysis Validation

Our first goal is to validate the results obtained in Section 3.4. In particular, we investigate the regular node collection and agent harvesting processes, as described respectively by Equations 3.2 and 3.7. Without loss of generality (see Section 5.5), let us assume that new summaries are synchronously generated by all regular nodes. A *generation epoch* is the time interval between two successive summary generations. In this set of experiments, every regular node continuously advertises the single summary it generated in the epoch



(a) RWP (b) MAN (c) RT

Figure 5.2: Fraction of *passively* harvested summaries by a regular node



(a) RWP (b) MAN (c) RT

Figure 5.3: Fraction of *actively* harvested summaries by an agent

$t = 0$ for the rest of the simulation run. Since Equations 3.2 and 3.7 characterize the spreading processes of all summaries generated in the same epoch, it is not necessary that regular nodes generate additional summaries. We remark that this assumption does not undermine the relevance of our results because the process is stationary as described in Section 5.5.

Figures 5.2 and 5.3 show results collected for number of nodes $N = 100/300$, average speed $v = 5/25$, and RWP, MAN, and RT mobility models. In particular, Figure 5.2 plots the cumulative distribution of summaries collected by regular nodes as a function of time. The figure shows that the process highly depends on the average node speed; in fact, the speed determines to a large extent how quickly nodes "infect" other participants with their own summaries. The results do not depend on node density as we have shown in Equation

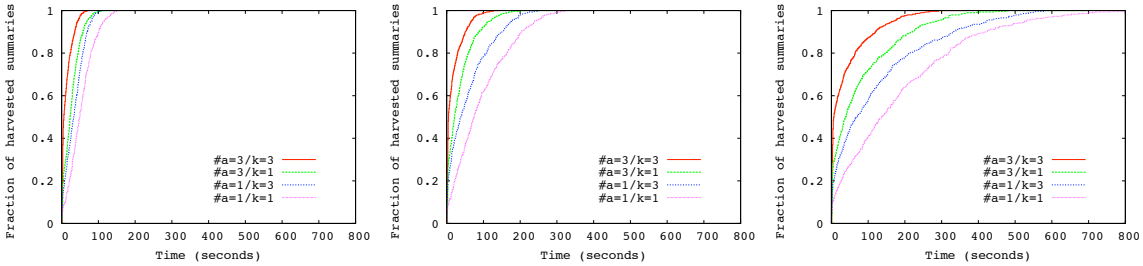
3.4. Our analytic model (Equation 3.2) accurately fits the simulation results for RWP and MAN. The curves of RT results exhibit worst fitting since they start deviating from that of analytical results after certain thresholds. RWP shows slightly better accuracy mainly due to node unconstrained motion and to their tendency to gather at the center of the field as time passes [12]. Although both MAN and RT show the restricted mobility patterns, their node distributions are different: MAN has almost uniform node distribution over the grids, whereas RT exhibits non-uniform node distribution over the map as shown in [79]. Since our model is based on uniform distribution, the curve fitting works well with MAN. Experimental results helped us to tune the constant compensation factor η we introduced in Section 3.4 to take into account the non-uniform movement patterns. In detail, the values we found are 0.97 and 0.9 respectively for $v = 15m/s$ and $v = 25m/s$ in RWP and 0.40 and 0.36 in MAN. As restrictions on node mobility grow due to the mobility model, the values of η decreases, thus representing a slower process.

Figure 5.3 plots the cumulative distribution of summaries harvested by a police agent as a function of time. The figure shows that the results are mainly dependent on the speed. Unlike passive harvesting, the density plays an important role in active harvesting. In the analysis section, we show that the higher the density, the faster is the harvesting progress (Equation 3.7). Intuitively, if there are more neighbors, the agent has a higher chance of getting a random summary. Our analytic model fits well the simulations, especially when we have large N and v. This set of results allowed us to tune the parameter γ accounting for the effect of overlapping of summaries contributed from regular agents (see Section 3.4). In detail, the values we found are 0.21 and 0.21 respectively for $v = 5m/s$ and $v = 25m/s$ in RWP and 0.15 and 0.20 in MAN.

5.3 Effect of k-hop Relay and Multiple Agents

The effectiveness of MobEyes harvesting can be measured in terms of the fraction of summaries harvested by the agent(s) in function of time. To enhance the validity of our conclusions, it is important to determine the dependence of the performance indicators on different mobility models. In [77] we only investigate RT mobility model; here, we extend the results to RWP and MAN. For every mobility model, we show plots for $1, 3$ agents ($a\#$) and for $1, 3$ relay hops (k). The summary harvesting latency is a crucial figure to determine the feasibility of the MobEyes approach, since it allows us to estimate the fraction of harvested summaries by the agent within a certain time t. This estimation is useful to decide the tuning of the parameters (k-hop relay scope and number of agents) to address application requirements. Figure 5.4 shows how the number of agents, the choice of the number of relaying hops k, and the average speed v of the nodes influence the process. Figure 5.4 plots the cumulative distribution of the summaries harvested for $N = 300$, $v = 15m/s$. In the case of multiple agents, the harvesting process considers the union of the summary sets harvested by agents. The figure clearly shows that k-hop relay scope and multiple agents highly impact harvesting latency.

By carefully inspecting the results in Figure 5.4, it is possible to obtain some guidelines for the choice of MobEyes parameters. For example, given as a baseline a network with $N = 300$ nodes moving with an average speed $v = 15m/s$, fixed $k = 1$, a single agent employs $530s$, $236s$, and $116s$ to harvest 95% of the summaries generated respectively in RT, MAN, and RWP mobility models. By increasing k to 3, times respectively reduce to $420s$, $176s$, $86s$, showing an improvement of about $20 - 30\%$ in all cases. Instead, increasing the number of agents to three, times become respectively $280s$, $123s$, and $68s$; in this case, the improvement is in the $40 - 50\%$ range. Finally, if we set $v = 25m/s$,



(a) RWP (b) MAN (c) RT

Figure 5.4: Fraction of *actively* harvested summaries by multiple agents with k-hop relay ($N = 300$ and $v = 15$)

times become $211s$, $67s$, and $43s$; the improvement is around $60 - 70\%$. Interestingly, the relative impact of the three parameters (harvesting team size, multihop forwarding, and speed) shows a limited dependence on the mobility model. This holds also for the results we collected for different cases (different values of N and v): in particular, speed has a larger impact than the number of agents, and k is the less decisive factor.

5.4 Summary Diffusion Overhead

The study of the diffusion overhead helps us understand the requirements imposed on the underlying vehicular communication technology and to determine if MobEyes can coexist with other applications. For example, the parameter k shows the largest impact on the performance; the effect due to a small number of agents is negligible, since they are only responsible for local single-hop traffic. Figure 5.5 shows the average received packets per node per second, obtained during a simulation time of $1000s$. In this set of simulations, we fixed $k = 1$, and changed all the other parameters, i.e., mobility model (RWP, MAN, RT), N (100, 200, 300), and v (5, 15, 25). As expected, the number of received packets linearly increases as the number of nodes increases. Therefore, for the sake of clarity, the figure only reports the case with $N = 300$. In addition, the number of received packets

exhibits no dependence on v. In all considered cases, the overhead is limited, on the order of few (two to five) packets per second, proving the low impact of MobEyes on the available bandwidth.

The latter result could mislead to conclude that speed increments would not impact the harvesting latency, since the number of received packets would not change. This apparently invalidates our previous results (see Figure 5.2) and has the following motivations. For a fixed advertisement interval, as average speed increases, the probability of useful meetings (i.e., of receiving a non-redundant summary) increases because there is more mixing among mobile nodes. For example, given an average speed v, let us assume that the average period that any two nodes are within their communication ranges simply be $2R/2v$. Then with v set to 5 and 25 m/s, and $R = 250m$, the periods can be estimated as $50s$ and $10s$ respectively. This implies that the cases of $5m/s$ has roughly 5 times higher chances of receiving redundant advertisements than the case of $25m/s$. It is interesting to note that, fixed the average speed, there exists an optimal advertisement period allowing to maximize non-redundant summary diffusion, while minimizing the overhead. It will be part of our future work to analytically determine this value.

Figure 5.6 shows the magnifying effect produced by an increase of the parameter k. k-hop relaying produces an enlargement of the area where summary packets are diffused intuitively proportional to k^2. Consequently, also the number of nodes affected by a single summary diffusion will be about k^2 larger than the single-hop case. Moreover, while in the single-hop case nodes receive any summary packet only once, with k-hop relaying any node within k-hops from the originator receives it a number of times proportional to the number of its neighbors. Thus, the total overhead is expected to increase by a factor larger than k^2 but lower than k^2 times the average number of neighbors (please note that

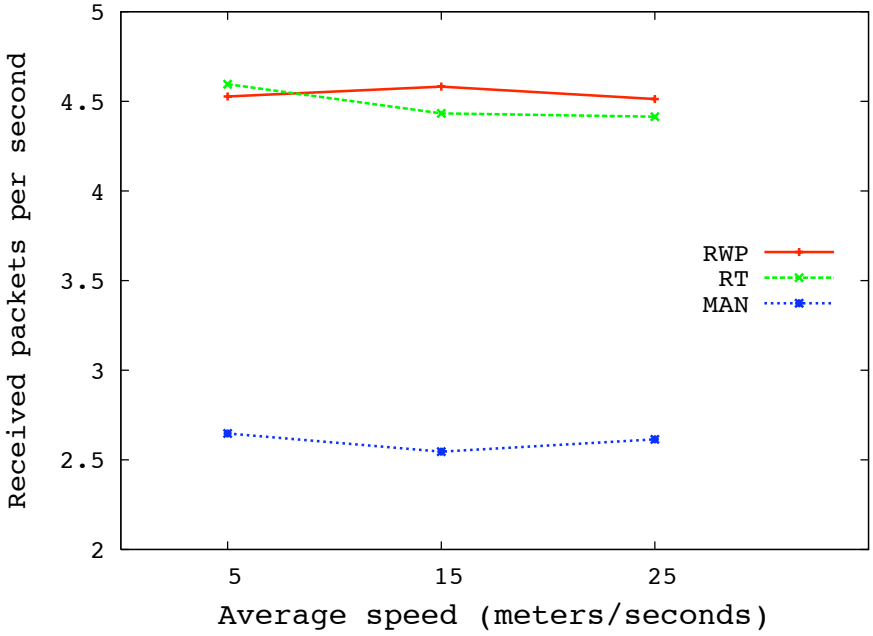


Figure 5.5: Total number of received packets *(k=1)*

k-hop distant nodes do not relay packets, thus reducing the latter factor for k-hop as well as $k-1$-hop distant nodes). The combination of these results with those in Figure 5.4 lead us to conclude that parameter k permits to decrease harvesting latency (about $20-30\%$ for $k=3$) at the price of relevant overhead increase (around $15-20$ times). The proper balance of latency/k tradeoff can be only decided depending on specific characteristics and requirements of the supported urban monitoring application.

5.5 Stability Check

In the following, we investigate the stability of MobEyes, by verifying that continuous summary injections do not influence its performance to a large extent. In particular, we show that the ratio of summaries harvested on longer periods remains acceptable and that the harvesting latency does not grow as time passes. With regard to the results presented so far,

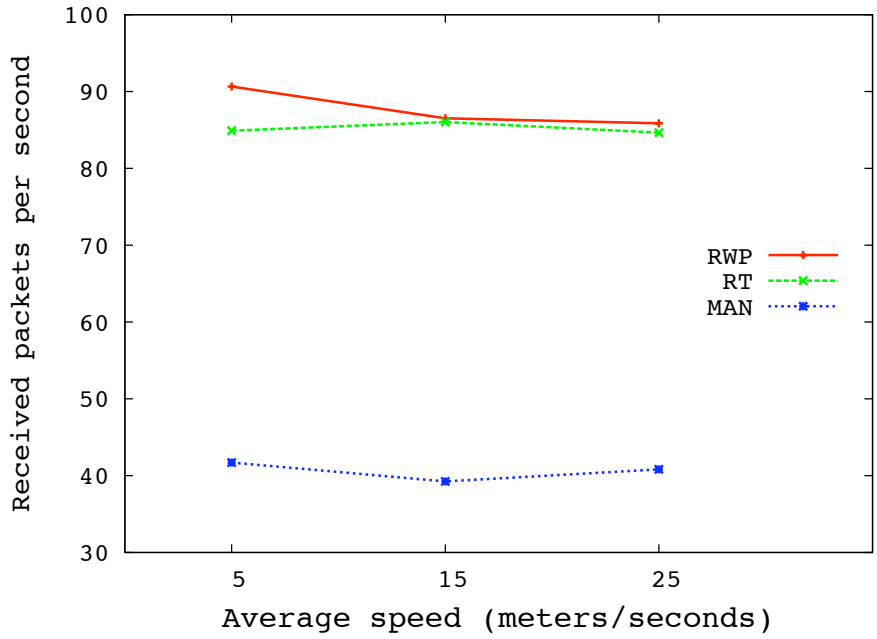


Figure 5.6: Total number of received packets *(k=3)*

here we remove the assumption about the single summary generation epoch at $t = 0$. Nodes generate new summaries with period $T = 120s$ and advertise the last generated summary: let us observe that according to the discussion presented in Section 3.2 this rate represents a practical worst case. For the sake of clarity of presented results, we hold the synchronicity assumption: all nodes simultaneously generate new summaries at intervals multiple of T. We obtained similar performance with differently distributed generation intervals, i.e., Poisson with average value T but plots (especially related to results in Figure 5.8) are far more jumbled. The following results are reported for the case of a single harvesting agent, $k = 1$, $N = 100$, $v = 15m/s$, and nodes moving according to the RT model. Figure 5.7 plots the cumulative distribution of the number of summaries generated and harvested as a function of time (we ran simulations for $6000s$). The graph shows that the harvesting curve

tracks the generation curve with a certain delay, which can be traced to the harvesting latency in Section 5.2. This also motivates the difference of the endpoints of the two plots. Figure 5.8 provides further evidence of the stability of the system; curves show the harvesting latency for summaries generated during some generation epochs. For the sake of figure clarity, the graph does not exhaustively represent every generation epoch, but only samples one generation epoch every $T * 7 = 840s$ till the end of the simulation time. The different curves show similar trends, without any performance degradation caused by the increase of the number of summaries in the network. The harvesting related to the last summary generation epoch is evidently incomplete (25% of the summaries are harvested within the timeline), since the epoch starts $120s$ before the end of the simulation. These results prove that MobEyes achieves completeness in harvesting generated summaries even in practical worst cases.

We also investigated if higher summary generation rates afflict MobEyes performance. We shortened T from $120s$ to $6s$ (with $T = 6s$, the chunk generation rate is $100ms$). Such a generation rate is largely greater than the one required for the set of applications addressed by MobEyes. Simulation results prove that MobEyes performance starts degrading only when $T < 30s$. Figure 5.9 shows the harvesting process for two epochs ($0s$ and $2520s$) and compares $T = 120s$ with $T = 6s$. The second case shows that MobEyes performance degrades gracefully as the generation epoch shortens, thus demonstrating the high stability of the system when operating in usual summary rate conditions.

5.6 Tracking Application

In the Section 3.1 we sketched some application cases for MobEyes. For the sake of proving its effectiveness in supporting urban monitoring, we also simulated a vehicle tracking

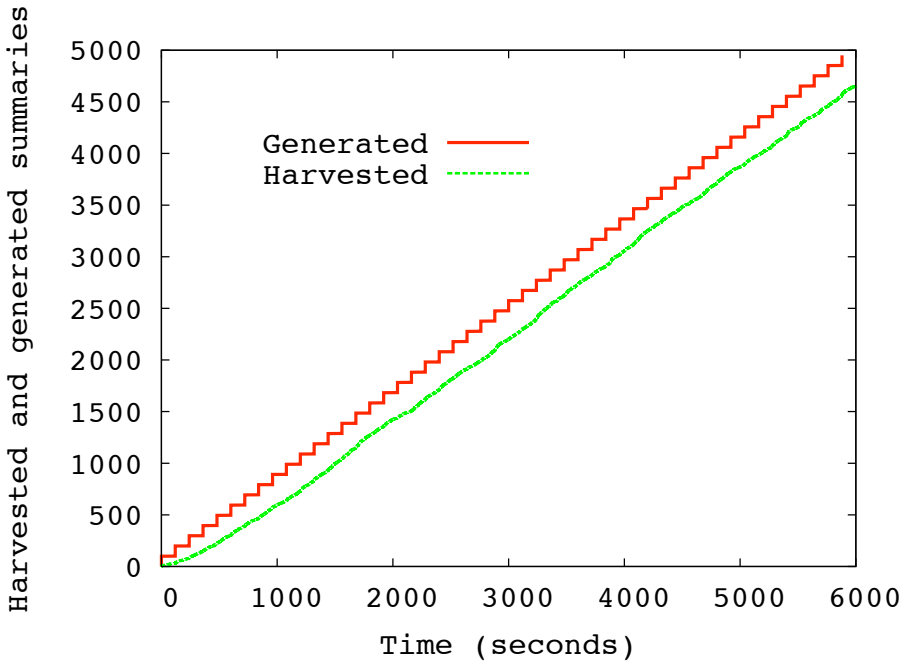


Figure 5.7: Cumulative distribution of generated and harvested summaries *over all epochs*

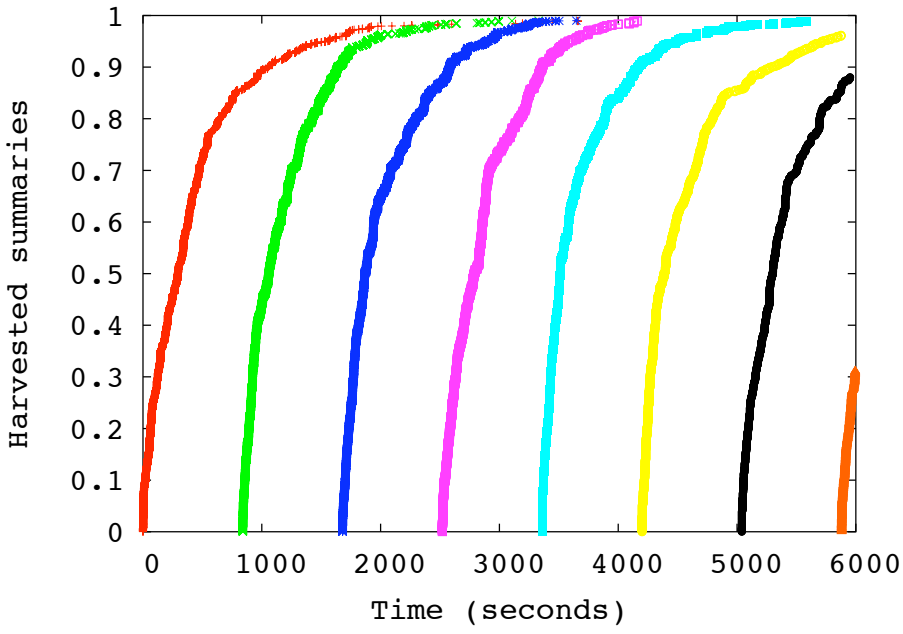


Figure 5.8: Cumulative distribution of harvested summaries *per epoch*

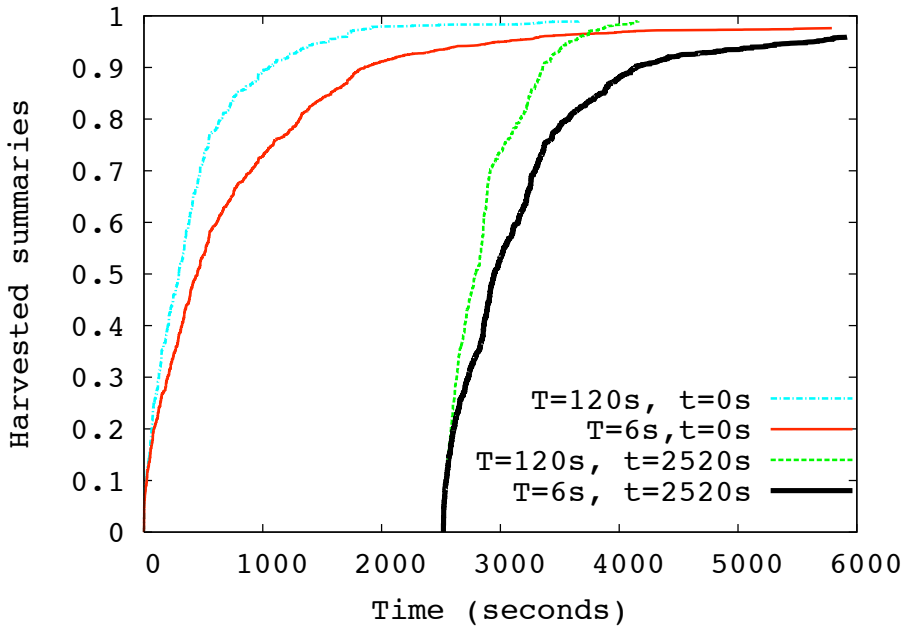


Figure 5.9: Cumulative distribution of harvested summaries *per epoch*

application where the agent reconstructs node trajectories exploiting the collected summaries. This is a challenging application, since it requires our system (1) to monitor a large number of targets, i.e., all participant vehicles, (2) to periodically generate fresh information on these targets, since they are highly mobile, (3) to deliver to the agent a high share of the generated information. Moreover, since nodes are generally spread all over the area, this application shows that a single agent can maintain a consistent view of a large zone of responsibility. More in details, as regular cars move in the field, they generate new summaries every $T = 120s$ and continuously advertise the last generated summary. Every summary contains 60 *summary chunks*, which are created every $ChunkPeriod = 2s$ and include the license plate and position of the vehicle nearest to the summary sender at the generating time, tagged with a timestamp. The application exploits the MobEyes diffusion protocol with $k = 1$ to spread the summaries and deliver as much information as possible to

a single agent scouting the ground. As the agent receives the summaries, it extracts the information about node plates and positions, and tries to reconstruct node trajectories within the area. This is possible by aggregating data related to the same license plate, reported from different summaries.

To determine the effectiveness of MobEyes we decided to evaluate the *average uncovered interval* and *maximum uncovered interval* for each node in the field. Given a set of summary chunks related to the same vehicle and ordered on time basis, these parameters measure respectively the average period for which the agent does not have any record for that vehicle and the longest period. The latter typically represents situations in which a node moves in a zone where vehicle density is low; thus, it cannot be traced by any other participant. We associated the average and maximum uncovered intervals to each simulated node, and present the results in Figure 5.10 (note the logarithmic scale on the Y-axis). Every point in the figure represents the value of the parameter for a different node. We sorted nodes on the X-axis so that they are reported with increasing values of *uncovered interval*. Results are collected along a $6000s$ simulation. The plot shows that in most cases the average uncovered interval floats between $[2.7s - 3.5s]$; the maximum uncovered interval shows that even in the worst cases the agent has at least one sample every $200s$ for more than 90% of the participants. A more immediate visualization of the inaccuracy is given in Figure 5.11. This figure shows, for the case of a node with a maximum uncovered interval equal to $200s$ (i.e., locating this node in the lowest $10th$ percentile), its real trajectory (the unbroken line), and the sample points the agent collected.

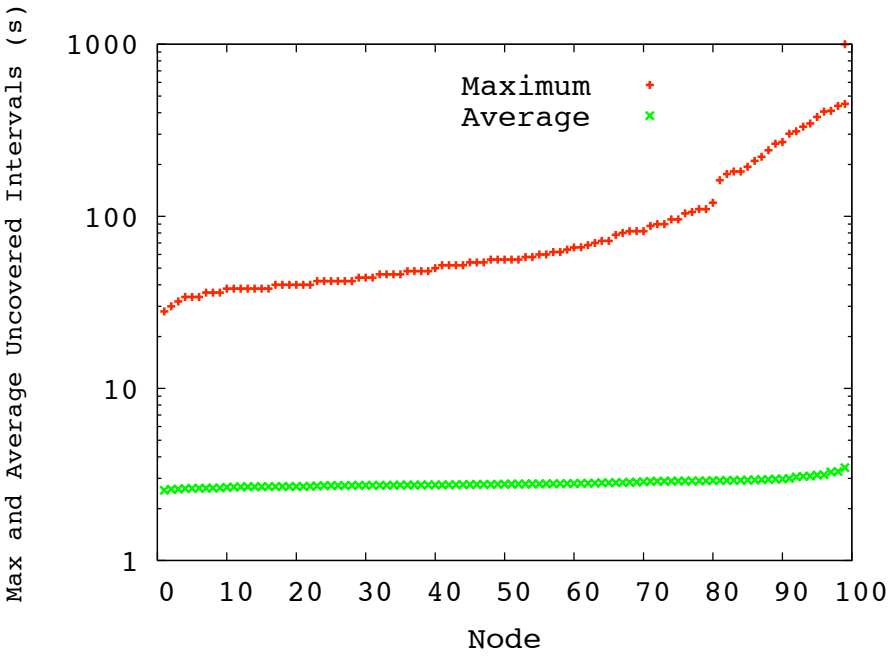


Figure 5.10: Maximum uncovered intervals per node

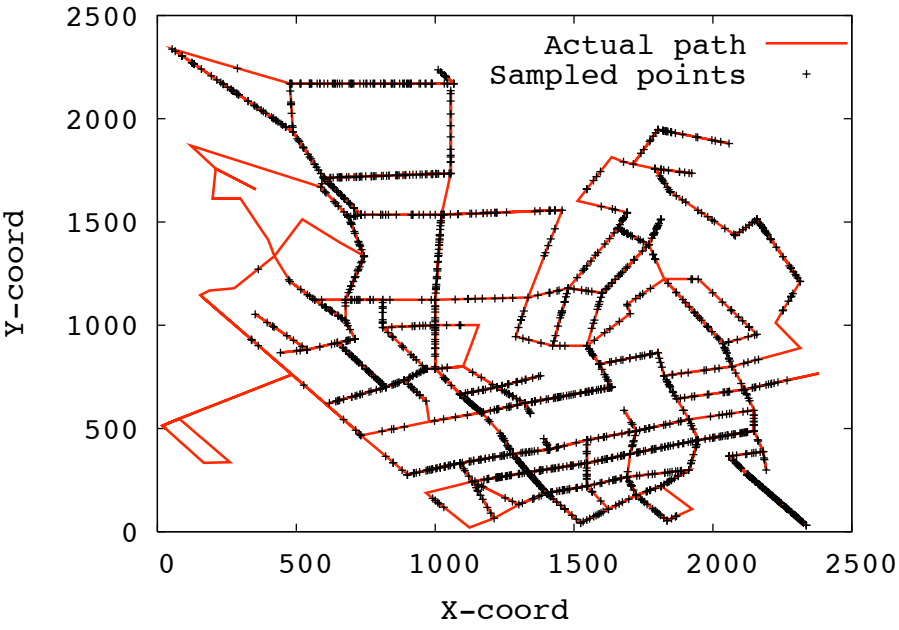


Figure 5.11: Actual node trajectory vs harvested sampled points

5.7 Border Effects and Turn Over

Usual mobility models [23], such as RWP, MAN, and RT, assume that nodes remain within the simulated area during the whole simulation (in the following, we indicate them as *closed* mobility models). Even if this does not necessary hold for MobEyes applications, we observe that this assumption does not invalidate our findings. First, if we consider a sufficiently large area, on the order of tenths by tenths km^2, the amount of time that nodes continuously reside within the area is likely very long, approximating for most nodes a closed mobility model. Second, the worst effect of dynamic scenarios takes place when nodes leave a specific area carrying several summaries (locally generated or collected) not harvested by the local agent yet. Nonetheless, we remark that carried information does not vanish as nodes leave, but can be harvested later by remote agents, responsible for the adjacent area the leaving nodes are moving into.

However, to estimate how node entrances/exits impact presented results, we tested MobEyes with a novel mobility model, *open-RT*, which takes these effects into account. In open-RT nodes follow the same patterns of RT, with one exception: as soon as a node reaches the endpoint of a track, close to the boundary of the area, it suddenly disappears. To maintain unchanged the number of nodes within the area, and obtain results comparable to the ones presented in the previous sections, we assume that the net vehicle flow in/out the area is null. Thus, any node exiting from the area is immediately replaced with one node entering; the latter is placed at the endpoint of a random road, close to the boundary of the area.

This dynamic effect is better evaluated for long simulation periods and periodic summary generation epochs. Thus, we confirm the settings used in Sections 5.5 and 5.6; in addition, we consider a single harvesting agent, $k = 1$, $N = 100$, $v = 15m/s$. Nodes

generate new summaries synchronously, and only as long as they remain in the area. To avoid that nodes stay within the area only for very short periods, we introduce a constraint on their minimum residing time equal to 10% of the whole simulation. Even with this assumption, more than 550 nodes need to take turns on the simulation area, to maintain 100 nodes always present. The agent does not follow open-RT model, but traditional RT, i.e., it always remains within the area.

Figures 5.12 and 5.13 present results corresponding to those in Sections 5.5 and 5.6, but obtained with the open-RT model. Significant conclusions can be drawn, especially from Figure 5.12: also under these unfavorable assumptions, the agent is able to collect more than 85% of any generated summary, and in most cases it reaches 90%. By inspecting simulation traces, we could find that missing summaries generally originate by vehicles leaving the area within a short interval from any epoch. In that case, the last generated summary is only advertised for that short interval and cannot spread enough to reach the agent. Let us remark once more that those summaries are not irreparably lost, but will be probably harvested by agents in charge of the adjacent areas. Figure 5.13 shows average and maximum uncovered intervals as obtained with the open-RT model. The quality of the reconstructed trajectories is only slightly degraded, given that the average uncovered interval is below $4s$ for more than 75% of the nodes (and below $10s$ for 90%), and that the $85th$ percentile of the vehicles can be tracked with a worst-case inaccuracy of $200s$.

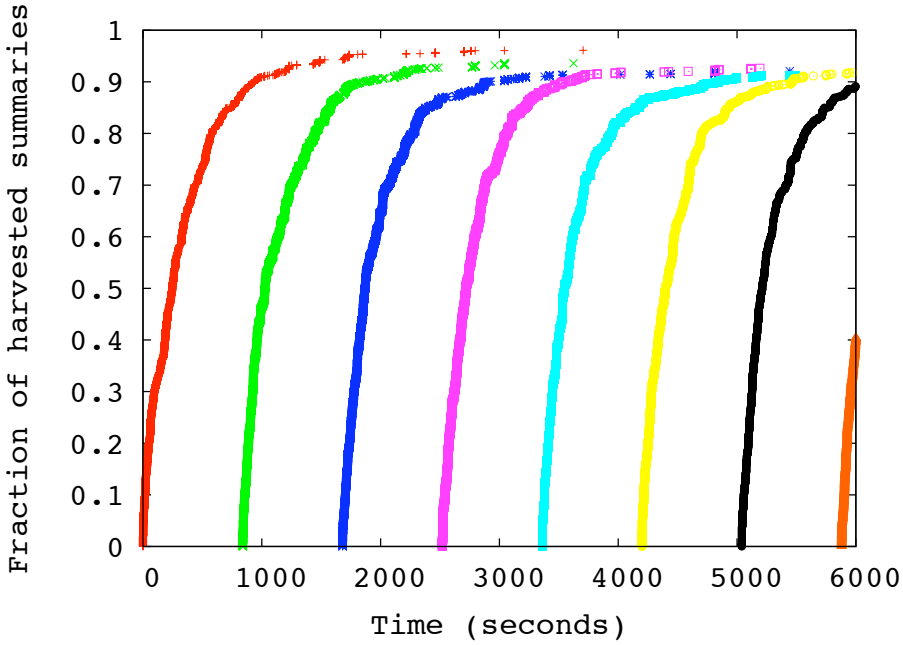


Figure 5.12: Cumulative distribution of harvested summaries *per epoch (open-RT)*

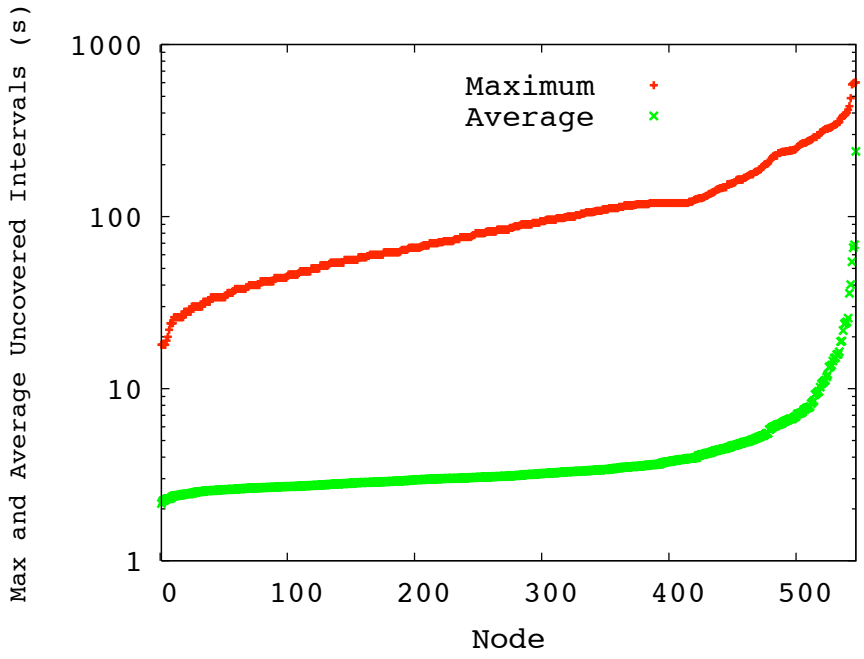


Figure 5.13: Maximum uncovered intervals per node *(open-RT)*

Chapter 6

REDMAN Implementation

To support REDMAN operations we have designed and implemented the architecture described in Section 2.2. We briefly recall that it is composed of four main components respectively in charge of carrying dense MANET configuration (DMC), replica dissemination (RD), retrieval (RR), and degree maintenance (RDM). These components can be logically organized into three macro-modules, according to the role of the entity they support. Delegates perform resource dissemination and retrieval, and are also partly responsible of degree maintenance; in fact, in case of exit from the dense MANET, they are in charge of notifying the manager, and getting rid of their shared resources via neighbor uploads. Managers only perform RDM operations; in particular, they decide resource replication degree as a new delegate enters the dense MANET, and counteract degree inconsistencies, by invoking new replications. To this end, they need a Shared Resource Table verifier, periodically checking actual vs. established degree. Finally, the Dense MANET Configuration module is implemented on all nodes participating in the REDMAN network. DMC identifies nodes belonging to the dense MANET and chooses suitable managers.

Before analyzing REDMAN design and implementation in more details, we rapidly identify the key requirements of our work.

- *Flexibility*: REDMAN components include distributed protocols to effectively perform replica management operations (e.g., dissemination and retrieval). We aim at designing an architecture that can easily adapt to protocol/strategy changes.

- *Portability*: addressed scenarios accomodate highly heterogeneous user devices; thus, to permit the seamless integration of REDMAN on most of them, we developed our system in Java, allowing the execution on all devices including a virtual machine implementation.

- *Lightweightness*: in waiting rooms, but even more in stadium-like scenarios, users are likely equipped with portable devices, such as PDAs and smart phones. These are resource limited devices unable to support the full functionality of a Java standard edition, but in most cases only a reduced version called Java Micro-Edition.

- *Energy-awareness*: resource limited devices emphasize energy issues. So far, we have only provided a way for a node to waive manager election if energy is not enough. However, our current research mostly focuses on global effective strategies, e.g., to place replicas according to accurate energy evaluations.

- *Modularity*: memory is likely limited on current portable devices. We address a modular design to permit to easily exclude the local burden of un-necessary components, e.g., for replica dissemination and manager coordination.

6.1 Design

This section discusses the logic design of REDMAN middleware. The presented components refine the high-level architecture in Figure 2.2, include details of some interfaces, and clarify system interactions. However, we observe that the described architecture is not yet

ready to be implemented: additional details will be specifically included in the next section. According to the introduction we separate the subject in three main modules: DMC, delegate and manager.

6.1.1 DMC

DMC represents the basic module for operating dense MANET, and provides functionality to access topology information to upper-layer modules. The two main components discussed in Section 2.2 reflect the design architecture (Figure 6.1); DMC coordinates their operations by hiding details from upper layers. The exposed function `setupDenseMANET` transparently performs dense MANET identification and manager election. The *Density Enquiry Mng* broadcasts a discovery message; then, when a timer expires, it starts the *Election Manager*, which in turn activates the *Farthest Node Identification*.

As soon as the operations end, the identifier of the currently elected manager is stored, and the control is returned to the caller. However, the *Identification Agent* continuously runs to perform dense MANET and manager role maintenance operations. In particular, it maitains a *Neighbor Table* via Hello message exchanges (see Section 2.3.1), which is specifically needed in RD/RR protocols. Upper layers (e.g., delegate/manager RDM) can register a "topology" callback (e.g., according to Java listener architecture) to be notified when the node is exiting the dense MANET (i.e., when the number of neighbors rapidly decreases or approaches the threshold value n). On the replica manager, this same execution flow periodically runs the *FarthestNodesIdentification* component to determine if the conditions to carry manager role still locally hold.

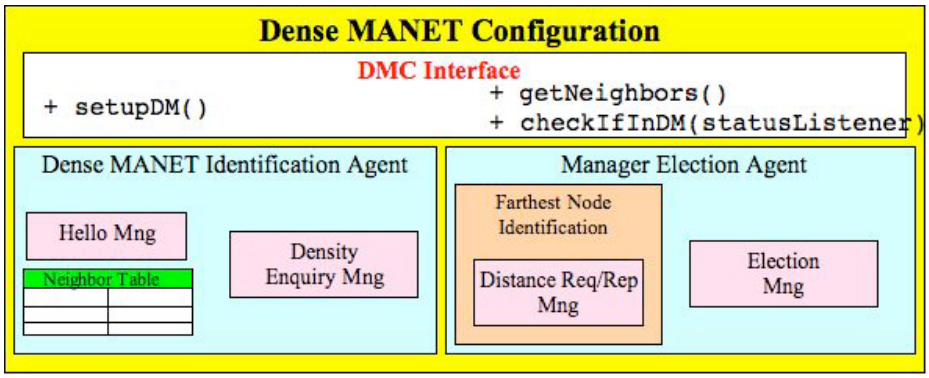


Figure 6.1: The modular architecture of REDMAN DMC

6.1.2 Delegate

The delegate architecture (Figure 6.2) builds upon DMC and is more complex. First, it includes a *ResourceTable* containing one entry for each shared resource. The user can populate this table via the `addResource` method; every time a new resource is added, the user should specify its importance level. As soon as a node enters a dense MANET (let us rapidly observe that the status of the node, whether belonging to the dense MANET or not, can be accurately monitored via the same topology DMC callback described few lines above.), the *ManagerCoordinator* delivers the list of carried resources, with an RDF descriptor [37], to the network manager to obtain suitable replication degrees.

Then, according to the replication degree, the *Replica Propagation Strategy* agent performs the distribution. This is provided as an abstract component, since it only represents an interface implemented by the actual agent (the instantiation takes place through the Factory pattern [49]). In the current prototype, a SID agent is included, performing the "straight" line deployment of IRPs (including the RDF description of the resource) and replicas (see Section 2.5.3); receiving nodes store IRPs in the *IRPTable*. Obviously, the

Replica Propagation Strategy agent needs to move the full resource along the dissemination path. However, especially for low-end portable devices (e.g., smart phones), access to local resources is device dependent (more details are in the next section). Thus, with portability goals in mind, we exploited the Adapter pattern [49] that permits to abstract the actual access protocol behind generic methods.

The delegate interface exports two replica retrieval methods, i.e., `search` to find a node sharing the needed resource, and `getResource` to effectively command the download. Similarly to replica dissemination, to maintain flexibility and permit to seamlessly change the strategy, we provide an abstract replica retrieval component, exposing a generic `retrieve` method. The prototype implements the SID retrieval agent, receiving a parameter related to the searched resource (i.e., its RDF description), and returning the identifier of a node sharing that resource. A *Replica Snd/Recv* component is responsible for downloading/uploading the needed replica. Two methods, `getResource` and `sendResource`, respectively implement the operations to download replicas and to upload. Both of them deal with the *ResourceManagerAdapter* to extract and store the resource.

If the receiving node decides on its turn to share the downloaded resource, `getResource` interfaces with the RD's *ManagerCoordinator* via `addResource` method (which invokes `notifyAddedResource`). Let us observe that the *Replica Snd/Recv* component is the only REDMAN component supposed to operate upon a routing protocol for the resource delivery. In fact, resource provider can be several-hop distant from requester. Our recent work consists in comparing the effectiveness of multi-hop routing solutions, with alternative path-building strategies using the information carried during IRP deployments. The latter likely involve a higher overhead due to the non-optimality of

the provider-requester paths implicitly found joining straight lines.

Finally, *RDMAgent* is responsible for performing the client-side RDM protocols. In particular, three situations in Section 2.6 lead to the following control flow:

- A node can realize it is leaving the dense MANET if either the user invokes the `exitFromDM` method or a notification is risen by the DMC. In fact, if the number of neighbors rapidly decreses, approaching the threshold n, DMC notifies the *StatusListener*. In that case, the `notifyExiting` method calls the *ReplicaPropagationStrategy* to immediately discharge carried resources (if any).

- The *StatusListener* can also determine whether the node has left the dense MANET, i.e., if the number of neighbors falls behind the threshold; in that case, it immediately invokes the `notifyExited` method.

- The RDM protocol to recover abrupt departures is based on delegate presence confirmation: this is accomplished through periodic invocation of the `notifyIAmAlive` method.

6.1.3 Manager

Since the manager is only a support role, without any direct interaction with the local user, it does not export any interface method, and behaves essentially like a daemon. We rapidly mention that actually a non-functional interaction with the user occurs as an explicit preventive confirmation, required before its own activation. The core data structure of the manager is the *SharedResourceTable* including resource descriptors, target replication degree and the list of the replica delegates with associated expiration timers. These are loosely consistent information used to re-establish consistency in case of abrupt delegate departures.

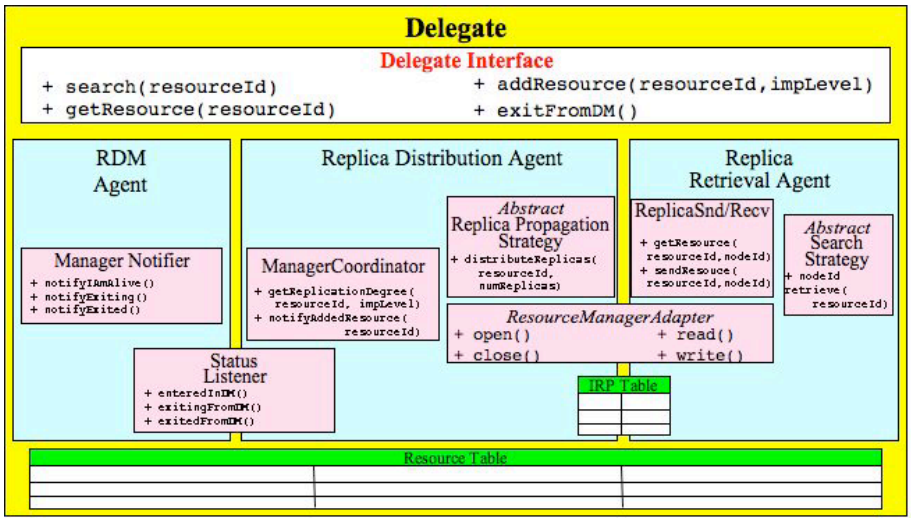


Figure 6.2: The modular architecture of the REDMAN delegate

Two main components accomplish manager functions. *RDMEngine* determines target replication degrees and implements communication protocols with delegates. *SRTMng* operates on the *SharedResourceTable*, by updating its content and periodically checking replication degree consistency. If the actual replication degree of any resource is below the target degree, then *SRTMng* invokes the `requireReplication` on the *RDMEngine*.

6.2 Implementation

According to the guidelines and key requirements stated in the introduction of this chapter we implemented REDMAN architecture on J2ME. The goal of this section is not to provide a mere description of classes and interfaces, but to discuss the main implementation issues encountered during the development. However, before probing this question it is helpful to rapidly review J2ME characteristics most relevant to the desciption (i.e., related to communication and persistent storage).

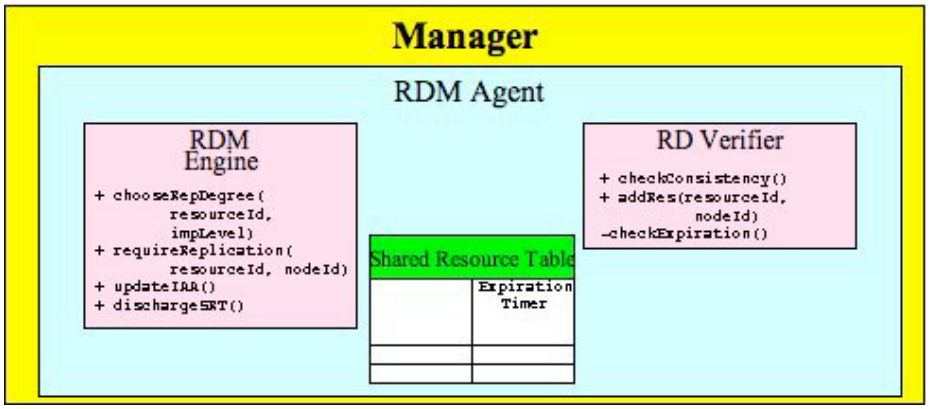


Figure 6.3: The modular architecture of the REDMAN manager

6.2.1 J2ME

To meet the constrained resources of portable devices, Sun started releasing a number of limited versions of the Java 1.1 platform, each tailoring a different group of devices with similar characteristics. J2ME was developed as a new architecture to unify these heterogeneous products. J2ME modular design permits to combine Configurations and Profiles, the former essentially defining virtual machines and basic Java classes, the latter extending Configurations with additional libraries. In this thesis work, we focus on CLDC, i.e., the Configuration with the lowest hardware/software requirements (it targets devices with as low as 160KB of non-volatile memory and 32KB of volatile), and MIDP Profile, providing in particular UDP support, persistent storage access, and a model for the development of applications (Midlet).

CLDC's Generic Connection Framework (GCF) jointly supports communication as well as I/O operations with a common set of abstractions. Essentially, GCF is an API refining a generic *Connection* into more specific abstractions: *DatagramConnection* defines packet I/O; *InputConnection*, and *OutputConnection* stream-based I/O, which is further refined in *ContentConnection*. GCF does not implement any protocol; MIDP extends GCF

by implementing TCP and UDP classes. To permit to instantiate new *UDPDatagramConnection*, users invoke the `open` method of a *Connector* factory class. This receives a string argument including used protocol, destination address, and protocol-specific parameters, and returns the connection object.

GCF abstraction can be seamlessly implemented to provide file support: *FileConnection* [69] permits to access files stored on device filesystem as well as removable memory cards. The operations should be carried exactly as for any other GCF connection. Unfortunately, *FileConnection* is available only for devices running operating systems with filesystem abstraction. If a device does not provide a filesystem, then it is still possible to persistently store resources, by exploiting MIDP's *RecordStore*. Differently from filesystem, *RecordStore* is strictly tied to the instantiating application and hardly permits resource sharing between a well-defined group of applications. *RecordStore* abstraction is by no means connected to GCF, but is separately defined [67].

6.2.2 Implementation Experience

This section reports some interesting issues we encountered and solved during REDMAN implementation. In particular, we focus on four main areas: i) packet processing; ii) communications; iii) resource packetization; iv) routing protocol interactions.

Packet Processing

REDMAN packets contain a common header including a Type field, which can vary over more than a dozen possibilities, SrcAddress/DestAddress, representing the actual sender and receiver of the message (let us recall that in multi-hop communications these can differ from the values in the IP-header), and DatagramId that, combined with the SrcAddress, provides a unique identifier for the packet. As soon as a packet is received, a common

REDMAN Dispatcher is in charge of determining how it should be managed by inspecting the Type field. Different choices are possible: the Dispatcher could sequentially elaborate the packet, or it can notify the packet to a single waiting Thread, or it can activate a brand new Thread for every received packet. Ideally, to maximally parallelize the execution, the latter solution would be the best; however, constrained devices implicitly limit the maximum number of active Threads, by degrading performance as more Threads are activated. Thus, we choose to differentiate packets requiring a complex (and generally blocking) elaboration, e.g., those delegating the execution of manager election or resource dissemination protocols, from those expecting a quick reply, e.g., shared neighbors probe in SID. The Dispatcher activates brand new Threads for the former, while it only notifies existing Threads for the latter, by placing the packet in the respective waiting queue.

Communications

Most packets are delivered with local broadcasts, e.g., hellos, neighbor probes, farthest node determination relaying. However, we found that limited broadcast (with destination "255.255.255.255") is not supported on J2ME. We collected the same experience on a number of different implementations: PalmOS and WindowsMobile versions of IBM Websphere, and Sun and IBM Wireless Toolkits. In particular, the limited broadcast destination address is not recognized as a valid argument in the *Connector*'s `open` method. This problem could be solved by replacing limited broadcast with direct broadcast (with destination "X.Y.Z.255").

Resource Packetization

Resources are locally accessed via *FileConnection* GCF APIs where a filesystem is supported, via *Record Store* elsewhere. During upload/download phases, resources need to be

carried in *Datagram* packets. Unfortunately, they often exceed datagram sizes; thus, they need to be split into a sequence of packets. REDMAN implements automatic methods to fragment resources at sender, and recompose at destination. In this case, it is important to determine the packet size allowing the best performance. We experimentally proved that, as expected, the biggest packet size supported by the communication device always leads to best performance (i.e., because this choice minimizes the communication overhead).

Routing Protocol Interactions

Even if we have not implemented any routing protocol yet, we realized that some of the operations we support would be identically repeated at the network layer. For instance, many multi-hop routing protocols exchange Hello packets to monitor local connectivity, and maintain a neighbor table. REDMAN repeats the same operations/data structures at an upper level. Interestingly, cross-layer design could avoid this unnecessary communication/memory waste, by allowing REDMAN to directly access network-layer information.

6.3 Test

We tested REDMAN prototype in small PDA and laptop network setups, on top of J2ME CLDC and MIDP provided by IBM J9 Websphere (for PDAs) and by Sun Wireless Toolkit (for laptops). We utilized different types of PDAs (Compaq, Palm, HP) with 400MHz Intel CPU, [64MB-128MB] RAM and PalmOS or Pocket PC operating systems. As for the laptops, we ran REDMAN on Dell Latitudes D600, equipped with Pentium M 1.4GHz, 512MB RAM and Windows XP operating system.

Due to the limited number of available devices, we only aimed at evaluating the basic mechanism of REDMAN protocols. In particular, we instantiated a 1 to 2-hop (i.e., 2 to 3-node) network and evaluated latency during basic DMC operations. Here we present only

Network hop diameter	1 hop	2 hops
Farthest Node Determination [ms]	214.5	238.1
Manager Election [ms]	1156.7	1881

Table 6.1: DMC protocol latency

a subset of the results we obtained (we are still working to extend our evaluation). We measured DMC farthest node identification latency in 1 and 2-hop networks. This step of the election algorithm is highly influenced by two timers: one on the flooding forwarder determining whether the node is the farthest in its direction or not ($FarthestReplyTimer$), one on the current initiator to stop the farthest identification process and proclaim its $INvalue$ ($StopFarthTimer$). Since we were interested only in the communication latencies, we set $FarthestReplyTimer = 0$ to collect replies from all the few nodes, while we did not take into account $StopFarthTimer$ at all, but we measured only the time needed to obtain all the replies. The first row of Table 6.1 shows that this value is on the order of few hundred milliseconds.

Then, we measured the election latency in the same network setup. In this case, the results are highly influenced by $StopFarthTimer$, which is set equal to $500ms$, and by *MaxConsecutiveEqualSolutions*, which is set equal to 2. We observe that the election latency is lower than a couple of seconds for 1-hop as well as 2-hop networks (second row of Table 6.1). As expected, the difference for the two cases approximates the value $StopFarthTimer$ meaning that for the second case one more election iteration is needed.

6.4 Security

REDMAN is supposed to operate in open environments and, as any other MANET middleware/application, potentially presents several security issues [123, 141, 59]. Even if not the primary focus of our research, after having demonstrated the feasibility and effectiveness of the REDMAN approach, we are now analyzing the main security weaknesses affecting our solution and investigating how to provide lightweight countermeasures against these issues.

By passing over link-layer security concerns, such as denial of service threats leveraging traffic jamming (which frequency hopping or spread spectrum techniques can eliminate), we have mainly identified two different protocol layers suffering from potential security problems. On the one hand, DMC protocols (in particular, manager election) undergo the same security issues of any MANET network-layer solution, such as in the case of routing. On the other hand, RD/RR strategies are prone to the security concerns typical of any peer-to-peer system for content distribution. In the following, we rapidly discuss the main characteristics of primary security concerns specific to dense MANET replication and identify possible guidelines of solution that are currently under investigation.

About the DMC layer, the main security issue is represented by malicious participants aiming at either becoming managers or saving their own battery/network resources. For instance, a malicious node M can force its election as manager by forging replies with a fake (higher) hop-counter during the REDMAN farthest node identification protocol. In that case, the current IN is likely to delegate the exploration to its neighbor placed in the M direction. M can repeat the same procedure until it becomes an IN, and then can falsely assume the manager role. In the case of non-colluding attackers, misbehavior detection

strategies can easily face up with this problem, as deeply investigated within ad-hoc routing research. Several solutions in the literature are based on passive acknowledgement, i.e., on the implicit control by each participant of the behavior of its successor through channel overhearing [17], even if there are well-known situations where the approach does not work properly [137]. Once identified a manager election attack, the culprit needs to be penalized to discourage further security attacks. That raises additional trust issues since it can be difficult to decide whether the accuser or the accused is the actual rogue. To this specific purpose, many solutions have been proposed [102], also based on reputation methods [137]. Selfish node behavior represents another potentially security issue for REDMAN [88]: nodes could aim at saving their battery, by refusing to forward farthest node detection packets or by forging distant node replies with a fake (lower) hop-counter, thus avoiding to be chosen as the next explored IN. Let us observe that the high node density of dense MANETs makes the problem less critical because there is sufficient redundancy in link connectivity to guarantee a correct protocol behavior anyway, at least when the number of simultaneous attackers is limited. Moreover, also in this case, overhearing techniques can help in defending from the attack. An additional approach to overcome selfish behavior is represented by incentives, sorts of virtual monetary rewards to stimulate and motivate user cooperation. For instance, intermediate nodes could be refunded for the energy dissipated in packet forwarding. [20] presents a tamper-resistant device in charge of maintaining a virtual wallet, by also preventing from selfish money forgeries. An auction-based solution aiming at establishing how much a node should pay for a MANET service is presented in [31].

About the security concerns of high-layer REDMAN protocols, RD/RR/RDM solutions are subject to well-known content distribution and peer-to-peer attacks, widely investigated

for traditional wired-network deployment scenarios [2]. With regards to resource distribution and retrieval, malicious intermediate nodes could exhibit a selfish misbehavior: they could not forward replica distribution/retrieval packets or could not store replicas when designated as resource delegates. Security solutions similar to the ones explained above can deal with the former attack. Again, let us observe that flooding-based REDMAN RR benefits from path redundancy within the dense MANET. The latter problem of "node storing quotas", i.e., the selfish behavior consisting in exploiting the storage of other participants while not lending one's own, has been addressed in many research works. For instance, [41] proposes to equip each device with an anti-tampering smartcard.

Additional problems relate to resource access control and authorization [2], by taking into account also the digital rights that replicated resources could be subject to. While generally disregarded in traditional peer-to-peer protocols, in wired networks these concerns can be solved by trusting a certification authority (CA) for key distribution [35]. In infrastructure-less MANETs, where it is unfeasible to assume CA availability, the problem is harder: recent research is proposing novel and completely decentralized key management systems for these scenarios [59, 4]. Finally, also Digital Rights Management (DRM) has undergone deep research in the last years: [84] proposes DRM architectures for infrastructure-based networks that can be easily adapted to MANET when assuming the temporary availability of wide-range connections, such as GPRS, to off-line download licenses directly from clearinghouses.

Chapter 7

MobEyes Implementation

In this chapter, we report our experience in developing and deploying a Java-based architecture for MobEyes, reflecting that described in 3.2. We recall that this is composed of three main components: The two key modules are MSI, which supports a portable and transparent access to heterogeneous sensing devices, and MDHP, which implements opportunistic summary diffusion/harvesting protocols. Both these facilities will be extensively described in the following sections. The third MobEyes component, less specific for the VSN research area, is MDP, which periodically collects sensed data from MSI, and extracts useful features, such as license plate numbers of cars in sensed video streams, through application filters. We rapidly observe that signal processing algorithms to support MobEyes target applications (e.g., accurate license plate recognition) have been recently developed and are out of the specific scope of this thesis [131, 27]. To create summaries, extracted features are then combined with relevant data read from other sensors and physically situated with corresponding timestamp and geographic location. Finally, MDP stores raw sensed data and summaries in the Raw Data Storage and Summary Database, respectively, via standard functions for persistency and database management.

Before delving into finer design details, we rapidly observe that the key requirements of

our work are *Portability* and *Openness*. Vehicles accomodate heterogeneous devices; thus, to permit the seamless integration of MobEyes on most of them, we developed our system in Java. Moreover, we considered important to interface with the highly heterogeneous world of sensing devices with standard and state-of-the-art open specifications, such as the Java Media Framework for cameras, the JSR179 Location API for possibly heterogeneous positioning systems, and the Java Communications API for interfacing with lower-layer environmental sensors. We note that with respect to REDMAN, MobEyes application scenario allows to relax some key requirements. First of all, computers installed on vehicles have likely far lower energy constraint than portable devices. Thus, for MobEyes we could exploit the full-fledged Java Standard Edition (however, let us rapidly observe that the design could be easily migrated to more limited platforms). Similar considerations allow to relax constraints on architecture modularity and energy-awareness, even if these remain second-order guidelines of our design.

7.1 Design and Implementation

7.1.1 MobEyes Sensor Interface

MSI aims to facilitate the access to possibly heterogeneous sensor devices, by providing a high-level interface that exposes generic functions. In this way, MobEyes guarantees access transparency and high adaptability to changes in the devices (and in their driver implementations) available in the deployment environment and possibly discovered at runtime. For instance, if the sequence of operations to access a camera sensor changes, developers working on top of MobEyes are completely hidden from the modification. Let us rapidly observe that this dynamicity is obtained by considering only the limited and invariant set of operations needed to MobEyes. In other words, MSI has been specifically designed

for MobEyes and, thus, does not permit general-purpose control operations and parameter settings on sensor devices.

MSI is built on top of the standard Java Virtual Machine. This choice grants wide portability to our implementation. At the same time, that design decision has facilitated our development and deployment work, by allowing us to adopt several standard Java API that provide useful contributions to supporting sensor communications, control, and management. Some API, as detailed in the following, are the result of the open Sun standardization process (Java Community Process - JCP), which warrants a widespread support for heterogeneous platforms by involving all stakeholders, from developers to companies, in the definition of novel Java specifications and extensions to increase consensus on crucial design decisions [65].

MobEyes deployment scenarios call for the support of a number of different sensing devices, depending on the kind of data that police patrol agents are going to retrieve from regular cars, e.g., audio/video streams or images of the streets, temperature, weather, and road conditions, all to be tagged with location information. We identified three primary classes of sensors, corresponding to three different standard Java API. MSI exploits the Java Media Framework (JMF) API to access the first class of sensed data, which is generated by multimedia devices such as cameras and microphones. JMF provides a widespread set of functions to perform acquisition, control, and management operations on multimedia sensors (e.g., to capture images or video streams with digital webcams, or to command/transfer the recording of audio streams with microphones) [66].

The second class includes all sensors (usually monitoring lower-layer environmental information if compared with audio/video sensors) that can be connected through an RS-232 serial interface. RS-232 can either provide access to an embedded board where sensors

report analog inputs or directly receive the output signal of a single sensor. In both cases, the values made available on the serial interface are retrieved by using the Java Communications API [64]. This standard API permits to operate with serial/parallel communication ports, by hiding the details of low-level platform-dependent drivers. The Java Communications API supports both synchronous and asynchronous (event-driven) programming models. In particular, it is possible to automatically raise/receive notifications every time a signal overcomes a specified threshold. For any different sensor type (temperature sensors, carbon-oxide detectors, ...) of interest for MobEyes, MSI currently implements an ad-hoc module for specialized serial data parsing. We are working on generalizing the parsing process so to provide a single parser module, possibly instructed by different XML-based descriptions of the data format provided by specific types of sensors. Mainly due to the proof-of-concept purpose of our current MobEyes prototype and to the non-negligible cost of pollution sensors, at the moment MSI includes only two specific parsing modules for temperature and hygrometer sensors.

Since in MobEyes any monitored data are useful only if tagged with space/time coordinates of the corresponding sensing location, the third crucial class of sensors includes positioning systems, i.e., “sensors” that can provide localization data. MSI can obtain geographic location of sensors by querying the positioning system hosted on board of the car. To interface with heterogeneous positioning solutions (satellite-based, such as GPS, but also signal-strength based, such as Ekahau [85]) in a standard way, MSI exploits the Java Location API (JSR 179) [68]. For instance, MSI invokes JSR 179 functions to select the positioning technique to use, by simply specifying the desired location accuracy and/or response time. Other JSR 179 functions are used to get position updates either synchronously or through an event-driven interface. The latter permits either to specify periodic updating

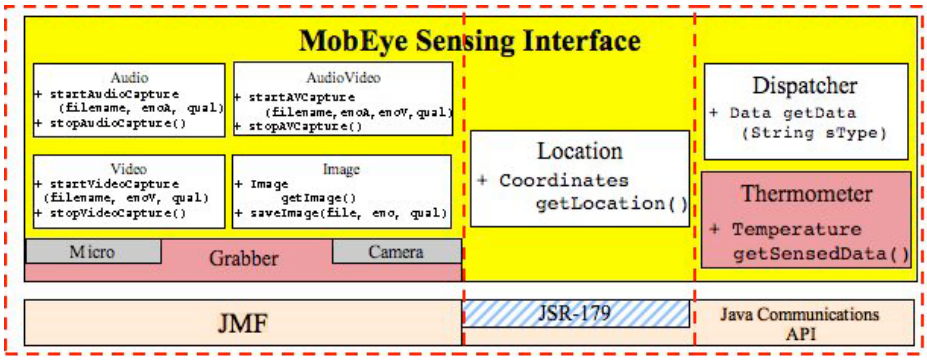


Figure 7.1: The modular architecture of MobEyes MSI

intervals or to be notified when located in proximity of a target. The exploitation of these standard API allows MSI to be independent of the implementation of the specific localization system available in the deployment environment.

In the following, the section specifically focuses on the main design choices behind our portable MSI realization, structured around the three previously sketched classes of sensors. Figure 7.1 shows the overall MSI architecture with, from left to right, the three subsystems supporting the three sensor classes.

Standard Interfacing with Audio/Video Sensors

The leftmost part of Figure 7.1 shows the audio/video sensor interfacing subsystem. This includes four components supporting access to audio, video, synchronized audio/video, and image data. A grabber module is responsible of the interaction with JMF functions. The grabber facilitates design maintenance by decoupling high-level MSI components from access procedures, which may be specific of each device. In short, the grabber is in charge of obtaining a JMF Player/Processor from an abstract input device (Microphone or Camera), and of connecting its output to a destination file where the sensed data will be stored.

MSI hides the details of actual data access operations, by exposing high-level methods

to grab the currently sensed image and to save audio, video, and audio/video streams. For image grabbing, MSI either returns an Image object or a file of the taken picture. With regards to data streams, MSI permits to command recording start/stop; the stream is initially stored in an uncompressed format (RGB video and LINEAR audio) for efficiency reasons as motivated in the experimental result section, and encoded only offline.

MSI provides two main parameters to simply control sensing processes: format and quality, which affect occupied memory, processing time, and reproduction accuracy. The suitable parameter choice depends on application-level requirements. For example, MDP may require high quality images for post-processing to extract license plate numbers. In the case of video streams finally watched by human operators, top quality is usually not needed and it is possible to configure MSI with less resource-consuming format/quality settings.

Through JMF, MSI can support many different formats, including PCM, MPEG Layer 2 and GSM for audio streams, MPEG-1, MJPEG and H.263 for video streams. The choice of MSI quality value (with a coarse granularity from 1 to 3) directly influences the adopted encoding parameters (see Section 5).

Standard Interfacing with Temperature Sensors

The rightmost part of Figure 7.1 shows the sensor interfacing subsystem. Its flexible and modular architecture is based on the Sensor abstraction representing the actual device. Sensors export a generic method, getSensedData(), which returns an object of abstract type Data. The MSI Dispatcher rules the interaction between upper layers and Sensors. The MSI Dispatcher API includes the method Data getData(String sType) that, based on the requested data type, returns a Data object with current reading. In case the sensor does not properly work, the method raises an exception.

MSI directly builds on the low-level Java Communications API to access and collect

sensed data. The Java Communications API permit both to synchronously read data from Sensors and to register listeners to be invoked every time new data are available on the communication port (serial and parallel ports). Both modes are fully supported and integrated in MSI.

Since MSI will likely need to support a growing number of sensors, extensibility is a crucial aspect for its design. To this purpose, the Dispatcher manages only Sensor and Data interfaces, without the need of any modification if a new sensor type is added. In that case, developers willing to extend the MSI prototype should only implement the new device class as a subclass of Sensor. The device-specific Sensor subclass is in charge of actually reading, parsing, and verifying the raw data present on the serial port.

Standard Interfacing with Positioning Systems

MSI permits to easily include in the set of sensed data also the geographic coordinates of sensors, in an open and standard way. Our Location module provides a simplified view of JSR 179 functions to MobEyes developers, by aggregating and composing API of the standard Java specification. In particular, the Location module can synchronously return the current ¡latitude, longitude, and (optionally) altitude¿ car coordinates. Similarly to the generic sensor case, the function either creates an object encapsulating the coordinates or raises an exception, e.g., in the case GPS is the only available positioning technique and cannot determine the position because the car is indoor in an underground car park.

To the best of our knowledge, no free implementation of JSR 179 was available for J2SE at the time of writing. Thus, two different design options were possible: either implementing the JSR 179 specifications, or interfacing the GPS as if it was a common sensing device, i.e., directly through the Java Communications API. Given the relevance of opening MSI via the extensive adoption of standard specifications, we decided to develop our partial

implementation of JSR 179. Our implementation of LocationProvider interfaces with GPS equipment via a serial port by exploiting the Java Communications API. Currently, we are working on a portable extension of our LocationProvider to support also USB-based GPS devices, by exploiting the Java USB API (JSR 80) [70].

7.1.2 MobEyes Data Harvesting Processor

MDHP manages communications for regular nodes as well as for police agents. It is in charge of extracting/storing summaries from/to the local database and of implementing opportunistic protocols for summary diffusion/harvesting. Figure 7.2 represents both regular node and police agent MDHP components; obviously, only the suitable ones will be installed on a single vehicle depending on its type. Figure 7.2. MDHP component architecture

MDHP functions can be split in two different layers. The upper layer (DB Interfacing Layer) interworks with the summary database that maintains summaries either locally generated or obtained from neighbors and not delivered to an agent yet. The lower one (Network Management Layer) consists of the components that actually implement communication protocol operations. While the DB Interfacing layer deals with instances of the Java Summary class, the Network Management layer marshals/unmarshals summaries into packets. Any summary includes a license plate number (6 bytes), additional sensed data (10 bytes, currently a 3-byte temperature/hygrometer info and a 7-byte placeholder), timestamp (2 bytes), and vehicle location (8 bytes). Thus, each 1500-byte packet can pack up to 58 summaries, without exploiting any additional aggregation or size-optimizing encoding technique.

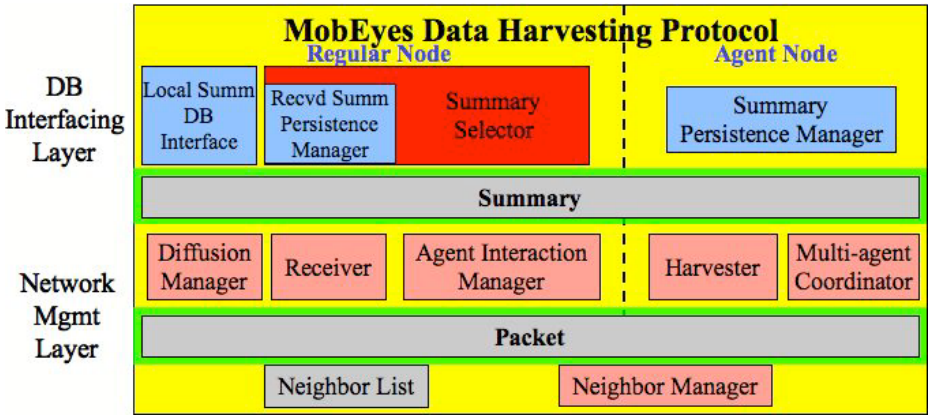


Figure 7.2: The MDHP modular architecture for regular and agent nodes

Periodically, the Diffusion Manager at regular nodes advertises recently generated summaries (one packet with the 58 last summaries provided by the Local Summary DB Interface). In the current MobEyes implementation, the diffusion period is set to 5 seconds and new summaries are expected to be generated with a maximum rate of 0.4 Hz (so, each summary is advertised at least 29 times). As future work, we are considering: i) to adapt the diffusion period to the changing rate of neighbor set (detected by Neighbor Manager); and ii) to combine summaries generated in different epochs in the same packet. The guideline is to maximize the usefulness of packets by advertising them to new neighbors expected not to have already collected the included summaries.

Any time a regular node receives new summaries, the Received Summary Persistence Manager updates the local database. Summaries are maintained until a police agent query is received: in that case, the Summary Selector component performs Bloom filter matching [43], prepares the set of summaries to send to the agent, ordered from the least to the most recent ones (in fact, due to random node trajectories, oldest summaries are likely to be the rarest in the neighborhood and consequently the higher priority ones), and then removes those summaries from the local database.

Actual communications between agents and regular cars are carried by the Harvester and the Agent Interaction Manager. The Harvester coordinates neighbor communications by exploiting unicast messages for queries. The Agent Interaction Manager, instead, handles summary delivery on regular nodes. In the current prototype, as soon as an agent encounters other agents (at 1-hop distance), the Multi-Agent Coordinator exchanges the list of harvested summaries. We are now extending the MobEyes architecture to support multi-hop inter-agent communications by exploiting IETF reactive routing protocols such as AODV [110, 105]. About low-layer communication support, we built MDHP on the standard .net package for sensing nodes equipped with J2SE. Limited nodes with J2ME will benefit from a MobEyes prototype version with a different implementation of the Packet component of the Network Management Layer, based on the standard Connection framework included in the official Sun virtual machine.

7.2 Test

We developed current MSI and MDHP implementations on top of J2SEv1.5. We adopted: 1) the official Sun release of JMF 2.1.1 with Windows Performance Package (including enhanced audio/video decoders/encoders for Microsoft platforms); 2) our own implementation of JSR 179; 3) the official Sun release of Java Communications API v2.0.

To verify the feasibility of our approach, we tested MobEyes components in real-world scenarios. Here we present some selected results, referring to the performance of the MSI audio/video capture functionality. Tests were performed by capturing streams at a trafficked intersection near the UCLA campus and were run on Dell Latitude D610 laptops, equipped with PentiumM2GHz, 512MB RAM, and Logitech Quickcam Chats. We aimed at evaluating three parameters: the size of generated files of sensed data, mainly dependent

Rec-Time	A/V RGB	A/V MJPG (Medium)	Audio LINEAR	Audio GSM
5s	4907	934	439	9
20s	16280	3295	1747	33
60s	46518	8749	5180	95

Table 7.1: Generated file size [KB]

on stream length and encoding type; the overhead time needed to capture the stream, i.e., the gross amount of time needed to start media processor and to close processor and output file; and the conversion time to encode the stream.

Tables 7.1, 7.2, and 7.3 show the results obtained while capturing audio and audio/video streams, either in raw or encoded formats. For RGB video, the frame rate was set to 4 fps and resolution to 320x240 (we are currently experimenting with higher-resolution cameras allowing 640x480 image capturing, as suggested in [27]); for LINEAR audio, sample rate was set to 44100, 16 bits per sample. Table 7.1 shows the average size of the files generated during our tests, for different recording time lengths. Both compression methods (audio/video MJPG, and audio GSM) achieve a significant file size reduction. MJPG permits to tune a "quality" parameter, influencing the conversion time, as well as the output file size. Table 7.1 shows only the results with a medium quality value: this permits to reduce the file size of about 5 times with regard to the raw version. Similar results are obtained in the case of streams including only the video track. Table 7.2 shows the overhead time needed to start the JMF processor capturing the stream and to terminate it. Results prove that audio/video capturing overhead is significantly greater than the one for a stream with only the audio track. Finally, Table 7.3 reports how long it takes to MSI to convert audio/video streams to the MJPG encoded format, depending on the chosen quality factor. Lower quality values (MobEyes quality=1, equivalent to MJPG quality=10%) impose a

Rec-Time	A/V RGB	Audio LINEAR
5s	4790	306
20s	4493	401
60s	7780	270

Table 7.2: Overhead time [ms]

Rec-Time	A/V MJPG (Low)	A/V MJPG (Medium)	A/V MJPG (High)
5s	802	1432	2099
20s	1901	2198	11354
60s	6726	6523	37429

Table 7.3: Conversion time [ms]

stable conversion time, largely compatible with typical MobEyes application requirements.

7.3 Security

MobEyes nodes continually generate and diffuse summaries containing private information, e.g., license plate numbers. Thus, privacy is of critical importance. On the one hand, non authorized nodes must not be allowed access to private information. On the other hand, the harvesting process should not reveal the information that is being sought, since this may tip the attackers or may cause unnecessary panic in the public. In general, we can summarize the security requirements of MobEyes as follows:

- *Authentication.* Harvesting agents (authority nodes) must authenticate summary senders and vice versa.

- *Non-repudiation.* A summary originator cannot deny the transmission of a summary (liability issue); in that way, upon request from the agent, the summary sender must

submit the full file with related sensed data.

- *Privacy*. Only legitimate users (authority nodes) can access summaries. Moreover, summaries must be privately advertised such that the attackers cannot track users.

- *Service Availability*. MobEyes summary diffusion/harvesting should be protected from Denial of Service (DoS) attacks.

- *Data Integrity*. MobEyes should be able to filter out false summary data injected by attackers.

- *Query Confidentiality*. In some cases, e.g., bio-attacks and search for crime suspects, even the nature of the query should not be disclosed, not to create unnecessary panic in the population or to avoid tipping the criminals.

One important aspect that sets apart MobEyes "forensic sensed data" security from conventional "safe navigation" VANET security is the "real time" and "criticality" of the safe navigation application. Consider for example a dangerous curve on the road monitored by an "e-mirror". If no car is coming, the e-mirror tells the driver to proceed at normal speed, else, it tells her to slow down. An adversary can "anticipate" the message from the mirror and tell her that the way is clear, while instead a truck is coming at high speed behind the curve. In this "safe drive" application, it is mandatory to authenticate alert messages. Thus, in safe navigation applications, message authentication is far more important than privacy. For instance, privacy concerns should not prevent a driver from alerting the vehicles behind her that there is a boulder on the road.

MobEyes has strongly different security requirements. A false report cannot create much damage since it is not acted upon immediately (for example, a wrong set of license

plates at the crime scene). There is plenty of time to detect and if necessary punish the "impostors." On the other hand, drivers that propagate summaries want to be assured that their privacy will not be violated. This major difference in security concerns leads to MobEyes security approaches that are quite different (and in fact much simpler and generally more efficient) than conventional VANET security solutions. Readers can find general security issues for VANET in [108]. For the sake of brevity, in this section we will simply outline several MobEyes security approaches, reserving the detailed, rigorous discussion of MobEyes security to future publications.

7.3.1 PKI Model

In MobEyes, we assume the existence of Public Key Infrastructure (PKI). Every node has a private/public key pair, which is issued through the Certificate Authority (CA) such as the police. Let PK_A and SK_A denote the node A's public and secret key pair. Let $\mathcal{H}(\cdot)$ denote a one-way hash function (e.g., SHA-1). For the sake of illustration, we assume that each node reads license plate numbers of nearby vehicles and prepares a summary which may contain a set of license plate numbers annotated with time and location. The summary should be encrypted by using the police public key (PK_{C_a}) so no one but the police can read it. Only the police need to authenticate the signature of the summary generator, while neighboring vehicles have no such need. Therefore, MobEyes does not need continuous access to the distributed PKI infrastructure. The digital signature and the certificate of the originator ($Cert_{C_x}$) are also encrypted in the same way. The verifier (i.e., the Police), upon decryption, uses the certificate $Cert_{C_x}$ signed by the authority where C_x is the node ID. Thus, for a given summary S_{C_x} generated at time T and position P, node C_x sends the

following advertisement to its one hop neighbors (denoted as $*$).

$$C_x \rightarrow * : M_x, T, P$$

where $M_x = \{S_{C_x}, T, P, \{\mathcal{H}(S_{C_x})\}_{SK_{C_x}}, Cert_{C_x}\}_{PK_{C_a}}$.

The parameters T = time and P = location, corresponding to summary collection, are in the clear as they are the indexes to the summary database kept by each private vehicle. They are necessary to process on demand queries by the police. Every time the originator reissues the same summary, it introduces "jitter" in T and P (say several seconds and several meters) so that the police agent can still retrieve the record (as it falls in its space time window of interest), but the eavesdropper cannot infer the presence of the same user along the path as detailed in Section 7.3.3. Then, neighbor nodes will store the message in their local database, indexed by T and P. Through Bloom filter set reconciliation, the agent will harvest the encrypted summary if it falls within the space-time window of interest. After decrypting the summary (and eliminating aliases), the agent stores it in its local database.

7.3.2 Attack Model

As just shown, standard PKI mechanisms provide authentication and non-repudiation. In this section we focus on the rest of the MobEyes requirements, namely privacy, service availability, and data consistency, by addressing the following MobEyes-specific attack models:

- *Location Tracking*. Periodic broadcasting of identical summaries could facilitate attackers in tracking the route of a vehicle.

- *Denial of Service (DoS)*. Attackers could inject a large number of bogus summaries in order to slow down correct summary harvesting by agents.

- *False Data Injection.* Attackers could inject fabricated summaries in order to mislead investigations or make the data inconsistent.

- *Query Confidentiality.* Attackers could infer "important" information from the content of police queries.

Let us finally note that in MobEyes we can exclude Sybil attacks, where a node illegitimately claims multiple identities. In fact, the certification authority issues a public/private key pair for a given vehicle (unique identifier per node). We will assume that the signature/certificate cannot be forged.

7.3.3 Location Tracking Attack

Let us consider a typical tracking scenario where the attackers can infiltrate base stations on the roadsides and listen to all the advertised summaries. In MobEyes, a node periodically advertises its encrypted summary. If the message remains the same, the attacker can infer the trajectory of the sender node. We have solved this problem in a very simple way. Each repeated summary is slightly altered by changing T and P (i.e., it is a slightly modified clone). Assuming that the traffic was moderately dense when the sample was collected (say a few cars within a 100m street segment), the attacker cannot recognize the presence of the same vehicle from the sequence of summaries. The overhead introduced by this solution is minimal. Since it is less likely that a node meets the same vehicles many times, a few vehicles will store two or more "clones." The police agent detects and discards clones upon collecting and decrypting summary messages.

7.3.4 Denial of Service

Apart from channel jamming (which can be easily detected at the physical layer and immediately stopped and punished by authorities), a serious DoS attack to MobEyes may be caused by the injection of a large number of summaries into the network, e.g., caused by sensor data interface malfunctioning. If the summaries are playback messages, they are immediately detected and dropped. If the summaries look like legitimate summaries, the attack can be handled by using *rate limited summary diffusion* which shares the same idea of RREQ rate limit in a secure routing protocol [58]. Each node keeps track of the incoming rate of summary for each MAC address. Recall that a node will periodically rotate its MAC address for privacy. However, the rate monitoring period (say, a few seconds) is much smaller than the rotation period. If the rate is above a certain threshold, intermediate nodes simply discard incoming summaries. The rate is a system parameter and is determined based on the types of sensed data. Suspicious activity is reported to authorities for further analysis. For instance, the authorities can map pseudorandom MAC addresses to vehicle numbers, thus requesting the owner to fix the device or (in case of malicious attack) by revoking key pairs and prosecuting the abuser.

7.3.5 False Data Injection

False data injection is a very serious attack in conventional sensor networks [125, 138, 142]. Fortunately, in a VSN designed for forensic investigation, this type of attack is easy to detect and neutralize thanks to the observations of other nodes. There are several possible attacks of this type. First, the attacker could aim to mislead the search for kidnapping criminals, say, and it reports that it was at the right place/time of crime and saw a set of "fabricated" license plates. The attacker (or colluding attackers) will be quickly uncovered

when the police investigate the phony license plates. A second false report attack is for the criminal to claim it was at a different place at the time of crime. Video taped records from crime witnesses will also permit to uncover the false report. A third type of attack could be the reporting of false sensed values. If the attackers collude, they may indeed create enough false reports to offset the scale. However, fluctuations in values would prompt the authorities to investigate, thus leading to the vehicle IDs of the cars injecting false reports, with their possible prosecution.

7.3.6 Query Confidentiality

Agents must retrieve information confidentially. Specifically, the agent tries to avoid a situation where criminals may take an evasive action if they realized the police are on their heels. Another possible concern is the investigation of chemicals that might be connected to a bio-attack. Since this investigation may be launched after a tip, and in most cases will turn out to be a false alarm, the public should not be told of the specific target of this investigation. Otherwise phenomenal traffic congestion may follow, with potentially very serious damage to vehicles and drivers. This requires a secure query such that the vehicles cannot tell what the agent is searching for. To this end, MobEyes exploits *private keyword searching*, proposed by Ostrovsky et al. [100].

Let us assume that the harvesting agent aims to retrieve images of some target vehicles. The agent has already harvested the summaries in the suspect area and has determined which vehicles were in the right place/time and might store the original images/licenses she is interested in. The agent cannot bluntly ask these vehicles if they have the target data. Rather, it prepares a dictionary of the license plate numbers in the nearby area (acquired from the harvested summaries). Each item is tagged $E(1)$ if interested; otherwise $E(0)$.

$E(x)$ is a homomorphic public-key encryption function which has two important properties: $i)$ $E(x)$ is probabilistic, in particular it will encrypt a single bit in many different ways, such that any instance of $E(0)$ and any instance of $E(1)$ cannot be distinguished; and $ii)$ $E(x)$ is homomorphic such that $E(x) \times E(y) = E(x + y)$.

The agent will broadcast this tagged dictionary as a query in the vicinity of each of the vehicles that hold the information. After receiving a query, a vehicle start resolving the query by processing each document in its local storage as follows. For each document D with license plate number x, compute the encrypted value g ($g = E(1)$ or $g = E(0)$) and then calculate g^D. If the agent is interested in D, g^D will result in $g^D = E(D)$; otherwise, $g^D = E(0)$. Given output g^D, the agent can find D exactly. Each vehicle has an output buffer initialized with $E(0)$. Hashing is used to find a slot to store the output. The output will be multiplied with the value in the slot, leading to the result that only interested documents will be stored in the buffer. The output buffers of all the local vehicles (neighbors of the vehicles with useful data) are later read by the agent. As a result, the police agent discovers the vehicles with useful information and instructs them to upload all of their data for a proper time-space window (not too narrow to raise suspicions, nor too large to bring in too much junk data) to the next police access point.

Conclusions

Pervasive and ubiquitous computing scenarios are calling for dynamic models, describing timely, temporary, and non-mediate interactions among autonomous nodes. The MANET paradigm imposes a communication pattern with message exchange in infrastructure-less environments. To allow remote communications between devices beyond direct radio coverage, MANET establish a co-operation principle, involving intermediate node support, in terms of message relaying. In most scenarios, MANET devices are both mobile, exacerbating multi-hop path stability issues, and energy-limited, thus requiring power conservation strategies. MANET model benefits applications in different areas, ranging from emergency relief in harsh environments, to opportunistic entertainment networking and resource exchange, to urban monitoring.

MANET features raise a number of common critical issues undermining the effectiveness of applications. To this end, it is crucial the design of middleware facilitating application development and deployment. In particular, middleware typically increases service portability over different lower-layer communication protocols. In this thesis work, we identified two primary challenges hampering effective remote resource access in MANET. First, mobile nodes can leave network area without any notice, disrupting availability of common interest resources. Second, lacking centralized server authorities, resources need to be distributedly discovered and located. This issue is aggravated in wide-scale, sparse

network, such as VANET.

This thesis work proposes original middleware solutions addressing mentioned problems. REDMAN provides the dissemination of resource replicas, to improve their availability. In particular, it proposes effective distributed protocols, to locate close resources and to maintain established replication degrees in spite of possible node mobility outside the service area. Simulative results prove that REDMAN approach is feasible and that designed protocols are lightweight, i.e., limit the number of diffused messages, and highly accurate and effective in addressing discussed goals. MobEyes specifically targets information indexing and dissemination for proactive urban monitoring on Vehicular Sensor Networks. MobEyes exploits mobility to opportunistically diffuse sensed data summaries among neighbor vehicles and to create a low-cost opportunistic index supporting queries on distributed sensed data storage. Results obtained simulating wide-scale, highly-dynamic scenarios show that MobEyes protocols are effective in supporting distributed indexing, even in case of very high sensed data generation rates.

We remark that this work has been an original investigation of the addressed themes, paving the way for promising further explorations. The encouraging results obtained via simulations and prototype experimentations disclose challenging directions for REDMAN and MobEyes research. First of all, a wider and more extensive deployment of both developed prototypes in-the-field is important to test and quantify performance. A practical and stimulating direction is in the development of applications on top of these platforms. Actually, a MIDlet exchange solution have been prototyped on REDMAN, while a target tracking solution have been simulated on MobEyes: more extensive development work is on the way. Security analyses reported in implementation chapters mainly review envisioned

attacks and general-purpose countermeasures. We will address the design of original solutions targeting specific scenarios as future investigations. From a theoretical point of view, the extension of protocol analysis is a very interesting direction. This allows to increase our insights of favorable operating conditions, and of applicability limitations of proposed middleware. Next results will address the evaluation of, e.g., how REDMAN overhead tradeoffs are affected by system parameters (e.g., dense MANET diameter, number of participants, node mobility, resource distributions/requests ratio, average replica size,...), and how multiple harvesting agents in MobEyes influence the proposed equations, depending on collaboration strategies. Finally, also replica dissemination/retrieval and information harvesting protocols will be carefully reviewed to find possible modifications leading to performance improvements. Further comparisons with feasible alternatives allow to extend protocol verification and suggest favorable integrations in REDMAN and MobEyes.

Bibliography

[1] IEEE 802.11e/D4.4, Draft Supplement to Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Medium Access Control (MAC) Enhancements for Quality of Service (QoS), June 2003.

[2] S. Adroutselli-Theotokis and D. Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, 36(4), Dec. 2004.

[3] ARGO - Global Ocean Sensor Network. `www.argo.ucsd.edu`.

[4] N. Asokan and P.Ginzboorg. Key agreement in ad hoc networks. *ACM Computer Communication Review*, 23(17), Nov. 2000.

[5] I. Aydin and C.-C. Shen. Facilitating match-making service in ad hoc and sensor networks using pseudo quorum. In *IEEE ICCCN*, Oct. 2002.

[6] F. Bai and A. Helmy. Impact of Mobility on Mobility-Assisted Information Diffusion (MAID) Protocols. Technical report, USC, July 2005.

[7] P. Bellavista, A. Corradi, and E. Magistretti. Comparing and evaluating lightweight solutions for replica dissemination and retrieval in dense manets. In *IEEE ISCC*, Jun. 2005.

[8] P. Bellavista, A. Corradi, and E. Magistretti. Lightweight autonomic dissemination of entertainment services in wide-scale wireless environments. *IEEE Communications Magazine*, 1(6), June 2005.

[9] P. Bellavista, A. Corradi, and E. Magistretti. Lightweight replication middleware for data and service components in dense manets. In *IEEE WoWMoM*, Jun. 2005.

[10] P. Bellavista, A. Corradi, and E. Magistretti. Redman: a decentralized middleware solution for cooperative replication in dense manets. In *IEEE PerWare Workshop*, Mar. 2005.

[11] P. Bellavista, A. Corradi, and E. Magistretti. Redman: an optimistic replication middleware for read-only resources in dense manets. *Elsevier Journal of Pervasive and Mobile Computing*, 1(3), Aug. 2005.

[12] C. Bettstetter, G. Resta, and P. Santi. The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad Hoc Networks. *IEEE Transactions on Mobile Computing*, 2(3), Jul.-Sept. 2003.

[13] BitTorrent. `http://bittorrent.com/`.

[14] M. Boulkenafed and V. Issarny. A middleware service for mobile ad hoc data sharing, enhancing data availability. In *4th ACM/IFIP/USENIX Middleware*, June 2003.

[15] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *ACM WSNA*, Sept. 2002.

[16] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. G. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *ACM MOBICOM*, Oct. 1998.

[17] S. Buchegger, C. Tissieres, and J.-Y. Le-Boudec. A test-bed for misbehavior detection in mobile ad-hoc networks - how much can watchdogs really do? In *IEEE WMCSA*, Dec. 2004.

[18] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine. MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks. In *IEEE INFOCOM*, Apr. 2006.

[19] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory Sensing. In *ACM WSW*, Oct.-Nov. 2006.

[20] L. Buttyan and J.-P. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *ACM/Kluwer Mobile Networks and Applications (MONET)*, 8(5), Oct. 2003.

[21] F. Cali, M. Conti, and E. Gragori. Ieee 802.11 wireless lan: Capacity analysis and protocol enhancement. In *IEEE INFOCOM*, Mar. 1998.

[22] M. Caliskan, D. Graupner, and M. Mauve. Decentralized discovery of free parking places . In *ACM VANET*, Sept. 2006.

[23] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing (WCMC)*, 2(5), 2002.

[24] G. Cao, L. Yin, and C. Das. Cooperative cache-based data access in ad hoc networks. *IEEE Computer*, 37(2), Feb. 2004.

[25] MIT's CarTel Central. `http://cartel.csail.mit.edu/`.

[26] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash. A feedback based scheme for improving tcp performance in ad hoc wireless networks. *IEEE Personal Communications*, 8(1), Jan. 2001.

[27] S.-L. Chang, L.-S. Chen, Y.-C. Chung, and S.-W. Chen. Automatic license plate recognition. *IEEE Transactions on Intelligent Transportation Systems*, 5(1), Mar. 2004.

[28] J.-H. Change and L. Tassiulas. Energy conserving routing in wireless ad-hoc networks. In *IEEE INFOCOM*, Mar. 2000.

[29] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Data gathering algorithms in sensor networks using energy metrics. *ACM/Kluwer Wireless Networks Journal*, 8(5), Sept. 2002.

[30] K. Chen and K. Nahrstedt. An integrated data lookup and replication scheme in mobile ad hoc networks. In *SPIE ITCom*, Aug. 2001.

[31] K. Chen and K. Nahrstedt. ipass: an incentive compatible auction scheme to enable packet forwarding service in manet. In *IEEE ICDCS*, Mar. 2004.

[32] Z. D. Chen, H. Kung, and D. Vlah. Ad Hoc Relay Wireless Networks over Moving Vehicles on Highways. In *ACM MOBIHOC*, Oct. 2001.

[33] I. Chlamtac, M. Conti, and J.-N. Liu. Mobile ad hoc networking: Imperatives and challenges. *Elsevier Journal on Ad Hoc Networks*, 1(3), Jul. 2003.

[34] L. Cox and B. Noble. Fast reconciliations in fluid replication. In *IEEE ICDCS*, Apr. 2001.

[35] N. Daswani, H. Garcia-Molina, and B. Yang. Open problems in data-sharing peer-to-peer systems. In *Int. Conf. on Database Theory (ICDT)*, Jan. 2003.

[36] A. Datta. Autonomous Gossiping: A Self-organizing Epidemic Algorithm for Selective Information Dissemination in Wireless Mobile Ad hoc Networks. In *ICDCS 2003 Doctoral Symposium*, May 2003.

[37] S. Decker, P. Mitra, and S. Melnik. Framework for the semantic web: an rdf tutorial. *IEEE Internet Computing*, 4(6), 2000.

[38] UMass' DieselNet. `http://prisms.cs.umass.edu/dome/`.

[39] M. D. Dikaiakos, S. Iqbal, T. Nadeem, and L. Iftode. VITP: an information transfer protocol for vehicular computing . In *ACM VANET*, Sept. 2005.

[40] L. Dlagnekov and S. Belongie. Recognizing Cars. Technical Report CS2005-0833, UCSD CSE, 2005.

[41] P. Druschel and A. Rowstron. Past: A large-scale, persistent peer-to-peer storage utility. In *IEEE HotOS-VIII*, May 2001.

[42] S. B. Eisenman, G.-S. Ahn, N. D. Lane, E. Miluzzo, R. A. Peterson, and A. T. Campbell. MetroSense Project: People-Centric Sensing at Scale. In *ACM WSW*, Oct.-Nov. 2006.

[43] L. Fan, P. Cao, and J. Almeida. Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol. In *ACM SIGCOMM*, Aug.-Sept. 1998.

[44] L. Feeney. Energy efficient communication in ad hoc wireless networks. In *S. Basagni, M. Conti, S. Giordano, I. Stojmenovic, Ad Hoc Networking, IEEE Press*, 2003.

[45] M. Freedman, E. Freudenthal, and D. Mazières. Democratizing content publication with coral. In *USENIX/ACM NSDI*, Mar. 2004.

[46] R. Friedman. Caching web services in mobile ad-hoc networks: Opportunities and challenges. In *ACM Workshop on Principles of Mobile Computing*, Oct. 2002.

[47] Z. Fu, P. Zerfos, K. Xu, H. Luo, S. Lu, L. Zhang, and M. Gerla. The impact of multihop wireless channel on tcp throughput and loss. In *IEEE INFOCOM*, Apr. 2003.

[48] C. Fullmer and J. G.-L. Aceves. Solutions to hidden terminal problems in wireless networks. In *ACM SIGCOMM*, Sept. 1997.

[49] E. Gamm, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Publisher, 1994.

[50] S. Garg, Y. Huang, C. Kintala, K. Trivedi, and S. Yajnik. Performance and reliability evaluation of passive replication schemes in application-level fault tolerance. In *IEEE ISFTC*, Jun. 1999.

[51] P. B. Gibbons, B. Karp, Y. Ke, S. Nath, and S. Seshan. IrisNet: An Architecture for a Worldwide Sensor Web. *IEEE Pervasive Computing*, 2(4), Oct.-Dec. 2003.

[52] M. Grossglauser and D. Tse. Mobility Increases the Capacity of Mobile Ad-hoc Networks. In *IEEE INFOCOM*, Apr. 2001.

[53] M. Grossglauser and M. Vetterli. Locating Nodes with EASE: Mobility Diffusion of Last Encounters in Ad Hoc Networks. In *IEEE INFOCOM*, Mar.-Apr. 2003.

[54] T. Hara. Effective replica allocation in ad hoc networks for improving data accessibility. In *IEEE INFOCOM*, Apr. 2001.

[55] W. Heinzelmann, A. Chandrakasan, and H. Balakrishnan. Eergy-efficient communication protocols for wireless microsensor networks. In *IEEE HICSS*, Jan. 2000.

[56] A. Helal, N. Desai, V. Verma, and L. Choonhwa. Konark - a service discovery and delivery protocol for ad-hoc networks. In *IEEE Wireless Communications Networks*, Mar. 2003.

[57] G. Holland and N. Vaidya. Analysis of tcp performance over mobile ad hoc networks. In *ACM MOBICOM*, Aug. 1999.

[58] Y.-C. Hu, A. Perrig, and D. B. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. In *MOBICOM*, Sept. 2002.

[59] J.-P. Hubaux, L. Buttyan, and S. Capkun. The quest for security in mobile ad hoc network. In *ACM MOBIHOC*, Oct. 2001.

[60] F. Hui and P. Mohapatra. Experimental characterization of multi-hop communications in vehicular ad hoc networks. In *ACM VANET*, Oct. 2005.

[61] B. Hull, V. Bychkovsky, K. Chen, M. Goraczko, A. Miu, E. Shih, Y. Zhang, H. Balakrishnan, and S. Madden. CarTel: A Distributed Mobile Sensor Computing System. In *ACM SenSys*, Oct.-Nov. 2006.

[62] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: a Scalable and Robust Communication Paradigm for Sensor Networks. In *ACM MOBICOM'00*, 2000.

[63] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol. In *IEEE INMIC*, Dec. 2001.

[64] Java Communication API. `http://java.sun.com/products/javacomm/`.

[65] Java Community Process. `http://jcp.org`.

[66] Java Media Framework. `http://java.sun.com/products/java-media/jmf/`.

[67] Java Record Management System. `http://developers.sun.com/techtopics/mobility/midp/articles/databaserms/`.

[68] Java Specification Requests 179. `http://jcp.org/en/jsr/detail?id=179`.

[69] Java Specification Requests 75. `http://jcp.org/en/jsr/detail?id=75`.

[70] Java Specification Requests 80. `http://jcp.org/en/jsr/detail?id=80`.

[71] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing Kluwer Academic Publisher*. Kluwer Academic Publishers, 1996.

[72] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.-S. Peh, and D. Rubenstein. Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet. In *ACM ASPLOS-X*, Oct. 2002.

[73] B. Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *ACM MOBICOM'00*, Aug. 2000.

[74] R. M. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. Randomized rumor spreading. In *IEEE Symposium on Foundations of Computer Science*, 2000.

[75] G. Korkmaz, E. Ekici, F. Ozguner, and U. Ozguner. Urban Multi-Hop Broadcast Protocols for Inter-Vehicle Communication Systems. In *ACM VANET*, Oct. 2004.

[76] U. Lee, E. Magistretti, B. Zhou, M. Gerla, P. Bellavista, and A. Corradi. Efficient Data Harvesting in Mobile Sensor Platforms. In *IEEE PerSeNS*, Mar. 2006.

[77] U. Lee, E. Magistretti, B. Zhou, M. Gerla, P. Bellavista, and A. Corradi. MobEyes: Smart Mobs for Urban Monitoring with Vehicular Sensor Networks. *IEEE Wireless Communications*, 13(5), Sept.-Oct. 2006.

[78] U. Lee, J.-S. Park, E. Amir, and M. Gerla. FleaNet: A Virtual Market Place on Vehicular Networks. In *IEEE V2VCOM*, Jul. 2006.

[79] U. Lee, J.-S. Park, E. Amir, and M. Gerla. FleaNet: A Virtual Market Place on Vehicular Networks. In *V2VCOM'06*, July 2006.

[80] J. Li, J. Jannotti, D. S. J. D. Couto, D. R. Karger, and R. Morris. A Scalable Location Service for Geographic Ad Hoc Routing. In *ACM MOBICOM*, 2000.

[81] L. Li and J. Halpern. Minimum-energy mobile wireless networks revisited. In *IEEE ICC*, Jun. 2001.

[82] Q. Li and D. Rus. Sending messages to mobile users in disconnected ad-hoc wireless networks. In *ACM MOBICOM*, Aug. 2000.

[83] S. Lindsey, C. Raghavendra, and K. Sivalingam. Data gathering algorithms in sensor networks using energy metrics. *IEEE Transactions on Parallel and Distributed Systems*, 13(9), Sept. 2002.

[84] W. Liu, R. Safavi-Naini, and P. Sheppard. Digital rights management for content distribution. In *Australasian Information Security Workshop (AISW)*, Feb. 2003.

[85] T. Manesis and N. Avouris. Survey of position location techniques in mobile systems. In *ACM MobileHCI*, Sept. 2005.

[86] A. Manjeshwar and D. P. Agrawal. Teen: A routing protocol for enhanced efficiency in wireless sensor networks. In *ACM/IEEE Workshop Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, Apr. 2001.

[87] A. Manjeshwar and D. P. Agrawal. An efficient sensor network routing protocol (apteen) with comprehensive information retrieval. In *ACM/IEEE Workshop Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, Apr. 2002.

[88] S. Marti, T. J., G. K., L. M., and Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *ACM MOBICOM*, Aug. 2000.

[89] University of Dartmouth MetroSense. `http://metrosense.cs.dartmouth.edu/`.

[90] P. Michiardi and R. Molva. Ad hoc networks security. In *S. Basagni, M. Conti, S. Giordano, I. Stojmenovic, Ad Hoc Networking, IEEE Press*, 2003.

[91] D. Nain, N. Petigara, and H. Balakrishnan. Integrated Routing and Storage for Messaging Applications in Mobile Ad Hoc Networks. In *WiOpt'03*, Mar. 2003.

[92] A. Nandan, S. Das, G. Pau, M. Gerla, and M. Y. Sanadidi. Co-operative Downloading in Vehicular Ad-Hoc Wireless Networks. In *IEEE WONS*, Jan. 2005.

[93] A. Nandan, S. Tewari, S. Das, G. Pau, M. Gerla, and L. Kleinrock. AdTorrent: Delivering Location Cognizant Advertisements to Car Networks. In *IFIP WONS*, Jan. 2006.

[94] S. Narayanaswamy, V. Kawadia, R. Sreenivas, and P.R.Kumar. Power control in ad-hoc networks: Theory, architecture, algorithm implementation of the compow protocol. In *European Wireless*, Feb. 2002.

[95] S. Nath, J. Liu, and F. Zhao. Challenges in Building a Portal for Sensors World-Wide. In *ACM WSW*, Oct.-Nov. 2006.

[96] V. Naumov, R. Baumann, and T. Gross. An Evaluation of Inter-Vehicle Ad Hoc Networks Based on Realistic Vehicular Traces. In *ACM MOBIHOC*, May 2006.

[97] S. Nesargi and R. Prakash. Manetconf: Configuration of hosts in a mobile ad hoc networks. In *IEEE INFOCOM*, Jun. 2002.

[98] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. In *ACM MOBICOM*, Aug. 1999.

[99] ns-2 (The Network Simulator). `http://www.isi.edu/nsnam/ns/`.

[100] R. Ostrovsky and W. Skeith. Private Searching on Streaming Data. In *CRYPTO*, Aug. 2005.

[101] P. Papadimitratos and Z. Haas. Secure routing for mobile ad hoc networks. In *CNDS*, Jan. 2002.

[102] K. Paul and D. Westhoff. Context aware detection of selfish nodes in dsr based ad-hoc networks. In *IEEE GLOBECOM*, Nov. 2002.

[103] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM*, 1994.

[104] C. Perkins, J. Malinen, R. Wakikawa, E. Belding-Royer, and Y. Sun. Ip address autoconfiguration for ad hoc networks. In *Internet Engineering Task Force*.

[105] C. E. Perkins and E. M. Belding-Royer. Ad-hoc On-Demand Distance Vector Routing. In *ACM WMCSA*, Feb. 1999.

[106] G. Pottie and W. Kaiser. Wireless integrated networks sensors (wins): Principles and approach. *Communication of the ACM*, 43(5), May 2000.

[107] T. Rappaport. *Wireless Communications: Principles and Practice*. IEEE Press Piscataway, NJ, USA, 1996.

[108] M. Raya and J.-P. Hubaux. The Security of Vehicular Ad Hoc Networks. In *SASN*, Nov. 2005.

[109] O. Riva and C. Borcea. The urbanet revolution: Sensor power to the people! *IEEE Pervasive Computing*, 6(2), 2007.

[110] E. M. Royer and C.-K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, 6(2), Apr. 1999.

[111] J. L. B. S. Buchegger. The effect of rumor spreading in reputation systems for mobile ad-hoc networks. In *IEEE WiOpt*, Mar. 2003.

[112] A. K. Saha and D. B. Johnson. Modeling Mobility for Vehicular Ad Hoc Networks. In *ACM VANET'04*, October 2004.

[113] F. Sailhan and V. Issarny. Energy-aware web caching for mobile terminals. In *ICDCS Workshops*, July 2002.

[114] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. B. Srivastava. Optimizing sensor networks in the energy-latency-density design space. *IEEE Transactions on Mobile Computing*, 1(1), Jan.-Mar. 2002.

[115] R. C. Shah, S. Roy, S. Jain, and W. Brunette. Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks. *Elsevier Ad Hoc Networks Journal*, 1(2-3), Sept. 2003.

[116] Sid Meier's Civilization. `http://www.civ3.com/`.

[117] S. Singh, M. Woo, and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. In *ACM MOBICOM*, 1998.

[118] A. P. Sistla, O. Wolfson, and B. Xu. Opportunistic Data Dissemination in Mobile Peer-to- Peer Networks. In *International Symposium on Spatial and Temporal Databases*, Aug. 2005.

[119] T. Small and Z. J. Haas. The Shared Wireless Infostation Model - A New Ad Hoc Networking Paradigm (or Where There is a Whale, There is a Way). In *ACM MOBIHOC*, June 2003.

[120] K. Sohrabi, J. Gao, V. Ailawadhi, G. J., and Pottie. Protocols for self-organization of a wireless sensor network. *IEEE Personal Communications*, 7(5), Oct. 2000.

[121] D. Sormani, G. Turconi, P. Costa, D. Frey, M. Migliavacca, and L. Mottola. Towards lightweight information dissemination in inter-vehicular networks . In *ACM VANET*, Sept. 2006.

[122] T. Spyropoulos, K. Psounis, and C. Raghavendra. Performance Analysis of Mobility-Assisted Routing). In *ACM MOBIHOC*, May 2006.

[123] F. Stajano, R. J., and Anderson. The resurrecting duckling: Security issues for adhoc wireless networks. In *Workshop on Security Protocols*, Apr. 1999.

[124] M. Tamori, S. Ishihara, T. Watanabe, and T. Mizuno. A replica distribution method with consideration of the positions of mobile hosts on wireless ad-hoc networks. In *IEEE ICDCS Workshops*, July 2002.

[125] S. Tanachaiwiwat and A. Helmy. Correlation Analysis for Alleviating Effects of Inserted Data in Wireless Sensor Networks. In *MobiQuitous*, Jul. 2005.

[126] A. Tanenbaum. *Computer Networks*. Prentice Hall, 2003.

[127] K. Tang, M. Gerla, and R. Bagrodia. Tcp performance in wireless multi-hop networks. In *IEEE WMCSA*, Feb. 1999.

[128] J. Tchakarov and N. Vaidya. Efficient content location in mobile ad hoc networks. In *IEEE MDM*, Jan. 2004.

[129] U.S. Census Bureau. TIGER, TIGER/Line and TIGER-Related Products. Available at. `http://www.census.gov/geo/www/tiger/`.

[130] M. Torrent-Moreno, D. Jiang, and H. Hartenstein. Broadcast reception rates and effects of priority access in 802.11-based vehicular ad-hoc networks. In *ACM VANET*, Oct. 2004.

[131] UNO-2160 in Mobile License Plate Recognition System. `http://www.advantech.com.tw/ia/newslet-ter/AutomationLink/January2005/Applica-tion_Story_UNO-2160.pdf`.

[132] CENS' Urban Sensing. `http://research.cens.ucla.edu/projects/2006/Systems/Urban_Sensing/`.

[133] A. Vahdat and D. Becker. Epidemic Routing for Partially-Connected Ad Hoc Networks. Technical Report CS-200006, Duke University, Apr. 2000.

[134] H. Wu, R. Fujimoto, R. Guensler, and M. Hunter. MDDV: a Mobility-entric Data Dissemination Algorithm for Vehicular Networks. In *ACM VANET*, Oct. 2004.

[135] Q. Xu, T. Mak, J. Ko, and R. Sengupta. Vehicle-to-vehicle safety messaging in DSRC. In *ACM VANET*, Oct. 2004.

[136] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *ACM MOBICOM*, Jul. 2001.

[137] P.-W. Yau, C. J., and Mitchell. Reputation methods for routing security for mobile ad hoc networks. In *Joint IST Workshop on Mobile Future and Symp. on Trends in Communications (SympoTIC)*, Oct. 2003.

[138] F. Ye, H. Luo, S. Lu, and L. Zhang. Statistical En-route Filtering of Injected False Data in Sensor Networks. In *INFOCOM*, Mar. 2004.

[139] J. Zhao and G. Cao. VADD: Vehicle-Assisted Data Delivery in Vehicular Ad Hoc Networks. In *IEEE INFOCOM*, Apr. 2006.

[140] B. Zhou, K. Xu, and M. Gerla. Group and Swarm Mobility Models for Ad Hoc Network Scenarios Using Virtual Tracks. In *IEEE MILCOM*, Oct.-Nov. 2004.

[141] L. Zhou and Z. Haas. Securing ad hoc networks. *IEEE Network*, 13(6), Nov.-Dec. 1999.

[142] S. Zhu, S. Setia, S. Jajodia, and P. Ning. An Interleaved Hop-by-Hop Authentication Scheme for Filtering False Data Injection in Sensor Networks. In *IEEE Symposium on Security and Privacy*, May 2004.