Sapienza University of Rome

Ph.D. program in Computer Engineering

XXIII Cycle - 2011

Information Gathering in Resource Constrained Wireless Networks

Ugo Maria Colesanti

Sapienza University of Rome

Ph.D. program in Computer Engineering

XXIII Cycle - 2011

Ugo Maria Colesanti

Information Gathering in Resource Constrained Wireless Networks

Thesis Committee

Prof. Andrea Vitaletti (Advisor)
Prof. Domenico Lembo

Reviewers

Prof. Stefan Fischer
Prof. Wendi Heinzelman

Author's address:
Ugo Maria Colesanti
Dipartimento di Informatica e Sistemistica
Sapienza Università di Roma
Via Ariosto 25, 00185 Roma, Italy
e-mail: colesanti@dis.uniroma1.it
www: http://wiserver.dis.uniroma1.it/cms/index.php?id=8

for Mauriana

Contents

Introduction

Motivations

In the near future, new types of systems will appear, designed or emerged, of massive scale, expansive and permeating their environment, of very heterogeneous nature, and operating in a constantly changing networked environment. Despite the dynamics of the environment, they will have to meet their clearly-defined objectives and provide guarantees about certain aspects of their own behavior. We expect that most such systems will have the form of a large society of tiny artifacts. Each such artifact will be unimpressive and with limited resources such as limited sensing, signal processing, and communication capabilities, and in particular, being battery-powered, it will be subject to severe energy constraints.

The challenges emerging from the management of such devices are faced by the European Project FRONTS (Foundations of Adaptive Networked Societies of Tiny Artifacts) [4]. A reference scenario considered in FRONTS, deals with a new class of devices named MURPESS - Multi Radio Pedestrian Energy Scavenging Sensor Network[49].

A MURPESS is a tiny battery-powered device equipped with heterogeneous sensors, that can scavenge energy from pedestrians' steps [103], enabling mobile users to collect data while moving around. MURPESSes are equipped with multiple radio technologies: passive ones, such as RFID tags, have less communication capabilities but do not require energy supply while active radios, e.g., 802.15.4, Bluetooth or Near Field Communication (NFC), are more powerful but also more power consuming. When the level of energy in the battery is sufficiently high, a MURPESS can exploit active radios. When the battery is low and the energy scavenging mechanism is recharging it, only passive communications are possible.

In figure 1 a possible application scenario for MURPESSes is shown. When the energy is sufficiently high, MURPESSes act as active mobile clients of a Wireless Sensor Network (WSN) generating an information flow to (request)

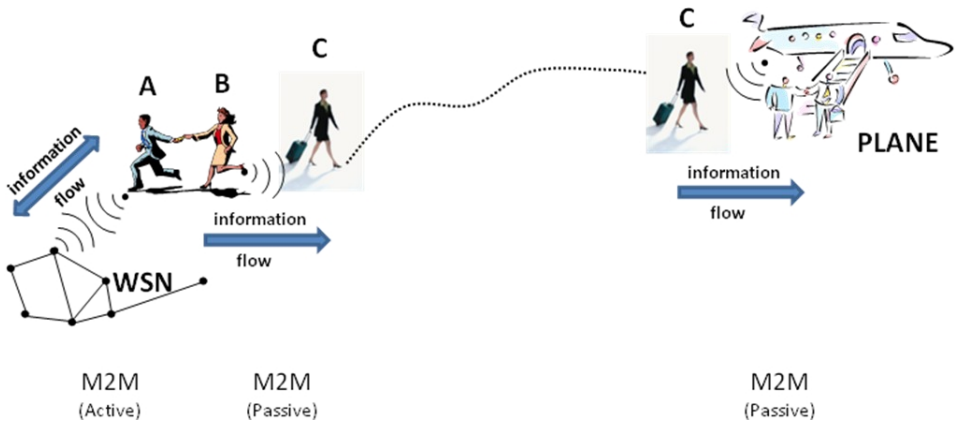


Figure 1: A unifying scenario of passive and active information gathering using low power communication techniques.

and from (answer) the network. In other words, the MURPESS consumes the information gathered by the WSN. At the same time, an active MURPESS can interact with passive ones (i.e., MURPESSes with momentary insufficient energy to start an interaction), to obtain relevant information and carry this information to other MURPESSes generating in such way an information flow in two otherwise disconnected areas. This is a typical example of *Opportunistic* and *Delay-Tolerant* networking [105].

The challenges emerging in the MURPESS scenario are faced by the FRONTS project members by proposing solutions in different research areas: self-stabilizing communication infrastructures, algorithm and tools to enforce a cooperative behavior, algorithms and protocols to efficiently share information and algorithms and protocols to protect user privacy. This thesis focuses on problems and solutions emerging from *information gathering* in resource constrained wireless networks. Considering the push (resp. pull) information exchange technique, we define information gathering as the effort made by an entity to deliver (resp. retrieve) some kind of information to (resp. from) another entity. As an example, in our context, information gathering involves the challenges the WSN must face to reliably gather information on the surrounding environment and exchange it with MURPESSes in its proximity. In passive communication, information gathering can relate on how small portions of potentially incomplete local data, opportunistically exchanged, can be used to derive useful global views on the overall system. From these examples, it is clear that the heterogeneity of the devices in resource constrained wireless networks will lead to heterogeneity in problem formulation, scope and solutions.

In our work, we have considered two main categories of resource constrained wireless networks:

- **Wireless Sensor Networks** are made of battery powered tiny devices with limited memory, computational power and communication capabilities. These devices, also known as *sensor nodes*, are equipped with heterogeneous sensors able to monitor physical phenomena of the surrounding environment. Nodes are able to communicate over long distances by dynamically building a multi-hop network topology. In such way, sensed data are forwarded to one or more collection points which make available this information to end-users. WSNs are expected to run unattended for several years without the need of battery replacement. However, because of the energy consumption of radio transceiver, information gathering cannot take place without the adoption of suitable energy-aware solutions. *Energy Awareness* in Wireless Sensor Networks is a research topic that, despite the extensive research made during the last decade, still presents several open issues.

- **Passive Pervasive Systems** are implemented by RfID and NFC technologies. Passive RfID tags are characterized by very low memory availability (8KB), the absence of a power supply and, contrary to WSN, they cannot actively establish and maintain a stable communication path. RfID tags draw power from the radio waves emitted by the readers, as a consequence, they are activated only when an RfID reader in their proximity starts a communication. The readers are active battery-powered devices able to read, elaborate and manipulate the information stored on passive RfID tags. Similarly to active readers, NFC-based devices are short-range high frequency wireless communication systems that are compatible with RfID infrastructure but are also able to exchange data with other NFC devices.

  The interaction between active readers or NFC devices and RfID tags enables information exchange between active-passive devices, but active devices allow also the mediated communication between two passive devices. In fact, an active device carried by a mobile user, can carry information taken from the first visited passive device to the next one and so on; in this context, the information exchange among passive devices is mediated by users' collective and unpredictable navigation patterns. Thus, despite the absence of direct links between passive tags, there is still an indirect multi-hop networking interaction. We refer to this mechanism as *opportunistic networking*. In such context, information gathering relies on the ability of distributing and sharing the information locally stored in passive devices through active devices, such as to

	WSN	PPS
Computational Power	Yes (4∼12Mhz)	Very limited (tags)
Memory	8∼40 KB	∼8 KB
Power Supply	Battery (2xAA-type)	No (passive)
Communication Type	Infrastructure (multi-hop)	Opportunistic (single-hop)
Autonomously initiate a communication	Yes	No

Table 1: Differences between WSNs and Passive Pervasive Systems (PPS)

obtain a common global view of the available information on top of which new services can be built.

Despite we do not cover the whole set of technologies involved in resource constrained wireless networks, WSNs and Passive Pervasive Systems are well representative of two distinct categories in terms of computational power, communication capability and energy constraints. The main differences are summarized in table 1.

Thesis Outline

Part I The first part of the thesis focuses on information gathering in Wireless Sensor Networks. Chapter I is an introduction to wireless sensor networks. Section 1.1 describes applications and scenarios of commonly known WSNs deployments. It is shown how, for most common deployments related to what we define in section 1.1 as *Periodical Environmental Monitoring* (PEM), energy constraints represents the main obstacle for information gathering in WSNs. In particular, the energy consumption of radio transceivers, required for data transmission and reception, quickly drains the batteries of sensor nodes and, thus, reduces the network lifetime to few days against a requirement of more than a year. Section 1.2 briefly describes common techniques to face energy limitations. From this section it emerges that increasing battery capacity or adopting energy scavenging techniques can mitigate the effect of power con-

sumption but it is not sufficient to meet lifetime requirements of PEM applications. Thus, energy awareness needs to be introduced on sensor nodes. Energy awareness in WSNs operates at different level. At the application level (section 1.3) *sensor selection* and *data aggregation* both try to reduce the overall quantity of data gathered by the network. Sensor selection algorithms allow selecting which nodes should participate in the sensing task to reconstruct the signal with sufficient accuracy. Data aggregation techniques aggregate or filter data sensed by nodes that are close to each other or in the same region of interest. In both cases, the semantic of the gathered data is essential for such protocols to run properly. In communication-based energy awareness (section 1.4), *energy-aware routing protocols* aim at reducing the number of hops and/or transmissions needed to propagate the sensed data from a sensing node to a collection point. On the other hand, *energy-aware MAC protocols* reduce the energy consumption of radio transceiver by leaving the radio into sleep state as soon as reception or transmission are not required.

Contributions Chapters 2, 3 and 4 present the main contributions of the first part of the thesis.

In chapter 2 we present a new Sensor Selection technique, namely Adaptive Random Selection (ARS) [113] based on *Random Sensor Selection* (RSS). The advantage of RSS approach in respect of the others described in section 1.3 relies on the minimal control overhead and the implicit load balancing. On the other hand, estimating the correct activation probability is not trivial furthermore, RSS suffers from non homogeneous network deployments. ARS randomly selects nodes, as in RSS, but uses locally computed values of the probability of activation. The rationale behind ARS draws upon the consideration that the probability of activation of a node should depend on its position with respect of its neighbors, and the number thereof. ARS is able to reduce the amount of active nodes for effective field reconstruction up to 70% in a 100 nodes environment (1-Dimensional case) still providing a root mean square error below 5%. In [50] we have shown how, reducing the number of active nodes, drastically improves the performance of underlying protocols in particular in terms of packet delivery and collection time.

These benefits directly affect the performance of *DISSense* [51] (chapter 3), an adaptive ultralow-power communication protocol for wireless sensor networks specifically designed for long-term PEM applications. DISSense implements a full communication stack allowing for both data collection and dissemination. It determines a schedule that alternates short activity phases during which dissemination, resynchronization and data gathering are executed, and long intervals during which nodes operate in an ultralow-power mode. DISSense provides an Adaptive Engine that computes, in the view of

collected statistics, the length of the active phase such as to minimize the protocol duty cycle. The advantage of DISSense in respect of energy-aware solutions described in 1.4 relies on the adaptivity that enables DISSense to reach ultra-low duty cycle that in several circumstances can compete with Dozer [47] which is, to the best of our knowledge, the most performing ultra-low data gathering protocol for environmental monitoring. However, as opposed to Dozer, DISSense is open-source and guarantees lower data latency. Additionally, the adaptive behavior of DISSense enables a WSN to be easily deployed without the need to fine tune protocol's parameters. Moreover, DISSense supports large networks deployment by adopting multi-sink gathering solutions.

In chapter 4 we applied ARS over DISSense. In particular, we observed in [50] that the lower number of active nodes, characterizing sensor selection schemes, reduces the number of collisions and retransmissions, thus, the overall gathering time of a collection protocol. This peculiarity is captured by the Adaptive Engine of DISSense which reduces the scheduled active time and consequently, the duty cycle. As a result, by applying ARS over DISSense, we have been able to greatly improve the energy efficiency of the communication protocol.

Part II The second part of the thesis focuses on information gathering in Passive Pervasive Systems. In chapter 5 we briefly describe the involved technologies and the emerging applications. We also describe the main differences in terms of network infrastructure, traffic pattern, communication type and memory availability with respect to WSNs. As a consequence of these differences, information gathering sharply differs in passive pervasive systems. In particular, information gathering cannot rely on a stable communication infrastructure, for this reason centralized solutions are not suitable. The only way nodes have to store the information is the opportunistic communication that takes place when a user interacts with a passive device while moving in the environment. At the same time, the user must compute the information locally to retrieve some useful information for the implemented application. Thus, information gathering in passive pervasive systems relates on the design and evaluation of new application scenarios that must deal with the constraints imposed by a fully decentralized approach.

Contributions Chapters 6 and 7 represent our contribution related to the second part of the thesis.

In chapter 6 we present a fully decentralized RfID-based recommendation system tailored for smart posters applications (i.e., new items recommendation based on the previously visited posters) that is able to suggest items of

potential interest on the basis of the succinct information obtained interacting with the smart poster. The system defines a probabilistic model of user behavior and uses statistics computed over past user transactions to estimate parameters of the model. The output of the model is a set of items that are most likely to meet a user's interests. The system is fully decentralized and easily matches the low-computational, low-memory and low-bandwidth requirements of passive pervasive systems.

In chapter 7 we present an SMS-based recommendation system able to provide social recommendations, e.g., new friendships, based on the exchange of succinct representation of the list of contacts dubbed *sketches* encoded in the residual space of Short Messages. Despite Short Messages rely on the mobile phone communication system, which sharply differs from passive pervasive systems, we show how the application can be easily implemented on NFC based devices.

Publications

Part of this thesis has been published in the following journal articles, conference and workshop proceedings:

Introduction

- Ioannis Chatzigiannakis, Ugo Colesanti, Spyros Kontogiannis, Guy Leshem, Alberto Marchetti-Spaccamela, Jan Mehler, Giuseppe Persiano, Paul Spirakis, and Andrea Vitaletti. Murpess - multi radio pedestrian energy scavenging sensor network. *In eChallenges2010.* Warsaw, (POLAND) 27-29 October 2010

Part I

- Ugo Colesanti, Silvia Santini, and Andrea Vitaletti. DISSense: An Adaptive Ultralow-Power Communication Protocol for Wireless Sensor Networks. ACCEPTED, to be presented at the *7th IEEE International Conference on Distributed Computing in Sensor Systems (IEEE DCOSS '11).* Barcelona (Spain), June 2011

- Daniele Puccinelli, Omprakash Gnawali, SunHee Yoon, Silvia Santini, Ugo Maria Colesanti, Silvia Giordano, and Leonidas Guibas. The Impact of Network Topology on Collection Performance. *In Proceedings of the 8th European Conference on Wireless Sensor Networks (EWSN 2011).* Bonn (Germany), February 23-25, 2011

- Ugo Colesanti, Silvia Santini. A Performance Evaluation Of The Collection Tree Protocol Based On Its Implementation For The Castalia Wireless Sensor Networks Simulator. *Technical Report n. 681*, Department of Computer Science, ETH Zurich, August 2010

- Silvia Santini, Ugo Colesanti. Adaptive Random Sensor Selection for Field Reconstruction in Wireless Sensor Networks. *6th International Workshop on Data Management for Sensor Networks (DMSN'09)*. Lyon, FRANCE, 2009

Part II

- Luca Becchetti, Ugo Maria Colesanti, Alberto Marchetti-Spaccamela and Andrea Vitaletti. Recommending Items in Pervasive Scenarios: Models and Experimental Analysis. ACCEPTED, to appear *In Journal of Knowledge And Information Systems (KAIS)*, Springer-Verlag

- E. Baglioni, L. Becchetti, L. Bergamini, U. Colesanti, L. Filipponi, A. Vitaletti, G. Persiano. A Lightweight Privacy Preserving SMS-based Recommendation System for Mobile Users. *ACM Recommender Systems (RecSys)*, 26-30 September 2010, Barcelona, SPAIN

- L. Becchetti, U. Colesanti, A. Marchetti-Spaccamela, A. Vitaletti. Fully Decentralized Recommendations in Pervasive Systems: Models and Experimental Analysis. *First International Workshop on Data Warehousing and Knowledge Discovery from Sensors and Streams (DKSS 2009)*, Marina del Rey, CA, USA, 2009

- Becchetti Luca, Colesanti Ugo, Marchetti-Spaccamela Alberto, Vitaletti Andrea. Self-Adaptive Recommendation Systems: Models and Experimental Analysis. *Second IEEE International Conference on: Self-Adaptive and Self-Organizing Systems (SASO '08)*. Venice (Italy), 2008

Part I

Information Gathering in Wireless Sensor Networks

Chapter 1

Wireless sensor networks

1.1 Applications and scenarios

Wireless Sensor Networks (WSNs) are made of battery-powered tiny devices with processing capabilities, equipped with sensors and a radio transceiver that enables communications over long distances through multi-hop traffic patterns. Nodes in WSN have the ability to coordinate themselves to reach a common goal which typically consists in retrieving some kind of information or sending commands to some actuators. WSNs are commonly used for monitoring applications. In *Area Monitoring* the network is deployed over a region and monitors different ambient values to detect possible intrusions, track hostile vehicles or detect unattended events [29, 68]. In *Environmental Monitoring*, applications are designed to periodically retrieve information from the surrounding environment (e.g., humidity, temperature or pressure) and collect them to one or more gateways for offline analysis [39, 86]. Environmental Monitoring can also integrate event-based detection that generates alarms when, for example, a given threshold is met. Event-based detection is useful for critical scenarios like fire-prevention or flood-detection [131]. In *Machine Health Monitoring*, the wireless network is deployed over an industrial area so as to monitor industrial machineries. By gathering mechanical parameters, the WSN is able to predict possible malfunctioning [75].

In the following of the thesis we will focus our attention on *Periodical Environmental Monitoring* (PEM) applications, i.e., Environmental Monitoring without event-detection support. PEM represents the most popular research area with several testbeds already deployed. In recent deployments like [38], two networks of 25 nodes each, deployed over the Jungfraujoch and Matternhorn mountains in the Alps, are able to periodically measure temperature,

electric conductivity, crack motion, ice stress and water pressure. The testbed is able to adjust the sampling interval from 1 to 60 minutes still guaranteeing over 99% of delivery rate and more than 3 years of network lifetime. In [36], 16 nodes were deployed over Le Génépi rock glacier in Switzerland. The network was able to retrieve, each 2 minutes, values related to air temperature and humidity, surface temperature, incoming solar radiation, wind speed and direction, precipitation, soil water content, and soil water suction. The network has been tested for 2 months but, thanks to the low power consumption and the built-in solar panel, several years of network lifetime is expected. In [48], 17 nodes monitoring a medieval tower in Trento (Italy) were periodically retrieving, each 10 minutes, temperature, relative humidity, light and deformation. Additionally, the testbed was able to support high frequency measurements related to tower vibrations and download them at the end of each sampling session with bursty downloads. The testbed which has been active for 4 months, reached over 99.9% of delivery rate and a year of expected lifetime.

Based on the already described scenarios and on testbeds deployed in the past [45, 86, 121], we can assess that in Periodical Environmental Monitoring the sampling period ranges from few minutes to some hours and the per-node amount of data generated during each sampling period is low (typically fits into a packet except for some special high frequency measurements [48]). Additionally, a high delivery rate of the sensed data is required (more than 95%, but recent testbeds reach 99%), while latency requirements may vary depending on the scenario: for statistical analysis, a per-day latency is tolerated, on the contrary, in other scenarios such as vineyard monitoring [45], the latency must be sufficiently low (few seconds) to enable possible human intervention in the system. As for most of sensor networks scenarios, battery lifetime is expected to exceed the year. In table 1.1 we report the PEM requirements already described.

It is important to note that unlike common WSN surveys where network size can reach thousands or millions of nodes [26, 130], practical nowadays PEM applications refer to network size ranging between ten and hundred of nodes.

1.2 Power management in WSN

In Periodical Environmental Monitoring, tens or hundreds of nodes may be deployed over a large area making battery replacement a cumbersome and time consuming task. Despite the distributed nature of wireless sensor networks and their scalability, node malfunctioning related to battery exhaustion may lead to connectivity problems such as network partitioning, i.e., network split

Network Size (nodes)	10 - 100
Sampling Period (minutes)	1 - 60
Expected Lifetime (years)	1 - 5
Data Delivery Rate (%)	> 95
Latency (time)	1s - 1day
Data Traffic (pkt/node/sampling period)	1

Table 1.1: Periodical Environmental Monitoring requirements

in two or more sets without connectivity between them. Minimizing energy consumption to extend network lifetime is thus a primary requirement entering the design of WSNs protocols and applications.

Battery capacity Nodes in a WSN are commonly powered by a pair of AA-type batteries. At full load (i.e., active radio and CPU), the power drain of most common nodes in a WSN is 21.8mA [107]. Assuming 2000mAh AA-type alkaline batteries, this translates in 4-5 days autonomy before reaching node's cut-off voltage (1.8 Volts). Increasing the battery size or the battery number can bring the overall capacity up to 20-30Ah which corresponds to 50-70 days of nodes lifetime. Thus, the lifetime remains far below PEM requirements while the cost, the size and the weight of such solutions increase. Additionally, as figure 1.1 shows, despite the improvements, over the last decade, of battery capacity, the relative improvement is far below those related to CPU, wireless communication and storage capacity.

Energy scavenging Energy scavenging techniques consist in recharging the batteries by converting power from ambient sources. Several sources may be available in the environment where the WSN is deployed: photovoltaic cells can capture solar energy in indoor/outdoor deployments while piezoelectrics can retrieve energy from vibrations or thermoelectric generators from thermal sources. However, different obstacles need to be faced when using such techniques. The main one is that energy sources are not always available or may not be distributed homogeneously. As an example, in a deployment of sensor nodes with built-in photovoltaic cells, the nodes deployed in a shadowed area will die much quicker than the others, thus, requiring specific design choice before deployment. Moreover, the energy retrieved from energy scavenging techniques is typically low and not sufficient to fully power a sensor node. Last but not least, the adoption of energy scavenging techniques increases the

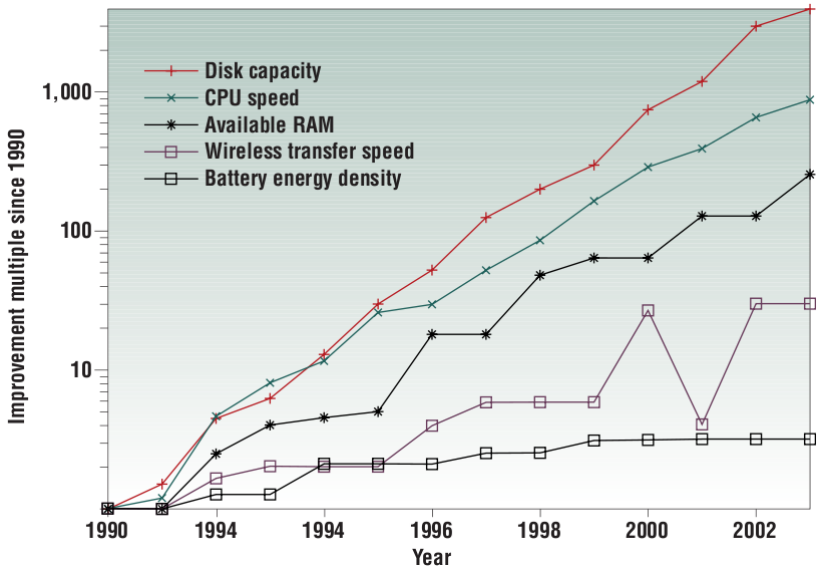


Figure 1.1: Relative improvements of different technologies (source [104])

overall cost of hardware nodes. As for battery capacity, energy scavenging techniques are not sufficient to meet PEM requirements.

Energy awareness The most effective solution to meet PEM requirements consists in adopting energy-aware software stack on sensor nodes. In particular, since the operation of the wireless transceiver causes high energy spending on sensor nodes [107], energy saving techniques for WSNs typically aim at optimizing the communication stack. Energy awareness in WSN is able to lower the current drain of sensor nodes below the 1% in respect of full powered nodes enabling a WSN to reach several years lifetime. Energy awareness affects different layers in the communication stack. In the remainder of this chapter, we will present the different existing techniques at application, routing and mac layers.

1.3 Application-based energy awareness

As depicted in the previous paragraph, typical application scenarios for WSNs envision a large number of sensor nodes being distributed at various locations over a region of interest to capture data about some physical quantity, like temperature, atmospheric pressure or a pollutant concentration [34, 46, 117, 118]. Sensor readings are then further processed locally and/or reported to a central

server to comply with specific application goals, like reconstructing the temporal and spatial developing of the observed physical quantities. To report their readings to one or more data collectors, sensor nodes communicate through their integrated radio-transceivers and collaboratively build an ad-hoc, possibly multi-hop relay network. The traffic generated to deliver the sensed data to the collectors increments collision probability and retransmissions, thus, it increases transceivers usage.

In application-based energy-aware protocols the semantic of the information encoded in data packets is used to determine the relevance of each packet for the accomplishment of the application's goal. The rationale is that in several circumstances, a subset of packets is sufficient to meet application's requirements, thus, the overall quantity of data packets can be reduced so as to decrease the average usage of the transceivers. In this chapter we focus on the two main application-based techniques: *Sensor Selection* and *Data Aggregation*.

1.3.1 Sensor selection

Sensor selection tries to limit communication among nodes so as to increase the overall reliability and data throughput of the network and consequently, reduce the overall energy consumption. To this aim, sensor selection techniques check whether all nodes should actually participate in a sensing task or not. In target detection and tracking application, for instance, one could select a subset of the nodes to guarantee spatial coverage and put the rest to sleep so as to save energy. Upon detection of a target, however, activating the sleeping nodes may guarantee better tracking performances, although at the cost of increased overall energy consumption. By trading off energy consumption with data granularity, sensor selection algorithms allow to optimize resource usage within a WSN and, consequently, to improve its lifetime and reliability. Sensor selection is usually performed at the application layer, since the question about which nodes should actively sense and report their observations clearly depends on the specific application requirements in terms of data granularity.

A particularly challenging scenario for sensor selection algorithms is that of typical long-term periodical environmental monitoring applications, in which the network's ultimate goal is the reconstruction, at a central server, of the temporal and spatial developing of a specific physical phenomenon. We refer to this issue as the *field reconstruction* problem in WSNs. To ensure a reliable reconstruction, a sufficiently large number of nodes must sample this physical quantity at sufficiently close time intervals. In other words, the spatial and temporal sampling rates of the network must be sufficiently high and

determining these values may be seen as an instance of the sensor selection problem. Indeed, the default values of these rates are often fixed a priori on the basis of conservative estimations of the data accuracy requirements of the application and compatibly with other constraints, like the total number of available nodes and the extension of the region to cover. However, the values of the sampling rates actually necessary to comply with the application quality requirements may change over time and across different sectors of the network, since they depend on the actual dynamics of the observed signal and even on the physical topology of the network. In particular, it may be possible to, at least temporarily, reduce these rates without affecting the overall data quality. Clearly, reducing the spatial sampling rate may allow for energy savings, since only a subset of the nodes will be active in each sampling interval. Similarly, reducing the temporal sampling rate preserves resources since the nodes are required to sense and communicate data less frequently. We refer to *spatial* and *temporal* sensor selection strategies to mean algorithms operating on either the spatial or the temporal sampling rate of the network.

Conch [115] uses a combination of both, spatial and temporal sensor selection named *Approximation Caching*. In [115], each node stores the sensed value as the difference between old and new measurements. During the sensing task, when the new value differs from the previous one, an *update* is broadcasted to the neighbors. Upon the reception of an *update*, the node stores the incoming value as the difference with respect to the local value. Thus, at each node, for each incoming edge (i.e., update from a neighbor), the difference between the local value and the incoming one is kept in memory. Conch sends all these values to the sink through multiple previously build spanning trees. The relationship between a monitored node (i.e., a node belonging to the spanning tree) and its edges enables the sink to reconstruct the whole sensing field.

Region Sampling [81] is a budget-based sensor selection protocol. Given a budget (i.e., power consumption), Region Sampling tries to minimize the approximation error of the reconstructed sensed field. The budget design of Region Sampling enables the protocol to keep an upper-bound on the power consumption of the WSN. Region Sampling runs an approximation algorithm that creates k clusters inside which a *region head* is responsible to select the nodes participating to the sampling phase. The sampled values are collected by each region head which computes the Cross Validation Error (CVE) for its region and forwards both, the number of sampling nodes and the CVE to the sink. Based on those collected statistics, the sink computes the first sampling plan that minimizes the approximation error within budget requirements. After the first sampling plan, an online approximation algorithm adjusts the plan at each round based on the newly collected statistics.

Event Contour[92] is a data-collection scheme for event monitoring. It is based on Contour maps construction. A contour map is composed of several

lines, where each line in a map connects points of equal value while neighboring lines have values that differ by a pre-determined threshold. More contour lines indicate fine-grained data, which comes at cost of collecting more information. By adjusting the step-values, the trade-off between information and cost of obtaining it can be tuned to suit situational requirements. For the construction of a contour map, Event Contour uses spatial and temporal suppression locally and along the routing path and implements a reconstruction algorithm at the sink using interpolation and smoothing of the received data. The algorithm makes no assumptions on the density of the network neither on the mac and routing protocol used, but on the other side, it requires that the knowledge of the sensor locations and the topology are available at the sink.

Backcasting [126] provides an initial estimate of the field being sensed by reporting only a subset of measurements to a fusion center. Based on the initial estimate, the fusion center asks additional nodes to report their measurements in order to reach a predefined error level. The key idea is that the preview step can detect correlations between data indicating that other sensors can avoid transmitting their one. The overall process is called backcasting to emphasize the role of the feedback of the preview step. The adaptive mechanism is based on the hierarchical field estimation: the sensor measurements can be viewed as sampling the field over a partition of n nested sub-squares of side length $1/\sqrt{n}$.

In [106], a spatially band-limited signal is reconstructed by a randomly deployed WSN using a blue noise sampling pattern. The advantage of a blue noise pattern, with respect to a common white noise sampling strategy, is that it achieves a higher accuracy with the same number of sampling points. The authors provide a distributed algorithm to generate blue noise sampling patterns. Each node, in the WSN, sets a backoff timer based on the node density and the filter used to create the blue noise sampling pattern. A shorter backoff will correspond to a higher deactivation priority. In fact, when the timer expires, the node broadcasts a deactivation beacon. Upon the reception of a deactivation beacon, a node updates its backoff timer. During a sampling round, those sensors whose timer exceeds a predefined threshold, will participate to the sensing task.

Two coverage preserving algorithms are presented in [125] and [129]. This family of algorithms is mostly used for tracking applications, but in [112] it is shown how, under some specific assumptions, field reconstruction can be reduced to a coverage problem.

In [125], each node running the Configurable Coverage Protocol (CCP) basically remains in sleep state. Each node periodically wakes up and checks for broadcast messages (*Hello*, *Withdraw* or *Join* messages). If one of those messages is received, the node checks its eligibility based on the covered area of its neighbors. If the node is eligible, it starts a *Join* timer, otherwise it

goes back to sleep. If the *Join* timer expires and the node is still eligible, it broadcasts a *Join* message and turns active. An active node periodically checks its eligibility such that, if it becomes ineligible, it can withdraw from the set of active nodes and turn back to sleep state.

In [129] the PEAS algorithm "selects only a necessary set of sensors in working mode. Sleeping nodes wake up once in a while to probe their neighborhood and replace any failed working nodes as needed". The probing frequency is self-adjusted using an adaptive sleeping algorithm in order to meet a good trade-off between energy consumption and working node density. Additionally, the probe is sent using a *Probe* broadcast message within its local probing range R_p. This feature is enabled assuming nodes with isotropic antennas and selectable power transmission. Upon the reception of a *Probe* message, any working node sends back a *Reply* message. If a probing node receives a *Reply* message, it goes back to sleep and adapts its probing rate.

Costs of sensor selection. The main drawback of the previously described sensor selection techniques is represented by the *control overhead* required to run those solutions. In particular, [115] and [81] require the construction and maintenance of complex routing topologies (i.e., multiple spanning trees for [115] and a clustered topology for [81]). In [92] the whole topology needs to be known by the collection point thus, topology information is periodically forwarded somehow. In [126] several preview steps (i.e., control traffic) need to be run before reaching the required quality while in [106], [125] and [129] local broadcast messages are periodically used to notify the neighborhood (i.e., deactivation beacons for [106], *Hello*, *Join* and *Withdraw* messages for [125] and *Probe* messages for [129]).

It is clear that the advantages of sensor selection approaches are mitigated by the cost of the control overhead required to run those solutions. To overcome these limitations, in chapter 2 we will introduce random-based sensor selection techniques and present an evolution of such kind of solutions.

1.3.2 Data aggregation

Similarly to sensor selection, data aggregation aim at reducing the overall network traffic so as to reduce network collisions and retransmissions affecting the overall power consumption of the network. To reach its goal, in-network data aggregation evaluates, at each intermediate node, whether the information coming from different sensors can be merged together before being forwarded to the sink. In [61] two different aggregation methodologies are defined:

- **In-network aggregation with size reduction** combines and compresses data coming from different sources and, as a result, sends a single

packet whose content is a function of the merged values. Functions as the minimum, maximum or average are commonly used in such context where sampled values are typically represented by temperature, humidity or other common physical values. Similarly, distinct counting functions can be used to recognize the same event sensed by two nearby nodes and consequently, activating packet suppression for duplicated events.

- **In-network aggregation without size reduction** is a technique that merges together different packets from different sources without data processing. In such context, two or more sampled values are concatenated inside a single packet payload. The longer packet size is compensated by the control overhead that is spread over two or more sensed values. Moreover, the lower number of packets reduces channel contention.

As opposed to in-network aggregation with size reduction, data concatenation used in the latter technique does not require data homogeneity. Actually, the semantic of transmitted data is useless for the in-network aggregation without size reduction. This latter technique, which is more related to routing layer rather than application layer, is commonly used in most of the routing protocols presented in section 1.4.1 and will not be further discussed. Rather, in this section we describe the Tiny AGgregation protocol (TAG) [85] which uses in-network aggregation with size reduction and is specifically designed for PEM applications. TAG builds an aggregation tree by periodically broadcasting beacon messages. In a first phase, named distribution phase, TAG disseminates a query where the selection and aggregation follows the SQL query language. During the collection phase, the data is transmitted to the sink following the aggregation tree. Data aggregation is performed at each intermediate nodes (for this reason each parent needs to wait for all its children before sending its own packet) and an extended set of functions is supported in addition to common aggregation functions of the SQL language (count,min,max,sum and average).

1.4 Communication-based solutions

1.4.1 Energy-aware routing protocols

Possible power failures and the well-known wireless channel instability [116] can cause significant topological changes in a multi-hop WSN. The high dynamics in wireless networks may require rerouting or retransmission of packets and reorganization of the network. Energy-aware Routing protocols in WSNs

for Periodical Environmental Monitoring aim at minimizing the overall quantity of (re)transmission required by a network to gather the sensed data from the nodes. Different approaches have lead to a classification of routing protocols defined in [25] and [27] that differentiates between *flat-based*, *hierarchical-based* and *location-based* routing protocols. Flat-based routing, also known as *data-centric* routing, assumes each node to play the same role in the network. There is no knowledge concerning the addressing of nodes rather, an attribute-based naming that specifies the property of the data is used. In general, in flat-based routing the sink requests data through the dissemination of a query and waits for nodes, having data matching the attributes of the query, to reply. In hierarchical routing clusters are formed so that only one node per cluster, the cluster-head, is responsible to forward the data to the sink. By this way, intra-cluster computation can reduce the overall quantity of data that need to be forwarded to the sink, thus, increasing the scalability and throughput of the network. Location-based routing assumes nodes being aware of their geographical position or, as an alternative, having an estimate of their relative distance. The absolute positioning enables a sink to efficiently retrieve information from a selected area of the network, while the relative distance allows a node to estimate the energy cost of a multi-hop packet transmission.

Directed diffusion (DD) [72] is a data-centric routing protocol that saves energy through data negotiation and elimination of redundant data. In DD, all the data is represented by attribute-value couples. An interest is generated by the sink and is propagated through the network. Upon the reception of an interest, gradients are setup such that data satisfying the query can be conveyed toward the requesting node. In fact, when interests fit gradients, paths of information flow are formed from multiple paths and best paths are reinforced so as to prevent further flooding. The interest dissemination and gradients generation enables the construction of a good aggregation tree in order to perform data aggregation on the way.

CTP [64] is a flat-based collection protocol for WSNs. Differently from more traditional data-centric approaches, CTP is not query initiated rather, beacon messages are used to build and maintain a routing tree, and *data* messages to report application data to the sink. The tree construction is based on a gradient dubbed Expected Transmissions (ETX) which estimates the average number of (re)transmissions required by a node to successfully transmit a packet to its neighbor. Based on this metric, the tree is build by choosing the minimum node-to-sink aggregated ETX for each node of the network. The standard implementation of CTP consists of three main logical software components: the *Link Estimator*, the *Routing Engine* and the *Forwarding Engine*. The Routing Engine takes care of sending and receiving beacons as well as creating and updating the routing table. The Forwarding Engine forwards data frames. Each transmitted data frame is acked at the link layer,

enhancing the reliability of the protocol. Furthermore, the FE implements a duplicate-detection mechanism and it has the ability to detect and repair routing loops. The Link Estimator is mainly responsible for computing the ETX by determining the inbound and outbound quality of a communication link.

The LEACH protocol [69] is a hierarchical-based protocol that randomly selects few sensor nodes as cluster-heads (CH) and rotate this role to distribute the energy load among the sensors in the network. Intra-cluster computation is performed by cluster-heads to suppress duplicates and compress data that need to be sent to the sink. After a given time interval, a randomized rotation of the CH is performed to load balance the power consumption of each node within the cluster. LEACH is organized in rounds. Each round is made of a setup phase so as to build clusters, and a steady-state phase for data collection. The round-based design of LEACH makes it suitable for Periodical Environment Monitoring applications. However, LEACH assumes 1-hop distance from CH to sink and for intra-cluster communications. This assumption represents a limit for large sensor networks deployments. Additionally, LEACH requires control overhead for cluster construction and maintenance that limits the power savings of this approach. LEACH represents one of the first hierarchical protocols for WSNs and has inspired several other solutions [83, 84, 88, 89].

Geographic Adaptive Fidelity (GAF) [128] is a location-based routing protocol that forms a virtual grid where nodes inside each zone collaborate with each other to efficiently perform the sensing task. Each node uses its GPS device to associate itself with a point in the virtual grid. In each zone of the grid the energy efficiency is performed, as an example, by demanding the sensing task to one node at time following a round-robin policy that enables the remaining nodes to sleep, thus, saving their energy.

Recent developments, related to the creation of suitable routing protocols for the *Internet of Things* (i.e., Internet communication of everyday objects), have led to the introduction of *address-based* routing protocols for resource constrained wireless devices. Along these lines, the IPv6 stack implementation for WSNs, namely μIPv6, has been introduced. The Routing Protocol for Low power and Lossy Networks (RPL) [10] is the reference protocol for IPv6 routing on WSNs. RPL builds Destination-Oriented Direct Acyclic Graphs (DODAG), i.e., Direct Acyclic Graphs rooted at one single node, based on the observation that most of the traffic flows through few nodes. RPL builds several DODAGs optimized on different criterion (e.g., latency, reliability, energy). For graphs construction and maintenance, RPL uses different types of ICMPv6 messages. Despite the promising outlook, the RPL protocol is still at an Internet draft stage and will not be discussed in the following of the thesis.

1.4.2 Energy-aware MAC protocols

There exist a plethora of energy-aware MAC protocols for WSNs. In [77], these are classified in *slotted*, *random*, *frame-based*, and *hybrid* protocols. Slotted protocols make the nodes share a common schedule that alternates sleep and active phases. The length of active slots typically ranges between tens to hundreds of milliseconds, while sleep slots last significantly longer resulting in low duty cycles. Within each active slot CSMA-based techniques are used to manage channel contention. Random access protocols avoid the use of a shared schedule. Instead, they demand most of the communication effort to the transmitter, which must inform the receiver if a transmission will take place. Frame-based protocols group slots into frames and assign one or more slots to each node. In this way, nodes can avoid collisions and channel contention. Keeping this schedule requires tight synchronization between nodes and induces a large memory footprint, making frame-based protocols hard to use in practice. Hybrid protocols aim at combining the advantages of both random and frame-based protocols. With respect to their random counterparts, hybrid protocols limit collisions but cause more control overhead.

T-MAC [122] is a slotted protocol that uses a CSMA/CA MAC with RTS/CTS mechanism for packet transmission. T-MAC achieves energy efficiency by keeping the active phase as short as possible with respect to the sleep period. If no traffic is sensed after a pre-specified timeout, a node can switch to sleep mode until the next active period will start. T-MAC sets the default length of the timeout to 15ms for a period of 610ms. Thus, the duty cycle of the radio is 2.4%, which is considerably high for PEM scenarios. Increasing the length of the sleep period introduces synchronization issues and high latencies that may hamper the correct operation of routing protocols.

BoX-MAC-2 [97] is a random protocol that is part of the standard low-power MAC of the TinyOS operating system [12]. Each node running BoX-MAC-2 wakes up periodically from the sleep mode and checks for channel activity. This mechanism, called *Low Power Listening* (LPL), enables a transmitter to communicate with a receiver by continuously transmitting data packets during an *activity* period. As soon as all transmissions are acknowledged, nodes go back to sleep mode. BoX-MAC-2 moves energy consumption for communication from receivers to transmitters and works well on 1-hop scenarios with low traffic loads. However, in multi-hop scenarios that require a routing protocol, overhearing and collisions as well as the transmission overhead of broadcast messages such as routing beacons, significantly degrade the energy efficiency of LPL-based protocols [77]. As an example, in [65] BoX-MAC-2 and CTP have been tested on a 100 nodes network with 5 minutes sampling period: despite the 1.08% duty cycle set for BoX-MAC-2 (1s sleep interval), the overall duty cycle related to the multi-hop routing activity increases as

high as 3.8%.

WiseMac [60] is a random protocol that uses LPL but, unlike BoX-MAC-2, it maintains a neighbor table with poll schedules, updated each time a packet is received, which allows a node to send short preambles only. The preamble length also takes into account the maximum clock drift from the last message exchange and, if no poll schedule is available, it simply falls back to long LPL preambles. Despite in [77] WiseMac is considered the most performing MAC protocol for low data rate applications, it is also mentioned that, similarly to Box-MAC-2, its performance quickly degrades when broadcast communication pattern is required.

Z-MAC [109] is a hybrid approach that works as a contention-based protocol for low traffic levels, but it turns into TDMA mode for high levels. Z-MAC uses global time synchronization once during setup. Subsequently, only local synchronization between sender and receiver is required. Despite the hybrid design, Z-MAC becomes energy efficient for high traffic load only (more than 3 packets per second per node), but it is still far from reaching the energy efficiency required by PEM applications.

1.4.3 Cross-layer energy-aware protocols

Designing a sensor network application that meets PEM requirements, based on the matching of a routing and MAC protocol presented in the previous sections, is not unproblematic. As observed in 1.4.2, several MAC protocols (i.e., LPL-based [60, 97]) quickly degrade their performance when transmitting broadcast messages. Other protocols [109, 122], being general-purpose, are not designed for ultra-low duty cycles (e.g., below 1%), thus, cannot reach the network lifetime required for PEM applications. To overcome these limitations, a new family of cross-layer communication protocols specifically designed for Periodical Environmental Monitoring applications has been recently introduced.

Koala [98] implements an efficient asynchronous wake up strategy and on-the-fly route computation whenever data download is requested by the sink. The energy saved avoiding the control overhead during inactivity periods compensates the higher cost of wake up and route construction. The sampled data is logged on the flash memory of each node and can be sent when requested. To ensure energy efficiency, Koala needs to log a significant amount of data before initiating the wake up strategy. However, the low data rate of PEM applications implies that several days are required to log a sufficiently large volume of data such that the energy efficiency of Koala becomes reasonably high.

Dozer [47] is a data-gathering protocol designed for Environmental Moni-

toring. It integrates MAC layer, topology control and routing to reduce energy wastage of the communication subsystem. The data gathering in Dozer relies on a tree-based network, while the data exchange is enabled by a TDMA protocol. To avoid global synchronization, each node has two independent TDMA schedules, one for its parent role and one for its child role. Link-layer acknowledgement is enabled for each packet transmission to enhance the protocol reliability. Dozer provides mechanisms for load balancing, parent selection and hidden-node collision avoidance. With its *lazy* TDMA approach, Dozer is able to reach an ultralow-power consumption, which increases network lifetime up to 8-10 years. To the best of our knowledge, Dozer is the most performing ultralow-power communication protocol available nowadays. However, Dozer is a commercial closed-source protocol available only for the TinyNode platform [11]. Moreover, it cannot provide any guarantees on data latency and requires a fine tuning of its parameters during setup.

Chapter 2

Adaptive random selection (ARS)

2.1 Introduction

One of the simplest ways to perform sensor selection is to make the nodes randomly decide upon their participation in the sensing task. In particular, each node can be assigned a *probability of activation* p, which can be computed locally or be disseminated by a central server. Each time data collection is required, sensor nodes can autonomously decide whether to participate in the sensing task or not by generating a random number r between 0 and 1. If $r < p$, the node collects and reports data, and it remains idle otherwise. This simple random sensor selection (RSS) protocol clearly requires very little control overhead. Indeed, once the value p has been fixed and disseminated to all the nodes, the protocol must intervene only to update possibly newly added nodes or to opportunely adapt the value of p. Its efficacy is indirectly demonstrated by the results reported in [81], which show that the RSS can perform comparably, or even far better, than other more elaborated (and costly) sensor selection protocols. As we will detail in section 2.2, however, the RSS also suffers some major drawbacks that may hamper its ability to continuously provide a reliable data base to reconstruct the developing of a physical phenomenon in time and space.

In this chapter, we provide an adaptive, random sensor selection (ARS) strategy to efficiently distribute sensing across the network. ARS can amend for the drawbacks of the simple RSS while keep its most desirable features.

In particular, ARS enables achieving high degree of (sensing) coverage of the region of interest (RoI) while limiting the number of nodes involved in sensing. We describe the advantages and drawbacks of the RSS strategy in section 2.2 and present our *ARS* algorithm in section 2.3. We finally investigate the performance of ARS through simulations in section 2.4.

2.2 Random sensor selection (RSS)

As we outlined above, a random sensor selection strategy may provide good performance while requiring minimal control overhead [81]. Additionally, the RSS strategy also shows an implicit load-balancing feature. Indeed, if all nodes use the same value of the probability of activation p, the participation in sensing and data communication will, on average, be the same for all nodes. On the long term, therefore, the RSS enables a balanced spending of energy across all nodes of the network.

Despite its many advantages, some relevant drawbacks hamper the ability of a simple RSS-algorithm to continuously provide a reliable data base to reconstruct the developing of a physical phenomenon in time and space. The choice of an adequate value of p, for instance, is all but trivial. Being N_{tot} the total number of nodes in the network, p controls the number of nodes $N_p = p \cdot N_{tot}$ that, on average, sample and report their data to a central collector. For the RSS to be effective, N_p should be as close as possible to the number N_r of readings the data sink needs reconstruct the sensor field within the desired accuracy. If $N_p << N_r$ the reconstruction error may grow unpredictably, while having $N_p >> N_r$ makes the network generate a disproportionately high amount of sensor data and, thus, waste energy and possibly also congest the communication channel. The goal, therefore, is to choose p so that $N_p \simeq N_r$, which in turn requires a good estimation of N_r to be available. The value of N_r depends on both the spatial bandwidth of the sensor field and the actual physical distribution of the nodes [67]. A rough estimation of N_r and, thus, of p, can indeed be computed on the basis of available a priori knowledge on the physical process of interest and can possibly be refined as actual sensed data become available.

A further critical issue that may undermine the practical usage of a plain RSS approach arises when the nodes are not evenly distributed throughout the RoI (as it is typically the case in WSNs settings). Due to peculiar characteristics of the terrain, for instance, nodes may aggregate in dense clusters and let some sectors of the field uncovered. This may happen also after a careful deployment due to the failure of some of the nodes or even due to environmental influences. If all the nodes use the same, fixed value of p for deciding upon their activation, areas of the network with high density of nodes

will provide too many readings while more scarcely populated sectors will end up delivering an insufficient amount of data. In section 2.3, we will show how our ARS scheme can cope with this problem by enabling each individual node to compute its probability of activation based upon the number and positions of its neighbors. By adapting the value of p to the actual local density of the nodes, the ARS can indeed better balance the number of readings generated by each sector of the network. On the other side, local adaptation of p weakens the automatic load-balancing effect obtainable with a unique, network-wide value of p.

2.3 Adaptive random selection

In section 2.2, we have praised the benefits of a plain random sensor selection strategy, but also outlined its inherent drawbacks. In our effort to preserve the first while mitigating the latter, we developed *ARS*, a spatial sensor selection strategy that randomly selects nodes, as in RSS, but uses locally computed values of the probability of activation p. The main rationale behind ARS draws upon the consideration that the probability of activation of a node should depend on its position with respect to its neighbors, and the number thereof.

The operations of the ARS may be divided in four main phases, to which we will refer as *dissemination*, *discovery*, *computation of p* and *sensing and update*. In this chapter, we mainly concentrate on the third phase, dealing with the computation of the local values of the probability of activation and we only briefly describe the function fulfilled by the other phases that will be further discussed in chapter 4. Before going into further details, however, we need to introduce some notation and clarify the assumption ARS relies upon.

2.3.1 Notation and assumptions

We assume the RoI to be 1- or 2-dimensional however, to keep the exposition simple, we will focus on the 1-dimensional case. Note that the following definitions can be easily extended to the 2-dimensional case as well. The RoI consists in the segment $[0,L_x]$. We indicate with N_{tot} the total number of deployed nodes and we assume all nodes within the RoI to have the same transmission range R_{tx} and can communicate according to a unit disk model. We further assume that a sensor node n_i can gather, at each time instant, a noisy value of the sensor field $f(x_i)$ at its specific location x_i. We also assume that each node is aware of its position x_i. The sampled value $\widetilde{f}(x_i)$ is given by the superposition of the "true" value $f(x_i)$ and a realization n_i of a gaussian

random variable of mean 0 and variance σ_n^2, which models the measurement error. Thus, $\widetilde{f}(x_i) = f(x_i) + n_i$. Additionally, we will refer to $d_{ij} = |x_i - x_j|$ as the (Euclidean) distance between any two nodes n_i and n_j . Further, we define the sensing range R_s of a node i as the segment $[x_i - R_s, x_i + R_s]$ representing the area within which a node can perform its sensing activity (typically, $R_s \leq R_{tx}$). Finally, the maximal allowed distance between any two consecutive sampling points is called *spatial resolution* Δ_s and in the context of this work we assume Δ_s to be constant and known a priori or estimated, along with the computation of the reconstruction, by running the ACT algorithm[1] [91, Chapter 6].

2.3.2 Dissemination

During the dissemination phase, ARS distributes the value of R_s to all nodes within the network. To this scope, ARS can resort to a standard dissemination protocol [79]. Alternatively, it can "misuse" the control traffic (beacons) of the data collection protocol of choice as a back-channel to the network, as proposed in [47]. In the context of this work, we will use this second strategy and assume that the network relies upon the Collection Tree Protocol (CTP) to safely route data to the central collector [63, 64]. This choice is motivated by the good performance showed by CTP in real-world experiments and the fact its beaconing mechanism enables nodes to keep information about their neighbors updated spending only limited amount of communication.

2.3.3 Discovery

During the discovery phase, each node running ARS gathers information about its neighborhood leveraging the information reported in CTP's control beacons. In particular, the computation of the local values of the probability of activation p_i of a node n_i requires knowledge of the number of neighbors of n_i and their positions. Gathering the latter information requires adding an additional value (the $< x, y >$ coordinates of the node) to the standard CTP beacon. To evaluate the feasibility and impact on this modification, as well as the whole functionality of our ARS scheme in networks with high number of nodes, we are currently implementing both CTP and the ARS on top of the Castalia wireless sensor networks simulator [1].

[1] Note that in general, $\Delta_s \leq 2R_s$

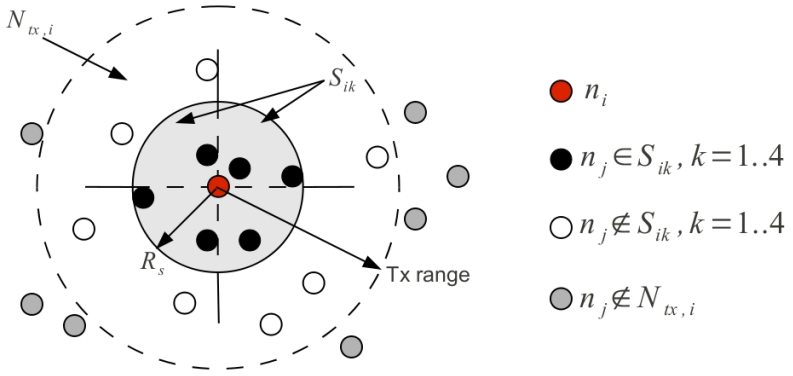


Figure 2.1: Nodes selection for the computation of p (2D case).

2.3.4 Computation of p

After the discovery phase, each node n_i holds an updated list of the $N_{tx,i}$ nodes that are within its transmission range, along with their positions s_j, $j = 1, .., N_{tx,i}$. To compute the probability of activation, we resort to the following heuristic. First, the neighbors are divided into N_{sets} sets $S_{ik}, k = 1, .., N_{sets}$. If the network is deployed on a line (1-dimensional case), the node n_i can assign each neighbor n_j to its "left" ($|s_i| >= s_j$) or "right" ($(|s_i| < s_j)$ neighborhood. If the nodes are deployed on a plane (2-dimensional case), the sets correspond to $N_{sets} = 4$ circular sectors spanning the circle centered on the node and having radius R_s (figure 2.1). In both the 1- and 2-dimensional case, only the $N_{R_s,i}$ neighbors whose distance d_{ij} to n_i is strictly smaller than R_s are included in the sets. If all sets are non-empty, the node computes, for each neighbor n_j, the quantities:

$$\phi_{ij} = 1 - \frac{d_{ij}}{R_s}$$

To understand the meaning of this factor one should recall that, considered alone, the node n_i is "responsible" for covering an entire sector of radius R_s and centered at n_i. But a node n_j with distance $d_{ij} < R_s$, can "relieve" the node n_i from part of its "sensing responsibility". The factor ϕ_{ij} is proportional to the distance occurring between node n_i and its neighbor n_j: the closer the neighbor n_j is to n_i, the higher the factor ϕ_{ij} will be[2].

In addition, the sensing responsibility of node n_i further decreases as the number of neighbors within its sensing range increases. As a consequence,

[2]Note that the factor ϕ_{ij} ranges between 0 and 1

by summing the quantities ϕ_{ij} for each neighbor of n_i, we obtain an aggregated value that represents the "sensing relevance" of n_i with respect to its neighborhood. Thus, the activation probability of node n_i will be inversely proportional to this aggregated value. To take into account the non-uniform deployment of the neighbors, the activation probability is computed separately for each sector k:

$$\Psi_{ik} = \frac{1}{1 + \sum_{j \in S_{ik}} \phi_{ij}}$$

For each set k the value Ψ_{ik} represents the probability of activation p_{ik} the node n_i should assume to "cover" the region span by the set k. An appropriate aggregate (e.g., minimum or average) of the Ψ_{ik} is then chosen as the activation probability p_i of the node n_i. For an empty set the probability of activation is 1 and in this case we force the p_i to be 1 too.

2.3.5 Sensing and update

Once each node n_i holds its own local value of the probability of activation p_i, the simple RSS algorithm is repeatedly executed. During operation, the node may also receive further updates from its neighbors and embed them on-the-fly to refine the value of p_i.

2.4 Experimental results

To investigate the ability of our *ARS* algorithm to efficiently and reliably perform sensor selection in the spatial domain, we performed a series of preliminary 1D experiments. In particular, we evaluated the ability of the ARS to provide a set on samples enabling stable and reliable reconstruction of a reference sensor field. In all our 1D experiments, we consider a sensor network of $N_{tot} = 50$ nodes distributed uniformly at random over a segment of length $L_x = 100m$. The transmission range R_{tx} is fixed and equal to $5m$, while the desired sensing range R_s is varied, so as to make the ratio $K_s = \frac{R_s}{R_{tx}}$ vary from 0.5 to 1. As an example of sensor field $f(x)$, we use the model for a physical process proposed within the Castalia [1] simulator (with parameters $N_{sources} = 1$, $V = 1$, $K = 0.05$, and $a = 3$). To reconstruct the physical process from its unevenly spaced samples we resort to nearest neighbor interpolation. The resulting reconstructed sensor field is indicated as $\hat{f}(x)$ and the correspondent reconstruction error signal as $e(x) = f(x) - \hat{f}(x)$.

In our experiments, we consider three main performance metrics: the percentage of active nodes $\%AN$, the percentage of the RoI that remains uncovered $\%URoI$ and the relative root mean square (rms) error of the reconstruction $Rrms$. $\%AN$ is defined as the ratio between the number of nodes that

participate in sensing and the total number of nodes. $\%URoI$ expresses the percentage of the RoI that is not covered by any active node, i.e., the percentage of points within the RoI whose distance to any of the active nodes is bigger than R_s. Finally, the $Rrms$ is defined as the ratio between the energy of the reconstruction error signal $e(x)$ and the energy of the original signal $f(x)$. To gather statistically significant values of the above described metrics, we generated 50 different random network configurations and, for each configuration, we run 50 trials of the ARS algorithm. In this preliminary study, we assume each node to have perfect knowledge of its neighborhood $N_{tx}(i)$ and of the value of R_s. The sharing of the sensing range and the neighbor discovery phase are faced in chapter 4.

For each configuration and trial, we also evaluate the performance of the plain RSS approach using two different values of the probability of activation p (we recall that for the RSS the value of p is fixed a priori and equal for all nodes). The first value, dubbed p_{min} is computed so that the expected number of active nodes is equal to the absolute minimum number of nodes that can guarantee R_s-coverage over the RoI. Clearly, this corresponds to the case in which the nodes are deployed on a regular grid and the required value of p_{min} is thus simply computed as $p_{min} = \frac{AN_{min}}{N_{tot}}$, where $AN_{min} = \frac{L_x}{R_s}$. The second value of the probability of activation considered in our experiments, dubbed p_{guess} is chosen as $p_{guess} = 2 \cdot p_{min}$. Previous work has indeed showed that random sampling typically requires more than twice the sampling rate of regular sampling to provide for a reliable reconstruction [101]. Additionally, if noise in the measurements and in the position estimates of the nodes is present, the number of required samples further increases. Therefore, p_{guess} can be seen as a realistic and conservative estimate of the required probability of activation in the RSS scheme. Finally, all the experiments reported in this section have been implemented and run on top of the Matlab[3] computing environment.

In the following, we will first show how the ARS modifies the probability of activation with respect to the RSS and then move over to the performance analysis based on the above defined metrics. For simplicity and without any loss of generality, we will present particular results obtained for the first of the 50 considered network configurations (configuration #1) and show complete results (over all configurations and trials) later in the section.

2.4.1 Local adaptation of the value of p

Figure 2.2 shows the values of the probability of activation p_{ARS} for the configuration #1, along with the values of p_{min} and p_{guess} for comparison. The

[3] www.mathworks.com

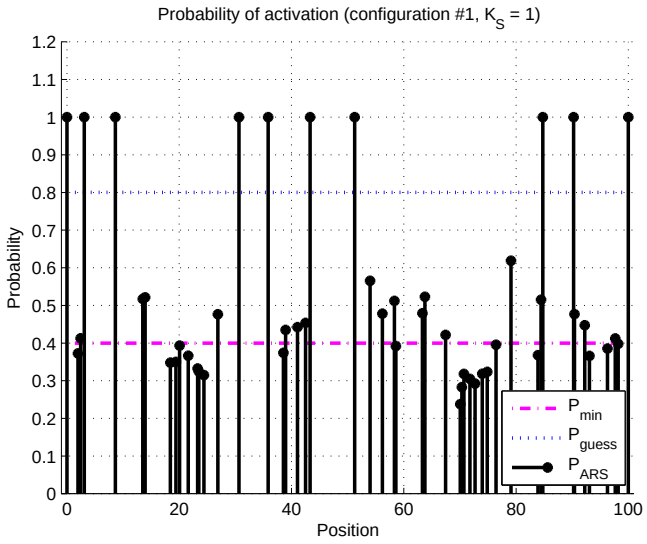


Figure 2.2: Probability of activation for the configuration #1 ($K_s = 1$).

horizontal axis reports the position of the nodes within the RoI and since the probability of activation only depends on the specific physical topology, it does not change across different trials. As expected, the value of p_{ARS} is higher for those nodes having less or far away neighbors and significantly lower for nodes lying in "crowded" neighborhoods. The average value of p_{ARS} is ~ 0.5, which foreshadows that the number of active nodes AN_{ARS} resulting by using the p_{ARS} will be, on average, about 10% higher the number resulting by the adoption of $p_{min} = 0.4$, but also considerable lower than the number of active nodes generated, on average, using the value $p_{guess} = 0.8$. As we will show later, this slightly higher number of nodes is largely compensated by the far better performance of the ARS in terms of obtainable sensing coverage and the resulting better signal reconstructions.

2.4.2 Number of actives nodes and sensing coverage

Figure 2.3 shows, again for the reference configuration #1, the average values of the percentage of active nodes, of the percentage of RoI being uncovered and the relative rms error of the resulting reconstruction (subfigure *(a)*). As we anticipated, the $\%AN$ is indeed higher for the ARS with respect to the case in which the RSS scheme with probability of activation p_{min} is used ($\sim 54\%$ against 43%, respectively). Despite this higher overhead, the ARS shows a far better ability to cover the RoI, since it obtains an average $\%URoI$ of $\sim 5\%$,

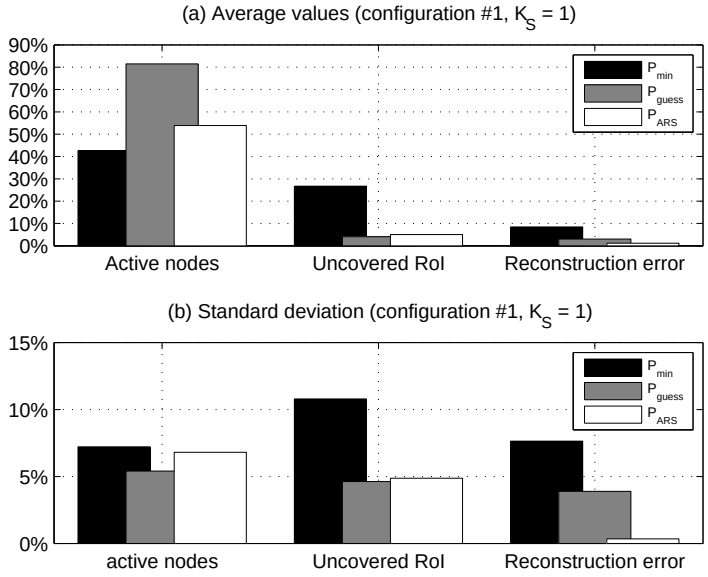


Figure 2.3: Average values of the $\%AN$, the $\%URoI$ and the $Rrms$ of the resulting reconstruction (a) along with the correspondent values of the standard deviation computed over 50 trials. All results relate to the reference network configuration #1.

while the RSS with p_{min} lets uncovered more than 26% of the RoI. In terms of sensing coverage, ARS has performance comparable to RSS with p_{guess}, which however requires more than 80% of the nodes to be active. The ability of ARS to provide sensing coverage of the RoI is also mirrored by the better performance in reconstructing the physical process $f(x)$. Figure 2.3 shows that while the average $Rrms$ for RSS with p_{min} is $\sim 8.5\%$, ARS is able to provide an average relative error of 1.1%. We should note at this point that the values of $Rrms$ reported here are computed using the true values of the physical process. In real settings, where the original values of the signal of interest are clearly unavailable, adequate estimations of the reconstruction errors must be used.

A significant feature of the ARS scheme is observable in the lower part of figure 2.3, which reports the standard deviations, computed over 50 trials, relative to the average values reported in the upper plot. The standard deviation of the reconstruction error is practically negligible for ARS ($\sim 0.34\%$), while the correspondent value for RSS with p_{min} is as much as 8%. Such a

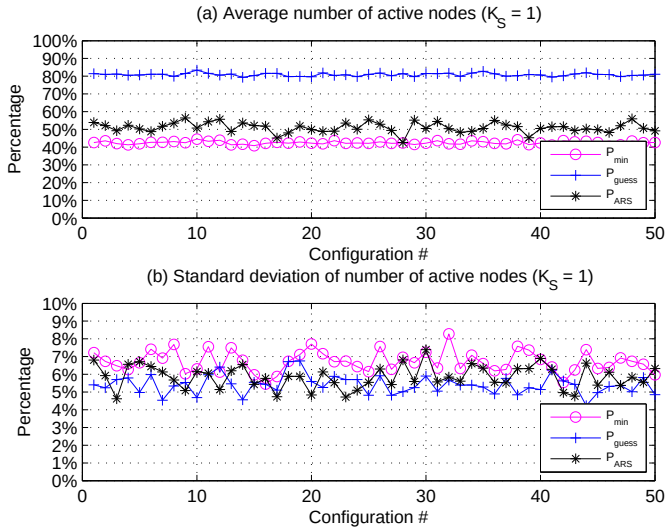


Figure 2.4: Average values of the $\%AN$ over 50 trials for 50 different configurations (a), and correspondent values of the standard deviation (b).

high value for the standard deviation shows that the reconstruction obtained through a completely random sampling pattern is typically highly unstable and, thus, unreliable. This drawback is particularly problematic if the application aims at controlling the number of active nodes (or, equivalently, the value of the sensing range R_s) necessary to reconstruct the physical process of interest within a given accuracy. To this scope, the application must estimate the quality of the reconstruction from the collected samples and decide upon one or more estimates whether to increase (or decrease) the sensing range R_s. Clearly, a safe controlling mechanism is unfeasible if the estimates are too unstable.

Until now, we only reported results relative to a specific reference configuration. Obviously, it is interesting to see whether the above praised abilities of the ARS remain valid for several different physical network topologies. Figure 2.4*(a)* shows the average $\%AN$ obtained for 50 different configurations along with the correspondent standard deviation (both computed over 50 trials). As we can see, the number of active nodes in the ARS is constantly slightly higher than the correspondent figure for the RSS with p_{min}, but significantly lower compared to the RSS with p_{guess}. This slightly higher effort is however praised by a significant better coverage, as showed in figure 2.5*(a)*, which reports the $\%URoI$ for all the three considered schemes. As we can see, the ARS schemes offer coverage performances comparable to those of the RSS

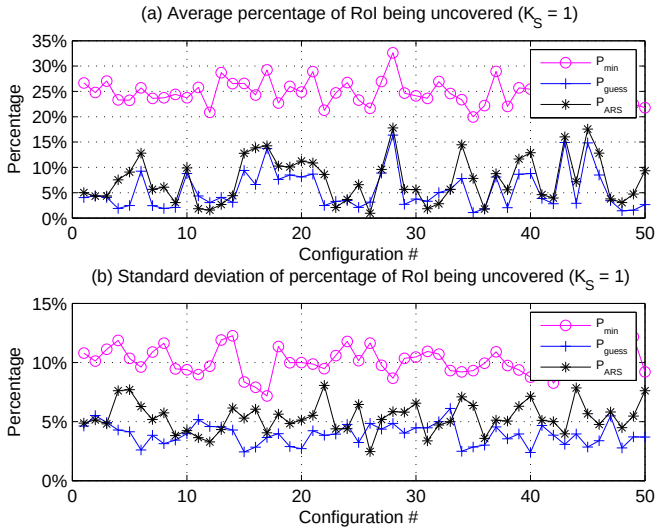


Figure 2.5: Average values of the $\%URoI$ 50 over 50 trials for 50 different configurations (a), and correspondent values of the standard deviation (b).

with p_{guess}, and behave in this context far better than the RSS with p_{min}. As a consequence of its ability to reliably provide sensing coverage, the ARS also enables to compute reconstruction of signals of interest with low and nearly constant error.

2.4.3 Number of active nodes for higher network density

Figures 2.6*(a)* and 2.6*(b)* report the average $\%AN$ obtained by increasing the total number of nodes N_{tot} to 100 while keeping all the other simulation parameters unchanged. The figure shows the effectiveness of random sensor selection approaches that, thanks to the higher density, enables the evaluated protocols to sensibly decrease the number of active nodes. As in the previous experiments, the number of nodes selected by the ARS is slightly higher than the correspondent figure for the RSS with p_{min}, but significantly lower compared to the RSS with p_{guess}. However ARS reach significantly better reconstruction performance, as showed in figure 2.6*(c)*, which reports the $\%Rrms$ for all the three considered sensor selection schemes. ARS thus offers reconstruction accuracy comparable to that obtained with the RSS with p_{guess} (and far better with respect to the RSS with p_{min}) but requires far less nodes to actively sample the signal.

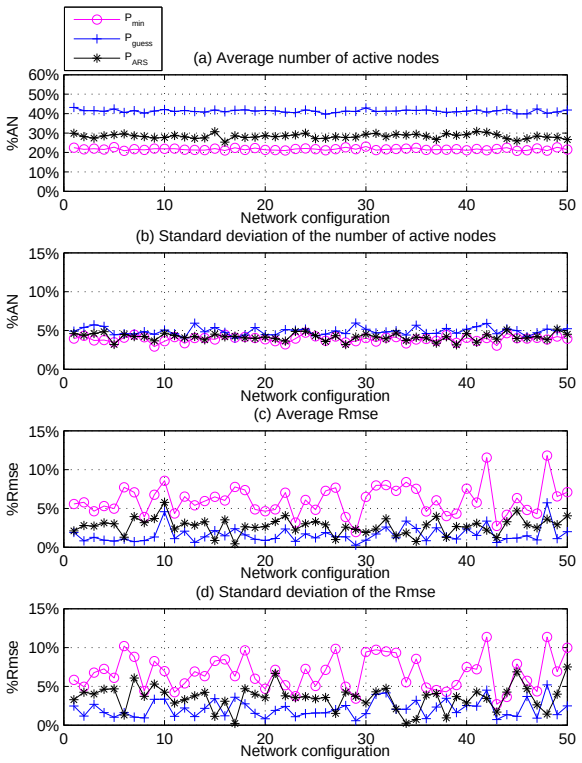


Figure 2.6: Average values of the $\%AN$ and $\%Rmse$ for 50 different network configurations (a), (c) along with the correspondent values of the standard deviation (average over 50 trials) (b), (d).

Path Loss Exponent	2.4
Reference distance d_0 (m)	1
Path Loss (d_0) (dBm)	61.4
Sigma	4
Bidirectional Sigma	1

Table 2.1: Castalia Wireless Channel Parameters

2.4.4 2D experiments

In [112] additional experiments have been performed using the 2D settings. Keeping the same metrics defined at the beginning of this section and extending the notation to the 2D case, the new experiments were made on a $L_x = L_y = 100$m square area with $N_{tot} = 200$ nodes distributed uniformly at random over it. The radio range of each node is set to $R_{tx} = 25$m and a varying sensing range, from $Rs = 9.375$m to $Rs = 12.5$m, is used. To gather statistically significant values, 25 different topologies have been generated and, for each one, 25 trials have been performed. As for the 1D experiments, the Matlab environment has been used to generate the results.

In [112, sect. 4.7.4] it is shown how ARS provides high degree of coverage even in the 2D settings, thus, validating the protocol design. In particular, in all the topologies, the RoI steadily remains over the 95% with a standard deviation below 1%. In addition, it is shown how, based on the sensing range set, the number of active nodes in such settings varies between 60% and 80% of N_{tot}.

In a recent work we provided a new implementation of the CTP collection protocol over the Castalia 3.1 framework for OMNeT++ 4 [52]. This has allowed us to replicate the experiments made in [112] in a more realistic scenario and to extend the tests with a comparison between ARS and RSS in terms of the percentage of active nodes $\%AN$, the percentage of the RoI that remains uncovered $\%URoI$ and the relative root mean square (rms) error of the reconstruction $Rrms$. As in [112], we generated 25 different topologies and for each one, we executed 25 trials. We set a sensing range $Rs = 12.5$m and kept the other simulation parameters unchanged. Differently from Matlab, Castalia provides an advanced wireless channel and radio implementation. The parameters used for the wireless channel and the radio model can be found in table 2.1 and 2.2 respectively.

In order to find suitable p values for the RSS approach, we ran Ars over the previously described setup and obtained the mean activation probability

Datarate (kbps)	250
Modulation	PSK
Bits per symbol	4
Bandwidth (Mhz)	20
Noise Bandwidth (Mhz)	4
Noise Floor (dBm)	-100
Sensitivity (dBm)	-95

Table 2.2: Castalia Radio Parameters

$p = 0.76$ computed by Ars. We then compared Ars to RSS applying activation probabilities ranging from 0.7 to 0.85. Figure 2.7 shows the results of our experiments.

As expected, the percentage of active nodes follows the activation probabilities used in RSS and the one computed by Ars. It is interesting to note the higher $\%AN$ standard deviation of Ars with respect to RSS approaches. This difference is related to the fact that Ars adapts the activation probability to the topology it runs on, thus, the value may fluctuate based on the spatial distribution of the nodes. On the contrary, the RSS approach uses the same activation probability irrespective of the underlying topology, hence, the standard deviation of the number of active nodes is lower.

Thanks to the adaptive computation of p, the uncovered RoI of Ars is by far the lowest one. In fact, the mean $\%URoI$ of Ars is 0.1% against 0.33% of RSS with $p = 0.76$ (i.e., the same activation probability than Ars but with RSS scheme) and 0.2% of RSS with $p = 0.85$. As in [112], the standard deviation of $\%URoI$ is much lower than 1%, therefore, negligible.

Finally, the relative root mean square error has been evaluated over the signal reconstruction of a static physical process generated by Castalia (with parameters $N_{sources} = 1$, $Position = (0, 0)$, $K = 0.15$, and $a = 1$) represented in figure 2.8. It is important to note that with the same mean activation probability $p = 0.76$ Ars outperforms RSS by 20% . Increasing the activation probability of the RSS approach, the $Rrms$ decreases and beats Ars starting from $p = 0.85$, thus, using 12% more nodes during the sampling phase. Nonetheless, the $Rrms$ standard deviation of Ars (18.9%) is sensibly lower than the one of RSS which ranges from 44% for $p = 0.7$ to 33.2% for $p = 0.85$. The reason of this difference resides on the capability of Ars to detect sparsely covered areas as much as nodes close to edges. In fact, recalling section 2.3.4, Ars keeps the activation probability to 1 if at least one of its spanning sectors

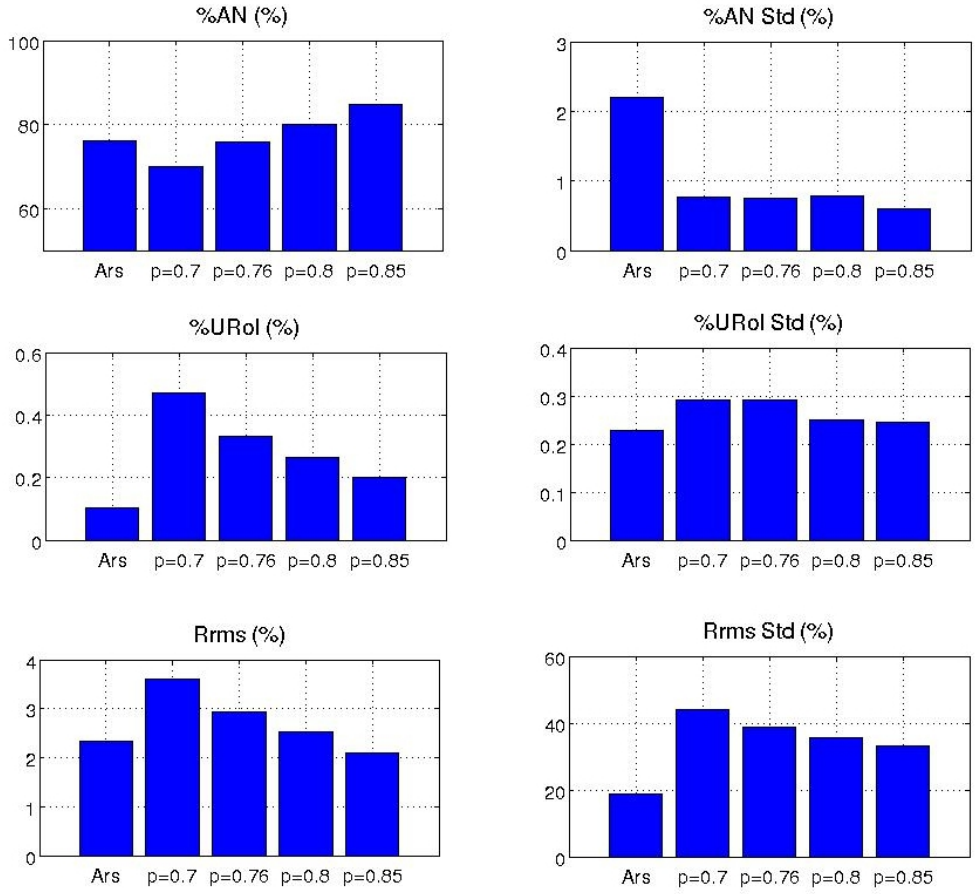


Figure 2.7: Average values of the $\%AN$, $\%URoI$ and $Rrms$ for 25 different network configurations along with the correspondent values of the standard deviation (average over 25 trials).

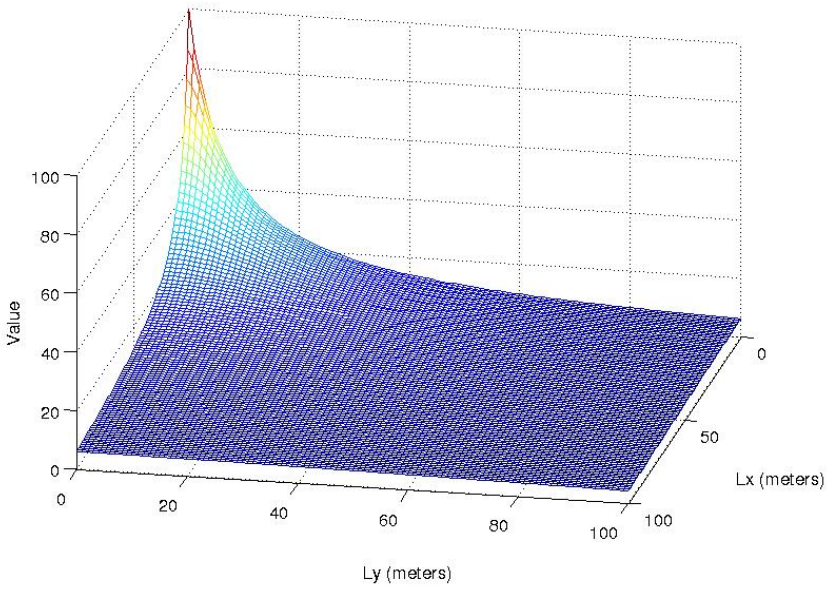


Figure 2.8: Physical Process used for $Rrms$ computation (Source (0,0), Alpha=1.0, K=0.15).

S_{ik} is empty. By this way, the activation probability has a role in densely covered areas only, thus mitigating $Rrms$ fluctuations.

2.5 Conclusion

In this chapter, we sketched the main component of ARS, an adaptive random spatial sensor selection protocol. Leveraging the neighborhood information gathered through the CTP data collection protocol, ARS can compute locally adapted values of the probability of activation p. Each node n_i can then decide autonomously whether to participate in the sensing task or not depending on the outcome of a random throw. Our simulation study shows the ability of ARS to provide for good sensing coverage of the region of interest while limiting the number of active nodes. Using the same number of nodes, Ars always outperforms RSS schemes in terms of signal reconstruction error and better adapts to all the underlying topologies.

As further steps, it would be interesting to check the different performance of both, Ars and RSS, with a varying sensing range, thus, different require-

ments in signal reconstruction's quality. Moreover, evaluating Ars and RSS in dynamic environments (i.e., with a physical process that varies over time) would certainly provide a more complete view of the advantages of our solution.

Chapter 3

DISSense: an adaptive energy-aware collection protocol

3.1 Introduction

As discussed in 1.4, several energy-aware routing and MAC protocols have been proposed in the past decade. However, because of their general-purpose design and because of inefficiencies when energy-aware MAC and routing protocols are combined together, most of the proposed solutions cannot reach energy requirements of Periodical Environmental Monitoring applications. To this aim, cross-layer communication protocols such as [47] and [98] have been recently introduced. The solution presented in this chapter, DISSense, falls within this category. In particular, DISSense is an adaptive, energy-aware communication protocol that has been developed taking into account cross-layer optimization issues. DISSense offers both a collection and dissemination service, and it is particularly suited to support long-term environmental monitoring applications. In this chapter we will show how thanks to its ability to adapt to changing network conditions, DISSense delivers, on average, more than 98% of the data packets injected into the network. At the same time, DISSense aggressively reduces the duty cycle of the network, enabling a mote-based WSN to reach several years of network lifetime that in many case it is comparable to [47] and [98] but with the advantage of an open source

multi-platform support, low data latency and scalability through multi-sink deployment.

3.2 Protocol overview

DISSense provides for both a data dissemination and collection service for WSNs. It targets environmental monitoring applications requiring periodic sampling of a given physical phenomenon. In particular, DISSense takes as input the desired sampling period and computes an adaptive time schedule for the nodes to coordinate in order to build a data collection tree. The schedule alternates short activity phases during which nodes deliver sensed data to the sink and long intervals during which nodes operate in an ultralow-power mode. Additionally, DISSense implements an efficient, one-to-many backward channel for disseminating the shared schedule to all nodes in the network.

3.2.1 Adaptation

DISSense achieves energy-efficient operation by adaptively shortening the length of the time interval during which nodes must activate their radio transceivers. Reducing the length of the active phase clearly enables DISSense to reduce the duty cycle of the network and, thus, to extend its lifetime. The main challenge arising in this context consists in making the protocol able to timely and reliably deliver data to the sink despite the shortening of the active phase. The diameter, density, and overall link quality of the network also affects protocol behavior. For example, reliable protocols such as DISSense may require several (re)transmission attempts over a bad link before at least one succeeds. Moreover, the denser the network the longer it takes to settle channel contention. And the higher the diameter of the network the higher the average number of hops packets must be relayed through before reaching the sink. By taking into account all these factors we make DISSense able to autonomously adapt its duty cycle to the actual dynamics of the network, and to ensure both high delivery ratios and energy efficiency. To control DISSense's adaptive behavior, we define two metrics: the Time To Resync (TTR) and Time To Receive Data (TTRD), which we will describe in detail in section 3.3.4.

3.2.2 Schedule

The sink is responsible for determining and disseminating the schedule according to which nodes send and receive their packets. Figure 3.1 illustrates the different phases of the DISSense schedule: active phases are scheduled at each

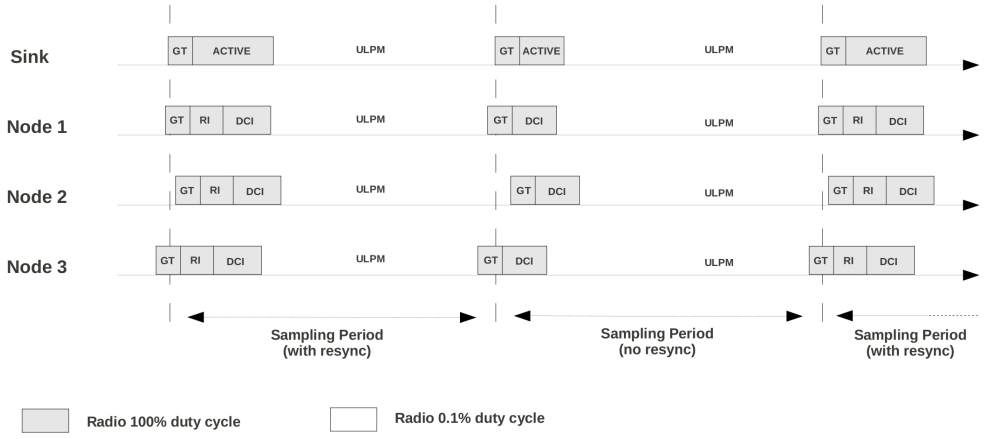


Figure 3.1: DISSense schedule with $\sigma_{sk} = 1$

sampling period. Because of clock drifts and of the long inactivity period between active phases, a GuardTime Interval (GT) is foreseen at the beginning of each phase. Moreover, a resynchronization procedure periodically takes place during the Resynchronization Interval (RI), so as to realign the schedule and compensate for clock drifts. Depending on the sampling period and intervals length, DISSense is able to skip the RI for one or more sampling periods, so as to optimize the overall protocol duty cycle. The skip functionality depends on the parameter σ_{sk} , whose computation is described in section 3.3.4. During the RI, nodes exchanges routing beacons and collect the information needed to build a collection tree having the sink as its root. At the end of the RI, DISSense ensure that the nodes share a common wake up time for the next active phase, and have a parent selected in the collection tree for data transmission. After the RI, the Data Collection Interval (DCI) begins. During the DCI each node sends its data over the already built collection tree, and also acts as forwarder for other nodes of the network. Between two active phases, DISSense turns into an Ultra-Low-Power State (ULPS) by switching the radio to LPL mode with a 0.1% duty cycle. In ULPS the radio is not turned fully off since some nodes may be added and other ones can go out of synchronization. Both these nodes need to retrieve the protocol schedule in order to participate to the network. The value of the duty cycle during ULPS is low such that it does not significantly affect the overall protocol duty cycle. Note that the sink schedule only adopts an *ACTIVE* interval, since it does not need to discriminate the different intervals of the active phase. The active phase of DISSense runs on a CSMA/CA MAC with 100% duty cycled radio. The benefits of such a solution are twofold. First, it accelerates the construction of the collection tree and the data collection process itself, thereby shortening the

length of the active phase. Second, it prevents the inefficiencies, described in [77, Paragraph 5.7] related to broadcast transmissions (e.g., routing beacons) under duty cycled MAC protocols.

3.2.3 Collection and backward channel

Data collection in DISSense is achieved by leveraging and extending the CTP Collection protocol [65]. CTP is a popular and highly reliable collection protocol. When running CTP each node computes a metric, called ETX, which represents the estimated number of transmissions a packet from this node will go through before reaching the sink. CTP supports link-layer acknowledgement for reliable data collection. It also supports loop detection, duplicate suppression and quick reaction to topology changes. However, CTP is not optimized for applications requiring short active phase sessions interleaved with inactivity periods, such as the scenario we are taking into account. Instead, DISSense allows to stop, start, pause, and reset the construction and maintenance of the collection tree at any time.

The backward channel in DISSense is used by the sink to resynchronize the network and to send schedule changes to nodes (e.g., intervals length and changes in sampling period). A node missing a schedule update is likely to loose synchronization with the other nodes. DISSense implements a backward channel, namely the Implicit Backward Channel (ICB), that guarantees that each node having selected a parent in the collection tree, also shares the same values sent by the sink over the ICB during the active phase. The ICB runs during the RI and uses the same beacons required for the collection tree construction. The main advantage of this solution is that the RI interval can be tailored on the collection tree construction only since the ICB execution does not require any additional time. Further details are discussed in section 3.3.1 and 3.3.2.

3.2.4 Architecture

Figure 3.2 illustrates DISSense's architecture. DISSense main modules are the Manager, the Adaptive Engine, the LplManager, and the NtpManager, which are described below. As mentioned above, DISSense relies on CTP to build and maintain the routing tree. Thus, DISSenses embeds CTP's Link Estimator, Forwarding Engine, and Routing Engine modules.

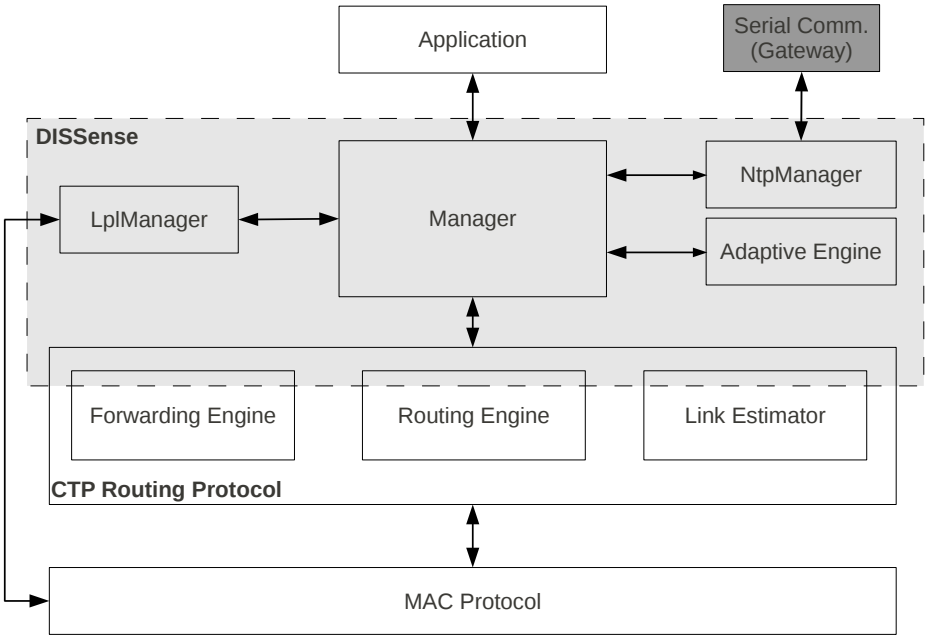


Figure 3.2: DISSense Architecture

Manager

The Manager handles DISSense core functionality, such as network resynchronization and schedule management. The module also provides a Send/Receive interface to the application layer that enables the send of a single data packet during each active phase and collects statistics for the Adaptive Engine. The Manager has the ability to start, pause and reset the underlying CTP protocol. It can also change the radio duty cycle.

Adaptive Engine

The Adaptive Engine has a dual functionality. On the sink, it computes intervals length for GT, RI, DCI and skip parameter σ_{sk}. These values are then transmitted over the ICB. On the nodes, the Adaptive Engine retrieves and stores the values, which can later be read by the manager module.

LplManager

The LplManager module is responsible for radio communications during Ultra-Low-Power Mode. As mentioned in section 3.2.2, during ULPM phase, the nodes turn their radio to a 0.1% duty cycle rather than switching it off. The LplManager enables a node that looses synchronization or a newly added node to efficiently retrieve synchronization information from its neighbors during their inactivity period. This mechanism avoids the need to scan for an active phase, which requires high power consumption. The LplManager also supports the additional functionality of state transmission, which consists in sending a snapshot of the node state for debugging purposes.

NtpManager

The NtpManager module that is active only on the sink, interacts with an external gateway to synchronize the sink with an external entity (e.g., Ntp Time). The NtpManager provides an additional functionality that enables the user acting on the gateway, to dynamically change the sampling period, and to determine the hour of the day at which data samples need to be generated.

3.3 Implementation

DISSense is implemented in TinyOS 2.1 [12], a lightweight, open-source operating system for WSNs. It supports the TelosB/TmoteSky and MICAz platforms. Support for Iris and TinyNode 184 is planned in future releases [6, 11]. DISSense also runs on the TOSSIM simulation environment.

As described in section 3.2, the sink in DISSense acts as an orchestrator for the network. In particular, the sink is responsible to determine the adaptive schedule, share it over the network, and periodically collect the sampled data from the nodes. To this end, DISSense runs a deeply modified version of CTP. On one hand, DISSense adapts CTP to run on the specific schedule described in section 3.2.2, which requires CTP to be paused and resumed periodically following the duty cycle of the protocol. On the other hand, DISSense embeds a backward channel in CTP namely, the Implicit Backward Channel, which enables DISSense to send controls for initiating synchronization and tree construction.

3.3.1 Data collection

In DISSense the CTP protocol is stopped during Ultra-Low-Power Mode and resumed during the active phase. When resumed, DISSense runs CTP in a

stateful or *stateless* mode. In stateful mode the routing information of the previous sampling period is used. This mode is employed when the DCI is scheduled without the RI and there is the need to directly collect the sampled data. Instead, in stateless mode the CTP state (neighbor table, selected parent) is reset in order to refresh the topology information of CTP and enable the IBC to share new synchronization information as much as the new schedule parameters values. The stateless mode is run when the RI is scheduled.

3.3.2 Implicit backward channel

The IBC guarantees, during each RI interval, that a node having selected a new parent also shares the values transmitted by the sink at the beginning of the RI. DISSense uses the IBC to share schedule's information updates. In particular, at each RI, the sink transmit over the IBC the sampling period, the length of RI and DCI intervals, the skip parameter σ_{sk} and the next wake up time. These values are appended as footer to each routing beacon. For each incoming beacon, the Routing Engine follows the algorithm in figure 3.3 which enables a node to store the information carried by the beacon (and retransmit it as a footer in each subsequent transmitted beacon) if and only if the sender has been selected as a parent.

Note that during beacons transmission, each node fills the footer with its local values, thus, at the beginning of the RI, each node that has not yet selected a parent, transmits stale content. The algorithm, however, implicitly solve this issue thanks to the parent selection algorithm of CTP. The parent selection algorithm guarantees that each parent candidate must already have selected a parent in the tree. Thus, each parent candidate must have run the IBC update process of algorithm 3.3, hence, carry updated values. As a consequence, at the end of the RI, each node will be in one of these two states:

- The node belongs to the collection tree, is resynchronized and have updated schedule parameters values;
- The node does not belong to the collection tree, is out of synchronization and needs to retrieve the schedule through the LplManager during ULPM.

3.3.3 Resynchronization

The resynchronization procedure uses the IBC to reliably propagate the timestamp related to the beginning of the next active phase. DISSense manages the Next Wake Up Time field of the IBC footer using the *TimeSyncAMSend* interface implemented in TinyOS 2.1 by the *CC2420TimeSyncMessage* component.

Require: node n

upon rx beacon b_i from node i

if parent(n) == null **then**

$temp \leftarrow footer(b_i)$

process(b_i)

if parent(n) == i **then**

resync($temp.nwu$)

store($temp.ri, temp.dci$)

store($temp.sp, temp.\sigma_{sk}$)

end if

$temp \leftarrow null$

else

process(b_i)

end if

Figure 3.3: IBC Algorithm

This component is a submodule of the *Flooding Time Synchronization Protocol* [90]. It has been written for ChipCon2420 transceivers and allows a sender to piggyback a local timestamp t_e, related to an event e, to each transmitted packet. The receiver, in turn, decodes the timestamp t_e as a new timestamp $t'_e = t_e + \delta$ representing the event e expressed as the receiver's timestamp t'_e with an error δ corresponding to the packet propagation time.

Using the *TimeSyncAMSend* interface for CTP beacons transmission enables DISSense to reliably propagate the next wake up time over the network with a maximum error $\Delta = d \cdot \delta$ where d is the network diameter. Note that, over short distances, the packet propagation time is in the order of few microseconds and, as it will be shown in section 3.4.2, Δ is several orders of magnitude lower than the size of the active phase. Thus, as figure 3.3 shows, when the parent is selected, the *resync* commands is called over the *Next Wake Up Time* field that realigns the schedule of the next wake up.

3.3.4 Adaptation

As described in section 3.2.1, DISSense uses two metrics, TTR and TTRD, in order to find a good trade-off between duty cycle minimization and correct protocol execution. These metrics are computed by the Manager, which sends them to the Adaptive Engine along with the protocol sampling period. Based on these values, the Adaptive Engine computes the intervals length of GT, RI, and DCI as well as the skip parameter σ_{sk} .

Time To Resynchronize (TTR) The TTR represents the maximum time required by DISSense to resynchronize all the nodes of the network. The value is computed by the Manager module of the sink during each sampling period where the RI is scheduled. Let n be the number of nodes of the network, the Manager retrieves TTR_i from each node $i = 1 \ldots n-1$ and computes the TTR as $max\{TTR_1, \ldots TTR_{n-1}\}$. The TTR_i of each node i is computed as the time elapsed from the beginning of the active period to the resynchronization event (c.f. *resync* command in algorithm of figure 3.3). Each locally computed TTR_i is piggybacked to data packets transmitted at each sampling period. Note that both events, the beginning of the active period and resynchronization, may vary from node to node. In fact, the former depends on node's clock drift which may induce nodes to wake up at different time intervals, while the latter depends on the resynchronization algorithm that, as seen in 3.3.3, depends on the node-to-sink distance and, hence, may differ between nodes.

Based on this metric, the Adaptive Engine will try to schedule a Resynchronization Interval (RI) short enough to reduce the overall active phase, thus, the Duty Cycle of the protocol, but long enough to allow all the nodes

participating into DISSense to correctly resynchronize.

Time To Receive Data (TTRD) The TTRD represents the time required by DISSense to collect the sampling data from the network. We recall that, as for most monitoring application, each node generates one packet per sampling period that is forwarded to the sink. As a consequence, the Manager module of the sink computes the $TTRD$, during each sampling period, as the time elapsed between the reception of the first and the last data packet during active phase. Note that the computed $TTRD$ is an approximation of the actual time required by DISSense to collect sampling period because it ignores the node-to-sink latency of the first packet received. However, the first packet usually arrives from a 1 hop neighbor of the sink and, thus, the node-to-sink latency of the first packet is very small compared to the overall computed TTRD.

The TTRD metric is used by the Adaptive Engine to schedule a Data Collection Interval (DCI) long enough to let the nodes in the collection tree to correctly forward their data packets to the sink. A longer DCI interval will extend the active phase of the schedule, thus, increase the Duty Cycle of the protocol. On the other side, a too short DCI interval will produce losses in packets forwarding due to the schedule that switches to ULPM phase in presence of nodes still attempting to forward data packets.

Duty Cycle Let T_{period} be the sampling period and T_{GTI} , T_{RI} , and T_{DCI} the GT, RI and DCI lengths. Also let $T_{on}(\sigma_{sk})$ be the fraction of time, within a sampling period, during which the radio is in active phase. Similarly, let $T_{ulpm} = 1 - T_{on}(\sigma_{sk})$ represent the same ratio for the Ultra-Low-Power Mode phase. We compute $T_{on}(\sigma_{sk})$ as:

$$T_{on}(\sigma_{sk}) = \frac{T_{GTI} + T_{RI} + T_{DCI}}{T_{period} \cdot (\sigma_{sk} + 1)} + \frac{T_{GTI} + T_{DCI}}{T_{period} \cdot (\sigma_{sk} + 1)} \cdot \sigma_{sk} \tag{3.1}$$

Recalling section 3.2.2, during $T_{on}(\sigma_{sk})$ the duty cycle is 100% while during T_{ulpm} it is set to 0.1%. We define the protocol duty cycle $P_{dc}(\sigma_{sk})$ as weighted sum of $T_{on}(\sigma_{sk})$ and T_{ulpm} with their corresponding duty cycle, hence:

$$P_{dc}(\sigma_{sk}) = 1 \cdot T_{on}(\sigma_{sk}) + 10^{-3} \cdot T_{ulpm} \approx T_{on}(\sigma_{sk}) + 10^{-3} \tag{3.2}$$

Guard Time Interval The Guard Time interval must be greater than the maximum drift produced within two resynchronization procedures. It depends on clock precision, sampling period and Skip value. For instance, the clock precision of TelosB motes is 50 ppm [6] that becomes $100ppm$ assuming clocks drift of two nodes in opposite direction. Further, resynchronization takes place

each $\sigma_{sk}+1$ sampling periods. Thus, if T_{period} represents the sampling period, the Adaptive Engine computes the Guard Time Interval as:

$$T_{GTI} = 3 \cdot 50 \cdot T_{period} \cdot (\sigma_{sk} + 1) \cdot 10^{-6} \tag{3.3}$$

Note that to ensure a safety margin (e.g., unpredictable behavior related to temperature changes) the interval is equal to the maximum drift increased by 50%.

Skip period The Adaptive Engine computes the skip value σ_{sk} so as to minimize the overall duty cycle of the protocol $P_{dc}(\sigma_{sk})$. Recalling equation (3.2), the minimum duty cycle corresponds to the minimum $T_{on}(\sigma_{sk})$. Thus, replacing (3.3) in (3.1) we have:

$$T_{on}(\sigma_{sk}) = \frac{150 \cdot (\sigma_{sk} + 1)}{10^6} + \frac{T_{DCI}}{T_{period}} + \frac{T_{RI}}{T_{period} \cdot (\sigma_{sk} + 1)} \tag{3.4}$$

It is easy to demonstrate that the minimum of (3.4) is reached for:

$$\tilde{\sigma}_{sk} = \sqrt{\frac{T_{RI} \cdot 10^6}{T_{period} \cdot 150}} - 1 \tag{3.5}$$

However, σ_{sk} must be an integer, thus, after having observed that the second derivative of (3.4) is positive, we can assess that the minimum duty cycle is reached for:

$$\sigma_{sk} = \arg\min(T_{on}(\lfloor \tilde{\sigma}_{sk} \rfloor), T_{on}(\lceil \tilde{\sigma}_{sk} \rceil)) \tag{3.6}$$

Resynchronization and Data Collection Intervals The RI must be long enough to allow DISSense to resynchronize all the nodes of the network and build the collection tree. Recalling the resynchronization algorithm in figure 3.3, a resynchronized node has already selected the parent thus, the RI depends uniquely on TTR. Similarly, the DCI must be long enough to allow each generated sample to be routed to the sink, thus, it depends on TTRD.

For each TTR and TTRD update, the Adaptive Engine increases them by 50% to catch values fluctuation and inserts them in an Exponential Weighted Moving Average (EWMA) obtaining:

$$T_{RI} = \frac{1.5 \cdot TTR_{new} + 9 \cdot T_{RI_{old}}}{10} \tag{3.7}$$

$$T_{DCI} = \frac{1.5 \cdot TTRD_{new} + 9 \cdot T_{DCI_{old}}}{10} \tag{3.8}$$

Where TTR_{new} and $TTRD_{new}$ are the newly inserted values while $T_{RI_{old}}$ and $T_{DCI_{old}}$ the intervals length before the updates.

3.4 Evaluation

We evaluated the performance of DISSense on both an indoor testbed and on the TOSSIM simulator. While the latter allowed us to test DISSense on different network topologies, the testbed-based evaluation demonstrates DISSense feasibility for real-world deployments. As detailed below, our experimental results show that DISSense can guarantee for reliable data delivery and, thanks to its power-efficiency, it allows to operate a Tmote Sky-based WSN with common AA type alkaline batteries for several years.

3.4.1 Metrics

We evaluate the performance of DISSense in terms of achieved duty cycle and data delivery, as well as in terms of number of duplicate packets. For the definition of DISSense's duty cycle we refer to equations 3.1 and 3.2. We derive an estimation of the lifetime of the network given a specific duty cycle using the methodology suggested in [47], thus, ignoring power consumption of application specific sensors. We further assume that a node is powered by two common AA Alkaline batteries of 2500mAh capacity each and that the current drain of the TmoteSky during 100% duty cycle and active CPU is 21.8mA [107]. Taking into account battery self-discharge equal to 15% in 4 years [2], a network running DISSense with a duty cycle of 1% can operate for 1 year. Reducing the duty cycle to 0.2% allows extending the network lifetime to 5 years. These considerations show the significant impact of the duty cycle on the lifetime of a WSN. However, these values represent rough estimates since temperature, intermittent power drain and chemical deterioration over time can increase or decrease the speed of battery discharge. Moreover, the lifetime can be further increased by using more performing type C batteries that can reach 8000mAh capacity [48] or some kind of energy harvesting technique such as photovoltaic panels [35].

Besides being able to operate a WSN for long periods of time, DISSense must provide for reliable data delivery. We thus also evaluate its performance in terms of data delivery ratio, or DDR, which we define as the ratio between the number of data packets injected by the nodes into the network and the number of those successfully collected at the sink. Last but not least we also consider the number of duplicate packets that eventually reach the sink as a further metric to describe the performance of DISSense[1]. In particular, this number must be kept low in order to increase DISSense's efficiency.

[1] Duplicated packets are generated during the collection phase due to possible false negative acknowledgements at the link-layer level.

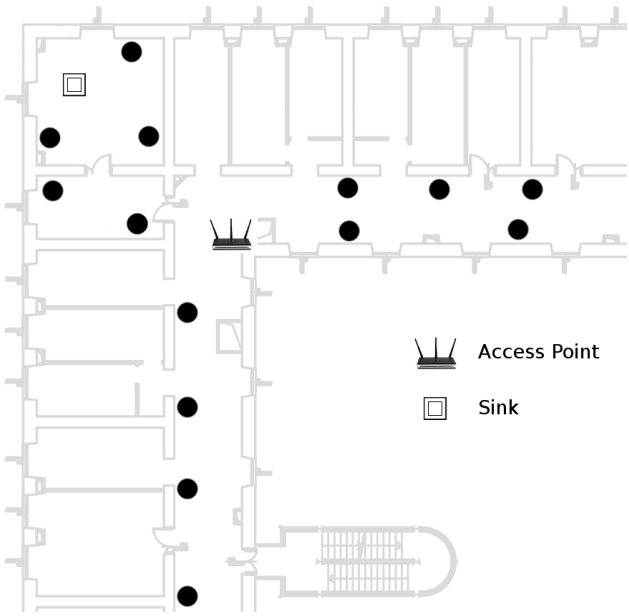


Figure 3.4: WSN Testbed (indoor)

We would like to point out that we do not report results about DISSense performance in terms of latency (i.e., the time elapsed between packet generation and its arrival to the sink), since packet latency in DISSense is upper-bounded by the length of the active phase, in all our experiments this length has never exceeded 4.5 seconds, which represents the default active phase interval at startup.

3.4.2 Testbed

Setup We run DISSense on a testbed of 15 nodes deployed on the first floor of an office building, as depicted in figure 3.4. The presence of walls, a WiFi access point and intense wireless communication activity contributed in creating an unreliable, unpredictable, and thus realistic wireless communication channel. We run DISSense on this testbed for 2 months keeping the sampling period to 1 minute for the first 31 days to 15 minutes for next 10 days, and to 1 hour for the last 20 days. In the following, we indicate with DISSense-x the instance of DISSense having a sampling period of x minutes.

Results Figure 3.5 shows how the length of the RI, DCI, and GT intervals, as well as the parameter σ_{sk}, varies over time (i.e., each time the Adaptive

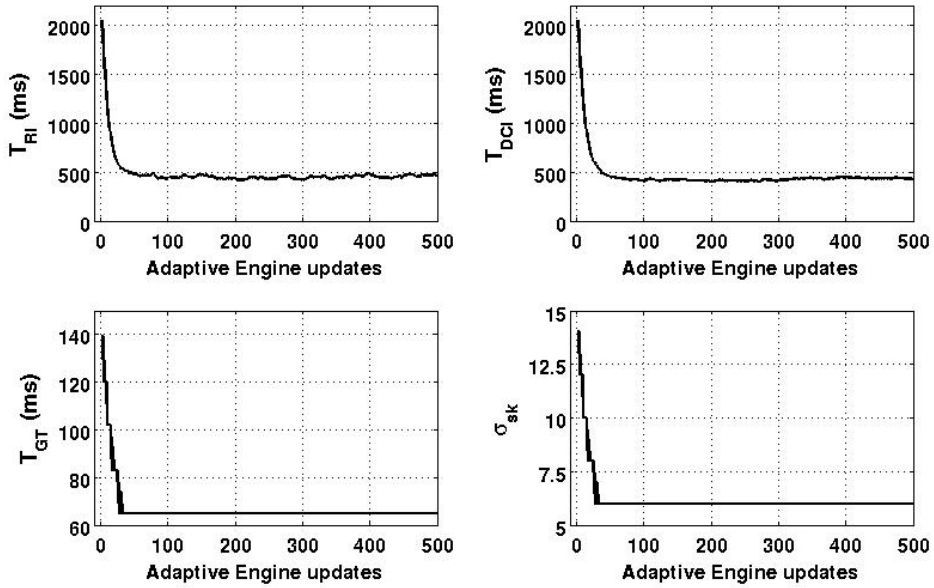


Figure 3.5: DISSense-1 Adaptive Engine Parameters (Testbed)

Engine performs an update) when running DISSense-1. The default startup value of both T_{RI} and T_{DCI} is 2 seconds. Using equations (3.6) and (3.3), the default values of σ_{sk} and T_{GT} result being 14 and 139ms, respectively. During each RI interval, the sink's Adaptive Engine recomputes the values of these parameters thus enabling DISSense adaptive behavior. Indeed, figure 3.5 shows that the length of both the RI and DCI intervals quickly converges to values included in the ranges $[450ms, 650ms]$ and $[400ms, 500ms]$, respectively. It is interesting to point out that since the sampling period is fixed and since equation (3.3), which controls the evolution of T_{GT}, depends on the values of the sampling period and skip parameter only, the evolution of T_{GT} follows that of σ_{sk} .

Figure 3.6 reports the same data as in figure 3.5 but for DISSense-15. In this experiment, the sink has been reset to make the nodes go out of synchronization. Due to this reset, the sink loads the default values for the schedule and shares them with the nodes. Figure 3.6 shows that the Adaptive Engine is able to quickly recompute new optimal values for the key protocol parameters.

Table 3.1 summarizes the performance of DISSense. As expected, increasing the sampling interval makes the duty cycle shrink significantly. In particular, it decreases from 1.09% for DISSense-1 to 0.15% for DISSense-60. This is due to the fact that, although a longer sampling period induces an increase in

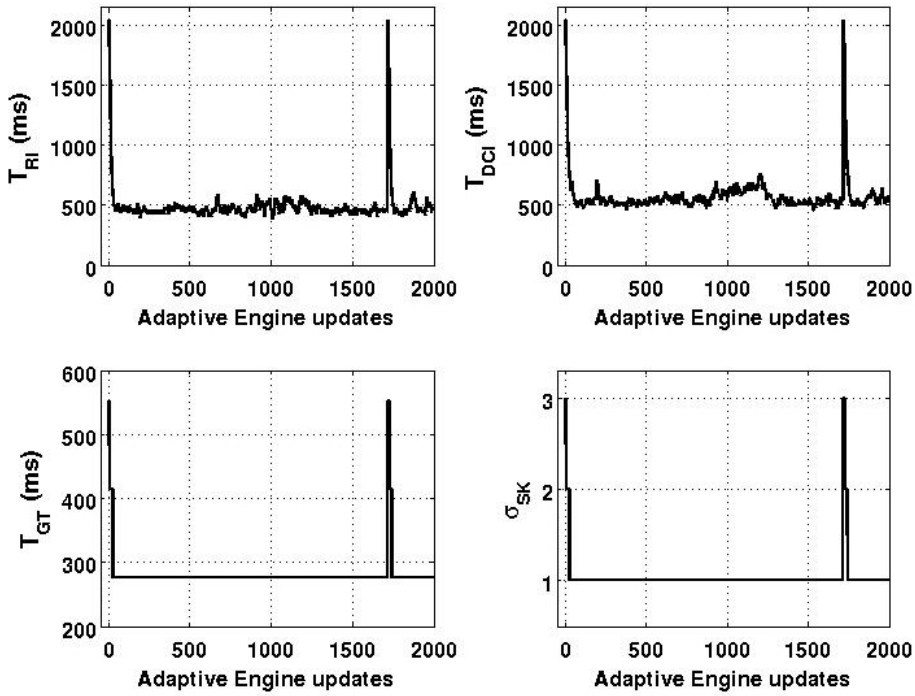


Figure 3.6: DISSense-15 Adaptive Engine Parameters (sink reset)

	Sampling Period (min.)		
	1	15	60
Duty Cycle (%)	1.09	0.22	0.15
D.D.R. (%)	97.8	98.6	98.9
Duplicated Packets (%)	0.16	0.24	0.03

Table 3.1: DISSense performance (Testbed)

the length of the guard interval T_{GTI} , the period during which the nodes are inactive has a proportionally higher increase. This results in an overall lower duty cycle. Table 3.1 also shows that DISSense's average delivery ratio oscillates around 98% irrespectively of the sampling period. Our analysis of the traces stored by the nodes during the execution of DISSense shows that most of the packet losses are related to the occurrence of routing loops. Although DISSense can rely on CTP's loop detection mechanism, the time necessary to re-establish a valid route is typically larger than the length of the active phase. Thus, looping packets cannot reach the sink before the network goes back to sleep. Finally, table 3.1 shows that the number of duplicated packets reaching the sink is negligible with respect to the total data traffic. This is due to the fact that DISSense relies on CTP's effective duplicates suppression mechanism.

3.4.3 Simulation study

Setup We run DISSense-1, DISSense-2 and DISSense-5 within the TOSSIM simulation environment. To this end, we generated networks having 10, 20, 30, 40 and 50 nodes (excluding the sink) using TOSSIM's network generator tool with parameters set as summarized in table 3.2. For each network size, we generated 20 different topologies. To reproduce the vagaries of the wireless channel we resort to the *casino-lab* noise model [13]. Furthermore, for larger networks, the queue size of CTP has been increased to 40 packets to avoid packet drops for full buffer.

Results Figure 3.7 shows the value of the duty cycle of DISSense as the number of nodes in the network increases. The duty cycle increases with the number of nodes but decreases as the sampling period increases. In particular, the duty cycle of DISSense-1 increases from 0.8% for 10 nodes to 3.3% for 50 nodes while for DISSense-2 it increases from 0.5% to 1.75% and for DISSense-5 from 0.35% to 0.8%. The duty cycle further decreases when the sampling

Path Loss Exponent	4.7
Shadowing Standard Dev.	3.2
Reference distance d_0 (m)	1
Path Loss (d_0) (dBm)	25.6
Noise Floor (dBm)	-105
White Gaussian Noise	4

Table 3.2: Network Generator Parameters

period progressively grows to 60 minutes. From these results, we observe the major strength and weakness of DISSense. As described in section 3.4.2, DISSense is very efficient for relatively long sampling periods (larger than 5 minutes), for which it can achieve duty cycles as low as 0.15%. However, the performance of DISSense degrades as the sampling period decreases and the network size increases. This is due to the fact that the CSMA/CA-based design of DISSense makes the time needed for channel contention increase with the density of the network. This, in turn, makes the length of DISSense synchronization and data collection intervals, and, thus, the length of the active phase, increase, causing the duty cycle to increase too. While for high sampling periods this effect is mitigated by the largely predominant ULPM interval, when the sampling period is low, the effect becomes less negligible. On the other hand, performance in terms of data delivery ratio are not affected by changes in sampling period or network size. In fact, as figure 3.8 shows, DISSense achieves a DDR higher than 99% irrespectively of the sampling period or network size. DISSense achieves this good performance by combining CTP's inherent reliability and the ability of the Adaptive Engine to estimate the appropriate length of the RI, DCI, and GT intervals.

3.4.4 Multi-sink support

To cope with DISSense's unsatisfactory performance for large network sizes, it is possible to resort to a multi-sink approach. If several sinks are defined, CTP will naturally construct several collection trees, each having one of the sinks as its root. This allows to reduce the overall diameter and density of the network and, thus, to reduce the length of the active phase. To this end, though, we have to assume that the sinks are all synchronized (through the NtpManager). Multi-sink DISSense works as the single-sink version except for the presence of multiple schedules, one for each sink. In fact, each sink shares

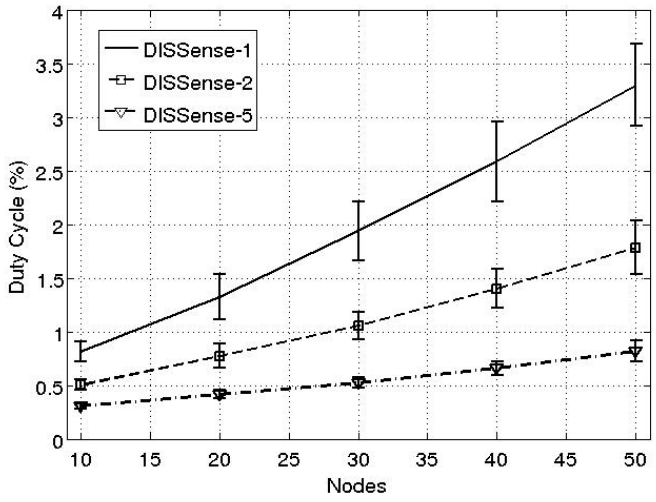


Figure 3.7: Duty Cycle with different sampling intervals

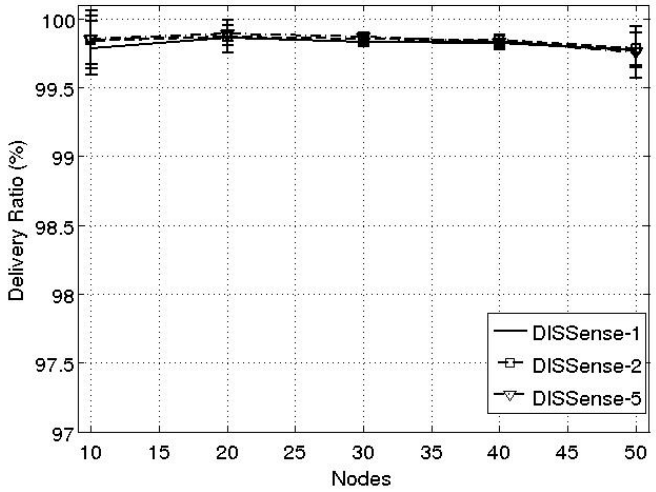


Figure 3.8: Delivery Ratio with different sampling intervals

its schedule with the nodes belonging to its sub-tree and adapts it following the same principles described in section 3.3.2 and 3.3.4. This means that the schedules will depend on the number of nodes participating to each sub-tree and, thus, we expect each schedule to have different values for T_{GTI} , T_{DCI} , T_{RI} and σ_{sk} . As a consequence, a node changing its sub-tree will have a different schedule and, with high probability, will go out of synchronization. This event typically affects nodes halfway between two sinks. One possible solution consists in enhancing the NtpManager functionality by adding a schedule sharing service that enables sinks to share the same schedule between them. In particular, a conservative choice is represented by the selection of the largest schedule within those computed by each sink. In this way, a node changing its sub-tree will still share the same schedule and will be easily integrated in the new sub-tree. This approach requires permanent communication link between the sinks of DISSense. Because we assume that each sink is connected to a gateway, from which it retrieves the Network Time Protocol (NTP), we can safely assume that this requirement is met. Another approach consists in leaving the sinks to adopt different schedules and let the nodes going in out of synchronization to retrieve the correct schedule through their LplManager. Assuming that the nodes will not change their sub-tree too frequently, this approach doesn't require any specific action. The choice within the two solutions will depend on application requirements in terms of reliability, energy efficiency and availability.

Figures 3.9 and 3.10 represent the duty cycle and delivery ratio reached by two multi-sink version of DISSense compared to the single-sink one. In particular we have selected one topology for each network size of the previous experiments and compared DISSense-1 to 2-DISSense-1 and 3-DISSense-1, where the prefix number corresponds to the number of sinks. No schedule sharing services has been adopted, thus, the nodes changing their sub-tree retrieve the new schedule through the LplManager. As figure 3.9 shows, the duty cycle is greatly reduced for multi-sink DISSense. In particular, for the 50 nodes topology, the duty cycle decreases from 3.8% of DISSense-1 to 1.75% of 2-DISSense-1 and 1.1% of 3-DISSense-1. Figure 3.10 shows that the DDR is not affected by the multi-sink version of DISSense since it remains always above 99%. This result demonstrates that the number of *out of synchronization* events is low.

3.4.5 Comparison to Dozer and Koala

We finally provide a qualitative comparison between DISSense and its closest competitors: Dozer [47] and Koala [98]. In particular, we compare the performance of the three protocols in terms of data delivery ratio, latency, duty

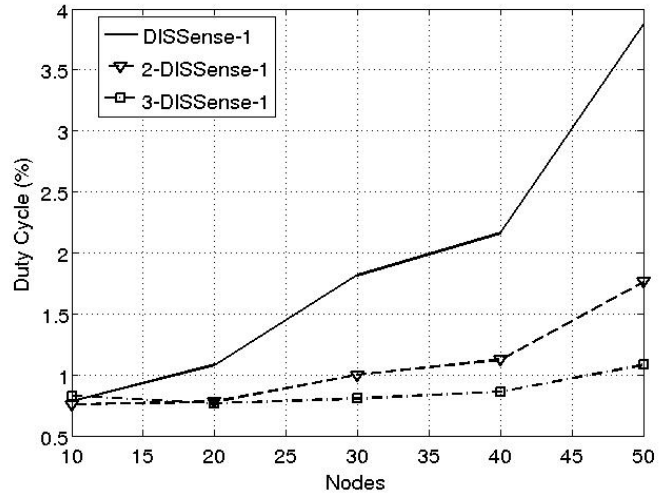


Figure 3.9: Duty Cycle with Multi-sink (topology # 1)

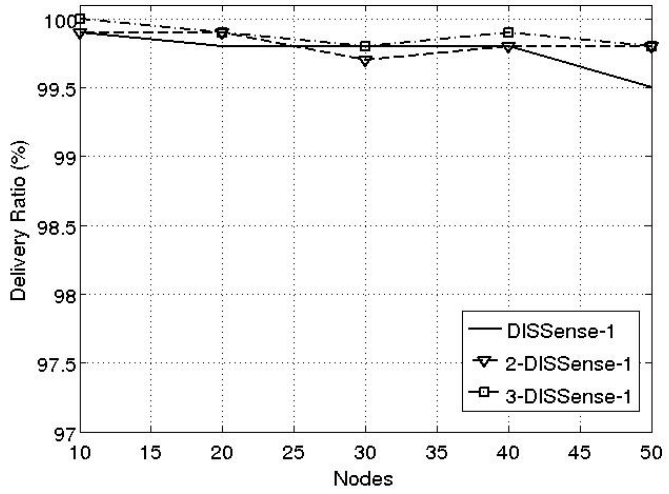


Figure 3.10: Delivery Ratio with Multi-sink (topology # 1)

cycle, adaptability, and openness. Table 3.3 summarizes our findings.

Data Delivery Ratio: DISSense, Dozer, and Koala all show high performance in terms of data delivery ratio. In particular, the end-to-end acknowledgement mechanism of Koala enables the protocol to achieve a DDR of 99.99%. Instead, both Dozer and DISSense make use of link-layer acknowledgements thereby implicitly accepting some packet losses. In particular, DISSense automatically drops packets not delivered during the past active phase while Dozer overwrites packets matching the same origin inside forwarding queues. Nonetheless, both protocols achieve data delivery ratios of 98-99%.

Latency: DISSense has an overall lower latency than Dozer and Koala. As mentioned in 3.4.1, latency in DISSense is upper-bounded by the length of the active period, which is typically lower than 5 seconds. Packet latency in Dozer (in the worst-case scenario) is equal to the number of hops in the collection tree times the length of the period of the TDMA schedule, the latter being fixed and equal to 30 seconds. Additionally, if an acknowledgement gets lost Dozer makes the transmitting node to stop and wait for the next TDMA round before retransmitting. This mechanism additionally increases the latency in Dozer. Koala has an overall higher latency. To keep its duty cycle short, Koala buffers packets and limits the number of time it needs to wake-up the network and perform a bulk download of the collected the data. The minimum buffer size is 32 KB [98]. Considering a 2-minutes sampling period and 35 bytes of payload per packet (as done in [47]), a 32 KB buffer would get filled in approximately 1.3 days. This value clearly increases as the length of the sampling period increases.

Duty Cycle: Dozer has an overall lower duty cycle than Koala and DISSense. As reported in [47], Dozer can achieve an average duty cycle of 0.168% on a network of 40 nodes and a 2 minutes sampling period. The actual duty cycle of each node varies depending on the role the node has in the collection tree. A node with many children needs to assign accordingly many communication slots to its children, thus, incurring in a high duty cycle. In the setup described in [47] leaf nodes have a duty cycle of 0.07% while nodes with many children achieve 0.32%. If the network topology does not change frequently enough, this difference may induce significantly uneven lifetimes on different nodes of the network. Koala manages to keep the duty cycle below 1% but, as discussed above, at the cost of high data latency. In particular, waking-up the network too often (e.g., each hour) would produce a sensible increase in Koala's duty cycle due to the high energy cost of network wake-up, route computation and bulk data download that the protocol requires. DISSense's duty cycle depends on the network size and on the sampling period. As discussed previously, DISSense's performance in terms of duty cycle is comparable to that of Dozer for small network sizes (<20 nodes) or high sampling period (>5 min.). Also, DISSense induces the same duty cycle to all nodes, which

	DISSense	Dozer	Koala
Duty Cycle	0.1% - 4%	0.168%	0.1% - 1%
Data Delivery Ratio	98-99%	98-99%	>99.99%
Latency	<5s	minutes	days
Platform Dependent	no	yes	no
Open Source	yes	no	yes
Adaptability	yes	no	no

Table 3.3: Protocols qualitative comparison

translates in homogeneous power consumption and, hence, a predictable overall network lifetime.

Adaptability: Through its Adaptive Engine DISSense can determine the optimal schedule for the network irrespectively of its size, topology, or state of the wireless channel. In particular, the Adaptive Engine autonomously collects statistics while the protocols runs and thereby determines the key protocol parameters. Instead, both Koala and Dozer depend on the a priori specification of crucial parameters. For Koala these include the probe interval for network wake-up and the buffer size, while Dozer relies on knowledge of the round period, parents update interval, overhearing phase length and frequency, and slot length.

Openness: Last but not least, DISSense and Koala are platform independent while, due to commercial agreements, Dozer is implemented on the TinyNode platform only. Also, Dozer is closed-source while DISSense and Koala are open-source protocols with publicly available implementations.

3.5 Conclusion

In this chapter, we described the design and implementation of DISSense, an adaptive, low-power communication protocol for WSNs-based periodical environmental monitoring applications. DISSense is easy to setup thanks to its adaptive engine that automatically updates the protocol parameters in order to minimize its power consumption. We tested DISSense on both a testbed and the TOSSIM WSN simulator. Our experimental results show that DISSense can guarantee for high data delivery and, thanks to its power-efficiency, it is able to operate a Tmote Sky-based WSN for several years. As for future work, we expect to test the performance of DISSense on wider testbeds

such as *Motelab* [7]. Additionally, in [108] we presented a new metric, dubbed Expected Network Delivery (END), that quantifies the delivery performance that a collection protocol can be expected to achieve given a network topology. We expect to provide a comparison between DISSense and other existing solutions such as CTP with BoX-MAC-2, Koala or Dozer in terms of duty cycle, DDR and latency with different network topologies classified following the END index.

Chapter 4

ARS over DISSense

4.1 CTP and activation probability

In [50] we evaluated the Collection Tree Protocol (CTP) [65] on Castalia framework for OMNeT++ Simulation Environment [8]. Castalia is a Wireless Sensor Network framework that features advanced channel and radio models, a MAC protocol with large number of tunable parameters and a highly flexible model for simulating physical processes [1, 42]. The goal of [50] was to investigate the interplay between specific collection services and application-level algorithms. The experiments were made by running CTP on 50 different topologies of 100 nodes deployed uniformly at random in a 250x250 meters field. Each run was further subdivided into 50 rounds where, during each round, the nodes of the network were sending one packet to the sink with activation probability p. Additionally, to simulate the behavior of a duty cycled protocol, each round was subdivided into an active phase, during which the CTP protocol was gathering the data, and a passive phase where the nodes were sleeping. Consequently, the data packets not gathered during the active phase were discarded.

Figure 4.1 represents one of the main results of the experiments made in [50]. The figure shows the Data Delivery Ratio (DDR), i.e., the ratio between the number of data packets received by the sink and the number of data packets actually sent by the nodes, for values of the activation probability p of 0.5 and 1. The figure shows how for several topologies, the DDR is sensibly affected by the activation probability. In particular, for $p = 0.5$ the map is homogeneously dark-red colored, indicating a DDR near 100%, while for $p = 1$ several horizontal lines becomes lighter as a consequence of the DDR decrease. The performance degradation is related to the increasing number of sensing

nodes that consequently increases the number of packets. The packet increase affects the limited size of CTP queues that, when full, generate several packet drops. Additional drops are caused by the increasing time required by CTP to gather the data that, in several circumstances, exceeds the scheduled active time.

In section 3.4.3, we solved the first issue by increasing the CTP queue length. However, the increasing gathering time related to the second issue was producing an increase in the DISSense duty cycle. In particular we find out how the CSMA/CA mechanism, on which DISSense is based, poorly scales when the number of nodes increases. As a result, the Adaptive Engine of DISSense was increasing the active time, thus, producing an increase of the duty cycle. A turnaround to this problem has already been faced in chapter 3 by increasing the number of sinks so as to reduce the number of nodes for each sub-tree. But, based on the results of the previous paragraph, and recalling that DISSense is CTP based, we want to study, in the remainder of this chapter, how a sensor selection mechanism such as ARS, which naturally reduces the number of active nodes, could provide benefits for the DISSense protocol.

4.2 Integration

In section 2.3 we outlined that ARS can be divided in four main phases: *dissemination*, *discovery*, *computation of p* and *sensing and update*. In section 2.3.4 we described the third phase rather, in this chapter, we provide the implementation of the remainders.

4.2.1 Dissemination

During the dissemination phase, ARS distributes the value of the sensing range R_s to all nodes within the network. By embedding ARS in the DISSense implementation, the R_s value can be disseminated using the Implicit Backward Channel (IBC) presented in section 3.3.2. We recall that the IBC is active during each Resynchronization Interval (RI) and guarantees that, at the end of the RI, each node is in one of the following states:

- The node belongs to the Collection tree (i.e., it has selected a parent) and shares the updated values carried by the IBC.
- The node does not belong to the Collection tree and is out of synchronization.

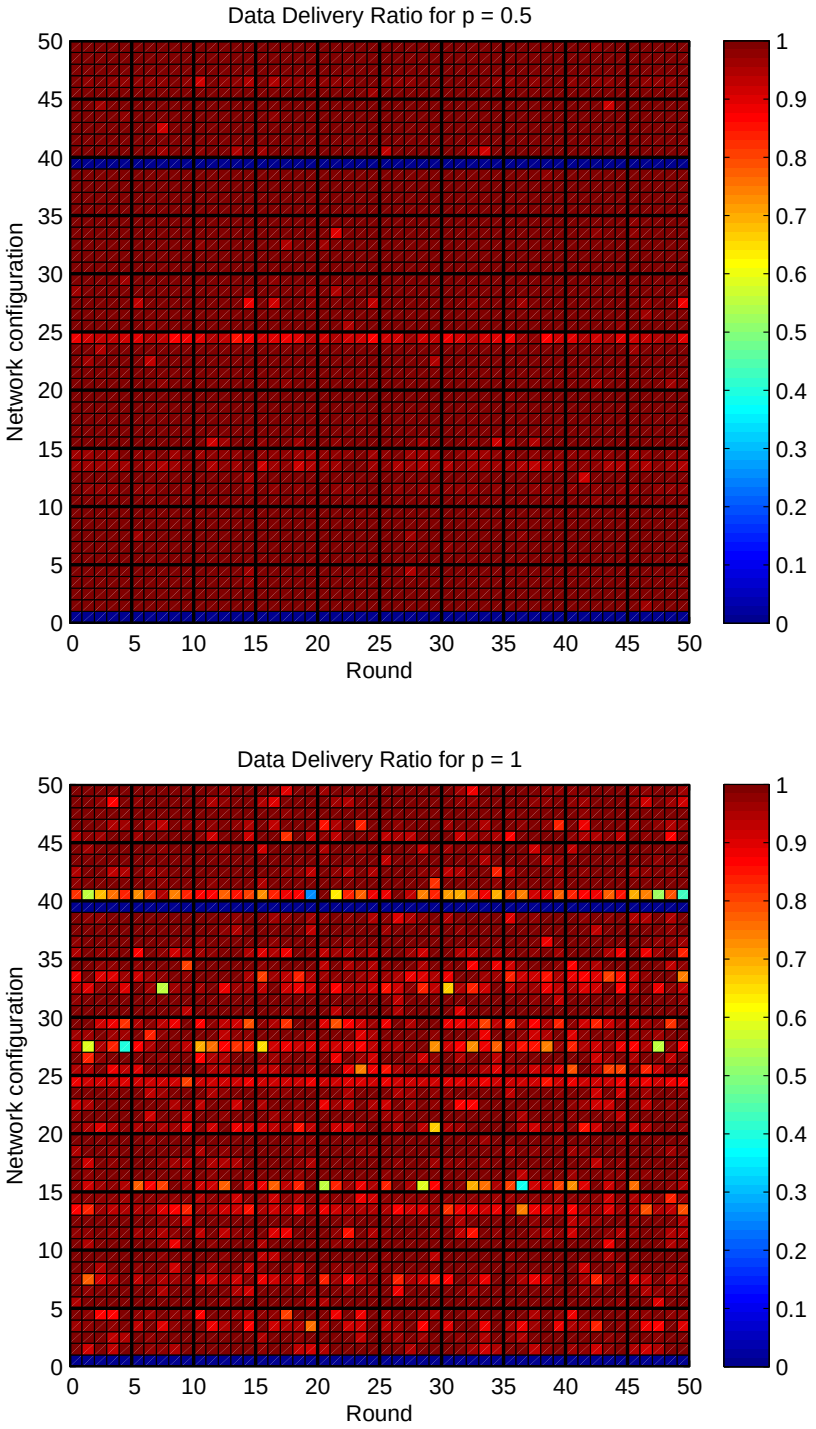


Figure 4.1: DDR for all the network configurations and rounds, for $p = 0.5$ and $p = 1$.

As a result, all the nodes running DISSense and participating to the collection tree have the sensing range already set before each gathering phase.

4.2.2 Discovery

Recalling section 2.3.4, the activation probability depends on the number of neighbors and their position. To retrieve these information, ARS can use the neighbor table of DISSense, which is built upon the reception of beacons from the node's neighborhood. To this aim, it might be necessary to increase the neighbor table size limit depending on the network density. Based on our experience, 25 entries represent a good trade-off between application scenarios and memory footprint. In fact, in most cases it is unlikely to have a network density higher than 25 nodes in a single hop range. Additionally, neighbor table and DISSense beacons need to be extended in order to store the $< x, y >$ coordinates of the neighbors.

4.2.3 Sensing and update

Once each node n_i holds its own local value of the probability of activation p_i computed following the procedure described in 2.3.4, the node executes the RSS algorithm of section 2.2. When active, the node sends the data packet following the Send interface implemented by the Manager of the DISSense protocol (section 3.2.4). By this way, the data of all the active nodes of the network is gathered by the sink which, in turn can run the ACT algorithm [91, Chapter 6] to refine the spatial resolution Δ_s and consequently update the sensing range R_s. Again, the updated R_s value is disseminated from the sink to the network following the IBC channel abstraction of DISSense. Note that, whenever a multi-sink DISSense is adopted, the ARS protocol requires the gathered data to be processed on a central gateway in order to correctly reconstruct the signal and refine the sensing range.

4.3 Experiments

4.3.1 Setup

We implemented ARS over the DISSense implementation of section 3.3. We also used the TOSSIM simulator for performance evaluation and the same 20 generated 50-nodes topologies of section 3.4.3. The monitored area is a square of 75 x 75 meters on which the 50 nodes and the sink have been deployed uniformly at random. The radio range is 30 meters and the casino-lab noise

model has been applied. As an example, figure 4.2 represents the first of the 50-nodes topologies. Based on the link-gain, this topology has an average number of hops equal to 2.82 and a maximum number of 4. However, considering the casino-lab noise model and the signal-strength-based radio model used in TOSSIM, these values represent a lower bound, while the actual values are sensibly higher. In the following of this section, we will refer to ARS-DISSense-1 when DISSense-1 is run with ARS activated.

4.3.2 Results

We first evaluated, over a single topology represented in figure 4.2, the Duty Cycle and the Data Delivery Ratio of DISSense-1 compared to the ARS-DISSense-1 for different values of the sensing range R_s. As figure 4.3 shows, the Duty Cycle quickly drops from nearly 4% of DISSense-1 (labeled as No Ars) to 3.23% of ARS-DISSense-1 with $R_s = 15$ meters. The Duty Cycle further decreases when the sensing range increases (2.85% for $R_s = 20$ meters and 2.65% for $R_s = 25$ meters). As opposed to DutyCycle, the Data Delivery Ratio holds steadily above 99.5%, thus, confirming the benefits of using a sensor selection approach, as ARS, over the DISSense protocol. It is important to note that arbitrarily increasing the sensing range produces a decrease of the activation probability and, as a consequence, a decrease of the duty cycle. However, a lower activation probability also generates a higher signal reconstruction error. A discussion on the existing trade-off between the quality of service and the energy efficiency of the protocol certainly represents an important research topic but goes beyond the scope of this thesis.

Figure 4.4 shows per-node ARS activation probability related to the evaluated topology with sensing range $R_s = 20$ meters. As expected, based on the neighborhood density, distance and polarization, the activation probability varies from a minimum of 0.5 for node 19 to a maximum of 1 for edge-nodes and nodes with no useful neighbors in at least one set S_{ik} (e.g., node 15).

From the previous experiment, we selected the sensing range $R_s = 20$ meters to compare the duty cycle of ARS-DISSense-1 and DISSense-1 over the whole set of 20 generated 50-nodes topologies. Figure 4.5 shows a constant improvement of ARS-DISSense-1 in respect of DISSense-1. The duty cycle of ARS-DISSense-1 ranges from a minimum of 2.09% for topology n.12 to 3.13% for topology n.3 while DISSense-1 lies between the 2.72% - 4% interval.

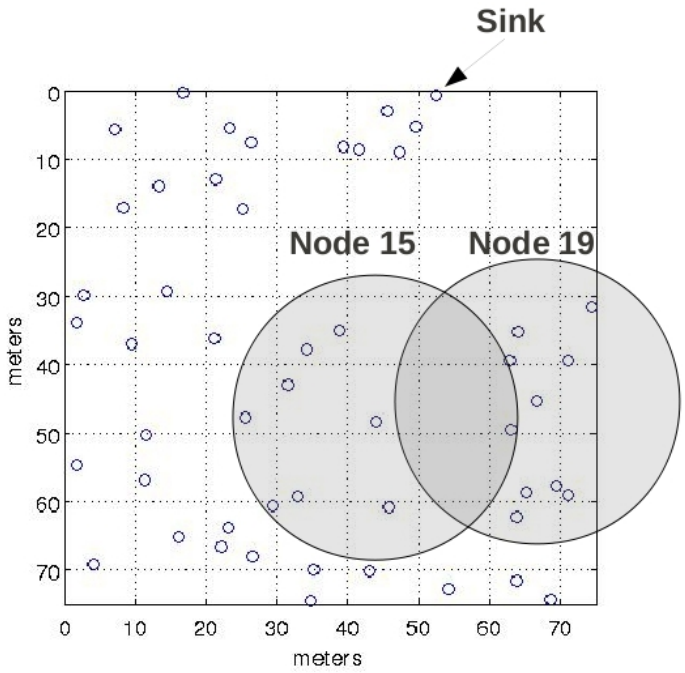


Figure 4.2: Topology #1 (50 nodes + sink) with sensing range $R_s = 20$ meters drawn for nodes 15 and 19

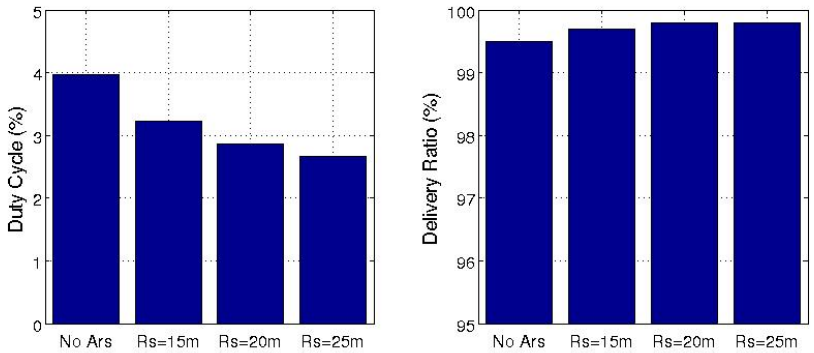


Figure 4.3: Ars-DISSense-1 performance for different sensing ranges R_s

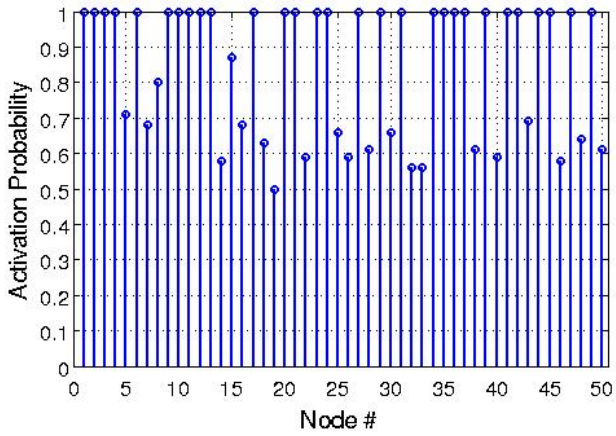


Figure 4.4: Per-node activation probability for ARS-DISSense-1 running on the first 50-nodes topology and sensing range $R_s = 20$ meters

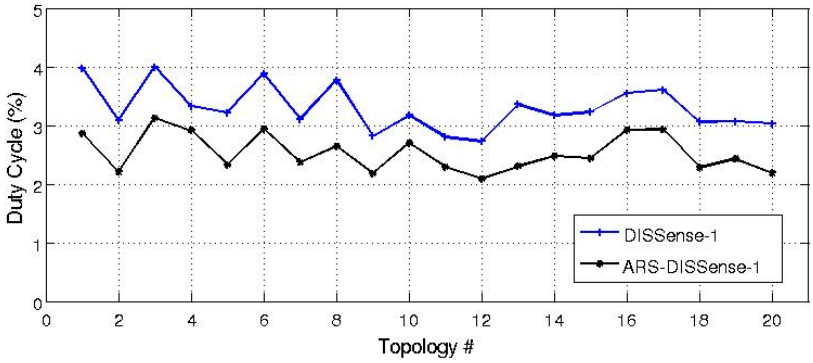


Figure 4.5: DISSense-1 and Ars-DISSense-1 duty cycle comparison for different topologies and sensing range $R_s = 20$ meters

4.4 Conclusion

In this chapter we observed how the adaptive sensor selection scheme presented in chapter 2 greatly improves the duty cycle performance of the DISSense protocol discussed in chapter 3. The reason relies in the lower traffic that, thanks to the specific design and the adaptive behavior of DISSense, reduces the active time scheduled for data gathering and consequently, the protocol's duty cycle.

Despite the good performance, the experiments made in the previous section do not take into account the signal reconstruction error and the covered region of interest. The reason basically resides in a lack of the TOSSIM simulation environment that we expect to overcome implementing DISSense over the Castalia framework for OMNeT++. The Castalia framework provides an advanced physical process that has already been used to evaluate ARS in chapter 2.

Part II

Information Gathering in Passive Pervasive Systems

Chapter 5

Passive pervasive systems

5.1 Technology

Nowadays, the pervasive deployment of tiny devices with minimum storage and limited or no computational capabilities appears a realistic perspective; one major obstacle is the strict energy constraints of battery-powered devices. We refer to a class of passive devices (i.e., not powered by batteries) that has emerged in the last decade, the most prominent examples being RfID and NFC tags. An RfID tag is an object that can be applied to or incorporated into a product, animal, or person for the purpose of identification and tracking using radio waves. Passive RfID tags have no internal power supply and draw power from the radio waves emitted by the reader. However, passive tags have minimal or no computational power, the memory is limited to only few KBs and the distance between the reader and the tag is at most few meters (ISO 18000-6), but it is typically only few centimeters (ISO 14443). Near Field Communication or NFC, is a short-range high frequency wireless communication technology which enables the exchange of data between devices within short distance aimed at usage in mobile phones. An NFC device can communicate with both existing ISO 14443 smartcards and readers, and it is thus compatible with existing RfID infrastructures, as well as with other NFC devices. The limited costs and the pervasiveness of these devices are paving the way for new pervasive solutions: mobile ticketing in public transport, mobile payment, smart shopping and social applications.

5.2 Communication abstraction

Wireless Sensor Networks discussed in the first part of the thesis were characterized by a multi-hop communication infrastructure and a many-to-one traffic pattern gathering the sensed data to a central collection unit dubbed sink. Additionally, the transmission of a data packet was node-initiated in the sense that each node was able to actively transmit a packet following the schedule of the underlying MAC protocol. Despite the restricted size of each single data packet, WSNs are able, over time, to collect large quantities of data and deliver them to some external entity.

In Passive Pervasive Systems, the communication abstraction radically changes. Since passive tags are powered by radio waves emitted by active readers, no communication can take place until a reader enters within the range of the tag and initiates a communication process (from now on, we will refer to this action as a *visit*). As a consequence, the communication cannot be initiated by the tag since it is the reader that decides whether to begin a communication. Moreover, the multi-hop traffic pattern adopted in WSNs is replaced by a simpler single-hop interaction represented by the active reader that reads, updates or stores data on the tag it is visiting. Nonetheless, we can still refer to multi-hop traffic if we consider the active reader visiting different tags, thus, potentially carrying the data from one tag to another. By the way, we cannot assume passive tags being connected through paths, as opposed to the WSN networked infrastructure, since the data exchange does not follow any kind of routing policy rather, it depends on the movement pattern of active readers that occasionally meets both, the source and the destination of a communication. We refer to this mechanism as *opportunistic and delay-tolerant networking*. The consequence of this communication abstraction is that the data is not uniformly spread following the unpredictable visit pattern of active readers. Additionally, the limited memory available (few kilobytes) limits the storage capacity of passive tags. Since the data is rarely exchanged in opportunistic networking, the memory of each tag quickly fills up. As a consequence, those devices will need to store aggregated information in order to increase their capacity. As a result, the aggregated value of each tag will potentially encode part of the active reader's history (i.e., the navigation pattern of the reader) in some form of compact representation.

5.3 Information gathering

It is clear that energy efficiency represents a secondary aspect in Passive Pervasive Systems since, in this context, power supply only holds on few active devices (the active readers). Thus, issues and solutions faced in the first part of

the thesis are no longer valid. Information gathering in such systems addresses a more general aspect related on how, the partial and non uniform distribution of the information can be processed to deliver some kind of services. The opportunistic networking communication abstraction does not allow the use of a centralized approach, thus, the information must be computed locally during each user's visit. Consequently, the emerging applications mentioned in the previous section will have to be developed following a fully-decentralized design in order to properly run on such systems. Providing fully decentralized services for mobile ticketing, mobile payment, smart shopping or social applications, based on the compact representation of the information in pervasive systems is an emerging research area and, as opposed to centralized approaches, only few examples are present in literature.

A pheromone-based object-tracking system adopting RfID tags has been presented in [87]. The system uses a stigmergy approach that leaves digital pheromones on tags dispersed in the environment. In particular, when moving, each object spreads digital pheromones, represented by a unique ID and a hop counter, all over the environment. Followers only need to search for a digital pheromone and follow subsequent digital traces with an increasing hop counter values to reach the tracked object. An application aiming at facilitating the finding of everyday objects has been developed based on this technique. In [14] a Decentralized Simultaneous Localization And Mapping (DSLAM) for pedestrians in the context of Urban Search And Rescue (USAR) based on RfID tags is presented. Simultaneous localization and mapping avoids pedestrians (e.g., firemen) to make loops while searching for victims. The decentralized RfID-based implementation relies on an (indoor) deployment of RfID tags that enables a fireman to update its knowledge of the graph by reading the data stored on the visited tag and, at the same time, update the tag with data collected so far by the fireman. The knowledge acquired by reading the data stored on the tag allows a fireman to learn a subgraph larger than the one represented by its own trajectory and potentially detect loops in the broader reconstructed subgraph.

In the next two chapters we present our contributions in developing fully decentralized applications for Passive Pervasive Systems. In chapter 6 we present a fully decentralized RfID-based recommendation system tailored for smart posters that is able to suggest items of potential interest on the basis of the succinct information obtained interacting with the smart poster. The system defines a probabilistic model of user behavior and uses statistics computed over past user transactions to estimate parameters of the model. The output of the model is a set of items that are most likely to meet a user's interests. The system is fully decentralized and easily matches the low-computational, low-memory and low-bandwidth requirements of Passive Pervasive Systems. In chapter 7 we present an SMS-based recommendation system able to pro-

vide social recommendations, e.g., new friendships, based on the exchange of succinct representation of the list of contacts dubbed *sketches* encoded in the residual space of Short Messages. Despite Short Messages rely on the mobile phone communication system, which sharply differs from Passive Pervasive Systems described in this chapter, we will demonstrate how the application can be easily applied to our reference scenario.

Chapter 6

A fully decentralized RfID-based recommendation system

6.1 Introduction

In this chapter we consider fully decentralized collaborative filtering strategies for item recommendation in passive pervasive systems. For example, the NFC consortium proposes smart posters for shopping by tagging items of interest posted in billboards, or any other form of advertising, with a passive tag, from which a user can exchange data by touching it with her NFC-enabled handset. The user can buy the item associated with the tag and receive information on the item or even recommendations on other items of potential interest on the basis of the succinct information obtained interacting with smart posters. The distinguishing features of decentralized computation in which low capability devices observe a local stream of events and have to maintain summary information about overall system behavior, while obeying stringent memory, computational and communication constraints. In this scenario, distributed, local algorithms appear a natural choice to address computational, communication and storage restrictions of the scenarios outlined above.

The importance of recommender systems has been widely recognized in e-commerce systems as a tool to suggest products to customers, providing relevant information in shopping (e.g., Amazon, eBay). In order to recommend

new items three main approaches have been proposed: collaborative filtering, content-based filtering and hybrid methods [40]. We consider Collaborative Filtering (CF for brevity) that classifies users and products in terms of their past interactions.

We assume that items of interest are advertised by smart tags (e.g., RfIDs or NFCs) distributed in an area, e.g., a city. In the sequel we will use the terms *smart tag* and *item* interchangeably and the term *smart reader* to denote a smart tag enabled reader device (e.g., an NFC enabled smart phone). Each user is characterized by a (unknown) ranking of items, describing her preferences; a smart reader stores her history (e.g., the set of items previously visited by the user) during the visit and has some computational capability. When a user interacts with item i the user's smart reader reads details on the item. We also assume that i stores a suitable summary of the histories of users that visited i in the past: smart readers interact with smart tags, by *transparently* (to the user) reading and updating the information stored by the latter. We stress that we do not assume any transmission capabilities between smart tags, which are assumed to be passive devices. When item i is visited by user j, her smart reader can recommend a new item (or a set of items) of potential interest to j, using current summary at i and j's history. The recommendation of an item is good if the proposed item is likely to meet j's preferences. The above scenario is technologically realistic and it is closely related to architectures proposed for smart shopping carts [74] and smart shelves [56]. We remark that it also complies with privacy issues, since only aggregated information is disclosed from which it is not possible to infer private information concerning specific users.

Our goal is to show that, even under the stringent constraints outlined above, simple heuristics allow to effectively profile users and provide good recommendations in the scenario described above. The recommendation algorithm we consider is simple: upon visiting item i, user j is recommended one item (or a subset of items) scoring highest among those not yet visited by j. The core of the whole problem is defining scores that i) can be efficiently estimated and updated locally by the smart reader upon visiting new items and ii) that effectively reflect the users' unknown preferences. Note that scores and rankings can statistically depend on users' visit patterns in complex ways. We tackle this issue by defining suitable models of user behavior.

On the theoretical side, we provide asymptotically tight lower and upper bounds on the number of examples required by the recommendation algorithm to compute good estimations of item scores and thus provide good recommendations. While theoretical analysis gives bounds that might be unfeasible in practical applications, our experiments compare the performance of our algorithms with that of a centralized, state-of-art recommendation algorithm that knows the overall system history. The tests are based on both synthetic data

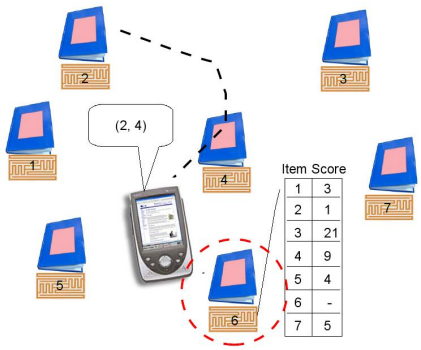


Figure 6.1: Exemplifying scenario: a smart library

Scenario: every item has an associated smart tag with a unique integer ID. The user is visiting item 6 after visiting items 2 and 4. The smart reader uses aggregate statistics concerning item 6 and user's history to assign scores to items other than 6 and to provide a recommendation (e.g., item 1 has score 3, while item 3 has score 21 and is thus the top item in the summary).

and real data sets provided by Netflix, a popular on-line DVD rental service; they show that the performance of our algorithm is very close to that of the centralized one, in terms of standard metrics normally adopted to assess the performance of Collaborative Filtering algorithms.

We observe that our algorithms can be considered as an application of stigmergy to recommendation algorithms. Stigmergy is a form of self-organization where traces left in the environment by the action of agents trigger the execution of subsequent actions, by the same or a different agent, thus, allowing spontaneous and indirect coordination between agents [32]. Stigmergy has been exploited in tracking objects tagged by RFIDs [87] and routing messages in mobile wireless ad-hoc networks [111]; however, to the best of our knowledge, this is the first paper presenting a recommendation system for pervasive systems based on this interaction paradigm.

6.2 Related work

In the last years recommendation systems have been recognized as an important research area and much work has been done both in industry and academia on developing new approaches. As a result, a number of recommender applications are used in a variety of e-commerce systems, e.g., for recommending books by Amazon [15, 82], movies by MovieLens [94], DVDs by Netflix [17]. A survey of the main approaches to recommendation appli-

cations can be found in [22]. On the other hand, future mass deployment of pervasive networks opens the possibility of new scenarios for recommendation systems. For example, we refer to MyGROCER [74], a recent proposal for a ubiquitous computing environment for supermarkets based on a smart shopping cart that exploits shopper's identity to provide a personalized service. As observed in [123], improvements are necessary to extend recommendation systems to new scenarios, *"including [...] products to purchase in a store made by a smart shopping cart"*.

Collaborative Filtering allows to extract useful information without requiring cooperation and identification of users and, for this reason, has emerged as the most effective approach to tackle the privacy issues and for mass deployment. We briefly review the main related results, referring to [22, 70, 114] for a thorough survey of literature on Collaborative Filtering. One of the main approaches to Collaborative Filtering, adopted in [57, 114], relies on the computation of similarity indices among items and on using them for prediction of user likely preferences. Namely, an $n \times m$ item-user matrix R stores binary information on users' choices: $R(i, j)$ is 1 if the j-th customer has purchased item i and zero otherwise. Using matrix R, items are classified and the user is suggested a set of items similar to items in U where, intuitively, two items are similar when most users that find one interesting tend to find the other relevant as well.

In many cases users can be clustered in groups: two users in the same group have similar preferences. Singular Value Decomposition (SVD) [93] was shown to be useful to cluster users; we remark that SVD is computationally intensive and requires centralized information and often requires additional conditions for its applicability that are not met in practical cases. In [31, 58] the goal is to approximately recover the latent structure of users' preferences. However, proposed solutions require extensive data on each user and a centralized, expensive computation. Kumar et al. [73, 76] study the off-line problem where preferences are identified with past choices; items are clustered and each user has a probability distribution over clusters: a user first chooses a cluster by her distribution and then chooses a product uniformly at random from that cluster. The goal is to recommend an item from the user's preferred cluster. A different approach is based on the use of ranking-based evaluation measures for the evaluation of regression models [110]; this is motivated by the fact that ranking can be the main underlying goal.

The contributions above consider centralized settings. Distributed recommendation strategies have also been considered in the recent past. In [37] the authors propose to partition item-user matrix R into smaller matrices: each new smaller matrix contains the ratings of all the users on the items belonging to a certain topic or domain, e.g., the movies having a particular genre. However, they assume that these systems can communicate with each other using

a simple request/response protocol. In [55], the authors explicitly consider the limitation imposed on CF by mobile devices, and incrementally update R by connecting near-by devices over Bluetooth without the need for constant connection to a central server. In [30], a distributed solution is proposed to on-line recommendation in which a user is in search of an item she likes; the algorithm is randomized: at each step, the user either selects an item uniformly at random or asks another user about her preferences. Although the above solutions are distributed, we remark that active cooperation between users is required. A similar remark applies to [28] where the goal of the users is to learn their complete preference vector (approximately) while minimizing the cost of probing.

Distributed CF has also been considered for P2P networks. We briefly discuss two representative approaches; [127] considers recommendation in P2P file sharing systems using a Distributed Hash Table to allocate the database of user past transaction among the nodes of the network. [124] uses a similar approach, but the storage and update of user information is performed differently and is determined by the navigation of users. Both strategies require explicit communication among nodes of the network to maintain information on past user transactions.

6.3 Models terminology and notation

We consider a set of n *smart tags*, passive devices tagging *items* in a shop (e.g., a library) or in a museum; every item has a unique integer identifier $i \in [n]$, where $[n] = \{1, \ldots, n\}$. To make terminology simpler, in the sequel we use the term item extensively when referring to the smart tags attached to them, since the scenario and the solutions we consider are oblivious to the nature of tagged items. There are m *users* and each user visits the shop over time carrying a *smart reader*, i.e., a device able to read and update information stored at smart tags. Every user j enters the system, visits a subset of the items and then leaves the system. We call this a *session*. In general, by *visiting an item* we mean an active and detectable interaction between a smart reader and the smart tag tagging an item (e.g., purchasing a tagged item).

The identities of users are not stored, hence, multiple visits of the same user to the shop are not individually tracked. However, we emphasize that information about multiple visits of the same user is stored in aggregate form at smart tags, as we shall see further. We assume that a user visits each item at most once during her permanence in the system, since this captures typical visiting patterns in many cases. The alternative model in which a user may perform multiple visits to the same item during the same session can be of interest in different scenarios and can be more easily modeled using random

walks (see, e.g., [66]). Note also that we assume that computation entirely occurs at the smart reader (e.g., a phone-like device), whereas smart tags only store the outcome of the computation. For this reason, we think of smart tags as passive devices (e.g., RFIDs), which are not battery operated. The results we present also apply to scenarios in which smart tags play an active role in computation. Clearly, in this case energy issues at smart tags can be no longer neglected.

Modeling user behavior in the system entails two aspects: i) describing the way in which users select items of potential interest and ii) the order of visits of items that determines the way in which information about users' past visits is spread across the pervasive system.

i) Cluster based item selection. We assume that every user $j = 1, \ldots, m$ has an associated vector $\mathbf{w}(j) = (w_{1j}, \ldots, w_{nj})$, w_{ij}, called *user profile* in the sequel, describing j's potential interest for item i. Note that $\mathbf{w}(j)$ is unknown to the system. In particular, $w_{ij} \leq 1$ gives the *absolute probability* that user j will select item i. Hence, $\sum_{i=1}^{n} w_{ij} \neq 1$ in general. The selection of items visited by user j proceeds as follows: for every $i = 1, \ldots n$, j visits item i with probability w_{ij}, independently of other items and of other users. Note that, by this definition, there is a user-dependent non zero probability that a user will visit no items.

Following a common assumption in the literature [22, 73, 76], we assume items are partitioned into disjoint *clusters*, $C_1, C_2, \ldots, C_s$, e.g., corresponding to different topics or categories. We further assume that, for item i and user j, w_{ij} satisfies $w_{ij} = p_{kj} w_i$, $i \in C_k$; p_{kj}, the *weight* of cluster k for user j, denotes the preference of user j for items in cluster C_k, and w_i, the *cluster weight* of item i, denotes the *popularity* of item i within cluster C_k, assumed to be the same for all users (i.e., $p_{kj} = p_{kj'}$ if j and j' belong to the same cluster). Put simply, this means that, if items were books for example, our model states that different users may have a different degree of preference for the topic 'Science fiction', but their preferences for science-fiction books mainly depend on item popularity. Note that, since p_{kj} is the absolute probability that user j visits cluster C_k, $\sum_{k=1}^{s} p_{kj} \neq 1$ in general.

We also assume that each item is aware of the cluster it belongs to. Namely we assume that each smart tag contains, among others, a unique label identifying the cluster it belongs to. Though a restriction, this assumption is perfectly realistic in many scenarios, such as the smart poster application we consider or also a bookshop, an e-shop or a supermarket, where items are (physically or virtually) arranged in groups defined by some notion of similarity (e.g., topic or use).

ii) Order of visit. We assume a *weighted visit model*. Namely, if S is the set of possible items and user j has already visited a subset X of items, then the probability that the next item visited is i, $i \in (S-X)$, is $w_{ij}/(\sum_{r\in S} w_{rj} - \sum_{r\in X} w_{rj})$ (for items in X this probability is 0); note that this probability is proportional to w_{ij} and depends on the sum of the total weights of the already visited items. [1] It follows that, the probability that a user visits a given subset of the items follows a distribution that is a special case of Fisher's noncentral hypergeometric distribution[2] [62].

Note that, while different users' visits are independent, the next items visited by a user clearly depend on the items he/she previously visited induced by her preferences. To consider the bookshop example, many people are likely to be first attracted by popular, recently published books in their fields of interest. Experimental evidence discussed in subsection 6.6.3 strongly supports this assumption, at least for the Netflix recommendation dataset.

Remarks. The above model is intended to strike a balance between simplicity and soundness. It is clear that this choice brings some simplification with respect to the scenarios of potential interest. The recommendation based on items' popularity can be sensitive to changes of users' visit patterns over time. Also, assuming that rankings inside clusters only depend on items' popularities may be unrealistic in some scenarios. Furthermore, visit patterns might depend on different (e.g., geometric and physical) constraints, such as the (physical or virtual) structure of the shop. Finally, in the description above, we have "artificially" separated the selection and the visit phases, since this way of looking at the model is useful in the analysis.

A few comments about independence of user visits are also in order. An aspect that is not taken into account in our model is the effect of recommendations themselves on future user behavior. We note that this is in fact a general problem in collaborative filtering and other recommendation approaches. Tackling this aspect easily brings to hardly tractable models. Furthermore, it can be hard to assess the soundness of such a model on publicly available datasets, since these (such the Netflix one) typically provide no information about the impact of recommendations possibly provided by the system on user behavior.

Another important point is that the structure itself of the shop may in

[1] Note that, consistently, $\sum_{i\in S-X} \frac{w_{ij}}{\sum_{r\in S} w_{rj} - \sum_{r\in X} w_{rj}} = 1$.

[2] Fisher's noncentral hypergeometric distribution arises in a "sampling balls from urns" model, in which each urn has an associated color and contains a number of balls of that color. Furthermore, balls are extracted according to weights that only depend on their colors. Our case is the special one in which every urn contains exactly one ball.

many cases "shape" the probabilistic distributions of user visits, intuitively making them look more "similar". Assessing whether this introduces dependencies in user visit patterns is in general hard and problem-dependent. We assume independence for simplicity, at the same time noting that this does not preclude the possibility of similar trends in user visiting patterns, induced by the shop structure.

We conclude by noting that experimental evidence on a real Netflix dataset, shows that at least in the scenario the data refer to, even this simple model captures important trends. For example, experimental evidence reported in section 7.4 supports the model we consider, showing that a significant correlation exists between the order of users' visits and items' popularities within the same movie cluster.

6.4 Recommendation algorithms

As we have already observed, new items are recommended to the user when she "visits" an item, e.g., as her smart reader reads the information contained in the smart tag attached to an item picked up in the shop. As we pointed out earlier, recommendation is performed locally by the smart reader itself, on the basis of information contained in the smart tag and of current user's history information stored in his/her smart reader. In the rest of this section, we address the following points: i) which is the nature of the information carried by the user and by smart tags; ii) how this information is maintained and updated; iii) how it is used to predict a user's likely preferences; iv) the rule followed to provide recommendations. We next discuss points i), iii) and iv) above. Point ii) poses most challenges; it will be briefly outlined in this section and addressed in detail in section 6.5.

User histories and item summaries. We assume that each user carries a vector $\mathbf{H}(j)$ (called *user history in the sequel*), whose components contain information about items visited by j earlier in her visit. More in detail, if j visited s at time t, $\mathbf{H}_s(j)$ contains $F_s(t)$, i.e., the number of users that visited s until time t, and the identifier of the cluster s belongs to. On the other hand, each smart tag also stores aggregate information about past user visits. More in detail, consider a generic item r belonging to cluster C_k. At any time t, r stores $F_r(t)$ and, for every other $s \in C_k$, r stores $N_{sr}(t)$, i.e., the number of users that visited r after visiting s. As we see more in detail in section 6.5, this information is read by smart readers to update their histories, while r's summary is updated as new users visit it. This information is propagated among cluster by stigmergy, as we discuss more in detail in the next section.

Summarizing, each item maintains a counter of the number of users that visited the item in the past, its cluster identifier and, in the worst case, a counter for every other item in its same cluster. We assume each smart tag stores this information, that thus becomes accessible to users as they visit the corresponding items. Note also that this information can be stored at every item (i.e., its smart tag) using a constant number of bits. We call the aggregate information stored at an item its *summary*. Note that items maintain no private information about specific users.

It is clear that, since each user only visits a small subset of a potentially large set, $\mathbf{H}(j)$ should be stored in compact form in a practical implementation, which we actually do. We only consider the definition given above to the purpose of simplifying notation. Furthermore, it is clear that even storing $O(n)$ bits at every tag can be overly expensive. Streaming techniques [99] can provide the necessary tools to address these issues. Since this is mainly an implementation aspect, it will be the focus of future work, our primary goal in this paper being to assess the feasibility of distributed recommendation algorithms using the models we consider.

Predicting user preferences. Upon visiting item r belonging to cluster C_k, the generic user j is recommended items of potential interest among those belonging to C_k, so that recommendations are cluster-based. Recalling the models discussed in section 6.3, in order to achieve this goal it is necessary to estimate $w_{sj} = p_{kj} w_s$, for every $s \in C_k$ other than r. Since, for a given user j, p_{kj} is the same for all items in C_k, this amounts to estimating w_s, for every $s \in C_k$. The problem is that user profiles are unknown to the system and it is unfeasible to estimate them accurately from aggregate information about past user behavior. Fortunately, in order to provide recommendations, it is not necessary to know the values of these weights, but only their relative order.

In fact, we show in the next section that user histories and item summaries as defined above provide enough information for j's reader device to locally, accurately and efficiently compute a suitable monotonic function $f(\cdot)$ of cluster weights such that, for items $i, s \in C_k$, $f(w_i) \geq f(w_s)$ if and only if $w_i \geq w_s$. In practice, upon visiting r, j's smart reader computes a vector $\mathbf{R}(r,t)$, whose i-th entry $\mathbf{R}_i(r)$ is j's estimation of $f(w_i)$ at time t, for every $i \in C_k$. In the rest of this paper, we drop t from $\mathbf{R}(r,t)$ whenever clear from context. Also, we set $f(w_i) = w_i^2 \sum_{j=1}^{m(t)} p_{kj}^2$, where $m(t)$ is the number of users that entered the system up to time t. $f(w_i)$ is monotonically increasing in w_i and, thus, can be used to rank items, as stated by the following lemma, whose proof is straightforward and therefore omitted.

Lemma 1. *For items i and r function f verifies $f(w_i) \geq f(w_r)$ if and only if $w_i \geq w_r$.*

A crucial aspect is the probabilistic nature of the available information: since the set of items and the order of visit of a user are random variables, $\mathbf{H}(j)$ is the outcome of a random process. As we see in next section, this implies that $\mathbf{R}(r)$ is generated from statistics over the histories of users that visited r in the past. This implies that the information available at item r does not allow to exactly compute function $f(\cdot)$; in fact, one of our contributions is algorithms to compute $\mathbf{R}(r)$, so that it is a good estimate of $f(\cdot)$.

Cluster-based recommendations. The general recommendation algorithm is the obvious one and is modular with respect to how ranking of items is computed: upon visiting r belonging to cluster C_k, j's smart reader recommends the $\hat{T}$ *top* ranking items in $\mathbf{R}(r)$ that i) belong to C_k and ii) have not yet been visited by j. The overall behavior of the smart reader is summarized in Figure 7.3, with `UPDATE(`r`, `j`)` implementing the core operation of item ranking.

Require: `Parameter` $\hat{T}$`: number of recommendations`

1: `When user` j `having history` $\mathbf{H}(j)$ `visits item` r `then`

2: $\mathbf{R}(r)$ `= UPDATE(`r`, `j`)`

3: `Recommend` $\hat{T}$ `top elements of` $\mathbf{R}(r)$ `not present in` $\mathbf{H}(j)$

Figure 6.2: Recommendation algorithm.

In the algorithm above, $\mathbf{R}(r)$ is read from the smart tag, updated and then it is used by the smart reader to recommend $\hat{T}$ top scoring elements. Then, the updated version of $\mathbf{R}(r)$ is stored back on the smart tag, replacing the older one. The key issue of how $\mathbf{R}(r)$ is maintained and updated is discussed in detail in section 6.5.

Remarks. The algorithm described above provides *cluster-based* recommendations. I.e., upon visiting an item, a user is recommended a set of items belonging to the same cluster as the current one. Of course, the above strategy can be easily generalized to recommend the top $\hat{T}$ items, regardless of the cluster they belong to. In this paper, we focus on the important case in which we recommend items belonging to the same cluster[3]. We also note here that the

[3]Notice that a user may visit items belonging to different clusters over time and thus be recommended items belonging to different clusters, even if the recommendation strategy is

general approach we consider is that of item-based recommendations, which has proved effective in practice (see for example [57, 82, 114] or [22] for a more general survey). Our main goal here is to carry this approach, appropriately adapted, over to the fully decentralized, stigmergy-based scenario we envision.

In some cases, it may be interesting to consider strategies that also recommend items not belonging to the set of the $\hat{T}$ top ones, so as to diversify the basket of recommendations and thus increase the chance of serendipity. Proceeding this way may increase the probability of recommending items that do not match the user profile and can negatively affect the average quality of the recommendations provided. In section 7.4, we test a simple randomized strategy, which recommends $\hat{T}$ items, each with probability proportional to its estimated weight in the user profile. This strategy essentially reproduces the probabilistic behavior of users in our model.

6.5 Prediction

We next discuss how $\mathbf{R}(r)$ is computed when the user visits the generic item r. Before, we give some additional notation that will be used in the sequel. In particular, we denote by $m(t)$ the number of users that entered the system up to time t. Recall that $F_i(t)$ denotes the number of users that visited item i up to time t. Considered two items i and r, we set $V_{ir}(j) = 1$ if j visits i *and* r *in this order*, 0 otherwise. Finally, we define $N_{ir}(t) = \sum_{j=1}^{m(t)} V_{ir}(j)$. If the user visits item r at time t, her smart reader reads r's summary and then, for every $i \neq r$ it computes $\mathbf{R}(r)$, where:

$$\mathbf{R}_i(r) = \left(1 + \frac{F_i(t)}{F_r(t)}\right) N_{ir}(t).$$

Here, $\mathbf{R}(r)$ is our estimator of $f(w_i)$ computed by the user's smart reader upon visiting r. Note that $F_r(t)$ and $N_{ir}(t)$ are available at r, while $F_i(t)$ is information carried and provided to r by the visiting agent itself (see Figure 6.4, also observe that, by definition, $N_{ir}(t) > 0$ implies $F_r(t) > 0$). The estimator above translates into the implementation of the `UPDATE(·, ·)` routine described in Figure 6.3, to dynamically update $\mathbf{R}(r)$ at a generic node r whenever a new agent j visits the node. In particular, whenever agent j visits node r, r initially updates its local counter (line 2), since it is receiving a new visit, while j records in its history the number of agents that visited r prior to its visit (line 3). This information will be provided by j to nodes it visits in the sequel, if any. Finally (`for cycle`), for every item i previously visited by j, the user's smart reader updates $\mathbf{R}_i(r)$ using the information carried by j.

cluster-based.

UPDATE(r, j)

Require: node r, agent j

1: r maintains a vector $\mathbf{w}(r)$ of estimates of nodes' weights
2: $F_r = F_r + 1$ {A new agent is visiting r}
3: $\mathbf{H}_r(j) = F_r$ {j must initialize $\mathbf{H}_r(j)$}
4: **for** i: 1 ... n do {and $i \neq r$}
5: if $\mathbf{H}_i(j) > 0$ then {j visited i}
6: $N_{ir} = N_{ir} + 1$
7: $\mathbf{R}_i(r) = \left(1 + \frac{\mathbf{H}_i(j)}{F_r}\right) N_{ir}$
8: end if
9: end for
10: return $\mathbf{R}(r)$

Figure 6.3: Update algorithm.

Computational aspects and memory requirements. The core of the computational complexity of the algorithm we propose lies in the update procedure of a smart tag's information upon a user's visit, described in Figure 6.3. The computational cost is clearly linear in the number of items for which the tag stores information (`for` cycle). It should be noted that this number will in general be smaller (possibly much smaller) than the total number n of items in the shop. On the other hand, in each iteration of the cycle, the smart reader is required to perform simple computations, which are compatible with many state-of-art devices (e.g., smart phones). Another issue (related to the former) concerns the amount of memory required at each smart tag. In the worst case, this will be the total number of items in the shop. Here, the limit comes from constraints imposed by the current state-of-art in passive devices. For example, commercial RFIDs can have capacities as large as 8Kbytes at the price of 15$, but 32KByte devices are ready for the market [21]. In our approach and without employing data streaming techniques, each smart tag needs to store one vector with n integer components (N_{*r}) and the counter F_r (note that $\mathbf{R}(r)$ is computed by the smart reader on the fly). Assuming 4 Bytes for each component, this implies the possibility of storing the necessary information for about 2000 items using an 8KByte RFID and four times so much in the foreseeable future. In fact, N_{*r} is a vector of frequency counts

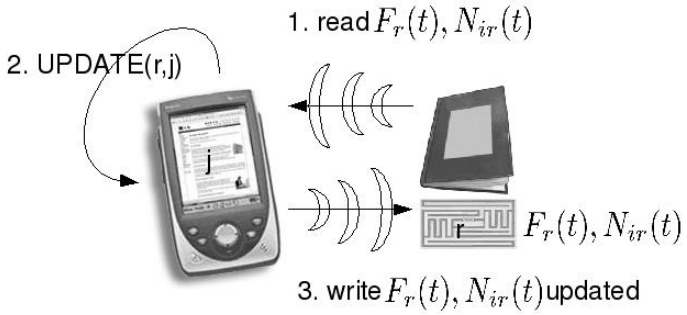


Figure 6.4: Reader-tag interaction.

and it could be maintained in small (polylogarithmic) space using streaming techniques [54].

Analysis. The main result of this section is the proof that, over time and for every $i \neq r$, $\mathbf{R}_i(r)$ provides an increasingly accurate estimation of $f(w_i) = w_i^2 \sum_{j=1}^{m(t)} p_{kj}^2$. In order to analyze the accuracy of the estimator above and the rationale behind, we assume that, if user j visits item i at time t_1 and item r at time t_2, no other users visit i in the interval $(t_1, t_2]$. This assumption is only done for the purpose of the analysis and is not required by our algorithm. In the analysis, it is equivalent to assuming that users visit the system one at the time, i.e., if user j visits i at time t_1 and r at time $t_2 > t_1$, we have $m(t_1) = m(t_2)$.

Note that we are interested in the system behavior as t grows and more and more users visit the smart shop, so as this approximation becomes increasingly accurate. The following result holds:

Theorem 1. *If i, r belong to the same cluster C_k for some k, $\mathbf{R}_i(r)$ becomes an increasingly accurate estimate of $f(w_i)$. In particular, accuracy becomes arbitrarily high as the number of users visiting r increases over time.*

Proof of Theorem 1. In the sequel, we set $\overline{f}_i(t) = \mathbf{R}_i(r)$, the estimate of $f(w_i)$ maintained at time t at node r. The proof of Theorem 1 is implied by proving the following statement:
If $i, r \in C_k$ for some k:

$$f(w_i) = \left(1 + \frac{\mathbf{E}[F_i(t)]}{\mathbf{E}[F_r(t)]}\right) \mathbf{E}[N_{ir}(t)]\,.$$

Furthermore, for every r and i belonging to the same cluster C_k, whenever t

is large enough that $\sum_{j=1}^{m(t)} p_{kj}^2 \geq \frac{3(w_i+w_r)}{\epsilon^2 w_i^2 w_r} \ln \frac{6}{\delta}$ with $\epsilon \leq 1/5$:

$$\mathbf{P}\left[(1-3\epsilon)f(w_i) \leq \overline{f}_i(t) \leq (1+4\epsilon)f(w_i)\right] \geq 1-\delta.$$

We first give the following Lemma that will be useful later:

Lemma 2. *For every $i \in C_k$: $\mathbf{E}[F_i(t)] = w_i \sum_{j=1}^{m(t)} p_{kj}$. Furthermore, for every δ, $\epsilon > 0$, as soon as t is such that $\sum_{j=1}^{m(t)} p_{kj} \geq \frac{3}{\epsilon^2 w_i} \ln \frac{2}{\delta}$:*

$$\mathbf{P}[|F_i(t) - \mathbf{E}[F_i(t)|] > \epsilon \mathbf{E}[F_i(t)]] \leq \delta.$$

Proof. We obviously have $F_i(t) = \sum_{j=1}^{m(t)} X_i(j)$ and $\mathbf{P}[X_i(j) = 1] = p_{kj} w_i$, where $X_i(j) = 1$ if user j visits item i, 0 otherwise. This immediately gives $\mathbf{E}[F_i(t)] = w_i \sum_{j=1}^{m(t)} p_{kj}$. Furthermore, agents visits are independent of each other. Hence, applying Chernoff bound to $\sum_{j=1}^{m(t)} X_i(j)$ [96] yields the result. □

In the sequel, we denote by $\mathcal{S}(j)$ the set of items visited by user j during its permanence in the system. If $l \leq |\mathcal{S}(j)|$, $\mathcal{S}_{<l}(j)$ denotes the subset of the first $l-1$ items visited by j. The following lemma holds:

Lemma 3. *For every S such that $\{i, r\} \subseteq S$, with $i, r \in C_k$ for some k, for every j:*

$$\mathbf{P}[V_{ir}(j) = 1 \,|\, \mathcal{S}(j) = S] = \frac{w_i}{w_i + w_r}.$$

Proof. Denote by $Y_l(j)$ the item visited at the l-th step of j's visit, where $Y_l(j) = \emptyset$ if $l > |S|$. We have:

$$\mathbf{P}[V_{ir}(j) = 1 \,|\, \mathcal{S}(j) = S] = \sum_{l=1}^{|S|-1} \mathbf{P}[(\mathcal{S}_{<l}(j) \cap \{i, r\} = \emptyset) \wedge (Y_l(j) = i) \,|\, \mathcal{S}(j) = S]$$

$$= \sum_{l=1}^{|S|-1} \mathbf{P}[Y_l(j) = i \,|\, (\mathcal{S}(j) = S) \wedge (\mathcal{S}_{<l}(j) \cap \{i, r\} = \emptyset)] \, \mathbf{P}[\mathcal{S}_{<l}(j) \cap \{i, r\} = \emptyset \,|\, \mathcal{S}(j) = S],$$

where the first equality follows since, given $\mathcal{S}(j) = S$, with $\{i, r\} \subseteq S$, $V_{ir}(j) = 1$ is equivalent to stating that i is visited at some step where r has not been

visited yet. On the other hand, denote by $\mathcal{S}_l(i,j,S)$ the set of all subsets of S that i) contain $l-1$ elements and ii) do not contain $\{i,j\}$. We have:

$$\begin{aligned}&\mathbf{P}[Y_l(j)=i\,|\,(\mathcal{S}(j)=S)\wedge(\mathcal{S}_{<l}(j)\cap\{i,r\}=\emptyset)]\\&=\sum_{W\in\mathcal{S}_l(i,r,S)}\mathbf{P}[Y_l(j)=i\,|\,(\mathcal{S}(j)=S)\wedge(\mathcal{S}_{<l}(j)=W)]\cdot\\&\cdot\mathbf{P}[\mathcal{S}_{<l}(j)=W\,|\,(\mathcal{S}(j)=S)\wedge(\mathcal{S}_{<l}(j)\cap\{i,r\}=\emptyset)]\,,\end{aligned}$$

where the equality follows since $\mathcal{S}_{<l}(j)=W\in\mathcal{S}_l(i,j,S)$ implies $\mathcal{S}_{<l}(j)\cap\{i,r\}=\emptyset$. On the other hand:

$$\mathbf{P}[Y_l(j)=i\,|\,(\mathcal{S}(j)=S)\wedge(\mathcal{S}_{<l}(j)=W)]=\frac{w_{ij}}{1-\sum_{f\in W}w_{fj}},$$

by the definition of the weighted visit process described above. Analogously:

$$\mathbf{P}[Y_l(j)=r\,|\,(\mathcal{S}(j)=S)\wedge(\mathcal{S}_{<l}(j)=W)]=\frac{w_{rj}}{1-\sum_{f\in W}w_{fj}},$$

and

$$\begin{aligned}&\mathbf{P}[Y_l(j)=r\,|\,(\mathcal{S}(j)=S)\wedge(\mathcal{S}_{<l}(j)\cap\{i,r\}=\emptyset)]\\&=\sum_{W\in\mathcal{S}_l(i,j,S)}\mathbf{P}[Y_l(j)=r\,|\,(\mathcal{S}(j)=S)\wedge(\mathcal{S}_{<l}(j)=W)]\cdot\\&\cdot\mathbf{P}[\mathcal{S}_{<l}(j)=W\,|\,(\mathcal{S}(j)=S)\wedge(\mathcal{S}_{<l}(j)\cap\{i,r\}=\emptyset)]\,.\end{aligned}$$

This implies that the expressions of $\mathbf{P}[V_{ir}(j)=1\,|\,\mathcal{S}(j)=S]$ and $\mathbf{P}[V_{ri}(j)=1\,|\,\mathcal{S}(j)=S]$ are the same up to multiplying factors, which are $w_{ij}=p_{kj}w_i$ and $w_{rj}=p_{kj}w_r$ respectively. Therefore:

$$\frac{\mathbf{P}[V_{ir}(j)=1\,|\,\mathcal{S}(j)=S]}{\mathbf{P}[V_{ri}(j)=1\,|\,\mathcal{S}(j)=S]}=\frac{w_i}{w_r}.$$

Furthermore, $\mathbf{P}[V_{ir}(j)=1\,|\,\mathcal{S}(j)=S]+\mathbf{P}[V_{ri}(j)=1\,|\,\mathcal{S}(j)=S]=1$, since $\{i,r\}\in S$ and, therefore, $(\mathcal{S}(j)=S)$ implies the event $(V_{ir}(j)=1\vee V_{ri}(j)=1)$. This yields the result. □

Notice that, at every node r and at any time t, we are in fact observing the variable $N_{ir}(t)=\sum_{j=1}^{m(t)}V_{ir}(j)$. As to $\mathbf{P}[V_{ir}(j)=1\,|\,r\in\mathcal{S}(j)]$, we have:

Lemma 4. *If $i, r \in C_k$:* $\mathbf{P}[V_{ir}(j) = 1] = \frac{p_{kj}^2 w_i^2 w_r}{w_i + w_r}$.

Proof. We have:

$$\mathbf{P}[V_{ir}(j) = 1 \,|\, r \in \mathcal{S}(j)] = \sum_{S: r \in S} \mathbf{P}[V_{ir}(j) = 1 \,|\, \mathcal{S}(j) = S]\, \mathbf{P}[\mathcal{S}(j) = S \,|\, r \in \mathcal{S}(j)]$$
$$= \sum_{S: \{i,r\} \subseteq S} \mathbf{P}[V_{ir}(j) = 1 \,|\, \mathcal{S}(j) = S]\, \mathbf{P}[\mathcal{S}(j) = S \,|\, r \in \mathcal{S}(j)]$$
$$= \frac{w_i}{w_i + w_r} \sum_{S: \{i,r\} \subseteq S} \mathbf{P}[\mathcal{S}(j) = S \,|\, r \in \mathcal{S}(j)] = \frac{w_i}{w_i + w_r} \mathbf{P}[\{i, r\} \subseteq \mathcal{S}(j) \,|\, r \in \mathcal{S}(j)] = \frac{p_{kj} w_i^2}{w_i + w_r},$$

where the second inequality follows since $V_{ir}(j) = 0$ deterministically if $i \notin \mathcal{S}(j)$, the third follows from Lemma 3 and the fifth follows since the events $(i \in \mathcal{S}(j))$ and $(r \in \mathcal{S}(j))$ are statistically independent. The claim then follows since $\mathbf{P}[r \in \mathcal{S}(j)] = p_{kj} w_r$. □

Lemma 5. *If $i, r \in C_k$:* $\mathbf{E}[N_{ir}(t)] = \frac{w_i^2 w_r}{w_i + w_r} \sum_{j=1}^{m(t)} p_{kj}^2$.
Furthermore: $\mathbf{P}[|N_{ir}(t) - \mathbf{E}[N_{ir}(t)]| > \epsilon \mathbf{E}[N_{ir}(t)]] \leq \delta$, *as soon as t is large enough that* $\sum_{j=1}^{m(t)} p_{kj} \geq \frac{3(w_i + w_r)}{\epsilon^2 w_i^2 w_r} \ln \frac{2}{\delta}$:

Proof. The first claim follows immediately from Lemma 4. The second claim follows from a simple application of Chernoff bound [96] to the variable $N_{ir}(t)$. □

The following holds:

Lemma 6. *If at most 1 agent visits the system at any time t and $i, r \in C_k$ for some k:*

$$f(w_i) = w_i^2 \sum_{j=1}^{m(t)} p_{kj}^2 = \left(1 + \frac{\mathbf{E}[F_i(t)]}{\mathbf{E}[F_r(t)]}\right) \mathbf{E}[N_{ir}(t)]\,.$$

Proof. The proof follows immediately, by observing that $\mathbf{E}[F_r(t)]\,/\mathbf{E}[F_i(t)] = w_r/w_i$ from Lemma 2 and substituting $w_r = w_i \mathbf{E}[F_r(t)]\,/\mathbf{E}[F_i(t)]$ in the expression of $\mathbf{E}[N_{ir}(t)]$ in Lemma 5. □

Lemma 6 shows that the estimator we are using is in fact a simple plug-in estimator for $f(w_i)$. We can finally prove the claim of the theorem, i.e., that the approximation of $f(w_i)$ becomes more and more accurate over time.

In the sequel of this proof we drop t from the notation, since it is understood from context. We also recall that F_i, F_r and N_{ir} are each the sum of binary independent variables by the independence of the agents' visits. Hence, if $\sum_{j=1}^{m(t)} p_{kj} \geq \frac{3(w_i+w_r)}{\epsilon^2 w_i^2 w_r} \ln \frac{6}{\delta}$, simple applications of Lemma 2 and Lemma 5 allow to conclude that each of the following events occurs with probability at most $\delta/3$: i) $|F_i - \mathbf{E}[F_i]| > \epsilon \mathbf{E}[F_i]$; ii) $|F_r - \mathbf{E}[F_r]| > \epsilon \mathbf{E}[F_r]$; iii) $|N_{ir} - \mathbf{E}[N_{ir}]| > \epsilon \mathbf{E}[N_{ir}]$. Hence, with probability at least $1 - \delta$ we have:

$$\overline{f}_i \leq \left(1 + \frac{1+\epsilon}{1-\epsilon} \frac{\mathbf{E}[F_i]}{\mathbf{E}[F_r]}\right) (1+\epsilon)\mathbf{E}[N_{ir}] < (1+4\epsilon) \left(1 + \frac{\mathbf{E}[F_i(t)]}{\mathbf{E}[F_r(t)]}\right) \mathbf{E}[N_{ir}(t)]\,,$$

where the first inequality follows since we have $N_{ir} \leq (1+\epsilon)\mathbf{E}[N_{ir}]$, $F_i \leq (1+\epsilon)\mathbf{E}[F_i]$ and $F_r \geq (1-\epsilon)\mathbf{E}[F_r]$, while the second inequality holds if $\epsilon \leq 1/5$. Analogously:

$$\overline{f}_i \geq \left(1 + \frac{1-\epsilon}{1+\epsilon} \frac{\mathbf{E}[F_i]}{\mathbf{E}[F_r]}\right) (1-\epsilon)\mathbf{E}[N_{ir}] > (1-3\epsilon) \left(1 + \frac{\mathbf{E}[F_i(t)]}{\mathbf{E}[F_r(t)]}\right) \mathbf{E}[N_{ir}(t)]\,,$$

where the first inequality follows from a similar argument as above, while the second inequality follows from trivial manipulations. Recalling Lemma 6 we complete the proof of Theorem 1.

Convergence. The result of Theorem 1 also describes the convergence properties of our algorithms. It is possible to prove that these bounds are asymptotically tight. A complete analysis is not the purpose of this paper. For the sake of completeness, we briefly address the simpler aspect of the estimation of $\mathbf{E}[F_r(t)]$ at a generic node r. Note that accurately estimating $\mathbf{E}[F_r(t)]$ is crucial for our estimator. It is possible to prove the following theorem:

Theorem 2. *Assume the cluster based model. For every $i \in C_k$ and for every $0 < \delta < 1$, $\Theta(\frac{1}{w_i} \ln \frac{1}{\delta})$ visits are necessary and sufficient to estimate $\mathbf{E}[F_i(t)]$ accurately.*

Proof. The definition of our cluster-based model immediately implies that, if x users visit C_k, then $\mathbf{E}[F_i(t)] = w_i x$. In particular, every user has an equal probability w_i of visiting the item i, independently of the others. We next prove that

$$\mathbf{P}[|F_i(t) - \mathbf{E}[F_i(t)]| \geq \epsilon \mathbf{E}[F_i(t)]] = \mathbf{P}[F_i(t) \leq (1-\epsilon) w_i x] > \delta,$$

whenever x is small enough. The case $F_i(t) > (1+\epsilon)w_i x$ is handled similarly. We have:

$$\begin{aligned}
&\mathbf{P}[F_i(t) \leq (1-\epsilon)w_i x] \geq \mathbf{P}[F_i(t) \leq \lfloor (1-\epsilon)w_i x \rfloor] \\
&= \sum_{y=0}^{\lfloor (1-\epsilon)w_i x \rfloor} \binom{x}{y} (1-w_i)^{x-y} w_i^y > \sum_{y=0}^{\lfloor (1-\epsilon)w_i x \rfloor} (1-w_i)^{x-y} w_i^y \\
&= (1-w_i)^x \frac{1 - \left(\frac{w_i}{1-w_i}\right)^{\lfloor (1-\epsilon)w_i x \rfloor + 1}}{1 - \frac{w_i}{1-w_i}} > (1-w_i)^x,
\end{aligned}$$

where the fourth inequality follows from simple manipulations, while the last inequality follows whenever $w_i/(1-w_i) < 1$. which is always the case whenever $w_i < 1/2$. This implies that

$$\mathbf{P}[|F_i(t) - \mathbf{E}[F_i(t)]| > \epsilon \mathbf{E}[F_i(t)]] > (1-w_i)^x.$$

Considered any $0 < \delta < 1$, simple manipulations shows that $(1-w_i)^x \geq \delta$, whenever

$$x \ln\left(1 + \frac{w_i}{1-w_i}\right) \geq \ln \frac{1}{\delta}.$$

Now, if we assume $w_i > 1/2$, we have $w_i/(1-w_i) < 1$ and the inequality above is true only if

$$x \frac{w_i}{1-w_i} \geq \ln \frac{1}{\delta},$$

which holds whenever

$$x \geq \frac{1-w_i}{w_i} \ln \frac{1}{\delta},$$

thus proving the claim. □

Considering the real case. The results above are obtained under the assumption that agents visit the system in sequence, so that at most one agent is present at any time t. In practice this is not the case. Still, as the number of agents that visited the system grows, the approximation we use becomes negligible. In practice, if some agent j enters the system at time t_1 and leaves at t_2, if t_1 is large enough then we can in practice expect $m(t_2) - m(t_1) << m(t_1)$. This means that, when j visits r at time t_2 after visiting i at time t_1, we can reasonably expect that $(\sum_{j=1}^{m(t_1)} p_{kj})/(\sum_{j=1}^{m(t_2)} p_{kj}) \simeq 1$, so that $F_r(t_2)/F_i(t_1)$ still provides an accurate estimation of w_r/w_i.

6.6 Experimental analysis

The experimental part of this paper focuses on assessing the soundness of our model (section 6.6.3) and the effectiveness of our recommendation algorithms (section 6.6.4). As to the second issue, we compared the performance of our solution with a standard centralized method [57].

6.6.1 Performance metrics for recommendations

We evaluate the performance of the system along two main axes: the ability to infer a ranking in user preferences and the quality of recommendations. In particular, we evaluate the former in terms of ranking similarity, while the latter is evaluated in terms of standard measures of quality used in information retrieval [33], in particular hit ratio, precision and recall.

Ranking similarity. As described in section 6.3, a user's preferences are described in terms of a vector of weights (i.e., the user profile), its i-th component measuring the degree of potential interest of the i-th item to the user. As already remarked in section 6.4, to the purpose of recommending items we are not interested in estimating the components of the user profile, but only their relative order, i.e., their ranking. In particular, if user j visits items r and is recommended a list of items of potential interest, the quality of recommendation depends on how close the ranking of items estimated by j's smart reader is close to the real, unknown one determined by j's profile. To measure how close the real and the estimated rankings are, we use a standard measure of the distance between rankings, i.e., Kendall's τ coefficient (KT for short). KT measures the degree of similarity between two rankings and it is defined as $\tau = 4P/(n(n-1)) - 1$, where $P = \sum_i P_i$. Here, if x is the i-th item in the first ranking, P_i denotes the number of items that follow x in both rankings (i.e., actual ranking and estimated one). KT enjoys the following properties: i) if the agreement between the two rankings is perfect (i.e., the two rankings are the same) the coefficient has value 1; ii) if the disagreement between the two rankings is perfect (i.e., one ranking is the reverse of the other) the coefficient has value -1; iii) for all other arrangements the value lies between -1 and 1, and increasing values imply increasing agreement between the rankings.

HitRatio($\hat{T}$). We recall that user j's smart reader, recommends the $\hat{T}$ top items in $\mathbf{R}(r)$ belonging to the same cluster as r and that are not present in $\mathbf{H}(j)$. There is a hit if at least one of the $\hat{T}$ recommended items will be eventually visited by j. *HitRatio($\hat{\mathrm{T}}$)* is defined as the ratio between the number of hits and the overall number of recommendations given.

Precision($\hat{T}$) and Recall($\hat{T}$). Precision and recall are standard measures of the accuracy in providing relevant documents. Define by D the set of items (called *corpus*) and by D_j the set of relevant items for user j and let $(d_1, d_2, .., d_t)$ be the recommendations provided by the visited item and $(r_1, r_2, .., r_t)$ where $r_i = 1$ if $d_i \in D_j$ and 0 otherwise. Then:

$recall(\hat{T}) = \frac{1}{|D_j|} \sum_{1 \leq i \leq \hat{T}} r_i$.

I.e., recall is the fraction of all relevant items included in the recommendation. Furthermore:

$precision(\hat{T}) = \frac{1}{\hat{T}} \sum_{1 \leq i \leq \hat{T}} r_i$.

I.e., precision is the fraction of the top $\hat{T}$ recommendations that are actually relevant.

In the following we assume that D_j is the set of items that will be visited in the future by a user.

6.6.2 Datasets

We validated our model through experiments on the Netflix movie dataset[4]. This dataset is made of 17770 files, one for each movie. Each file stores a set of ratings represented as a list of tuples <userId, rating, date>, for an overall number of 100480507 ratings made by 480000 users.

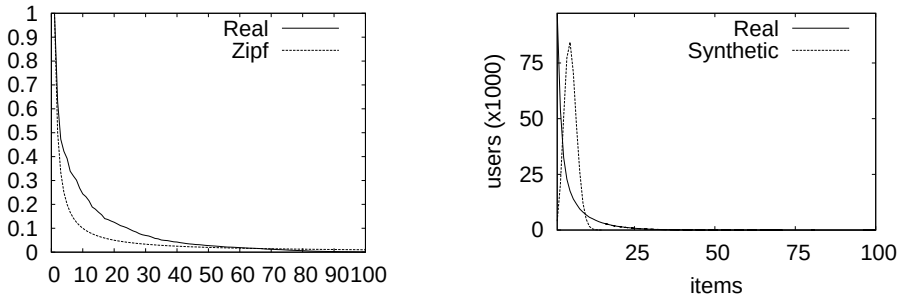
In our cluster based model, movies are clusterized by genre. Unfortunately, movies' genre is not provided by Netflix, but we extracted this information from the MovieLens dataset[5].

6.6.3 Model validation

Distribution of weights. In our experiments on cluster-based recommendation, we considered the Comedy genre, which is one of the most populated clusters. In particular, we uniformly sampled 100 items out of 744 available comedy movies. Considered an item i from the above sample, we used its popularity, obtained by dividing the number of users that visited i by the total number of users, as a proxy for w_i. We did not use users' ratings directly to compute weights, since we did not observe a clear connection between the a priori choice of the movie based on its popularity and user's taste and the a posteriori rating of the movie. Our model reasonably fits actual user behavior if we assume a Zipf's distribution of the weights, with exponent equal to 1 (see figure 6.5(a)).

[4] Available at http://www.netflixprize.com/.

[5] Available at http://www.grouplens.org/node/73



(a) weight distribution

(b) visit length

Figure 6.5: Validation of our model

Visit patterns. As far as the order of visits is concerned, we calculated the correlation coefficient between the cluster weights and the order of visit on real data and we obtained a value of -0.6. This strongly supports the use of the weighted visit model. In contrast, we observed that the distribution of the number of visited items per user predicted by our model using a Zipf's distribution for item weights differs with respect to real data (see Figure 6.5(b)). In fact, Figure 6.5(b) shows that the number of items visited also follows a Zipf law. This is due to our simplifying assumption that users have the same profile within each cluster (though having different preferences for the same cluster). This aspect could be easily taken into account in the model, by suitably redefining weights in the cluster-based model. Since the purpose of the probabilistic model we consider is to infer a ranking in user preferences and not to describe user behavior in its complexity, we opted for simplicity and neglected this issue.

6.6.4 Performance

Algorithms. As a benchmark to evaluate the quality of our recommendation algorithm *alg*, we compare its performance to that of a centralized recommendation algorithm *deshp* and a baseline algorithm *rnd* that simply recommends an item chosen uniformly at random among the ones not yet visited by the user. Furthermore we considered *prob*, a variant of *alg*, where instead of recommending the $\hat{T}$ top items, each item is recommended with a probability proportional to its weight.

More formally, considering item r and denoted by Q_f the first f items already recommended to user j in the current interaction with r, the $f+1$-th rec-

ommendation is for item $i \notin \{r\} \cup \mathbf{H}(j) \cup Q_f$ with probability $\frac{\sqrt{\mathbf{R}_i(r)}}{\sum_{l \notin \{r\} \cup \mathbf{H}(j) \cup Q_f} \sqrt{\mathbf{R}_l(r)}}$[6]. The possibility of also selecting items not belonging to the set of the $\hat{T}$ ones, addresses the issue that the top $\hat{T}$ recommendations are probably the most accurate, but they might in part correspond to very popular or obvious choices. Providing some degree of diversification may alleviate this potential issue at the cost of a loss in accuracy of prediction. Algorithm (*prob*) tries to achieve some degree of diversification in a "controlled" way, by essentially reproducing the probabilistic behavior of users as predicted by our model.

Algorithm (*deshp*) is a state-of-art centralized recommendation algorithm, based on conditional probability similarity and it is described in [57, Subsection 4.1]. The algorithm was implemented adopting the optimizations suggested in [57] and tuning parameters for best performance under the datasets we consider.

Note that, differently from our algorithm, (deshp) can access the whole dataset of user histories. This allows to define a similarity value among any pair of items i and j, such that they were both visited by at least one user. For our decentralized algorithm, this is not the case. For example, if x users visited first i and then j in this order and y users visited them in the inverse order, estimating the similarity between i and j using (deshp) would require knowledge of $x + y$ at both i and j, which is unfeasible in the scenario we envision. Furthermore, the performance of (deshp) depends by the choice of a frequency scaling parameter α [57, formula (2), page 152], which can have "a significant impact on the recommendation quality" [57, par. 6.2.1.4, page 164]. In our experiments, we optimized the choice of α for the specific dataset we considered, but this would be unfeasible in practice in the decentralized scenario we consider. On the other hand, our model assumes visiting patterns that probabilistically depend on item popularities within a topic and statistically infers them. This simple model seems to capture important trends in user behavior that somewhat compensate the lack of information mentioned above, as experimental evidence suggests.

Ranking similarity. Our first goal is to evaluate the ranking similarity between the actual cluster weights and the estimated ones by means of KT computed over 10000 users. It is worth noting that, each item i can estimate only a subset S_i of the other item's weights, depending on the number of users that visited the system and their visiting patterns. We calculate the KT of each item with cardinality $|S_i| \geq 2$. As predicted by our analysis, figure 6.6 shows

[6] Recall from section 6.4 that $\mathbf{H}(j)$ and $\mathbf{R}_i(r)$ are respectively j's history and the i-th component of r's summary and that $\mathbf{R}_i(r)$ is proportional to w_i^2, up to a factor which is constant for items belonging to the same cluster.

that KT tends to one as the number of users increases and, more importantly, that ranking similarity is significantly high (0.7 for synthetic and 0.8 for real data) already after collecting statistics over a very small number of users (i.e., 500). Note that this relatively small number of users supports the feasibility of our proposal in practice.

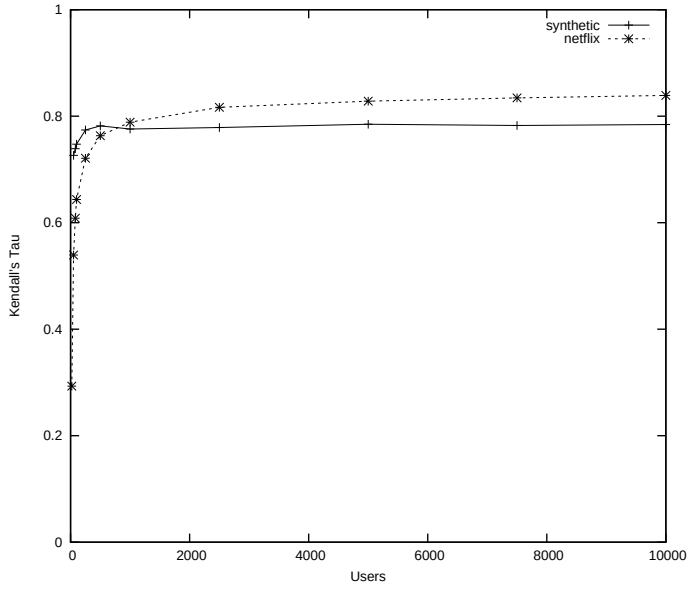


Figure 6.6: Kendall's τ: cluster-based, 100 items.

Quality of recommendation. We evaluated hit ratio, precision and recall of our recommendation algorithm on both a real Netflix dataset and synthetic data generated according to our model. Each performance index has been computed by averaging the results over 10 independent runs with 100 items and 10000 users. To generate synthetic inputs, we assume the weighted visit model with item weights within a cluster distributed according to Zipf's law; since we are considering the single cluster of comedy movies, we can assume that p_{kj}, the *cluster preference*, is one for all users j.

We distinguish two phases in the execution of our algorithms: the *training phase*, during which user profiles are computed (for *alg* and *prob*, item weights are estimated) and the *recommendation phase*, in which recommendations are actually given. In light of the results above, we limit the training phase to very few users. In particular, we consider 100 (tp=100) and 2000 (tp=2000) users, so as to better evaluate how the considered metrics improve as the length of the training phase increases.

For hit ratio, the performance of *alg* is always sensibly better than *rnd*

and *prob* and close to *deshp* as the length of the training phase increases (see figure 6.7). HitRatio($\hat{T}$) of *alg* and *deshp* on real data is between 4 and 2.5 times better than *rnd* (up to 2 for *prob*) when tp=2000. It is interesting to note that the absolute performance of the algorithms is worse on synthetic data (compare figures 6.8 and 6.7). This fact can be explained considering the average length $\tilde{v}$ of the number of visited items per users. In fact, $\tilde{v}$ on real data is about 7 while in synthetic data is about the half. Since the probability of a hit for a user clearly also depends on the number of visits of the user, it follows that the higher the $\tilde{v}$, the higher is the hit ratio (similar considerations can be made for precision and recall). For precision and recall, the performance of our fully decentralized algorithm is very close to the centralized one and both of them significantly out-perform *rnd*. The precision of *rnd* is constant and does not depend on the number of suggestions provided (i.e. $\hat{T}$), as it can be easily proved considering that the random recommendation process is governed by a hypergeometric distribution. The precision of *alg* tends to decrease as the number of recommendation increases; this is expected since, as we observed previously, the accuracy of recommendations depends on the popularity of the items and increasing the number of recommended items forces the algorithm to choose items of decreasing popularity, so more unlikely to meet user expectations. Finally, observe that, as expected *prob* exhibits performance that are worse than those of *alg* and *deshp*, but still well above the baseline *rnd*.

Remark. It should be noted that the values of precision and recall that we obtain on real datasets are indeed relatively high, since they refer to the prediction of really existing links and not to the judgment given by the user about the quality of the recommendations provided. In fact, the data we have do not allows us to directly infer the impact of recommendations on user behavior.

6.7 Conclusion

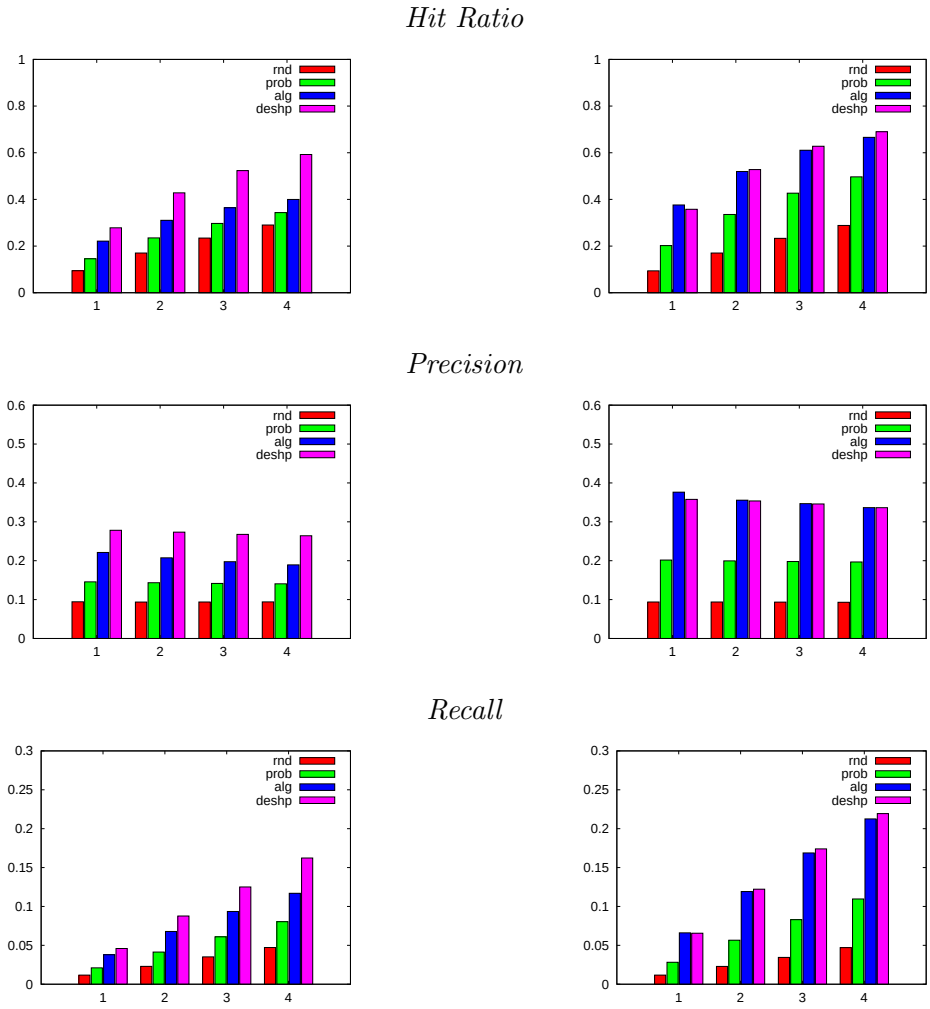
The main contribution of the work presented in this chapter is a model of user behavior that seems to capture important trends in real user data derived from commercial recommendation systems, enabling recommendation strategies that are fully decentralized and seem suitable to meet average user expectations. The proposed model is simple enough as to allow the statistical estimation of parameters from real user activity logs. The resulting recommendation strategy achieves a performance that is comparable to that of state-of-art centralized solutions.

Since available data did not allow to assess the impact of recommendations

directly, results on the quality of recommendation (i.e., precision and recall) have been obtained in a worsening scenario, i.e., checking the extent to which items that were judged of potential interest for a user by the system were actually chosen by that user, which of course leaves out items of potential interest that did not appear in the user log.

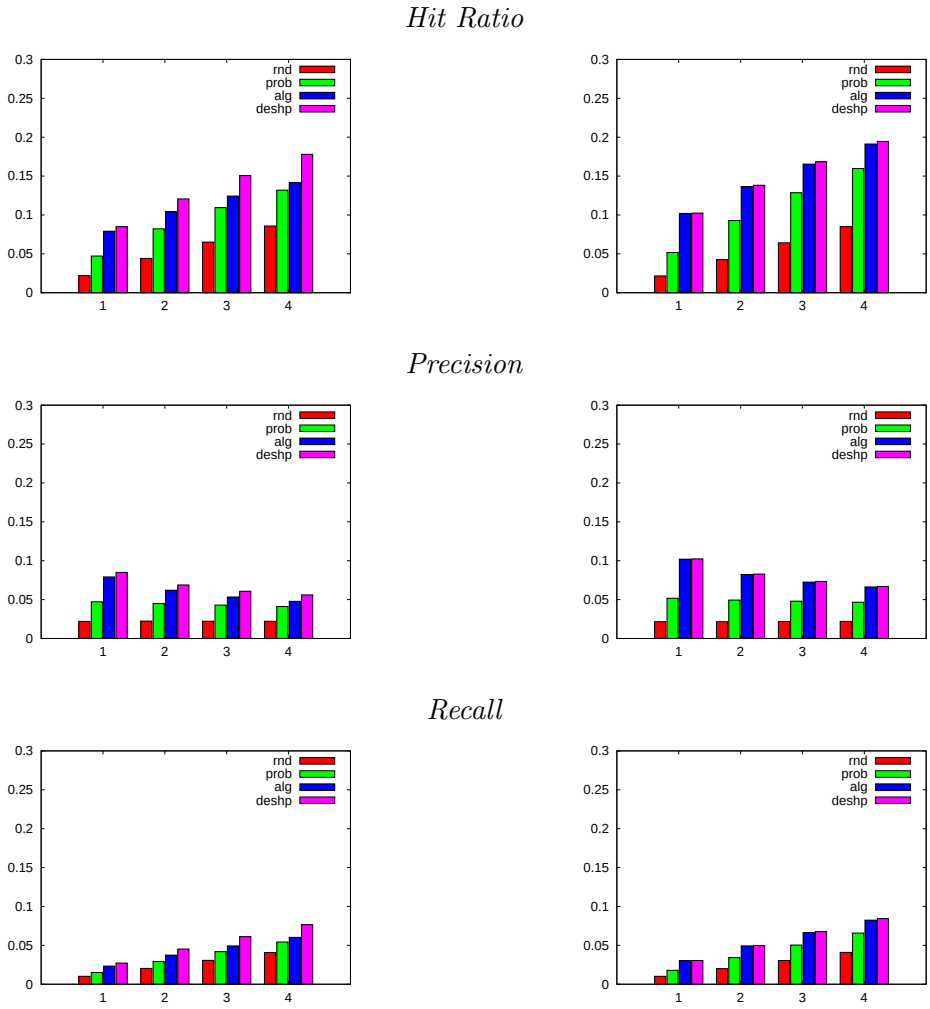
A number of issues remain, which will hopefully encourage further research in the area. As to the model, while physical constraints can be incorporated into the model, as discussed at the end of section 6.3, other aspects to address remain. A first issue has to do with item popularity. Popularity can indeed change, sometimes rapidly, over time. A best selling book might be much less popular within the next month, as the initial wave of interest fades. Proposing simple ageing mechanisms, while keeping the model simple enough is an interesting point. On the other hand, we have proposed strategies that work well when items are clustered, as described in section 6.3. Extending the approach of this paper to the general case is an interesting issue. Of course simple heuristics (e.g., recommending the most popular items) can be easily derived from our approach, but more sophisticated strategies based on more solid theoretical foundations are needed. Another important extension is to include the social context in providing recommendations; social context points on the user's community such as friends, neighbors and colleagues. According to the social dimension, adapting retrieval aims at leveraging the search according to implied preferences of the user's community rather than just the individual. Social context is used in recommender systems based on collaborative filtering techniques [78, 119] and it will be important to include it in our approach.

We conclude noting that, as experimental evidence also suggests, our fully decentralized approach is competitive with state-of-the-art centralized solutions and it is technologically realistic.



(a) training 100 users **(b) training 2000 users**

Figure 6.7: Netflix dataset. x coordinate is the number of provided recommendations ($\hat{T}$).



(a) training 100 users **(b) training 2000 users**

Figure 6.8: Synthetic data. x coordinate is the number of provided recommendations ($\hat{T}$).

Chapter 7

A lightweight sms-based recommendation algorithm

7.1 Introduction

Mobile social networking is an emerging trend. eMarketer forecasts [3] that mobile social networking will grow from 82 million users in 2007 to over 800 million worldwide by 2012. In most mobile communities, mobile users can create their own profiles, make friends, create and participate in chat rooms, hold private conversations, share photos and videos. Major players in social networking, such as Facebook, MySpace and LinkedIn, have already deployed mobile versions of their applications.

Moreover, mobile applications can be extended to support physical presence detection and thus eventually create a link and some kind of convergence between the virtual and real world. For example, Centrl (centrl.com/mobile) is a smart-phone application that lets you see which of your Facebook friends are around and Pelago (www.pelago.com) provides a similar application for Twitter users.

On the other hand, while western countries are experiencing the increasing availability of high speed connections and the diffusion of last generation smart phones with advanced interfaces to access mobile social networks, many still consider Short Messages the most convenient means for instant message exchange [1]. In any case, SMS traffic is still a consistent part of non-voice traf-

[1] "If you look at instant messaging, e-mail or even social networking, they don't have the ubiquity and the reach to replace messaging" - Bill Dudley, Sybase 365's group director for

fic. According to Lloyd's [19], overall Person-to-Person SMS traffic has been 4.5 trillion of messages in 2008. These figures seem to justify the investments of some companies in social networking applications based on Short Messages, such as Jyngle[2] [16] and Peekamo [18]. Furthermore, in large parts of the world, in particular Asia and Africa, SMS are expected to remain the primary means for data communication, at least in the near future. In 2007, nearly 1.5 trillion mobile messages were sent in the Asia-Pacific region [53].

Mobile social networks are thus rising in popularity, but along with clear benefits for users and companies, some concerns primarily related to privacy issues are arising. In the last W3C Workshop on the Future of Social Networking [20], several position papers on this issue appeared. For example, the basic operation of establishing a "friendship" in a social network, whatever the term means for the specific application, is a simple operation (e.g., a mouse-click), but it necessarily entails trust in the likely exchange of private information. As a matter of fact, privacy is one of the main concerns in mobile communities. As Jeff Chester, executive director of the Center for Digital Democracy business model they have developed for mobile advertising is one where lots of user data is collected and user profiles are analyzed" and "You're talking about multiple layers of surveillance at the heart of the mobile marketing business model that raise serious privacy concerns."

In this chapter we propose an approach that uses Short Messages (SMs) and local information available on mobile phones to design a fully decentralized application for recommending new contacts in the social network of mobile phone users. Recommending new contacts is a basic service provided by virtually every social network application. With respect to existing solutions, our approach is characterized by some distinguishing features:

The social networking application we propose is completely decentralized. This implies that the social network is not maintained in a centralized fashion, as usually done in nowadays social networking applications, but it is updated and managed in a fully distributed way by the collective effort of user devices. It transparently collects and processes user information that is accessible in any mobile phone, such as the *log of calls*, the *list of contacts* or the *inbox/outbox of short messages* possibly enriched by user profile information. This information is used to recommend new contacts.

The techniques we propose greatly reduce the amount of personal information that is disclosed, since it is exchanged with other users in the form of a compact summary that allows limited extraction of private data. In addition, we provide a simple and practical cryptographic protocol that can be used to ensure that the computation required by our recommendation system is

product management.

[2]Jyngle closed in August 2009.

performed without revealing any additional private information.

Information necessary to implement the application is exchanged transparently and opportunistically, by using the residual space in standard short messages occasionally exchanged by users. As a consequence, we do not ask users to change their habits in using SMS.

Past research has also considered decentralized systems for item recommendation, such as in [95], where the authors propose the P2P-based PocketLens architecture. We are aware of this body of work, but recommending items using statistical information about past user transactions is not the focus of our work, which is rather on the related but well distinguished "social matching" problem [120], in which we want to infer the latent structure of a social network.

The rest of the chapter is organized as follows: in section 7.2 we describe the approach we follow. In particular, we discuss some social networks naturally arising when analyzing the behaviour of users in a (mobile) telephone network. We then discuss the issues arising in the recommendation of new contacts in such networks, in the first place the notion of *similarity* between users. In section 7.3 we review and discuss the application of sophisticated hashing techniques that allow to estimate the degree of similarity between users in a fully decentralized and privacy preserving way. In section 7.4 we discuss experimental work assessing the effectiveness of our approach on real, publicly available datasets.

7.2 Social networking over SMS messaging

In this work, a node in the social network of mobile phone users is a mobile phone subscriber generating some amount of user-to-user communication. A link connecting two nodes represents an ongoing social relationship (e.g., nodes are friends, colleagues, classmates, etc.) between the corresponding users. In our approach, this social relationship can only be inferred estimating the users' *social profiles* similarity. Speaking in general terms, two users are similar when their social profiles are similar. In fact, the profile of a user is a general notion that depends on the information available to the system. In some cases this includes some biographical data, such as date of birth, sex, information about tastes, interests or activities. A profile is also completed by information that can be extracted transparently from the system, without explicit user intervention, such as the *log of calls*, the *list of contacts* or the *inbox/outbox of short messages*.

We stress that in many cases, even limited information, e.g., the address book or the log of calls, can be used to infer possible relationships: for example,

two users appearing in each other's address books are likely to be socially related, be it through a shared interest, a professional relationship, or simply because they are friends.

Mining the social network underlying telephone traffic has been considered in the past, for example in [23, 24]. Here, there is a (possibly labelled) link from A to B if A calls B at some point. The main goal in [23, 24] was to study the way in which such networks evolve over time, so as to infer and analyze probabilistic generative models [100] describing their evolution.

7.2.1 Recommending social relationships

Recommending new social relationships is one of the most basic services provided by social network applications. In our context, we are interested in strategies to recommend new contacts of potential interest to users. The challenge here is clearly to find contacts that are likely to share some common traits or, put differently, that are in some way "similar" to the user to whom the recommendation is being provided. As stated, this problem is very close to the link prediction problem studied by Liben-Nowell and Kleinberg [80], whose focus is on statistical indicators of social closeness and not on their efficient and decentralized computation.

More formally, if a node A recommends a node B to a third node C, A is suggesting a potential interest or utility for C in establishing a contact with B (unless this contact already exists). Recommendation is performed on the basis of knowledge about the social profiles $L(B)$ and $L(C)$, which are used to estimate the extent to which B and C are "similar". The underlying assumption, made more precise in Subsection 7.2.2, is that the more similar B and C, the more likely it is that they either have a contact, or they might benefit from establishing one.

Privacy requirements make the explicit exchange of private profile information or user contact lists unrealistic for applications. Furthermore, data must fit into the residual space of person-to-person short messages and thus they must be represented in a compact form (i.e., a *sketch*). Figure 7.1 outlines the general application scenario we consider. In step 1, users A and B compute the sketches $sk(L(A))$ and $sk(L(B))$ of their respective social profiles. As observed before, this is a compact representation of the user's social profile preserving her privacy. In step 2 and 3, A and B occasionally send a short message to C. The message space is partially filled with some personal text (e.g., SM Text = "shall we meet this evening?") while the residual space is exploited to deliver the sketches. Observe that users interact with the SMS as usual, while the residual space is transparently managed by a suitable application. In step 4, user C (i.e., the recommender) infers a high degree of similarity between A

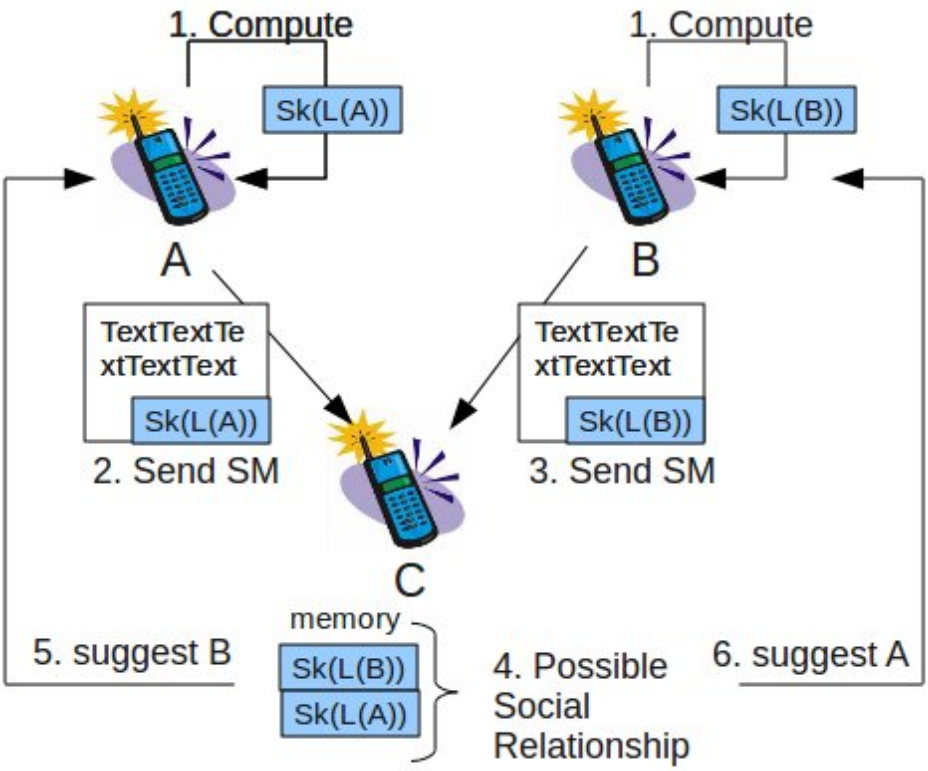


Figure 7.1: A scenario

and B on the basis of their respective sketches. In steps 5 and 6, C eventually recommends a possible friendship to users A and B.

7.2.2 Locally inferring community structure

One of the main issues in recommendation systems for social networking is predicting the potential benefit of new links between users. In the fully decentralized scenario we consider here, this amounts to answering the following question: when should a user A recommend a contact between two other users B and C she is aware of. This in turn implies a number of other issues: i) What information about B and C does A combine in order to decide whether or not she should suggest a contact between B and C if not existing already; ii) how is this information obtained, manipulated and exchanged; iii) how are computational, storage and communication constraints met; iv) how is privacy preserved.

Like many networking applications, we recommend new contacts on the basis of similarities between users. Thus, A will recommend B and C to establish a contact if A assesses that B and C are "similar". In particular, if we view profiles as feature sets, we say that two users A and B are *similar* when their social profiles $L(A)$ and $L(B)$ overlap significantly. In this perspective, we estimate user similarity by the Jaccard coefficient $J(L(A), L(B)) = \frac{|L(A) \cap L(B)|}{|L(A) \cup L(B)|}$, a widely accepted measure of similarity between sets. In the social networking scenario we consider, it captures the well known fact [80, 100] that social networks are densely connected at a local level or, roughly put, the folklore that two friends of the same person are significantly more likely to be friends than any two randomly chosen people.

7.3 Mining the social network of SMS users

A key aspect in the applications we consider is estimating the size of the intersection between the social profiles of two users in a fully decentralized way. More precisely, if a user C receives short messages from A and B, she should be able to estimate $J(L(A), L(B))$ from summary information about $L(A)$ and $L(B)$ piggybacked in the messages themselves. It is clear that short message size poses stringent constraints on the amount of information that can be piggybacked. This is at most 140 bytes, but recalling that we only use the residual space on the message, a variable number of those bytes will be occupied by the message body itself. We show below how to address these issues in the following way: i) we adapt a technique initially conceived for Web page similarity estimation to the scenario we consider. The adoption of this technique allows to compute *compact summaries* or *sketches* of each social profile, which in turn allows efficient estimation of the Jaccard coefficient between social profiles. The space required by the proposed sketches is in the order of a few tens of bytes; ii) we address the issue of variable SMS size under the assumption [132] that SMS sizes are (approximately) uniformly distributed. Specifically, for those messages created in person-to-person communications, the length seems to evenly span the whole range of the allowed message sizes [132], whose maximum value depends on the encoding that is used for each message, but it is typically 140 bytes. In the following, we refer to profiles based on users' contact lists, since contact information is locally available on virtually every recent commercial device. For this reason, the terms *social profile* and *contact list* will be used interchangeably. We emphasize that the techniques described in the remainder can be extended to more general notions of user profile, as described in section 7.2.

7.3.1 Estimation of the Jaccard coefficient.

Consider the set of possible contact identifiers. Recall that, as motivated further in this section, they can be regarded as integer numbers falling in the range $[n] = \{0, \ldots, n-1\}$ for suitable n. The only assumption we need is that they are unique, a constraint that is met in practice in the applications we consider; they are users' telephone numbers or a suitable representation of them. As a consequence, considered any two users A and B, their contact lists $L(A)$ and $L(B)$ may be simply regarded as two subsets of $[n]$. Our goal is to measure their overlap using the Jaccard coefficient: $J(L(A), L(B)) = \frac{|L(A) \cap L(B)|}{|L(A) \cup L(B)|}$.

A very simple and elegant technique to estimate the Jaccard coefficient has been proposed in several equivalent forms by Broder et al. [43, 44]. Assume we are able to choose a permutation $\pi(\cdot)$ mapping $[n]$ onto itself uniformly at random. For every $X \subseteq [n]$, denote by $\pi(X)$ the set of the images of elements in X when $\pi(\cdot)$ is applied and let $\min(\pi(X))$ denote their minimum. Then it can be shown [43] that (i) considered a set $S \subseteq [n]$ and for every $a \in S$, $\mathbf{P}[a = \arg\min(\pi(S))] = 1/|S|$; (ii) for every S_1, $S_2 \subseteq [n]$: $\mathbf{P}[\min(\pi(S_1)) = \min(\pi(S_2))] = J(S_1, S_2)$. This property immediately yields a technique to estimate $J(S_1, S_2)$.

The algorithm consists in performing m independent executions of the following procedure: i) pick one permutation $\pi(\cdot)$ of [n] uniformly at random from the $n!$ possible ones; ii) in the i-th iteration, let $\min(S_1) = \min(\pi(S_1))$ and $\min(S_2) = \min(\pi(S_2))$. We increment a counter C_m whenever $\min(S_1) = \min(S_2)$. At the end of the process, our estimation of $J(S_1, S_2)$ is C_m/m. Standard tools from probability theory tell us that C_m is an increasingly (with m) accurate estimation of $J(S_1, S_2)$.

7.3.2 Computing and maintaining contact list sketches.

Unfortunately, generating permutations uniformly at random requires a number of truly random bits that is in the order of n [43]. Fortunately, suitable families of simple, linear hash functions perform well in practice (e.g., see [41, 71]). In particular, we use linear permutations [41], i.e., functions of the form $h(x) = ((ax + b) \mod p) \mod n$. Here, p is large prime, while a and b are integers belonging to the intervals $[1, p-1]$ and $[0, p-1]$ respectively.

We next describe how each node A of the network maintains the local *sketch* $sk(A)$ associated to $L(A)$. As pointed out before, we assume below that every number in $L(A)$ is an integer falling in $[n]$. To this purpose, it is enough to perform a first step in which each contact identifier (e.g., a user's mobile phone number) is regarded as a string and this string is mapped onto an integer in $[n]$, using any hash function, as long as the probability of collision

UPDATE($sk(A)$, pn)

Require: Sketch $sk(A)$, number pn

1: x = hash(pn) {Hash pn to an integer in [n]}
2: **for** i: 1 ... m do
3: M_i = $h_i(x)$ {Map x according to a random permutation}
4: if M_i < $\min_i(A)$ then
5: $\min_i(A)$ = M_i
6: end if
7: end for
8: return $sk(A)$

Figure 7.2: Update algorithm.

is sufficiently small. This is for instance the case if we hash contact identifiers to 32-bit integers using a good hash function, e.g., implemented in Java standard classes. As a second step, m hash functions are generated. The i-th hash function has the form $h_i(x) = ((a_i x + b_i) \mod p) \mod n$. The integers $\{a_1, b_1, \ldots, a_m, b_m\}$ are generated *independently* and uniformly at random, respectively in the interval $[1, p-1]$ for the a_i's and $[0, p-1]$ for the b_i's. Finally, for $i = 1, \ldots, m$, let $\min_i(A) = \min_{x \in L(A)}\{h_i(x)\}$. The sketch of $L(A)$ is the *ordered* vector $sk(A) = (\min_1(A), \ldots, \min_m(A))$. A version of this algorithm that allows dynamic updates when new numbers are added to the contact list is given in Figure 7.2.

The cost of algorithm UPDATE($sk(A)$, pn) is $O(m)$. The deletion of items from the contact list is more expensive, since the element removed might be the one achieving minimum value on one or more of the hash functions. Therefore, in the case of deletions $sk(A)$ has to be recomputed from scratch and the cost becomes $O(m|L(A)|)$. Note however, that m is in the order of a few tens at most (10 in our experiments). This complexity is therefore fully compatible with standard commercial mobile phones.

In addition to $sk(A)$, A's device stores $sk(B)$, if available, for every B in her contact list. The required amount of additional memory, as discussed further in greater detail, is a few tens of bytes for each entry in the contact list (40 in the current implementation), thus, perfectly compatible with standard commercial devices.

RECOMMEND(A, $sk(B)$, $sk(C)$, θ)

Require: Node A, Sketch $sk(A)$, $sk(C)$, threshold θ

1: Estimate $J(L(B), L(C))$ from $sk(B)$ and $sk(C)$ {Node must have both}
2: **if** $J(L(B), L(C)) > \theta$ **then**
3: A recommends B to C or viceversa
4: **end if**

Figure 7.3: Recommendation algorithm.

7.3.3 Exchanging sketches.

In the scenario we envision, if both user A and B run the application and B sends an SMS to A, B will use the available free space of the message to send its own sketch $sk(B)$, or part of it, to A. Let's assume for the moment that there is enough residual space in the message to send the whole $sk(B)$. Note that this is likely to be often the case since, as we see later, the size of a sketch is typically a few tens of bytes, 40 in the present implementation. Moreover, we discuss how to address cases in which the SMS free space is not sufficient to contain $sk(B)$ in a further paragraph of this section. Whenever A's device receives the message, it transparently extracts $sk(B)$ from the message body. If B is one of A's contacts, then $sk(B)$ is stored in A's contact list, associated to B, possibly replacing an older copy of $sk(B)$.

7.3.4 Fully decentralized recommendation of contacts.

Recall that we assume that two users are *similar* to the purpose of the application whenever their contact lists overlap significantly. The algorithm in Figure 7.3 implements this general idea. In particular, the algorithm describes the behaviour of the generic, mobile terminal of some user A. If A has the sketches of both B's and C's contact lists, A will recommend B (C) to C (B) whenever the local estimation of $J(L(B), L(C))$ exceeds some given threshold θ. In section 7.4 we study, among others, how the choice of the threshold affects the quality of recommended contacts.

7.3.5 Implementation issues.

We discuss in this paragraph several implementation issues.

If we consider the generic node A, the amount of memory needed to store its contact list is $\Theta(L(A))$. In our implementation, A also needs to store i) its own sketch $sk(A)$ and, in the worst case, ii) $sk(B)$, for a subset of nodes from which A received SMS messages in the past. If we assume that A stores the sketch of every contact, the required amount of memory is $O(m(|L(A)|))$. In practice, if we use $m = 10$, the additive amount of bytes required for each contact is about 40. This is in the same order of magnitude of an entry in any address book of a commercial device.

The computational cost of maintaining sketches and providing recommendations is also compatible with current commercial devices. In particular, adding a new contact to the contact list of a node A requires updating $sk(A)$ (algorithm `UPDATE(`$\cdots$) in Figure 7.2) and has cost $(O(m))$. Removing a contact from $L(A)$ (typically a less frequent operation) is more expensive but it has (up to m) still linear cost, i.e., $O(m|L(A)|)$. Finally, for two nodes B and C other than A, deciding at A as to whether recommending each of them to the other requires estimating $J(L(B), L(C))$, which has cost $O(m)$. Computation is performed at user devices. Nowadays, these are typically small computers, whose computational capabilities are perfectly compatible with the computational effort required by the proposed techniques.

The number of hash functions required (i.e., m) is chosen, so that probability that the estimation of the Jaccard coefficient differs from the true value by more than a chosen constant is below a suitably small constant. We refer the reader to specific work (e.g., [43, 44]) for technical details. In our case, experimental evidence suggests that 10 hash functions are sufficient to strike a reasonable balance between accuracy of the estimation and memory requirements.

A further constraint is that all user devices use the same set of hash functions. In practice, hash functions and the algorithms we propose will be implemented and maintained in the device's memory. This in turn requires storing, for each hash function, its coefficients and p in binary form. In our implementation, coefficients are 32-bit integers, while p is the well-known Mersenne prime $2^{31} - 1$, which does not need to be stored explicitly. So, it turns out that the actual storage requirements for maintaining hash functions is around 80 bytes. The overall implementation (code, hash functions, runtime data structures) requires less than 1Kbyte space. To this, we must add the (variable) size of the user's (modified) contact list. Thus, the storage requirement of the modified contact list is in the same order of magnitude as in a standard implementation.

We observed earlier that we cannot always assume that the message body of an SMS sent from some node A to another node B has enough free space to host $sk(A)$. The most direct way to circumvent this problem is for A to send its sketch whenever the available free space in the message body exceeds $|sk(A)|$.

In fact, the distribution of SMS message sizes seems to be approximately uniform [132]. Assuming for the sake of simplicity that it is exactly uniform and that message sizes of different messages are independent variables, we have that half of the messages have 80 bytes available space in the average, more than 75% have at least 40 bytes available to carry sketches and so on. This means that, in the average, 1.34 message are enough for A to send its sketch to B, which means that, in practice, if A sends 2 SMS to B, the latter is very likely to receive A's sketch. [3]

7.3.6 Enforcing privacy

As described in the previous sections, a sketch is a representation of the contact list that, besides reducing the amount of data to be exchanged, does not fully discloses a user's contact list. As an example of the type of information that is leaked by the sketch $sk(A) = (\min_1(A), \ldots, \min_m(A))$ of contact list $L(A)$, we point out that if $h_i(x) < \min_i(A)$ then certainly $x \notin L(A)$. In this section we show how to securely compute the Jaccard coefficient of two contact lists, $L(A)$ and $L(B)$, without revealing any information except what can be deduced from the Jaccard coefficient itself.

Let us consider two parties A and B, each holding a vector of length m; with a slight abuse of notation we identify each party with his/her input vector. In our application to the computation of the Jaccard coefficient, the vectors will be the sketches of the respective contact lists. A and B wish to compute the number of positions i for which $A[i] = B[i]$ without revealing any additional information on the vectors. We will describe a protocol that uses an additively homomorphic encryption scheme $(E; D; K)$ like Paillier cryptosystem (see [102] for further information).

Homomorphic encryption scheme. Let $(E; D; K)$ be a homomorphic encryption scheme and assume that the message space for a public key pk returned by the key generator algorithm K on input security parameter m is $\mathbb{Z}_p$ for some integer p of length m. The following additive homomorphic properties hold: ii) the product of two ciphertexts is a ciphertext for the sum of the plaintexts; that is, for all messages $a; b \in \mathbb{Z}_p$ and public keys pk, we have $D(E(pk, a) \cdot E(pk, b), sk) = a + b$; ii) raising a ciphertext for message a to power r gives a ciphertext for $r \cdot a$; that is, for all $r \in \mathbb{Z}_p$ we have that $D(E(pk, a)^r, sk) = r \cdot a$.

[3] An alternative solution is that A sends to B part of its sketch, compatibly with the available space in the SMS message body. This solution requires bookkeeping both at A and B, to keep track of the portions of $sk(A)$ still missing at B. In fact, the former solution can be more easily implemented than the latter and it requires no additional data structures.

The protocol. The protocol can be described as follows:

1. A picks a pair of public and secret key (pk, sk) for encryption scheme (E, D, K) by running the key generator algorithm K on input 1^m; for $i \in [n]$, A computes encryption $a_i = E(pk, A[i])$ of $A[i]$; A sends pk and $(a_i)_{i\in[n]}$ to B;

2. for $i \in [n]$, B computes encryption $b_i = E(pk, -B[i])$ of $-B[i]$, picks random $r_i \in \mathbb{Z}_p$ and sets $c_i = (a_i + b_i)^{r_i}$. Notice that by the homomorphic properties of (E, D, K), c_i is a ciphertext for $r_i \cdot (A[i] - B[i])$. Therefore if $A[i] = B[i]$, then c_i is an encryption of 0; otherwise c_i is an encryption of a random element of $\mathbb{Z}_p$. B randomly permutes the c_i's and sends them to A.

3. A decrypts the m ciphertexts received from A, counts the number s of ciphertexts that are an encryption of 0 and sends s to B.

Properties of the protocol. We make the following simple observations: *Correctness.* The value s computed by the protocol is the number of indices i for which $A[i] = B[i]$, with probability exponentially close to 1. *Privacy of the input.* Each of A and B gets no information on the other party's vector, besides what can be obtained from the output of the protocol. For A, this can be easily seen by exhibiting a probabilistic polynomial-time simulator S that, for all vectors A and B, on input vector A and the number s of positions in which A and B coincide (but not vector B) outputs A's view of the protocol. Similarly, we can construct a simulator for B.

Coming back to the recommendation system, we have two parties A and B, each with a private contact list, $L(A)$ and $L(B)$, that wish to compute the Jaccard coefficient $J(L(A), L(B))$. Obviously, the Jaccard coefficient can be computed by applying the above protocol to the characteristic vector of the two sets. The protocol will then run in time linear in the size of the underlying universe set. A much more efficient protocol is instead obtained by running the above protocol with each party holding as an input the sketch of his/her contact list computed using the same sequence of random (or min-wise independent) permutations.

7.4 Experimental analysis

In this section we present the results of experimental on real, publicly available data sets, in our opinion supporting the effectiveness of the approach we propose.

7.4.1 Experimental setting

Objectives

Our experimental work had the following main goals. In the first place, we wanted to understand the intrinsic effectiveness of the Jaccard coefficient to infer social relationships in the mobile phone user network. A further issue was to assess whether the techniques we use to approximate the Jaccard coefficient and discussed in section 7.3, are compatible with the hard space constraints, imposed by Short Message size. In particular, these severely limit the number of hash functions we can use to compute contact list sketches (we considered the use of 10 or 20 hash functions in our implementation). This in turn affects the accuracy of the estimation, especially when the value of the Jaccard coefficient is relatively small in absolute terms, as is the case for the data set we consider. Finally, we wanted to investigate the effectiveness of our overall approach in suggesting contacts to users. The problem here is that the data do not allow us to directly assess the *a posteriori* effect of recommendations. For this reason, in our experiments we considered the ability of our approach in predicting existing links as a proxy of its effectiveness in providing useful recommendations.

Data sets and call graph

Accessing telephone traffic data is far from trivial, since very few public datasets are available. The Reality Mining project [9, 59] represents the largest mobile phone experiment ever attempted in academia. Its dataset contains thousands hours of continuous data on daily human behavior and contains information on *call logs*, Bluetooth devices in proximity, cell tower IDs, application usage, phone status.
We used call logs to build a *call graph* where nodes are mobile users characterized by unique ids (i.e., telephone numbers) and there is an edge connecting two users i and j if and only if the call log contains a number of calls between i and j that exceeds a given threshold[4]. For the purpose of this experimental analysis, we exploited the call graph to build the contact list of users in the network. To avoid problems related to data incompleteness, we restricted our experiments only to the people actually participating in the Reality Mining project (around 100 people), whose logs are complete and accurate.

[4]Note that this threshold is the number of calls above which we declare the existence of a link between the involved users. It is different from the threshold θ in Figure 7.3 of section 7.3, i.e., the value of the estimated Jaccard coefficient above which a contact is recommended.

Require: $G_w(V, E)$

1: **for** $v_1 \dots v_n \in V$ `do`
2: `for each` v_a, v_b `among the contacts of` v_i `do`
3: `retrieve` $sk(A)$, $sk(B)$ `by emulating an SMS reception`
4: `RECOMMEND(A,` $sk(A)$, $sk(B)$, θ`)`
5: `end for`
6: `end for`

Figure 7.4: Simulation algorithm.

Formally, the call graph $G_w(V, E)$ is built as follows:

- V is the set of users appearing in the log of calls
- for each pair $(i, j) \in V$, edge $(i, j) \in E$ if and only if at least w calls occurred between i and j.

Note that the graph G_w is *undirected*, i.e., we assume that contact lists are *symmetric*, i.e., i belongs to the contact list of j (and viceversa) if at least w calls occurred between i and j during the period of observation. Following the definition above, changing values of w can originate different graphs modeling stronger or weaker relations (i.e., a higher w can filter out occasional contacts). In our experiments we considered $G_1(V, E)$ and thus worked directly on the call graph, since the data set is relatively small and higher values of w further reduced the data set size.

Experimental scenario for recommendations

We assessed the quality of our technique in providing recommendations of good quality by using its ability to uncover existing relationships as a proxy. In particular, for each node v_i and for each pair $\{v_a, v_b\}$ both belonging to v_i's contact list, we ran our algorithm to predict the existence or non-existence of link (v_a, v_b). We then checked whether the link existed or not. This is synthetically described by the algorithm in Figure 7.4, which simulates the general recommendation algorithm described in section 7.3.

Performance indices

The error of the recommendation strategy we propose is potentially affected by two factors: i) inaccuracy in the estimation of the actual value of the

Jaccard coefficient; ii) error in the recommendation itself, i.e., the contact we recommend is not interesting to the user. These two aspects are clearly interrelated in complex ways. We treat them separately, which corresponds to the worst-case assumptions that the effects of the two sources of error sum up. As for the former aspect, assessing the accuracy of our approximation of the actual Jaccard coefficient poses some issues. In the first place, our data show that even values of the Jaccard coefficient related to a significant degree of social relationship can be low in absolute terms. This makes an accurate estimation harder to attain given the stringent constraints we have to comply with. In particular, if we use m hash functions, we only have m possible values for our estimation of the Jaccard coefficient. When $m = 10$ as we assume, this provides very little granularity. Namely, possible values of the estimated Jaccard coefficient are $0.1j$, with $j = 0, \ldots, 10$, whereas values of the true Jaccard coefficient corresponding to a significant degree of social interaction are around $[0.05, 0.1]$ in the dataset collection we consider. On the other hand, our algorithm is threshold-based: it recommends a contact between two nodes A and B whenever $J(L(A), L(B))$ is above the threshold. For this reasons, we consider the *Jaccard-Estimation Performance* (JEP), defined as the fraction of times that our algorithm gives the same recommendation as it would give if it knew the exact values of the Jaccard coefficient. We call the two versions of the algorithm *apxJacc* and *exactJacc* in the definitions that follows. Formally, for every node i, let C_i denote the number of times that *apxJacc* and *exactJacc* take the same recommendation decision for pairs of nodes belonging to i's contact list.

The *Jaccard-Estimation Performance* (JEP) is formally defined as:

$$JEP = \frac{\sum_{1 \leq i \leq |V|} C_i}{t}$$

where V is the vertex set, i.e., the overall number of users and t is the overall number of node pairs evaluated.

To assess the quality of our recommendations, we checked to which extent the contacts that are recommended correspond to actual links evaluating *precision* (i.e., the fraction of existing links that have been recommended over the total number of given recommendations) and *recall* (i.e., fraction of all existing links that have been recommended over the total number of actual links) of our recommendation algorithm.

7.4.2 Experimental results

In this section, we provide experimental results that address the following issues: i) whether or not the Jaccard coefficient is a good indicator of social ties in the datasets we consider and which are reasonable threshold values

for the recommendation heuristic we propose; ii) how good is our estimation of the Jaccard coefficient, at least in the sense made precise in the previous subsection; iii) how good are the recommendations we provide, which we indirectly answer by to which extent we are able to infer existing contacts between node pairs. Since our algorithms contain a probabilistic part in the selection and use of the hash functions used to estimate the Jaccard coefficient, all results reported below refer to averages taken over 5 independent runs of the algorithm.

Jaccard coefficient and social ties

Figure 7.5 synthetically describes the correlation existing between values of the Jaccard coefficient and existence of links between node pairs. More in detail, the x-axis is divided into interval of width 0.05 each, starting at 0.0 and ending at 0.2. For the j-th interval ($j = 0, 1, 2, 3$), the ordinate represents the fraction of pairs (A, B) of users such that i) $J(L(A), L(B))$ falls in the interval $[0.05j, 0.05(j + 1)]$ and ii) A and B are contacts, i.e., they are in each other's contact lists. The x-intervals stops at the value 0.2, since we observed too few pairs with Jaccard coefficient beyond this interval, to be statistically meaningful. This picture clearly shows that the Jaccard coefficient is a good indicator of social ties in mobile user networks. Furthermore, at least in the datasets we considered, the Jaccard coefficient allows to identify a sharp transition around the value 0.05, from a region characterized by sporadic ties to one characterized by frequent social relationships. In light of these observations, we chose the value 0.05 as a threshold in our recommendation algorithm.

Jaccard estimation performance

Figure 7.6 shows the behaviour of the Jaccard-Estimation Performance, as defined in the previous subsection, as a function of the threshold, both when 10 and 20 hash functions are used to estimate the Jaccard coefficient. In particular, the function has been computed in 5 points for 20 hash functions. Each point represents an independent run of the algorithm. More precisely, for $j = 1, \ldots, 5$, the j-th run is executed with threshold value $0.05j$. For 10 hash functions we only considered two points, since for values of the threshold above 0.1 the distance between the two curves becomes smaller and smaller. Results show that the algorithm that estimates the Jaccard coefficient using hash functions takes the same decisions as the one knowing the exact value of the Jaccard coefficient in most cases. This means that, even under the stringent constraints for sketch sizes, we are able to follow the ideal algorithm

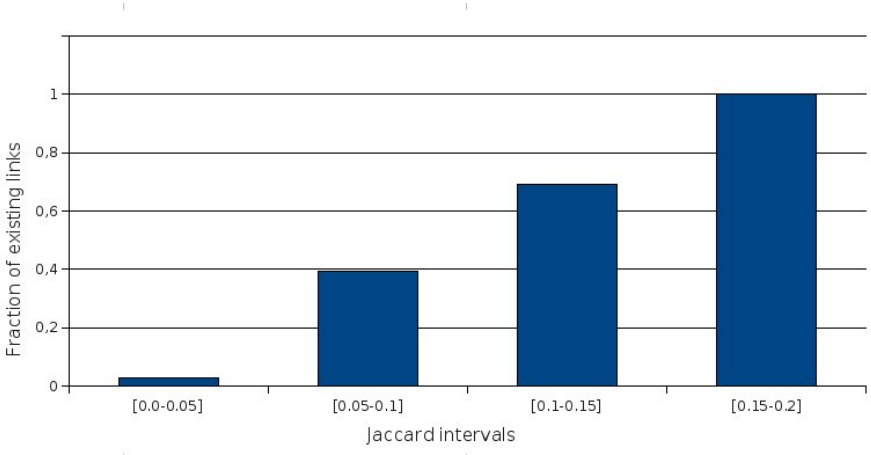


Figure 7.5: Existing linked pairs over total pairs

pretty close, as far as the recommendation decision is concerned.

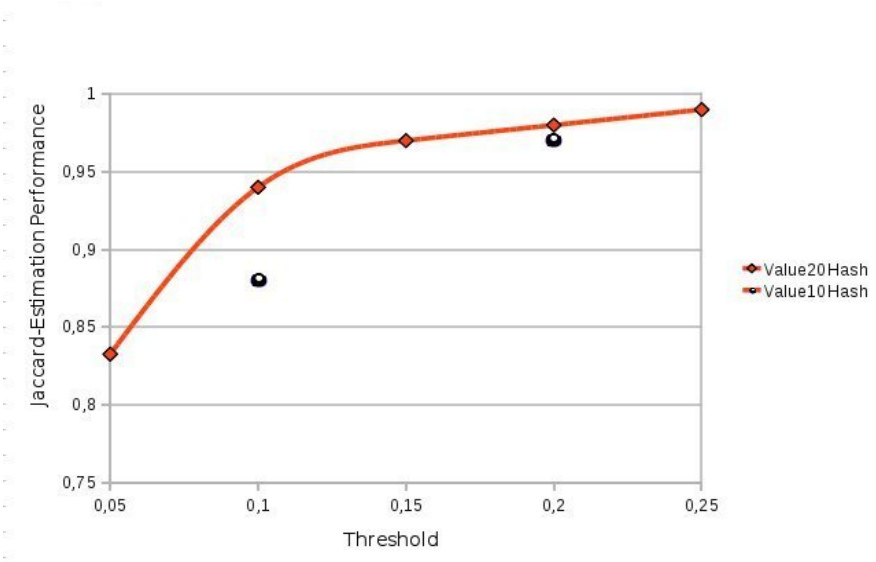


Figure 7.6: Jaccard-Estimation Performance

Quality of recommendation

Figure 7.7 shows the effectiveness of our algorithms in predicting the existence of contacts in the social network of mobile users. In particular, Figure 7.7 is a

scatter-plot showing the trade-off between precision and recall as the threshold and number of hash functions used vary[5]. For a better reading, values with $precision < 0.2$ or $recall < 0.2$ have been filtered out[6]. The following remarks are in order: i) the best trade-off between precision and recall is struck near the interval $[0.05, 0.1]$ of the threshold, both for the algorithm using exact Jaccard coefficient and for our heuristics; ii) For higher values precision increases and recall decreases, meaning that on one hand, a similarity beyond the threshold implies a contact with increasing probability, but we omit to recommend many contacts that fall below the threshold; iii) the values of precision/recall we obtain for the best choice of the threshold fall in the interval $[0.4, 0.6]$. Such values are indeed relatively high, since they refer to the prediction of really existing links; if we were only recommending links that already exist, there would be no point in providing recommendations. These results in our opinion provide an indication that our fully decentralized strategies might prove effective in providing recommendations of good quality.

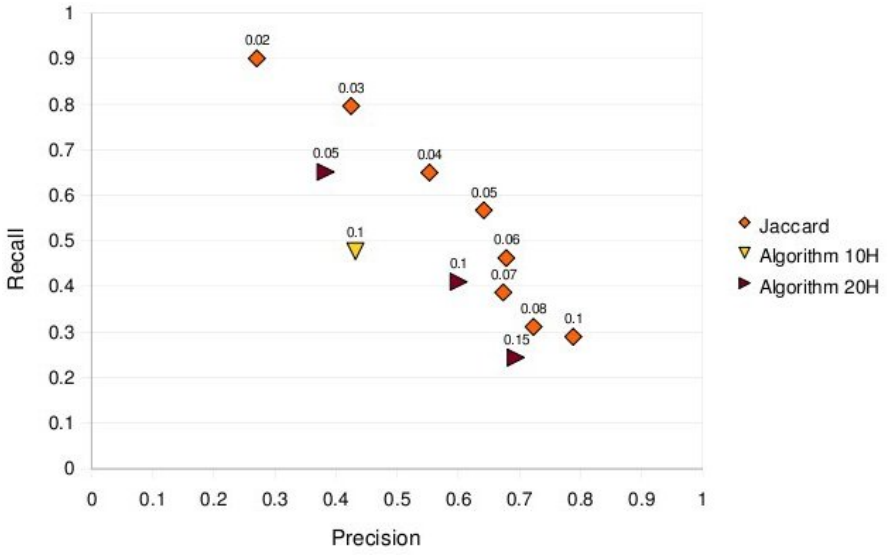


Figure 7.7: Precision vs. recall.

[5]The threshold is represented as a label over each point in the scatterplot.

[6]This is the reason why only one point of the 10 hash algorithm is represented on the scatterplot.

7.5 Recommendation in the MURPESS scenario

Despite the recommendation algorithm presented in this chapter is based on a centralized infrastructure, in this section we give a flavor of how the same approach can be applied in passive pervasive systems as well.

Sketch generation In the MURPESS scenario mentioned in the introduction of this thesis, it is easy to imagine that, instead of a list of contacts $L(A)$, each MURPESS will carry a more general social profile $S(A)$ where each entry in $S(A)$ represents a social information of user A (e.g., sex, political views, hobbies, etc. . .). Recalling section 7.3.1, in order to produce a sketch $sk(a)$ from the list of contacts $L(A)$ of user A, each entry in $L(A)$ must fall in $[n]$. But this principle applies to $S(A)$ as well since, as an example, each entry can be associated to the binary representation of the ASCII characters used to fill the entry. Thus, the same kind of sketch $sk(a)$ can be generated in the MURPESS scenario based on the social profile $S(A)$ carried by the user A.

Sketch exchange In the SMS application, sketches are exchanged using the residual space of Short Messages. Thus, user A and user B, both friends of user C, send their sketches, appended to Short Messages, to user C. Once user C receives both sketches, she is able to derive possible social relationships between user A and B by applying the recommendation algorithm described in section 7.3.4. In the MURPESS scenario the sketch exchange is enabled by the short-range communication techniques supported by MURPESSes (Bluetooth, NFC, RfID). In particular, it is suitable that most of friends occasionally get close to each other. Thus, at some point in time, user C will meet user A and the same will happen between user C and user B. Consequently, as in the SMS application, user A and B will be able to share their sketches with user C choosing the most appropriate communication technique.

Recommendation algorithm Since the SMS application and MURPESSes both produce sketches with the same properties, the recommendation algorithm will be the same. Thus, it will be based on the approximation of the Jaccard Coefficient described in section 7.3.1 and the recommendation algorithm described in 7.3.4.

7.6 Conclusion

In this chapter we have presented a fully decentralized approach for recommending new contacts in the social network of mobile phone users. The application does not assume any centralized coordination: it transparently collects

and processes user information that is accessible in any mobile phone, such as the *log of calls*, the *list of contacts* or the *inbox/outbox of short messages* and exchanges it with other users. This information is used to recommend new friendships to other users. We demonstrated the effectiveness of our approach with a set of experiments. In particular we demonstrated how the Jaccard coefficient is a good estimator of social relationships. We provided a succinct representation of contact lists on which our algorithm well approximates the Jaccard coefficient computed on the original lists. As a result, recommendations given using our fully decentralized approach competes both, in terms of precision and recall, with recommendations given using the Jaccard coefficient.

Conclusion

This thesis has focused on problems and solutions emerging from information gathering in resource constrained wireless networks. We have investigated the issues related to information gathering in the two most representative areas, i.e., Wireless Sensor Networks and Passive Pervasive Systems. We presented the main differences characterizing WSNs and Passive Pervasive Systems and showed how the issues related to information gathering tightly differ between these two research areas. As a consequence of these observations, we organized the thesis in two main parts.

Part I In Wireless Sensor Networks we identified the power management as the main obstacle for information gathering. In particular, the energy consumption of radio transceivers, required for data transmission and reception, quickly drains the batteries of sensor nodes and, thus, reduces the network lifetime. We outlined how the energy awareness in the software running on sensor nodes is the most effective solution facing the obstacle of energy consumption in WSNs. Energy awareness operates at different levels. At the application level, Sensor Selection and Data Aggregation both try to reduce the overall quantity of data gathered by the network. Sensor Selection algorithms allow to select which nodes should participate in the sensing task to reconstruct the signal with a sufficiently low error. Data Aggregation techniques aggregate or filter data sensed by nodes that are close to each other or in the same region of interest. In communication based energy awareness, energy-aware routing protocols aim at reducing the number of hops and/or transmissions needed to propagate the sensed data from a sensing node to a collection point. On the other hand, energy-aware MAC protocols reduce the energy consumption of radio transceiver by leaving the radio into sleep state as soon as reception or transmission are not required.

In chapter 2 we presented a new Sensor Selection technique, namely Adaptive Random Selection, an adaptive random spatial sensor selection protocol.

Based on the neighborhood information gathered through the CTP data collection protocol, ARS can compute locally adapted values of the probability of activation. Each node in ARS can then decide autonomously whether to participate in the sensing task or not depending on the outcome of a random throw. We demonstrated the capability of ARS to provide for good sensing coverage of the region of interest while limiting the number of active nodes.

In chapter 3 we described the design and implementation of DISSense, an adaptive, low-power communication protocol for WSNs-based periodical environmental monitoring applications. We tested DISSense on both a testbed and the TOSSIM WSN simulator. The experimental results show that DISSense can guarantee for high data delivery and, thanks to its power-efficiency, it is able to operate a Tmote Sky-based WSN for several years.

In the last chapter of the first part, we combined the two previously presented solutions. We observed how ARS greatly improves the duty cycle performance of the DISSense protocol. The reason relies in the lower traffic that, thanks to the specific design and the adaptive behavior of DISSense, reduces the active time scheduled for data gathering and consequently, the protocol's duty cycle.

Part II The second part of the thesis has focused on information gathering in Passive Pervasive Systems. We described the main differences in terms of network infrastructure, traffic pattern, communication type and memory availability with respect to WSNs and introduced the concept of *Opportunistic Networking* in such systems. These differences have led to a problem formulation that sharply differs from the WSN one. In particular, information gathering in Passive Pervasive Systems focuses on how to provide services for the emerging applications, taking into account the opportunistic networking behavior and memory limitations characterizing passive tags.

In chapter 6 we presented a fully decentralized RfID-based recommendation system that relies on a model of user behavior to capture important trends in real user data derived from commercial recommendation systems. The proposed model is simple enough as to allow the statistical estimation of parameters from real user activity logs. We demonstrated, through experimental evaluation, that the resulting recommendation strategy achieved a performance that is comparable to that of state-of-art centralized solutions.

In chapter 7 we presented a fully decentralized approach for recommending new contacts in the social network of mobile phone users. The presented application transparently collects and processes user information that is accessible in any mobile phone, such as the *log of calls*, the *list of contacts* or the *inbox/outbox of short messages* and exchanges it with other users. This in-

formation is used to recommend new friendships to other users. Experiments demonstrated how recommendations given, based on the succinct representation of the contact lists exchanged between users, competes both in terms of precision and recall, with the recommendations made managing whole sets of lists of contacts.

Future Work As for the first part, a deeper investigation of the interactions between ARS and DISSense need to be made. To this aim, the implementation of CTP, DISSense and ARS on the Castalia 3.1 simulation framework is ongoing. Additionally, DISSense needs to be validated on wider testbeds (e.g., Motelab) so as to test its performance in more challenging scenarios. Finally, in [108] we presented a new metric, dubbed Expected Network Delivery (END), that quantifies the delivery performance that a collection protocol can be expected to achieve given a network topology. We expect to provide a comparison between DISSense and other existing solutions such as CTP with BoX-MAC-2, Koala or Dozer in terms of duty cycle, DDR and latency with different network topologies classified following the END index.

For what concerns the second part of the thesis, the model presented in the RfID-based recommendation system can be extended by taking into account the changes, over time, of items' popularity. A best selling book might be much less popular within the next month, as the initial wave of interest fades. Proposing simple ageing mechanisms, while keeping the model simple enough is an interesting point. Additionally, the model has been evaluated with intra-cluster items recommendation; a more general inter-cluster extension is an interesting issue. As for the second contribution, a proof-of-concept has already been implemented on the Sim Application Toolkit on the Gemalto Developer Suite [5], but an NFC-based implementation that better approximates the MURPESS scenario would represent an interesting issue.

Bibliography

[1] Castalia homepage. http://castalia.npc.nicta.com.au. 28, 30, 67

[2] Duracell alkaline technical bulletin. chapter 5. p.9. http://www1.duracell.com/oem/Pdf/others/ATB-full.pdf. 54

[3] eMarketer forecasts online. http://technokitten.blogspot.com/2008/05/big-growth-predicted-for-mobile-social.html. 109

[4] Fronts project homepage. http://fronts.cti.gr/. 1

[5] Gemalto homepage. http://www.gemalto.com/. 131

[6] Memsic wireless modules. http://www.memsic.com. 48, 52

[7] Motelab project homepage. http://motelab.eecs.harvard.edu/. 65

[8] Omnet - project website. www.omnetpp.org. 67

[9] Reality mining project. http://reality.media.mit.edu/. 121

[10] Rpl draft. http://tools.ietf.org/html/draft-ietf-roll-rpl-18. 21

[11] Tinynode wireless modules. http://tinynode.com. 24, 48

[12] Tinyos web site. http://www.tinyos.net. 22, 48

[13] Tossim tutorial. http://docs.tinyos.net/index.php/TOSSIM. 58

[14] *Decentralized SLAM for pedestrians without direct communication*, 2007. 79

[15] Amazon web site. http://www.amazon.com, 2008. 83

[16] Jyngle web site. http://www.jyngle.com/, 2008. 110

[17] Netflix web site. http://www.netflix.com, 2008. 83

[18] Peekamo web site: http://peekamo.com/, 2008. 110

[19] Market intelligence on messaging and marketing. http://www.lloydsmiu.com/pdf/Jun-2005/16/mma_sample_issue_june.pdf, 2009. 110

[20] W3C workshop on the future of social networking. http://planb.nicecupoftea.org/2009/01/14/w3c-workshop-on-the-future-of-social-networking/, 2009. 110

[21] Airbus signs contract for high-memory RFID tags. RFID journal. http://www.rfidjournal.com/article/view/7323, 2010. 92

[22] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.*, 17(6):734–749, 2005. 84, 86, 91

[23] W. Aiello, F. Chung, and L. Lu. A random graph model for massive graphs. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 171–180. ACM, 2000. 112

[24] W. Aiello, F. Chung, and L. Lu. Random evolution in massive graphs. In Bob Werner, editor, *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 510–521, 2001. 112

[25] K. Akkaya and M. Younis. A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, 3(3):325 – 349, 2005. 20

[26] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *Computer Networks*, 38(4):393–422, 2002. 12

[27] J. N. Al-Karaki and A. E. Kamal. Routing techniques in wireless sensor networks: a survey. *IEEE Wireless Communications*, 11(6):6–28, December 2004. 20

[28] N. Alon, B. Awerbuch, Y. Azar, and B. Patt-Shamir. Tell me who i am: an interactive recommendation system. In *SPAA '06: Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 1–10, New York, NY, USA, 2006. ACM. 85

[29] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A line in the sand: a wireless sensor network for target detection, classification, and tracking. *Computer Networks*, 46(5):605 – 634, 2004. Military Communications Systems and Technologies. 11

[30] B. Awerbuch, B. Patt-Shamir, D. Peleg, and M. R. Tuttle. Improved recommendation systems. In *SODA*, pages 1174–1183, 2005. 85

[31] Y. Azar, A. Fiat, A. R. Karlin, F. McSherry, and J. Saia. Spectral analysis of data. In *STOC*, pages 619–626, 2001. 84

[32] O. Babaoglu, G. Canright, A. Deutsch, G. A. Di Caro, F. Ducatelle, L. M. Gambardella, N. Ganguly, M. Jelasity, R. Montemanni, A. Montresor, and T. Urnes. Design patterns from biology for distributed computing. *ACM Trans. Auton. Adapt. Syst.*, 1(1):26–66, 2006. 83

[33] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, May 1999. 99

[34] A. Baggio. Wireless sensor networks in precision agriculture. In *Proceedings of the First Workshop on Real-World Wireless Sensor Networks (REALWSN 2005)*, Stockholm, Sweden, June 20-21 2005. 14

[35] G. Barrenetxea, F. Ingelrest, G. Schaefer, and M. Vetterli. Wireless Sensor Networks for Environmental Monitoring: The SensorScope Experience. In *The 20th IEEE International Zurich Seminar on Communications (IZS 2008)*, 2008. Invited paper. 54

[36] G. Barrenetxea, F. Ingelrest, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange. Sensorscope: Out-of-the-box environmental monitoring. ipsn, 2008. 12

[37] S. Berkovsky, T. Kuflik, and F. Ricci. Distributed collaborative filtering with domain specialization. In *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pages 33–40, New York, NY, USA, 2007. ACM. 84

[38] J. Beutel, S. Gruber, S. Gubler, A. Hasler, M. Keller, R. Lim, L. Thiele, C. Tschudin, and M. Yücel. The permasense remote monitoring infrastructure, 2009. ISSW 09 Europe. 11

[39] J. Beutel, S. Gruber, A. Hasler, R. Lim, A. Meier, C. Plessl, I. Talzi, L. Thiele, C. Tschudin, M. Woehrle, and M. Yuecel. Permadaq: A scientific instrument for precision sensing and data recovery in environmental extremes. In *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, IPSN '09, pages 265–276, Washington, DC, USA, 2009. IEEE Computer Society. 11

[40] B. L. D. Bezerra and F. de Assis Tenorio de Carvalho. Symbolic data analysis tools for recommendation systems. *Knowl. Inf. Syst.*, 26:385–418, 2010, on-line. 82

[41] T. Bohman, C. Cooper, and A. M. Frieze. Min-wise independent linear permutations. *Electr. J. Comb*, 7, 2000. 115

[42] A. Boulis. Castalia: revealing pitfalls in designing distributed algorithms in wsn. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems (SenSys 2007)*, pages 407–408, Sydney, Australia, 6-7 November 2007. Demonstration session. 67

[43] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. In *ACM Symposium on the Theory of Computing*, New York, NY, USA, 1998. 115, 118

[44] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. In *Proc. of the World Wide Web Conference*, 1997. 115, 118

[45] T. Brooke and J. Burrell. From ethnography to design in a vineyard. In *Proceedings of the 2003 conference on Designing for user experiences*, DUX '03, pages 1–4, New York, NY, USA, 2003. ACM. 12

[46] P. Buonadonna, D. Gay, J. M. Hellerstein, W. Hong, and S. Madden. Task: Sensor network in a box. In Sebnem Baydere Erdal Cayirci and Paul Havinga, editors, *Proceedings of the Second IEEE European Workshop on Wireless Sensor Networks and Applications (EWSN 2005)*, Istanbul. Turkey, February 2005. 14

[47] N. Burri, P. von Rickenbach, and R. Wattenhofer. Dozer: Ultra-low power data gathering in sensor networks. In *Proceedings of the Sixth International Conference on Information Processing in Sensor Networks*, Cambridge, MA, USA, April 2007. 6, 23, 28, 43, 54, 61, 63

[48] M. Ceriotti, L. Mottola, G. P. Picco, A. L. Murphy, S. Guna, M. Corra, M. Pozzi, D. Zonta, and P. Zanon. Monitoring heritage buildings with wireless sensor networks: The torre aquila deployment. In *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, IPSN '09, pages 277–288, Washington, DC, USA, 2009. IEEE Computer Society. 12, 54

[49] I. Chatzigiannakis, U. Colesanti, S. Kontogiannis, G. Leshem, A. Marchetti-Spaccamela, J. Mehler, G. Persiano, P. Spirakis, and A. Vitaletti. Murpess - multi radio pedestrian energy scavenging sensor network. In *eChallenges2010*, 2010. 1

[50] U. Colesanti and S. Santini. A performance evaluation of the collection tree protocol based on its implementation for the castalia wireless sensor

networks simulator. Technical Report 681, ETH Zurich, August 2010. 5, 6, 67

[51] U. Colesanti, S. Santini, and A. Vitaletti. DISSense: An adaptive ultra-low power communication protocol for wireless sensor networks. DCOSS 2011 (to be presented), 2011. 5

[52] U.M. Colesanti and S. Santini. Ctp-castalia project. http://code.google.com/p/ctp-castalia/. 37

[53] Gartner Consulting. Gartner says mobile messages to surpass 2 trillion messages in major markets in 2008, 2008. 110

[54] G. Cormode and S. Muthukrishnan. What's hot and what's not: tracking most frequent items dynamically. *ACM Transactions on Database Systems*, 30(1):249–278, 2005. 93

[55] R. Cöster and M. Svensson. Incremental collaborative filtering for mobile devices. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 1102–1106, New York, NY, USA, 2005. ACM. 85

[56] C. Decker, U. Kubach, and M. Beigl. Revealing the retail black box by interaction sensing. In *ICDCS Workshops*, pages 328–333, 2003. 82

[57] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004. 84, 91, 99, 102

[58] P. Drineas, I. Kerenidis, and P. Raghavan. Competitive recommendation systems. In *STOC*, pages 82–90, 2002. 84

[59] N. Eagle and A. Pentland. Reality mining: sensing complex social systems. *Personal and Ubiquitous Computing*, 10(4):255–268, 2006. 121

[60] A. El-Hoiydi and J.-D. Decotignie. Wisemac: an ultra low power mac protocol for the downlink of infrastructure wireless sensor networks. In *Proceedings of the Ninth International Symposium on Computers and Communications 2004 Volume 2 (ISCC"04) - Volume 02*, ISCC '04, pages 244–251, Washington, DC, USA, 2004. IEEE Computer Society. 23

[61] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi. In-network aggregation techniques for wireless sensor networks: a survey. *Wireless Communications, IEEE [see also IEEE Personal Communications]*, 14(2):70–87, 2007. 18

[62] A. Fog. Sampling methods for wallenius' and fisher's noncentral hypergeometric distributions. *Communications in Statistics - Simulation and Computation*, 37(2):241 – 257, 2008. 87

[63] R. Fonseca, O. Gnawali, K. Jamieson, S. Kim, P. Levis, and A. Woo. TinyOS Enhancement Proposal (TEP) 123: The Collection Tree Protocol (CTP). http://www.tinyos.net/tinyos-2.x/doc/pdf/tep123.pdf. 28

[64] O. Gnawali, R. Fonseca, K. Jamieson, and P. Levis. Ctp: Robust and efficient collection through control and data plane integration. Technical report, The Stanford Information Networks Group (SING), 2008. available at: http://sing.stanford.edu/pubs/sing-08-02.pdf. 20, 28

[65] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys 2009)*, November 2009. 22, 46, 67

[66] M. Gori and A. Pucci. Itemrank: a random-walk based scoring algorithm for recommender engines. In *IJCAI'07: Proceedings of the 20th international joint conference on Artifical intelligence*, pages 2766–2771. Morgan Kaufmann Publishers Inc., 2007. 86

[67] K. Groechenig and T. Strohmer. *Nonuniform Sampling: Theory and Practice*, chapter Numerical and Theoretical Aspects of Non-Uniform Sampling of Band-Limited Images. Kluwer Academic / Plenum Publishers, 2001. 26

[68] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh. Energy-efficient surveillance system using wireless sensor networks. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, MobiSys '04, pages 270–283, New York, NY, USA, 2004. ACM. 11

[69] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of 33rd Hawaii International Conference on System Sciences (HICSS '00)*, Hawaii, HW, USA, 2000. 21

[70] Z. Huang, D. Zeng, and H. Chen. A comparison of collaborative-filtering recommendation algorithms for E-commerce. *IEEE Intelligent Systems*, 22(5):68–78, 2007. 84

[71] P. Indyk. A small approximately min-wise independent family of hash functions. In *SODA*, 1999. 115

[72] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. Netw.*, 11:2–16, February 2003. 20

[73] J. M. Kleinberg and M. Sandler. Convergent algorithms for collaborative filtering. In *ACM Conference on Electronic Commerce*, pages 1–10, 2003. 84, 86

[74] P. Kourouthanasis, D. Spinellis, G. Roussos, and G. Giaglis. Intelligent cokes and diapers: MyGrocer ubiquitous computing environment. In *First International Mobile Business Conference*, pages 150–172, July 2002. 82, 84

[75] L. Krishnamurthy, R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, N. Kushalnagar, L. Nachman, and M. Yarvis. Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, SenSys '05, pages 64–75, New York, NY, USA, 2005. ACM. 11

[76] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Recommendation systems: A probabilistic analysis. *J. Comput. Syst. Sci.*, 63(1):42–61, 2001. 84, 86

[77] K. Langendoen and A. Meier. Analyzing mac protocols for low data-rate applications. *ACM Trans. Sen. Netw.*, 7:10:1–10:34, August 2010. 22, 23, 46

[78] C. W. Leung, S. Chan Chi-fai, and F. Chung. A collaborative filtering framework based on fuzzy association rules and multiple-level similarity. *Knowl. Inf. Syst.*, 10(3):357–381, 2006. 105

[79] P. Levis and G. Tolle. Tinyos enhancement proposal (TEP) 118: Dissemination of small values. http://www.tinyos.net/tinyos-2.x/doc/pdf/tep118.pdf. 28

[80] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *J. Am. Soc. Inf. Sci. Technol.*, 58(7):1019–1031, 2007. 112, 114

[81] S. Lin, B. Arai, D. Gunopulos, and G. Das. Region Sampling: Continuous Adaptive Sampling on Sensor Networks. In *Proceedings of the 24th IEEE International Conference of Data Engineering (ICDE 2008)*, pages 794–803, Cancun, Mexico, April 2008. 16, 18, 25, 26

[82] G. Linden, B. Smith, and J. York. Industry report: Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Distributed Systems Online*, 4(1), 2003. 83, 91

[83] S. Lindsay, C. S. Raghavendra, and K. M. Sivalingam. Data gathering in sensor networks using the energy delay metric. In *Proceedings of the 15th International Parallel & Distributed Processing Symposium*, IPDPS '01, pages 188–, Washington, DC, USA, 2001. IEEE Computer Society. 21

[84] S. Lindsey and C. S. Raghavendra. PEGASIS: Power-efficient gathering in sensor information systems. In *Aerospace Conference Proceedings, 2002. IEEE*, volume 3, pages 3–1125–3–1130 vol.3, 2002. 21

[85] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, 36:131–146, December 2002. 19

[86] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, WSNA '02, pages 88–97, New York, NY, USA, 2002. ACM. 11, 12

[87] M. Mamei and F. Zambonelli. Pervasive pheromone-based interaction with rfid tags. *ACM Trans. Auton. Adapt. Syst.*, 2(2):4, 2007. 79, 83

[88] A. Manjeshwar and D. P. Agrawal. TEEN: a routing protocol for enhanced efficiency in wireless sensor networks. In *Parallel and Distributed Processing Symposium., Proceedings 15th International*, pages 2009–2015, 2001. 21

[89] A. Manjeshwar and D. P. Agrawal. APTEEN: a hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks. In *Parallel and Distributed Processing Symposium., Proceedings International, IPDPS 2002, Abstracts and CD-ROM*, pages 195–202, 2002. 21

[90] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi. The flooding time synchronization protocol. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 39–49, New York, NY, USA, 2004. ACM. 51

[91] Farokh Marvasti, editor. *Nonuniform Sampling: Theory and Practice*. Information Technology: Transmission, Processing and Storage. Springer, Berlin / Heidelberg, 2001. ISBN: 978-0-306-46445-4. 28, 70

[92] X. Meng, L. Li, T. Nandagopal, and S. Lu. Event contour: An efficient and robust mechanism for tasks in sensor networks. Technical report, UCLA, 2004. 16, 18

[93] Carl D. Meyer, editor. *Matrix analysis and applied linear algebra*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000. 84

[94] B. N. Miller, I. Albert, S. K. Lam, J. A. Konstan, and J. Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *Intelligent User Interfaces*, pages 263–266, 2003. 83

[95] B. N. Miller, J. A. Konstan, and J. Riedl. Pocketlens: Toward a personal recommender system. *ACM Trans. Inf. Syst.*, 22(3):437–476, 2004. 111

[96] M. Mitzenmacher and E. Upfal. *Probability and Computing : Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, January 2005. 94, 96

[97] D. Moss and P. Levis. Box-macs: Exploiting physical and link layer boundaries in low-power networking. Technical report, Stanford University, 2008. 22, 23

[98] R. Musaloiu-E, C.-J.M. Liang, and A. Terzis. Koala: Ultra-low power data retrieval in wireless sensor networks. In *Information Processing in Sensor Networks, 2008. IPSN '08. International Conference on*, pages 421 –432, 2008. 23, 43, 61, 63

[99] S. Muthukrishnan. Data streams: Algorithms and applications. In *Foundations and Trends in Theoretical Computer Science, Now Publishers or World Scientific*, volume 1. 2005. 89

[100] M. E. J. Newman. The structure and function of complex networks, March 2003. 112, 114

[101] A. Nordio, C.F. Chiasserini, and E. Viterbo. Performance of linear reconstruction techniques with noise and uncertain sensor locations. *IEEE Transactions on Signal Processing*, 56(8):3535–3547, August 2008. 31

[102] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances In Cryptology - EUROCRYPT 1999*, pages 223–238. Springer-Verlag, 1999. 119

[103] J. A. Paradiso and T. Starner. Energy scavenging for mobile and wireless electronics. 4(1):18–27, 2005. 1

[104] J.A. Paradiso and T. Starner. Energy scavenging for mobile and wireless electronics. *Pervasive Computing, IEEE*, 4(1):18 – 27, 2005. 14

[105] L. Pelusi, A. Passarella, and M. Conti. Opportunistic Networking: Data Forwarding in Disconnected Mobile Ad Hoc Networks. *IEEE Communications Magazine*, 44(11):134–141, November 2006. 2

[106] M. Perillo, Z. Ignjatovic, and W. Heinzelman. An energy conservation method for wireless sensor networks employing a blue noise spatial sampling technique. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, pages 116–123, Berkeley (CA), USA, 2004. 17, 18

[107] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling Ultra-Low Power Wireless Research. In *Proceedings of the 4th International Conference on Information Processing in Sensor Networks: Special track on Platform Tools and Design Methods for Network Embedded Sensors (IPSN/SPOTS 2005)*, pages 364–369, April 2005. 13, 14, 54

[108] D. Puccinelli, O. Gnawali, S. Yoon, S. Santini, U. Colesanti, S. Giordano, and L. Guibas. The impact of network topology on collection performance. In *Proceedings of the 8th European Conference on Wireless Sensor Networks (EWSN 2011).*, 2011. 65, 131

[109] I. Rhee, A. Warrier, M. Aia, and J. Min. Z-mac: a hybrid mac for wireless sensor networks. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, SenSys '05, pages 90–101, New York, NY, USA, 2005. ACM. 23

[110] S. Rosset, C. Perlich, and B. Zadrozny. Ranking-based evaluation of regression models. *Knowl. Inf. Syst.*, 12(3):331–353, 2007. 84

[111] M. Roth and S. Wicker. Termite: ad-hoc networking with stigmergy. *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, 5:2937–2941 vol.5, Dec. 2003. 83

[112] S. Santini. *Adaptive Sensor Selection Algorithms for Wireless Sensor Networks*. PhD thesis, ETH Zurich, Zurich, Switzerland, October 2009. 17, 37, 38

[113] S. Santini and U. Colesanti. Adaptive Random Sensor Selection for Field Reconstruction in Wireless Sensor Networks. In *Proceedings of the 6th International Workshop on Data Management for Sensor Networks (DMSN 2009)*, August 2009. 5

[114] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001. 84, 91

[115] A. Silberstein, R. Braynard, and J. Yang. Constraint Chaining: On Energy-Efficient Continuous Monitoring in Sensor Networks. In *Proceedings of the 22nd ACM SIGMODInternational Conference on Management of Data*, pages 157–168, Chicago, IL, USA, June 2006. 16, 18

[116] K. Srinivasan, M. A. Kazandjieva, S. Agarwal, and P. Levis. The β-factor: measuring wireless link burstiness. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, SenSys '08, pages 29–42, New York, NY, USA, 2008. ACM. 19

[117] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler. An analysis of a large scale habitat monitoring application. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*, Baltimore, Maryland, USA, November 3-5 2004. 14

[118] I. Talzi, A. Hasler, S. Gruber, and C. Tschudin. Permasense: Investigating permafrost with a wsn in the swiss alps. In *Proceedings of the Fourth ACM Workshop on Embedded Networked Sensors (EmNets 2007)*, Cork, Ireland, June 25-26 2007. 14

[119] L. Tamine-Lechani, M. Boughanem, and M. Daoudontact. Evaluation of contextual information retrieval effectiveness: overview of issues and research. *Knowl. Inf. Syst.*, 2009, on-line. 105

[120] L. Terveen and D. W. McDonald. Social matching: A framework and research agenda. *ACM Trans. Comput.-Hum. Interact.*, 12(3):401–434, 2005. 111

[121] P. Tiano and C. Pardini, editors. *In Situ Monitoring Of Monumental Surfaces*. Edifir - Edizioni Firenze, October 2008. 12

[122] T. van Dam and K. Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the First International Conference on Embedded Networked Sensing Systems (SenSys'03)*, pages 171 – 180, New York, USA, 2003. 22, 23

[123] W. Wade. A grocery cart that holds bread, butter, and preferences. *New York Times*, Jan(16), 2003. 84

[124] J. Wang, J. Pouwelse, R. Lagendijk, and M. J. T. Reinders. Distributed collaborative filtering for peer-to-peer file sharing systems. In *21st Annual ACM Symposium on Applied Computing*, pages 1026–1030, 2006. 85

[125] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated coverage and connectivity configuration in wireless sensor. In *Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, Los Angeles (CA), USA, November 2003. 17, 18

[126] R. Willett, A. Martin, and R. Nowak. Backcasting: Adaptive sampling for sensor networks. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN 2004)*, pages 124–133, Houston (TX), USA, April 2004. 17, 18

[127] B. Xie, P. Han, F. Yang, R. Shen, H. Zeng, and Z. Chen. DCFLA: A distributed collaborative-filtering neighbor-locating algorithm. *Information Sciences*, 177(6):1349–1363, 2007. 85

[128] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, MobiCom '01, pages 70–84, New York, NY, USA, 2001. ACM. 21

[129] F. Ye, G. Zhong, J. Cheng, S. Lu, and L. Zhang. Peas: A robust energy conserving protocol for long-lived sensor networks. pages 28–37, 2003. 17, 18

[130] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292 – 2330, 2008. 12

[131] L. Yu, N. Wang, and X. Meng. Real-time forest fire detection with wireless sensor networks. In *Wireless Communications Networking and Mobile Computing 2005 Proceedings 2005 International Conference on*, volume 2, pages 1214–1217. Ieee, 2005. 11

[132] P. Zerfos, X. Meng, S.H.Y. Wong, V. Samanta, and S. Lu. A study of the short message service of a nationwide cellular network. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 263–268, New York, NY, USA, 2006. ACM. 114, 119