MOBILITY IN WIRELESS NETWORKS: FRIEND OR FOE

Network Design and Control in the Age of Mobile Computing

THÈSE N^{O} 3456 (2006)

PRÉSENTÉE LE 7 AVRIL 2006
À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

Laboratoire pour les communications informatiques et leurs applications 1

SECTION DES SYSTÈMES DE COMMUNICATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Jun LUO

Master's Degree in Electrical Engineering, Tsinghua University, Chine
et de nationalité chinoise

acceptée sur proposition du jury:

Prof. E. Telatar, président du jury
Prof. J.-P. Hubaux, directeur de thèse
Prof. M. Grossglauser, rapporteur
M. C. Perkins, rapporteur
Prof. M. Srivastava, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Lausanne, EPFL
2006

To my wife Zhiling

— for her unconditional love, constant support, and endless patience

To my parents

— for their love and support across the two continents

Acknowledgements

First and foremost, I want to thank Prof. Jean-Pierre Hubaux, for his excellent guidance through the long march of my PhD research, for his help to clear my thoughts and grasp problems from the right sides, and for the great deal of freedom he allowed me in my research work.

I would also like to thank to Prof. Patrick Eugster, for our fruitful collaborations, and for his great support during the initial stage of my PhD research.

I am grateful to two professors: Pierre Brémaud and Emre Telatar; they, with their talented lectures, demonstrated to me how beautiful mathematical formulas can be when applied to engineering problems.

Many thanks to all the colleagues and friends with whom I shared an office or floor: Aleddine El Fawal, Dr. Božidar Radunovic, Dr. Emre Koksal, Gianluca Rizzo, Hung Nguyen, Dr. Imad Aad, Jacques Panchard, Prof. Jean-Yves Le Boudec, Dr. Ljubica Blazevic, Maciej Kurant, Mario Cagalj, Mark Felegyhazi, Mathilde Durvy, Prof. Matthias Grossglauser, Maxim Raya, Michal Piorkowski, Dr. Milan Vojnovic, Naouel Ben Salem, Natasa Sarafijanovic-Djukic, Dr. Olivier Dousse, Prof. Patrick Thiran, Ruben Merz, Slavisa Sarafijanovic, Dr. Sonja Buchegger and Dr. Srdjan Čapkun. Special thanks go to my former office-mate Milan, who dragged me out of the jam of stochastic process, also to Naouel, who helped me in many cases to overcome my inabilities in French (e.g., the writing of the "Version abrégée").

Many thanks go to our secretaries, Angela Devenoge, Danielle Alvarez and Holly Cogliati, and our system administrators, Jean-Pierre Dupertuis, Marc-André Lüthi and Phillipe Chammartin; I bothered them frequently, but they never lost patience with me.

Abstract

Wireless networking technologies allow computing devices to move while keeping them online, but the device mobility poses considerable technical challenges. Network designers, on the one hand, consider mobility to be harmful because it leads to dynamic and unpredictable links among devices. Compared with traditional wired networks, it indeed becomes much more difficult to design network control functions, such as routing path maintenance, in an efficient way under mobility. On the other hand, some researchers also realize that, by involving mobility into network design (i.e., controlling mobility in a desirable way), mobility can improve the performance of wireless networks. Inspired by these two seemingly contradictory thoughts, we focus this thesis on the understanding, tackling, and exploiting of **mobility** in wireless networks.

In the first part of this thesis, we take an antagonistic view on mobility and try to cope with its harm. We are concerned with the design of reliable protocols that support many-to-many communications (also called group communication service) under random node mobility constraints in mobile ad hoc networks. Our approach deviates from the conventional point of view (where protocols tend to predict the consequence of mobility and adapt to it), i.e., we "fight fire with fire" by exploiting the non-deterministic nature of ad hoc networks. In line with this concept, we propose a group communication system, Pilot, that guarantees reliability in a probabilistic manner. Pilot includes Route Driven Gossip (RDG, a probabilistic multicast protocol) and two services built upon: Probabilistic quorum systems for Ad-hoc Networks (Pan) to provide reliable storages and Reliable RDG (R^2DG) for supporting multicast streams. Our proposal is innovative because 1) it makes reliable group communication service available in ad hoc networks, and 2) it also provides predictability of the service such that we can fine tune the protocol parameters to obtain the desired tradeoff between reliability and overhead. In addition, we also demonstrate the application of Pilot by proposing a distributed certification authority, DICTATE, that makes use of its services.

In the second part of this thesis, we look at the favorable aspects of mobility. We involve controllable mobility in designing energy-conserving protocols in static wireless networks (in particular wireless sensor networks). We observe that the many-to-one transmission style in sensor networks poses a major threat to the network lifetime, because the sensor nodes located near a sink have to relay data for a large part of the network and thus deplete their batteries very quickly. Existing energy efficient/conserving routing protocols cannot completely solve the problem as they only balance the load among nodes whose distances from a sink are roughly the same. Inspired by a recent trend of using mobile sinks in sensor networks, we propose a solution that suggests the sinks be mobile; in this way, the role of "hot spots" (i.e., the nodes around the sinks) is distributed over time, which evens out the load. Our proposal differs from the existing approach that we term *mobile relay* approach, in that the sink does not rely only on mobility to retrieve data from sensors; the data collection procedure continues through multi-hop routing wherever a sink stays. We hence optimize data collection protocols by taking both sink mobility and multi-hop routing into account. In order to evaluate the practicality of our theoretical proposal, we further engineer a routing protocol, MobiRoute, that effectively supports sink mobility. Through intensive simulations in TOSSIM with a real implementation, we prove the feasibility of our mobile sink approach by demonstrating the improved network lifetime in several deployment scenarios.

Keywords: Mobile ad hoc networks, wireless sensor networks, reliable multicast, network lifetime, data collection, quorum systems, certification authority.

Version Abrégée

La technologie sans fil permet aux machines d'être mobiles tout en restant en ligne, cependant, la mobilité pose de gros problèmes en ce qui concerne la conception et le contrôle du réseau. D'une part, les concepteurs de réseaux considèrent la mobilité comme négative parce qu'elle mène à des liens radio dynamiques et imprévisibles entre les machines sans fil ; il est en effet beaucoup plus délicat de concevoir les opérations du réseau, par exemple le routage, dans un contexte de mobilité. D'autre part, certaines équipes de recherche se sont rendues compte qu'en considérant la mobilité dans la conception du réseau (c.-à-d., utiliser la mobilité à bon escient), il est possible d'améliorer la performance des réseaux sans-fil. Dans cette thèse de doctorat, nous nous concentrons sur la compréhension, l'analyse et l'exploitation de la **mobilité** dans les réseaux sans fil.

Dans la première partie de cette thèse, nous considérons la mobilité comme négative et essayons de nous en accommoder. Nous nous intéressons à la conception de protocoles fiables qui permettent des communications plusieurs-à-plusieurs (également appelées service de communications de groupe) et ceci dans le cadre de réseaux ad hoc avec une mobilité aléatoire des nœuds. Notre approche diffère du point de vue conventionnel (celui où les protocoles tendent à prévoir l'effet de la mobilité et à s'y adapter), vu que nous "combattons le feu avec le feu" en exploitant la nature non-déterministe des réseaux ad hoc. Dans cette optique, nous proposons PILOT, un système de communication de groupe qui garantit la fiabilité d'une façon probabiliste. PILOT inclut un protocole probabiliste de multicast RDG (Route Driven Gossip) et deux autres services construits en se basant dessus : PAN (Probabilistic quorum systems for Ad-hoc Networks) qui fournit un moyen de sauvegarde fiable et R^{2}DG (Reliable RDG) qui supporte les flux multicast. Notre proposition est innovatrice puisque 1) elle rend le service fiable de communication de groupe disponible dans les réseaux ad hoc, et 2) elle fournit la prévisibilité du service de façon à ce qu'il soit possible d'ajuster les paramètres du protocole pour obtenir le compromis désiré entre la fiabilité et le coût. En plus, nous démontrons que PILOT est applicable en proposant une autorité de certification distribuée qui utilise ses services.

Dans la deuxième partie de cette thèse, nous considérons les aspects positifs de la mobilité. Nous tenons compte de la mobilité contrôlable lors de la conception de protocoles à conservation d'énergie pour les réseaux sans fil statiques (en particulier les réseaux de senseurs sans fil). Nous avons observé que les communications du genre plusieurs-à-un posent un problème pour les réseaux de senseurs, en ce qui concerne la durée de vie du réseau, car les senseurs qui sont proches d'un collecteur d'informations doivent relayer des données pour une grande partie du réseau, ce qui tend à épuiser leurs batteries très rapidement. Les protocoles de routage à conservation d'énergie existants ne peuvent pas résoudre le problème complètement puisqu'ils ne font que balancer la charge parmi les nœuds qui sont plus ou moins à la même distance du collecteur de données. Inspiré par une tendance récente d'employer des collecteurs de données mobiles dans les réseaux de senseurs, nous proposons dans cette thèse de doctorat une solution qui suggère que les collecteurs de données soient mobiles ; ainsi, le rôle des nœuds qui se trouvent autour des collecteurs de données est réparti sur le temps, ce qui a pour but d'égalise la charge. Notre proposition diffère de l'approche existante, que nous appelons approche à relais mobile, par le fait que le collecteur de données ne se base pas uniquement sur la mobilité pour recueillir les données des senseurs ; le procédé de collecte de données utilise également le routage multi-hop pour permettre à tous les senseurs d'envoyer leurs données et ce indépendamment de la position dans le réseau du collecteur de données mobile. Nous optimisons par conséquent les protocoles de collecte de données en tenant compte de la mobilité du collecteur de données et du routage multi-hop. Afin d'évaluer le caractère pratique de notre proposition théorique, nous développons le protocole de routage MobiRoute, qui supporte efficacement la mobilité des collecteurs de données. Par des simulations intensives dans TOSSIM, nous prouvons la faisabilité de notre approche de collecteurs de données mobiles en démontrant l'amélioration de la durée de vie du réseau et ce dans plusieurs scénarios de déploiement.

Mots-Clés: Réseaux ad hoc mobiles, réseaux de senseurs sans fil, multicast fiable, temps de vie d'un réseau, collection de donnés, systèmes de quorum, authorité de certification.

Contents

Acknowledgements **v**

Abstract **vii**

Version Abrégée **ix**

1 Introduction **1**

I Fighting Fire with Fire – Probabilistic Protocols to Cope with Mobility in Large-scale Ad Hoc Networks 5

2 PILOT: Probabilistic Group Communication System **7**
- 2.1 Introduction . . . 7
- 2.2 Related Work . . . 8
  - 2.2.1 Deterministic Reliable Multicast in Wired Networks . . . 8
  - 2.2.2 Gossiping in Wired Networks . . . 9
  - 2.2.3 Gossiping in Ad Hoc Networks . . . 9
  - 2.2.4 Stateless Multicast . . . 10
  - 2.2.5 Probabilistic Quorum Systems . . . 10
  - 2.2.6 Data Management in Ad Hoc Networks . . . 11
- 2.3 Goals and Assumptions . . . 11
  - 2.3.1 Model . . . 12
  - 2.3.2 Problem Statement . . . 12
- 2.4 PILOT: Probabilistic Group Communication System . . . 13
  - 2.4.1 Overview: Layered Architecture of PILOT . . . 13
  - 2.4.2 RDG: Basic PILOT Multicast Protocol . . . 14
  - 2.4.3 R^2DG: Continuous Packet Multicasting Service . . . 18
  - 2.4.4 PAN: Reliable Data Sharing Service . . . 19
  - 2.4.5 Examples of Protocol Operations . . . 21

2.4.6 Comparing PILOT with Randomized Database Group 23
2.5 Analysis 23
2.5.1 Model 23
2.5.2 Stochastic Behavior of RDG 25
2.5.3 Stochastic Behavior of PAN 27
2.6 Simulations 29
2.6.1 Model and Parameters 29
2.6.2 Single Packet Dissemination Reliability $\mathcal{R}_{ds}$ 30
2.6.3 Comparing Anonymous Gossip (AG) and RDG 30
2.6.4 Continuous Packet Dissemination Reliability $\mathcal{R}_{dc}$ and Network Load $\mathcal{N}_l$ . . . 31
2.6.5 Impact of λ_o on PAN Performance 32
2.6.6 Access Reliability $\mathcal{R}_{da}$ and Network Load $\mathcal{N}_l$ 33
2.6.7 Adapting PAN to a New Environment 34
2.6.8 Sensitivity to Server Unavailability p_e 34
2.7 Conclusion 35

3 DICTATE: Distributed Certification Authority with Probabilistic Freshness 37
3.1 Introduction 37
3.2 Related Work 38
3.2.1 PKIs in Wired Networks 39
3.2.2 PKIs in Ad Hoc Networks 39
3.2.3 Lessons from the Past 40
3.3 Goal and Model 41
3.3.1 Problem Statement 41
3.3.2 System and Adversary Model 42
3.4 Our Solution: Joint Authority and DICTATE 42
3.4.1 Overview of PILOT 44
3.4.2 Protocol Overview 45
3.4.3 Protocol Rationale 49
3.5 Security Analysis 52
3.5.1 Protection from Malicious/Compromised Clients 52
3.5.2 Defense against Compromised Servers 53
3.6 Simulations 55
3.6.1 Parameters and Assumptions 56
3.6.2 Impact of t on DICTATE Performance 57
3.6.3 Tuning the Freshness Degree 57
3.7 Conclusion 58

II Turning Harm into Good – Optimizing Network Performance with Controllable Mobility in Wireless Sensor Networks **61**

4 Joint Routing and Sink Mobility for Network Lifetime Elongation **63**
- 4.1 Introduction 63
- 4.2 Related Work 65
  - 4.2.1 Energy Conservation Protocols 65
  - 4.2.2 Moving Sinks to Improve Network Lifetime 65
  - 4.2.3 Mobility Helps in Other Aspects 66
- 4.3 Optimization under a Graph Model 67
  - 4.3.1 System Model 67
  - 4.3.2 Problem Formulation 69
  - 4.3.3 Hardness of the Problem 71
  - 4.3.4 Induced Sub-Problems 72
  - 4.3.5 Maximizing Network Lifetime for a Single Mobile Sink (MNL-SMS) 74
  - 4.3.6 Numerical Experiments 79
  - 4.3.7 Summary 84
- 4.4 Optimization under a Continuum Model 85
  - 4.4.1 System Model 85
  - 4.4.2 Problem Formulation 85
  - 4.4.3 To Move or Not To Move 87
  - 4.4.4 Optimal Mobility and Improved Routing Strategy 92
  - 4.4.5 Simulations 97
- 4.5 Implementation Issues 101
- 4.6 Conclusion 102

5 MobiRoute: Routing Towards a Mobile Sink for Improving Lifetime **105**
- 5.1 Introduction 105
- 5.2 Related Work 106
- 5.3 Problem, Metrics and Methodology 107
- 5.4 MobiRoute: Routing Data Towards a Mobile Sink 108
  - 5.4.1 MintRoute 110
  - 5.4.2 Detecting Link Breakage 110
  - 5.4.3 Conveying Topological Changes 111
  - 5.4.4 Minimizing Packet Losses 111
- 5.5 Algorithm for Adaptively Controlled Mobility 112
- 5.6 Simulations 113
  - 5.6.1 Grid Networks 113
  - 5.6.2 Ring Network 118
- 5.7 Conclusion 119

Conclusion **121**

A Estimations of the Network Parameters **139**

B Another Proof of Claim 1 **141**

C Proof of Claim 3 **143**

D Load Model for Wireless Sensor Networks **145**
D.1 Modeling the Sensor Load: The Case of a Static Sink 145
D.2 Modeling the Sensor Load: The Case of a Mobile Sink 146

Chapter 1

Introduction

Mobile computing, as suggested by its name, features the introduction of mobility into computing, or more specifically, into computer networks for our networking research community. Host mobility becomes possible thanks to wireless networking technologies that deliver untethered connections among computing devices such as PCs, handhelds, and smart phones. The possibility of moving computing devices while still keeping them online, on one hand, greatly facilitates the human users of these devices and thus promotes mobile computing, but on the other hand, poses considerable challenges to network design and control. While the research on IP networking has already stepped up to the phase where people are considering fitting mobility into the protocol stack [Per96, PC00, Edd04, EIA04], the understanding of the influence from mobility to multi-hop wireless networks is still in its infancy.

In this thesis, we intend to bring forth our insight into this understanding, by investigating the way of **coping with** mobility and the methodology of **exploiting** mobility in protocol design and network control for ad hoc and sensor networks. Naturally, the thesis is divided into two parts. In the first part, we demonstrate how randomized protocols (in particular, multicast protocols) can be applied to overcome the unpredictable dynamics incurred by node mobility in mobile ad hoc networks. In the second part, we show concrete examples of using a few mobile devices to improve the performance of wireless sensor networks; we prove in both theory and practice that the network lifetime can be prolonged by introducing controllable mobility into the networks.

Scope of Research

Dealing with mobility is a recurring issue for all protocol layers in mobile ad hoc networks. At the network layer, several unicast routing protocols, either proactive or reactive (e.g., [PBRD03, JMH04, PB94, PH99, CJD03]), are proven to be feasible in coping with the network dynamics. However, multicast protocols are usually based on a sender-initiated and proactively-maintained routing structure (e.g., [RP99, LSG02]). Such protocols work fine with the one-to-many transmission style but do not provide necessary supports to the upper-layer protocols (e.g., reliable group communication systems [Sch93]) that require **many-to-many communication**s and incur **dynamic membership change**s in a multicast group. In the first part of this thesis, we are concerned with the design of a multicast protocol that satisfies the aforementioned needs. In addition, we aim at protocols that provide reliable group communication service in mobile ad hoc networks with the support of our multicast protocol. Finally, demonstrating the usefulness of our proposals is also one of our goals.

Introducing node mobility into wireless networks is like using a "double-bladed sword", i.e., it brings both disadvantages and advantages to the design and control of such a network. Therefore, as network designers, we should try to take advantage of mobility while coping with its downside. Existing methods of exploiting mobility consist in making use of mobility 1) to transport data for improving throughput [GT02] or network lifetime [SRJB03, CSA03, KSJ+04] and 2) to configure network topology for enhancing power efficiency in data communications [GLM+04]. While the former methods trade data delivery latency for their goals, the latter actually uses the energy spared in data communications to move network nodes. In the second part of this thesis, our goal is to show that mobility can become a tool to improve network lifetime in wireless sensor networks without sacrificing other aspects of network performance. Apart from theoretical analysis, we also aim at devising and evaluating practical protocols to support mobility.

Thesis Overview and Outline

Part I: Fighting Fire with Fire

The reliable group communication service is an important basis of many network functions, both in wired networks or wireless networks. However, the problem becomes more challenging in ad hoc networks due to highly dynamic and unpredictable topology changes. In fact, even guaranteeing the reliability of multicast, a key building block of group communication systems, becomes prohibitively hard. Some previous work tried to adapt certain existing acknowledgement-based reliable multicast protocols for wired network to ad hoc networks, but the results are quite disappointing [PR99, GS99]. As a result, many network functions traditionally depending on reliable group communication service, such as host configuration [NP02] and group key management [ZH99], have to either rely on the fragile "reliability" provided by flooding or make assumptions about such a service while waiting for it to appear.

In Chapter 2, we describe PILOT (ProbabilisticC LIghtweight grOup communication sysTem) for mobile ad hoc networks. PILOT is a two layer system; it takes Route Driven Gossip (RDG) as the underlying multicast protocol and further builds two services upon RDG: Probabilistic quorum systems for AD-hoc NEtworks (PAN) for data storage and Reliable RDG (R^{2}DG) for multicast streaming. RDG is a gossip-based multicast protocol; it is designed precisely to meet the many-to-many communication style in ad hoc networks. In the spirit of "fighting fire with fire", RDG embraces the non-deterministic nature of mobile ad hoc networks, providing analytically predictable (probabilistic) reliability guarantees at a reasonable overhead. PAN uses the service provided by RDG. Consequently, it also guarantees the reliability in a probabilistic manner. We present an analysis and simulations for both RDG and PAN, in terms of both reliability and overhead; these results help us to fine tune protocol parameters to obtain the desired tradeoff between efficiency and fault tolerance.

In Chapter 3, we demonstrate one important application of PILOT. We consider the case where an ad hoc network is under the responsibility of a *mother certification authority* (mCA). Since the nodes can frequently be collectively isolated from the mCA (e.g., for a remote mission) but still need the access to a certification authority, the mCA pre-assigns a special role to several nodes (called *servers*) that constitute a *distributed certification authority* (dCA) during the isolated period. We propose DICTATE (DIstributed CerTification Authority with probabilisTic frEshness) to manage the dCA. This solution ensures that the dCA always processes a certificate update (or query) request in a finite amount of time and that an adversary cannot forge a certificate. Moreover, DICTATE is built upon PILOT and thus guarantees that the dCA responds to a query request with the most recent version of the queried certificate in a predictable and tunable probability. We describe DICTATE in detail and, by security analysis and simulations, we show that it is robust against various attacks.

Part II: Turning Harm into Good

Wireless sensor networks, apart from the great advantages they can bring, are subject to many limitations, among which the energy constraint has become an increasing concern. In the past several years, many energy efficient/conserving routing protocols have been proposed for multi-hop wireless networks (e.g., [CT00, KKLT03, SL04]). Protocols as such, by default, try to balance the traffic load among nodes with the assumption of a many-to-many transmission style. Therefore, these protocols, when being applied to sensor networks (whose transmission style is usually many-to-one) only balance the load among nodes whose distances from a sink are roughly the same. As a result, the concentration of data traffic towards a small number of sinks remains a major threat to the network lifetime; the sensor nodes located near a sink have to relay data for a large part of the network and thus deplete their batteries very quickly.

In Chapter 4, inspired by a recent trend of using mobile sinks in sensor networks [LBK+02, SRJB03, CSA03, KSJ+04], we propose a solution that suggests the sinks be mobile; in this way, the role of "hot spots" (i.e., the nodes around the sinks) is distributed over time, which evens out the load. Our proposal differs from the existing approach (e.g. [SRJB03, CSA03], which we term *mobile relay* approach) in that the sink **does not** rely **only** on mobility to retrieve data from sensors; the data collection procedure continues through multi-hop routing wherever a sink stays. Data collection protocols can hence be optimized by taking both sink mobility and multi-hop routing into account. We analyze the problem of joint sink mobility and routing for lifetime optimization under both a graph model and a continuum model. Although the problem under a graph model is intractable in general, certain relaxations have a practical significance in directing protocol design.

We are able to find the optimal solution under a continuum model. The validity of our analytical results are confirmed by simulations.

In Chapter 5, we further investigate the approach that makes use of a mobile sink for balancing the traffic load and in turn improving network lifetime. We engineer a routing protocol, MobiRoute, that effectively supports sink mobility. Through intensive simulations in TOSSIM with a real implementation, we prove the feasibility of our mobile sink approach by demonstrating the improved network lifetime in several deployment scenarios. The theoretical results obtained in Chapter 4 are applied to improve the adaptability of MobiRoute and also to choose the moving traces of the sink in simulations.

Part I

Fighting Fire with Fire

— Probabilistic Protocols to Cope with Mobility in Large-scale Ad Hoc Networks

Chapter 2

PILOT: Probabilistic Group Communication System

2.1 Introduction

A *Group Communication System* (GCS) [Pow96] is a useful infrastructure on which various reliable distributed computing functions can be built. The need for such a system arises not only in wired networks but also in mobile ad hoc networks. Even some mechanisms traditionally relying on a centralized service have to be implemented in a distributed way in ad hoc networks, since the service provided by a single node is not dependable enough. Mobility management [HL99b, PG01], for instance, relies on a special group of nodes to continuously track locations of mobile nodes and to serve requests to these location data. The distributed management of cryptographic keys or certificates [ZH99, CBrH03] and group security functions like access control or key agreement [BrH02, AG00] represent another class of applications. Last but not least, distributed dynamic host configuration protocols such as naming or addressing services [NP02], which are essential to build a functional network, need to make agreements within the whole network.

Unfortunately, the complexity of building *reliable* GCSs, which is prohibitively high already in wired networks, is further amplified in ad hoc networks due to highly dynamic and unpredictable topology changes incurred as a result of **node mobility**. In fact, even guaranteeing reliability of multicast, a key building block of GCSs, becomes extremely hard. As a consequence, many distributed computing functions that would depend on reliable GCSs have to either rely on the fragile "reliability" provided by flooding [NP02] or make assumptions about such a service while waiting for it to appear [ZH99].

In this chapter, we identify two fundamental problems in the context of group communication, namely 1) multicast and 2) data sharing, and we define notions of probabilistic reliability for these problems, aiming at coping with node mobility in ad hoc networks. We then present our protocol suite, called ProbabilistIc Lightweight grOup communication sysTem (PILOT) for ad hoc networks, as a solution. Innovating on the principles of gossip mechanisms and probabilistic quorum systems, PILOT provides probabilistic reliability for multicasting and data sharing, based only on a unicast primitive (rather than a multicast primitive) in order to gain in flexibility. We present analytical results predicting the performance of PILOT in terms of message overhead and reliability degree. We then compare these results with simulation results obtained with the *ns-2* simulator to show that we can have useful predictions on the performance of PILOT. To the best of our knowledge, the work presented in this chapter, as a cornerstone of the MICS project [HGBV01], is the first to provide a complete solution to the problems of reliable multicast and data sharing in ad hoc networks, along with both analytical and simulation results. It smoothly integrates, expands and completes our previous individual results [LEH03, LHE03] into a compound group communication system.

The remainder of this chapter is structured as follows. Section 2.2 surveys related work. Section 2.3 details the network model and the problem to be solved. Section 2.4 presents our PILOT system. Section 2.5 analyzes PILOT in terms of reliability and efficiency. Section 2.6 compares analytical results with simulation results, and also investigates other aspects of PILOT, such as its sensitivity to node failures. Finally, Section 2.7 concludes the chapter.

2.2 Related Work

The prosperous research on group communication toolkits has led to a multitude of results in wired networks, such as Ensemble [Hay97] and Spread [ADS00]. However, similar systems have not yet appeared in ad hoc networks, although certain supporting mechanisms like token circulation [MVW02], random walk agent [DSW02], reliable broadcast [Vol03], and membership management [RHH01] have been proposed. Our PILOT system is a first step towards building a prototype for a group communication toolkit. Rather than emphasizing the discussion in the framework of GCSs, we will hence focus on the relevant underlying building blocks.

2.2.1 Deterministic Reliable Multicast in Wired Networks

In *wired* networks, reliable multicast protocols strive for strong, “all-or-nothing”-like, reliability guarantees semantics for the successful delivery of a message to a group of node despite the failure of a certain number of these nodes (cf. *Reliable Broadcast* [HT93]). These protocols scale poorly with an increasing group size even in a very stable network.

Protocols that indeed offer some practical reliability, but are not reliable in the metric of the above-defined category and lack an alternative measure of their reliability, includes typically protocols building on top of IP multicast, such as [FJL$^+$97, PSLB97]. The ack/nack mechanisms employed by such protocols to improve reliability, unfortunately, also tend to compromise their scalability by heavily loading the network (e.g., leading to *ack implosion*).

2.2.2 Gossiping in Wired Networks

Probabilistic multicast protocols are a family of protocols that has been re-discovered rather recently. Roughly, the basic idea is to have each node in a multicast group periodically "talk" to a random set of other nodes in the group about its knowledge of the "state" of the group, e.g., the multicast packets that it has received. Missing packets can then be recovered by nodes in a peer-based style (e.g., [BHO$^+$99, LM00, EGH$^+$03]). These protocols equally distribute the load over the nodes in a group and thus also make themselves very resilient to arbitrary node failures. Stochastic models derived from epidemiology enable the protocols to obtain 1) a performance prediction and 2) the desired tradeoff between reliability and overhead by adjusting protocol parameters.

The *Probabilistic Broadcast* (*pbcast*) [BHO$^+$99] protocol has in much rejuvenated the interest in gossip-based protocols that find their origins at Xerox where they were initially used for replicated database maintenance [DGH$^+$87]. The *pbcast* protocol consists of two phases: a first phase based on an unreliable multicast primitive and a second one making use of gossips for repairing packet losses. These phases are merged into one phase by the *Lightweight Probabilistic Broadcast* (*lpbcast*) [EGH$^+$03] protocol. By gossiping uniformly about data packets, digests, as well as membership information, *lpcast* provides reliability similar to *pbcast* without imposing a complete membership view on the members.

Taking the network topology into account when gossiping, *Directional Gossip* (DG) [LM00] gains in efficiency. In short, a *weight* is computed for each neighbor node, representing the connectivity of that given node. The larger the weight of a node, the higher the possibility for it to receive a given packet from other nodes. When gossiping, nodes with higher weights are hence chosen with a smaller probability, reducing redundant sends. In particular, LANs are represented by single nodes to distant LANs, and "long" routes between two such representatives are seldom chosen.

While the DG protocol does not provide any analytical evaluation, protocols such as *pbcast* and *lpbcast* are analyzed in much detail based on a recurrence relation establishing the probability for the possible number of infected nodes at all gossip rounds. Alternatively, protocols are modelled by differential equations (e.g., [DGH$^+$87]), or random graph theory (e.g., [KMG03]). The latter protocol is tightly coupled to its analysis, in the sense that a particular packet is gossiped only once by a given node. Roughly, in such a model, there is a sharp threshold for the required fanout around $\log n$ (n being the number of members in a multicast group) to ensure that, with very high probability, all nodes will receive a given multicast packet despite node and transmission failures.

2.2.3 Gossiping in Ad Hoc Networks

The benefits of gossiping techniques have, rather recently, also been exploited in ad hoc networks. In this context, gossip-based protocols are not favored for obtaining an analytical prediction of their performance in terms of reliability, but more for the practical observation that they 1) perform in a more reliable way than unreliable protocols such as MAODV [RP99] and 2) generate less traffic than, for instance, flooding approaches.

Anonymous Gossip (AG) protocol [CRB01], a descendant of the *pbcast* [BHO$^+$99] protocol, pioneered the recent research efforts on gossip-based multicast protocols for ad hoc networks. Through the concept of anonymous gossip, any agreement on membership is avoided during the gossip-based repair phase. This however shifts the responsibility for the membership management to the MAODV layer, which the AG protocol also relies upon for a preliminary, rough packet dissemination. These prerequisites make the AG protocol more difficult to apply in a broader context than the one offered by MAODV. Furthermore, the property of predictable behavior, an important merit of gossip-based protocols, is lost due to the dependence on MAODV to guide the gossips.

The exploitation of the observation 2) is briefly mentioned in [HKB99, NTCS99], and then more closely investigated by Haas et al. in [HHL02] for the dissemination of routing messages in mobile ad hoc networks and by Culler et al. in [LPCS04, HC04] for propagating and maintaining code updates in wireless sensor networks. Since deterministic flooding techniques do not necessarily ensure that, in practice, every node sees a given information either, gossiping techniques yield results close to those of flooding protocols, yet imposing far less load on the network.

Prior to that, Vahdat and Becker [VB00] have also employed gossiping techniques for unicast routing. Their idea is to ensure that packets are eventually delivered even if there is no path between the source and the destination for some time. Such an approach is very interesting, but tends to require relatively high buffering capacities at individual nodes if all unicast traffic is handled that way. Just like all other gossip-based protocols for ad hoc networks we know of, this effort does not include any analytical performance estimation.

2.2.4 Stateless Multicast

Another recent paradigm shift is given by *stateless multicast* protocols [JC01, CN02]. While the *Differential Destination Multicast* (DDM) [JC01] protocol explicitly calls the unicasting function to disseminate multicast packets, the protocol presented in [CN02] builds an overlay multicast packet distribution tree on top of the underlying unicast routing protocol, and multicast packets are encapsulated in a unicast envelop and transmitted between the nodes in the group. Though reducing the control overhead of the multicast session, the protocol leads to overweighted packet headers. This problem, known from unicast source routing but amplified in the case of multicasting, limits the protocol's scalability in terms of the group size.

2.2.5 Probabilistic Quorum Systems

Quorum systems [BGM87] have been proposed as an alternative to the *state-machine* approach [Sch93] for reliable data sharing. They improve the efficiency of the replication of the stored data by better balancing the overhead between updates and queries. Unfortunately, "original" quorum systems, also termed *strict quorum systems*, do not apply well to highly dynamic environments. This is because the very construction of these quorums is not a trivial task, the outcome of this task being strongly subject to membership changes. By introducing *probabilities* for the intersection of individual quorums, *probabilistic quorum systems* [MRW01] relax the construction rules for quorums and leave more freedom for trading protocol overhead for reliability. While this smoother tradeoff has constituted the driving force behind probabilistic quorum systems, it turns out that the resulting reduced determinism makes such an approach also more viable for ad hoc networks than a strict approach. The overhead considered in [MRW01] is the charge of computation for individual servers. Our definition of overhead, however, focuses on the consumption of network resources, because computation is much cheaper than communication in wireless networks.

Haas and Liang [HL99a] first introduced probabilistic quorum systems into ad hoc networks for mobility management, under the name of *randomized database groups*. They propose a very interesting way to express both fault tolerance and load as costs of their system, and optimize those costs numerically. Considering the similarity between their system and PILOT, we provide some comparisons between the two solutions in Section 2.4.6.

2.2.6 Data Management in Ad Hoc Networks

The 7DS system presented in [PS01] shares certain features of our PILOT system, with respect to the diffusion scheme used for data dissemination. However, since the two systems are designed for different network environments (7DS assumes a rarely connected network, whereas PILOT considers networks of relatively high density), the underlying diffusion mechanisms are quite different. Whereas 7DS passively exploits node mobility to relay data from one node to the other, which can result in a considerable delay for data spreading but has the potential to improve power and bandwidth usages, PILOT more actively "pushes" data to other nodes with a gossip-based protocol. As a result, the analytical models for the two diffusion processes are also different (diffusion controlled process for 7DS and epidemic model for PILOT).

Both [Har01] and [WL02] try to guarantee data accessibility upon network partitioning in a replication system by investigating the problem of dynamic replica allocation. While [Har01] makes assumptions (e.g., data items are not updated) that seem to be too strong to capture the reality of mobile networks and hence has limited application scope, the approach in [WL02] is more practical in the sense that it takes into consideration topology information (e.g., connection stability) when replicating data; and data replication only happens when necessary, according to certain partition detection schemes. As far as system models are concerned, the problem we solve is somewhat orthogonal to the one of [WL02]. The mobility model they propose assumes strong correlations between different nodes (e.g., nodes are organized into mobility groups), which might lead to frequent network partitions. We, however, consider a purely random mobility pattern, in which network partitions seldom happen and mobility prediction does not make much sense.

2.3 Goals and Assumptions

This section models the considered environment and states the problem to be solved.

2.3.1 Model

We consider an ad hoc network consisting of a set $\mathbf{N}$ of nodes and assume that every node $i \in \mathbf{N}$ has a unique *id*. Nodes may fail only by crashing, i.e., stopping to function. Failures are not permanent and can be recovered from.[1] All communications between different nodes are assumed to rely on the underlying unicast protocol. We use DSR [JMH04] as an example in this chapter but, in practice, our solution can be made to work with any unicast routing protocol. In addition, we assume a CSMA/CA-like MAC layer protocol (e.g., IEEE 802.11) that provides a RTS/CTS-Data/Ack handshake sequence for each transmission.

2.3.2 Problem Statement

We consider an ad hoc network where reliable group communication primitives are required by mobile nodes. Within the broad scope of group communication, we address two fundamental problems, namely multicast and data sharing, and associate each of them with a notion of *probabilistic reliability.*

Reliable Multicast Protocol The multicast protocol disseminates packets within a multicast group $\mathbf{G} \subset \mathbf{N}$, which, for brevity, will be referred to as *group* hereafter. We define the following two metrics to measure the probabilistic reliability achieved by this protocol:

- *Reliability Degree of Single Packet Dissemination* $\mathcal{R}_{ds}$*:* The fraction of group members that receive the packet sent by a certain member.
- *Reliability Degree of Continuous Packet Dissemination* $\mathcal{R}_{dc}$*:* The fraction of all packets that are received by a certain member, assuming that packets are continuously sent from the same member with rate λ_o.

Both metrics are described by respective *cumulative distribution function* (cdf) $\mathcal{F}(x) : [0,1] \rightarrow [0,1]$; it means that $\mathcal{F}(x)$ is the probability that $\mathcal{R}_{ds}$ (or $\mathcal{R}_{dc}$) is at most x.

Reliable Data Sharing Service Let StS[2] $\subset \mathbf{N}$ be a storage entity and ρ be a set of access protocols for StS. The StS holds shared data in a replicated fashion, and the consistency model for data replication is considered to be *shared-private* [FC94], i.e., the service does not commit itself to any access ordering except FIFO order.[3] Given access rates λ_u and λ_q for updates and queries, respectively, the data sharing service is probabilistically reliable in nature if a query access

---

[1] This failure model also captures the case where nodes are deliberately switched off (e.g., for the purpose of battery replacement or operating system rebooting, or because the users do not intend to make use of their devices for a while).

[2] StS is an abbreviation for *Storage Set*, a special group in the network. The algorithm used to initialize the StS will not be discussed here since it is out of the scope of this thesis. Refer to [XHG02, SSB99] for examples of initialization algorithms.

[3] All the applications we have mentioned in the introduction comply with this model.

$\rho_q(\textsc{StS}, \lambda_q)$ obtains, with a certain probability, the latest version of a data object resulting from an update access $\rho_u(\textsc{StS}, \lambda_u)$. The metric for the service is:

- *Reliability Degree of Access $\mathcal{R}_{da}$:* The probability that a query operation acquires the most recent update of the corresponding data object, considering both node and channel failures.

The overhead is measured by the *Network Load* $\mathcal{N}_l$, which is the average number of *unicast packet×hop* per multicast packet to achieve a certain $\mathcal{R}_{ds}$ or per unit time to achieve a certain $\mathcal{R}_{dc}$ or $\mathcal{R}_{da}$. This definition is adapted to ad hoc networks by taking into account the number of hops to route a particular packet. $\mathcal{N}_l$ considers only the load generated by our protocols, which is independent of the various possible implementations of the underlying networking functions.

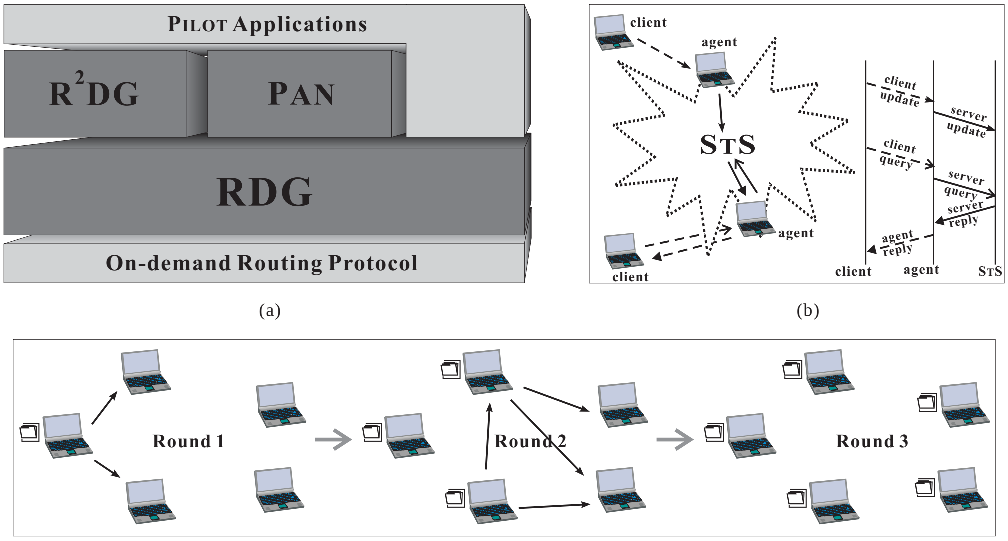
Our goal is to design a set of protocols that achieve a high reliability degree $\mathcal{R}_d$ (representing $\mathcal{R}_{ds}$, $\mathcal{R}_{dc}$, and $\mathcal{R}_{da}$ hereafter) even under large arrival rates λ_o (the sum of λ_u and λ_q for data sharing), while incurring reasonable overhead $\mathcal{N}_l$. We target relatively large scale networks, i.e., networks with tens or even hundreds of nodes and a random mobility pattern. Under a certain λ_o, the optimal performance with respect to both $\mathcal{R}_d$ and $\mathcal{N}_l$ does not exist, since one can always be sacrificed to improve the other. Hence, we will study the trade-off between the two metrics and show how to fine tune parameters to trade either for the other.

2.4 PILOT: Probabilistic Group Communication System

In this section, we first present the structure of our PILOT system, then we detail each component of PILOT separately.

2.4.1 Overview: Layered Architecture of PILOT

PILOT is a two layer system, illustrated by the dark grey part in Fig. 2.1(a). It has a probabilistic multicast protocol, Route Driven Gossip (RDG), as its basis. The protocol is gossip-based [BHO+99] in nature: it proceeds round by round while the receivers in each round are randomly chosen and they *relay* packets to the receivers of the later round(s), as shown in Fig. 2.1(c). Upon this layer, two dedicated services are built. R^{2}DG (Reliable RDG) is devised for continuous packet dissemination. It exploits the fact that packet losses can be detected by observing gaps in the *pid* (see Section 2.4.2) sequence, and thus piggybacks a negative acknowledgement with each packet sent (or relayed) to *pull* the lost packet back. The other service, Probabilistic quorum system for AD hoc NETWORKS (PAN), provides reliable data sharing by assuming the existence of an STS to store the shared data in a replicated manner. Any node $i \in \textsc{StS}$ is termed *server*, whereas the rest of the nodes are termed *clients* of the STS. Data queries and updates are directed to an arbitrary server in the STS while the message dissemination within the STS is performed by RDG, as shown in Fig. 2.1(b). According to their requirements, applications can either use the upper layer services or directly call RDG if only single packet dissemination service is required.



(a) (b)

(c)

Figure 2.1: Principles of PILOT. (a) Architecture of PILOT: the basic probabilistic multicast protocol (RDG) is at the bottom; R^{2}DG and PAN are built upon the basic protocol. (b) Message exchanges for updating and querying the STS in PAN. (c) Gossip-based multicasting in RDG.

2.4.2 RDG: Basic PILOT Multicast Protocol

Our RDG protocol uses a *pure* gossip scheme, as it is not built upon any underlying multicast protocol, in contrast to [CRB01] (the only related approach we are aware of). As opposed to “traditional” gossip protocols that only consider the membership information of a group, RDG adapts to the peculiarity of ad hoc networks by also taking the availability of routing information into account. Although the resulting membership view for each member is just a random subview due to the randomness of routing information that nodes can have, the protocol still works very well in the sense that the reliability is in practice very high and also predictable. We first briefly introduce DSR [JMH04], the basis of our current RDG implementation, then we elaborate RDG in detail.

Overview of DSR Protocol

Dynamic Source Routing (DSR) is an on-demand routing protocol making use of source routing and an aggressive caching policy. The protocol is on-demand since it floods route requests in the network upon routing packets to a destination without an available corresponding routing path. The source routing mechanism makes the routing paths loop-free, while providing certain topological information. With the aggressive caching policy, DSR tries to cache all routing paths that it learns (it even taps such information from the MAC layer if the "promiscuous" receive mode is enabled.).

Overview of Route Driven Gossip (RDG)

Each packet multicast by RDG is uniquely identified by its identifier *pid*, defined as a tuple [group ID (*gid*), source ID (*sid*), pkt seq. no. (*seq*)]. The protocol has four data structures. In the data management part, *pidList* stores the *pid*s of the received packets, and *Buffer* temporarily stores these packets. The other two are for the membership management of the protocol. *gidList* indicates which group(s) the node belongs to. *View* is composed of three fields:

1. *AView* stores the *id*s of known members, whose corresponding routing or location information is known,
2. *PView* stores the *id*s of known members, whose corresponding routing or location information is currently unavailable, and
3. *RView* stores the *id*s of members having indicated their willingness to leave.[4]

All these records are divided into several subsets with each subset being dedicated to a certain group, i.e., each $node_i$ has four subrecords ($pidList_i^{gid}$, $Buffer_i^{gid}$, $gidList_i^{gid}$, and $View_i^{gid}$) for a certain group **G** (with identifier *gid*) that it belongs to. In addition, each record is of limited size, noted $|R|_{max}$, for a given record R.

RDG offers seven operations, which are grouped into three sessions corresponding to their functionality. The *join* session defines the behavior of the node interested in joining a group and the reactions of other group members. The *leave* session defines the behavior of the node intending to leave the group and the reactions. In the *gossip* session, newly received packets are periodically propagated by a node. Furthermore, nodes respond to the gossip messages received. In relation to the Gossip task, three protocol parameters are defined here:

1. The *fanout* (F) is the number of gossip destinations randomly selected from the *AView* for each gossip emission,
2. The *quiescence threshold* (τ_q) is related to each data packet: a packet will be removed from *Buffer* after having been gossiped for τ_q rounds by individual nodes, and
3. The *age threshold* (τ_a) limits the propagation range of each packet.

These parameters are set by the upper layer to control the behavior of the protocol (see Section 2.4.2).

---

[4] *AView*, *PView*, and *RView* stand for *active* view, *passive* view, and *remove* view, repectively.

Join session (Fig. 2.2)

A node intending to join a group floods the network with a GROUPREQUEST message to search for other group members while announcing its existence (Fig. 2.2 (a) lines 1–2). Upon receiving such a message from a certain member, all members update their *AView* with the new *id*. They also return a GROUPREPLY to the request initiator with probability P_{reply} (Fig. 2.2 (b)). The probability is set by each node, according to its own estimation of the group size, in order to avoid GROUPREPLY storms. The initiator of the GROUPREQUEST also updates its *AView* after receiving the GROUPREPLY (Fig. 2.2 (a) lines 3–4).

```
1: procedure JOIN($gid$)
2:     GROUPREQUEST($id_i$, $gid$)

3: upon receiveGROUPREPLY($id$, $gid$) do
4:     $AView_i^{gid} \leftarrow AView_i^{gid} \cup \{id\}$
```

(a) *Join* indication emission and reply reception

```
1: upon receiveGROUPREQUEST($id$, $gid$) do
2:     if $gidList_i^{gid}$ = true then
3:         $AView_i^{gid} \leftarrow AView_i^{gid} \cup \{id\}$
4:         GROUPREPLY($id_i$, $gid$) with probability $P_{reply}$
```

(b) *Join* indication reception

Figure 2.2: *Join* session at node i

By recording the route of each incoming packet, DSR ensures that a new element in *AView* has a corresponding route entry in the DSR routing table. The validity of this relationship is periodically checked and the *AView* and *PView* are updated accordingly. When the size of *AView* drops below some threshold, the node has to reinitiate a *join* session.

Gossip/Leave Session (Fig. 2.3)

When a node wants to multicast a packet p, it inserts the packet in its *Buffer* as shown in Fig. 2.3 (a). A node intending to leave a group resets the corresponding flag in *gidList* as shown in Fig. 2.3 (b). Each member of the group periodically (every T ms)[5] gossips packets stored in *Buffer* to F other nodes randomly chosen from *AView* (see Fig. 2.3 (c) lines 8–16). It also piggybacks part of its view of the membership. A data packet is removed from *Buffer* after having been gossiped for τ_q times. The SEND primitive is a direct call to the underlying unicast protocol, which will also be used by other parts of PILOT for the same purpose. If the node intends to leave, only the field of *rmb* is used (see Fig. 2.3 (c) lines 3–6). As illustrated in Fig. 2.3 (d), a group member receiving a gossip packet will 1) update the *Buffer* with new packets (lines 3–8), 2) remove the obsolete member from its *View* (lines 10–14), and 3) add the new member to the *View* (lines 16–20). Note that a packet relayed τ_a times will not be gossiped again.

---

[5] In order to save bandwidth, we apply the *binary exponential backoff* algorithm to adjust the period when there is no new packet to be sent.

```
procedure RDG-CAST($gid$, $p$)
    $p.gid \leftarrow gid$
    $Buffer_i^{gid} \leftarrow Buffer_i^{gid} \cup \{p\}$
```

(a) Multicast

```
procedure Leave($gid$)
    $gidList_i^{gid} \leftarrow$ false
```

(b) Node leave

```
task Gossip($gid$)                /* Executed every $T$ ms */
    if $gidList_i^{gid} =$ false then
        $p.rmb \leftarrow id_i$
        $DS^{(1)} \leftarrow set \subset_{\Re} AView_i^{gid}$ s.t. $|set| = F^{(2)}$
        for all $id \in DS$ do
            Send($id_i$, $id$, $p$)
    else
        while $Buffer \neq \emptyset$ do
            $p \leftarrow pkt \in Buffer_i^{gid}$
            if $pkt$ has been gossiped $\tau_q$ times then
                $Buffer_i^{gid} \leftarrow Buffer_i^{gid} \backslash \{pkt\}$
            $p.rmb \leftarrow rmb \in_{\Re} RView_i^{gid}$
            $p.mb^{(3)} \leftarrow mb \in_{\Re} AView_i^{gid} \cup PView_i^{gid}$
            $DS \leftarrow set \subset_{\Re} AView_i^{gid}$ s.t. $|set| = F$
            for all $id \in DS$ do
                Send($id_i$, $id$, $p$)
```

(1) DS stands for *destination set*.
(2) A subscript $\Re$ stands for random selection.
(3) *mb* and *rmb* stands for member and removed member.

(c) Packet emission

```
upon receiveGossip($id_s$, $id_i$, $p$) do
    /* Step 1: Update $Buffer$ with new packets */
    if $p.pid \notin pidList_i^{p \cdot gid}$ then
        $pidList_i^{p \cdot gid} \leftarrow pidList_i^{p \cdot gid} \cup \{p.pid\}$
        $p.age \leftarrow p.age + 1$
        if $p.age < \tau_a$ then
            $Buffer_i^{p \cdot gid} \leftarrow Buffer_i^{p \cdot gid} \cup \{p\}$
        Deliver($p$)                /* to the upper layer */
    /* Step 2: Remove obsolete member from $View$ */
    $AView_i^{p \cdot gid} \leftarrow AView_i^{p \cdot gid} \backslash \{p.rmb\}$
    $PView_i^{p \cdot gid} \leftarrow PView_i^{p \cdot gid} \backslash \{p.rmb\}$
    $RView_i^{p \cdot gid} \leftarrow RView_i^{p \cdot gid} \cup \{p.rmb\}$
    while $|RView_i^{p \cdot gid}| > |RView_i^{p \cdot gid}|_{max}$ do
        $RView_i^{p \cdot gid} \leftarrow RView_i^{p \cdot gid} \backslash \{rmb \in_{\Re} RView_i^{p \cdot gid}\}$
    /* Step 3: Add new member to $View$ */
    if $p.mb \notin (AView_i^{p \cdot gid} \cup PView_i^{p \cdot gid})$ then
        if there exists a route to that node then
            $AView_i^{p \cdot gid} \leftarrow AView_i^{p \cdot gid} \cup \{p.mb\}$
        else
            $PView_i^{p \cdot gid} \leftarrow PView_i^{p \cdot gid} \cup \{p.mb\}$
```

(d) Packet reception

Figure 2.3: *Gossip/leave* session at node *i*

Nodes along routing paths to gossip destinations belonging to the same group as those destinations, when forwarding a packet they have not received yet, also deliver the packet and update their data buffers (not shown in the code). Due to its unpredictability, this operation will not be taken into account in our analysis, making the protocol perform better than expected. Note that the packet salvaging function of DSR is disabled while a gossip message is on its way, i.e., packets are dropped immediately whenever the routing path becomes obsolete or the sending buffer overflows. In fact, the redundancy provided by our RDG protocol automatically offsets the packet loss.

RDG performs message dissemination and membership tracking at the same time. Due to the node mobility and frequent membership changes, it is not practical to have a full membership view for each member. In fact, even if it is possible to have the *id*s of all members, there is no guarantee that the corresponding routing or location information is available. Our routing/location oriented membership management scheme tries to provide each member with a partial view, approximately random in nature, by exchanging membership information between members. The underlying scheme, together with sporadic losses and discoveries of the routing or location information[6], has a similar effect as the reshuffling of the partial view.

Considering that the locality of network traffic can reduce the network load, we apply a general optimization by raising the awareness of the topology. This optimization is based on the assumption that the underlying routing protocol can provide partial topological information (e.g., we can have the information about the lengths of paths from the routing table of DSR). Our heuristics in the case of DSR work like this: for a given group member, different weights are assigned to the members in *AView* according to the lengths of the routing paths to them, i.e., the longer a path the lower its weight, such that a "near" member is chosen with higher probability to relay a packet.

2.4.3 R^2DG: Continuous Packet Multicasting Service

If a stream of packets is multicast from a source, the *pid* sequence of received packets, at a certain group member, provides important information about packet loss. Based on RDG, our R^2DG protocol exploits this feature to enhance the reliability of multicasting.

R^2DG has its own data structures that are the same as for the data management part of RDG, except that the *Buffer* is much larger than that of RDG in order to have enough packets to respond to a negative acknowledgement (or *pull*). Before invoking the RDG primitive, R^2DG (as shown in Fig. 2.4 (a) lines 1–3) inserts the information about a missing packet into the packet header. In addition (task PULL in Fig. 2.4 (a) lines 4–6), a packet with an empty payload (*pull*-packet) is periodically sent to the lower layer with similar information attached to it. The period is dynamically adjusted according to the number of missing packets. A group member receiving such a packet will try to respond to the pull with the packets it has, see Fig. 2.4 (b).

```
1: procedure P-RDG-CAST($gid$, $p$)
2:     $p.pull \leftarrow pid$ of the most recent missing packet
3:     RDG-CAST($gid$, $p$)
4: task PULL($gid$)            /* Executed periodically */
5:     $p.pull \leftarrow pid$ of the most recent missing packet
6:     RDG-CAST($gid$, $p$)
```

(a) Multicast and *pull* task

```
1: upon RECV($p$) do
2:     $pidList_i^{p.gid} \leftarrow pidList_i^{p.gid} \cup \{p.pid\}$
3:     $Buffer_i^{p.gid} \leftarrow Buffer_i^{p.gid} \cup \{p\}$
4:     if $p.pull \in pidList$ then
5:         SEND($id_i$, $p.sid$, $pkt_{p.pull}$)
6:     DELIVER($p$)                /* to the upper layer */
```

(b) Packet reception and the response to *pull*

Figure 2.4: Multicast and *pull* session at node i

Considering that R^2DG passes pull-packets to RDG irregularly, RDG behaves intelligently when gossiping, in the sense that it tries to piggyback the pull information along with a data packet instead of sending the pull-packet directly.

---

[6] The information could be lost due to the node mobility or the timeout of route cache timer. On the other hand, a node can also obtain new information by requesting it or tapping it from packets under transmission.

2.4.4 PAN: Reliable Data Sharing Service

Our PAN system relies on the underlying RDG to provide reliable data sharing services. It includes two protocols: a client protocol and a server protocol, as shown in Fig. 2.1(b). In both cases of update and query, a client sends a request to an arbitrary server in the STS.[7] This server, termed *agent* for that client, then performs a corresponding operation of the server protocol. We assume that all messages (updates and queries) for our protocols have relatively small sizes such that they can be fit into single network packets. This requirement is justified by considering the applications we aim at. For example, a public key is about one thousand bits long and location information might be just a coordinate in a three-dimensional space. We further require that each message be uniquely identified by its identifier *mid*, which is a tuple [source ID (*sid*), object ID (*oid*), version no. (*ver*)][8], and that there is a way to establish a FIFO order among *mid*s.[9] Since the client protocol, a one-to-one connection, can always implement certain mechanisms (e.g., ARQ [AW00]) to ensure reliability, we will not consider this protocol in our analysis and simulations. In the rest of this section we focus on the server protocol.

The server protocol maintains a quorum system building upon the STS with the support from the underlying RDG protocol. We distinguish two types of quorums within the quorum system. A quorum can be a *write quorum*, accessed by an update, or a *read quorum* in the case of an access by query. Throughout the presentation, as well as in the analysis and simulations of the server protocol, we will use two symbols $\xi_?$ and $\hat{\xi}_?$ to represent the *nominal quorum size* and the *real quorum size*, where "?" can be "W" for a write quorum or "R" for a read quorum. The nominal size is the number of servers that a certain update or query attempts to access, while the real size is the number of servers effectively accessed.

Server Update Protocol

The agent diffuses an update message m_u within the STS by invoking the RDG protocol, as shown in Fig. 2.5 (a). Two parameters F and τ_a (see Section 2.4.2 for the definition of these parameters) are set in order to control the size of the resulting quorum. In this chapter, the value of τ_a is always set to ∞ for the server update protocol to simplify the analysis and simulations.

In order to keep track of the data access, each server keeps a record *midList*. It stores the *mid*s of the most recent updates. Each server receiving a new update, including the agent, substitutes the old *mid* with the *mid* of the new update message in its *midList* before delivering the message to the upper layer, as shown in Fig. 2.5 (b). At last, all servers that effectively receive the update form a write quorum. The size of the quorum, $\hat{\xi}_W$, is predictable thanks to the epidemic nature of the underlying gossip-based protocol, as we will see in Section 2.5.3.

---

[7] A client may have several ways to acquire information (e.g., identity and routing) about servers, depending on certain implementations of the STS initialization algorithm.

[8] The elements *oid* and *sid* stand for the identity of the data object to be queried or updated and of the object owner, respectively.

[9] $mid_1 > mid_2$ implies that $mid_1.sid = mid_2.sid \wedge mid_1.oid = mid_2.oid \wedge mid_1.ver > mid_2.ver$.

```
1: upon UPDATE($m_u$)   /* $m_u$ infused by client update */ do
2:     $p.data \leftarrow m_u$
3:     RDG-CAST($gid_{sts}$, $p$, $F$, $\tau_a = \infty$)
```

(a) UPDATE emission

```
1: upon RECEIVEUPDATE($m_u$) do
2:     if $\nexists$ $mid \in midList$ s.t. $mid \geq m_u.mid$ then
3:         for all $mid \in midList$ s.t. $mid < m_u.mid$ do
4:             $midList \leftarrow midList \setminus \{mid\}$
5:         $midList \leftarrow midList \cup \{m_u.mid\}$
6:         DELIVER($m_u$)
```

(b) UPDATE reception

```
1: upon QUERY($m_q$)   /* $m_q$ infused by client query */ do
2:     if $\exists$ $mid \in midList$ s.t. $mid > m_q.mid$ then
3:         $m_q.ver \leftarrow mid.ver$
4:         $Counter_{m_q} \leftarrow 0$
5:         $p.data \leftarrow m_q$
6:         RDG-CAST($gid_{sts}$, $p$, $F = \xi_R - 1$, $\tau_a = 1$)
7:         $timer_{m_q} \leftarrow 0$   /* set a timer for the query */
8: upon RECEIVEQUERY($m_q$) do
9:     if $\exists$ $mid \in midList$ s.t. $mid > m_q.mid$ then
10:        $m_q.mid \leftarrow mid$
11:        $m_q.data \leftarrow$ queried data object$^{(1)}$
12:        SEND($id_i$, $id_{agent}$, $m_q$)
```

(1) The data object is retrieved from the upper layer with certain callback procedures.

(c) QUERY emission and reception

```
1: upon RECEIVEQUERYREPLY($m_q$) do
2:     $Counter_{m_q} \leftarrow Counter_{m_q} + 1$
3:     if $\nexists$ $mid \in midList$ s.t. $mid \geq m_q.mid$ then
4:         for all $mid \in midList$ s.t. $mid < m_q.mid$ do
5:             $midList \leftarrow midList \setminus \{mid\}$
6:         $midList \leftarrow midList \cup \{m_q.mid\}$
7:         DELIVER($m_q$)
8:     if $Counter_{m_q} = \xi_R - 1$ then
9:         Invoke the client query protocol
10: upon $timer_{m_q} = timeout$ do
11:     Invoke the client query protocol
```

(d) REPLY reception at an agent

Figure 2.5: UPDATE/QUERY operation at node i

Server Query Protocol

In the case of a query, the agent again uses RDG to disseminate the query message to other servers. The value of τ_a is set to 1 to simplify the protocol evaluation later on. Also, since we consider that the arrival rate of queries is higher than that of updates in most cases, it is justifiable to have a relatively small read quorum.[10]

After receiving a query message from a client, the agent sends it to other servers immediately, along with the version number of the corresponding local data object. The agent also sets a counter and a timer in order to guarantee proper termination of the query session (Fig. 2.5 (c) lines 1–7). Each server belonging to the read quorum, upon receiving the message, responds with its own copy of the data object, if its version is more recent than the one of the agent (Fig. 2.5 (c) lines 8–12). The agent always delivers a new update returned from other servers. It invokes the corresponding client protocol, after every request either yields a reply or times out, as illustrated in Fig. 2.5 (d).

---

[10] By setting $\tau_a = 1$, the nominal read quorum size ξ_R is directly determined by F (Fig. 2.5 (c) line 6), since a server receiving the query will not relay it further.

2.4.5 Examples of Protocol Operations

We provide several illustrations to demonstrate the behaviors of RDG and Pan. For RDG, We assume a single group G of size 10 within a 20 nodes network. Fig. 2.6 shows the behaviour of the protocol with respect to the dissemination of one packet. Assuming $F = 2$ and $\tau_q = 2$, the packet initiated by member 15 infects the whole group in only 3 rounds in spite of the fact that no member has a full view of the membership while nodes move and even fail. By comparison, Fig. 2.7 illustrates the behavior of R^{2}DG with respect to the dissemination of two consecutive packets, assuming $F = 2$ and $\tau_q = 1$. Note that the strength of gossiper-pull becomes evident. The figures intuitively show that using gossiper-pull is a cheaper way to improve the protocol reliability than having $\tau_q \geq 1$ in the case of continuous packet dissemination; this intuition is proven in Section 2.6.

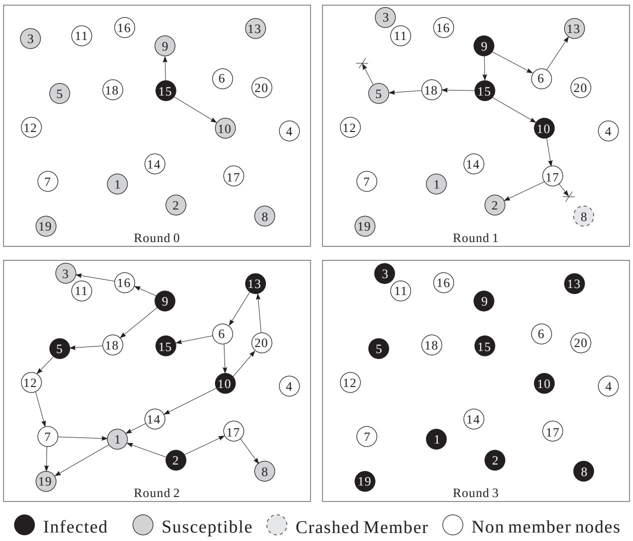


Figure 2.6: An example of one "run" of the protocol with $F = 2$ and $\tau_q = 2$ within a group of size 10. A member may receive duplicates of the same packet (e.g., member 1 at round 2). On the other hand, the packet can get lost at a certain round due to nodes crashing or moving (e.g., members 8 and 3 in round 1), but these losses will be compensated with high probability at a later round.

An example for Pan is shown in Fig. 2.8; we assume, in a network of 50 nodes, an StS consisting of 25 nodes. It is clear that, since the quorums are created as the result of using RDG, the existence of intersections between two quorums is probabilistic in nature and is not very sensitive to the topological changes (and thus node mobility).

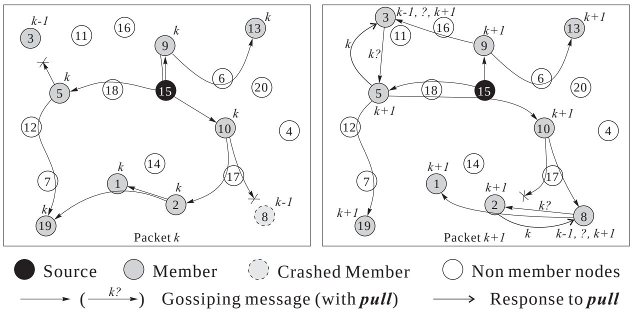


Figure 2.7: An example of two "run"s of the protocol with $F = 2$ and $\tau_q = 1$ within a group of size 10. The likelihood of receiving duplicates of the same packet is reduced due to the smaller value of τ_q, which implies a lower overhead, but at the cost of a reliability degradation in a run; this cost is, however, compensated in the next run.

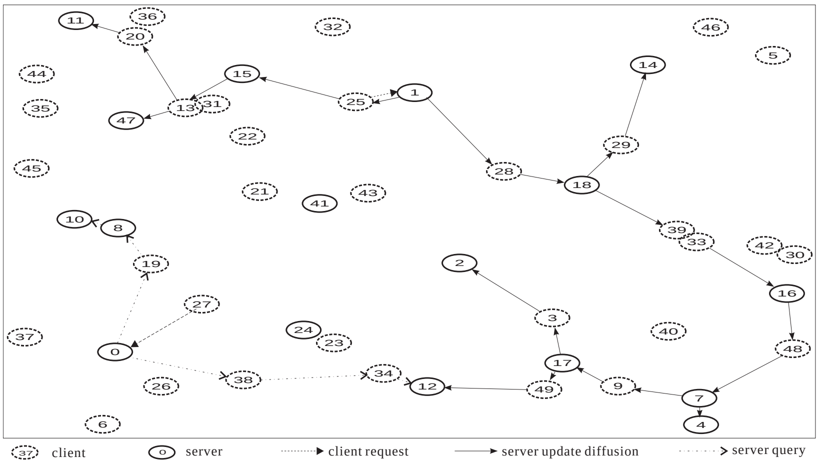


Figure 2.8: Illustration of an operation pair within a network of 50 nodes located in a square area of 1km^2 (aspect ratio is not kept for the sake of space). When node 25 wants to perform an update, it sends a request to its agent, node 1. The request of this update is diffused to other servers by node 1, using gossip-based scheme (Only the valid transmissions are shown here. Duplicated transmissions are omitted to simplify the visualization.). If node 27 wants to access the data, it also asks its agent, node 0. Node 0 in turn requests other servers, nodes 8, 10, and 12. In this case, node 12 is the intersection of the read and write quorums. It is able to reply the requested data of node 25 to node 27. The query reply is omitted here for simplicity.

2.4.6 Comparing PILOT with Randomized Database Group

In this section, we compare the work of Haas and Liang [HL99a] with PAN. The comparisons are qualitative rather than quantitative, because [HL99a] does not provide simulation results to evaluate the system performance and to confirm the precision of their numerical analysis. In a nutshell, PAN outperforms the randomized database group in two aspects. On one hand, the protocol used to access the database group in [HL99a] consists in multiple unicasts, based on the assumption of perfect routing information. Obviously such an approach fails under the more realistic assumption of incomplete routing information, while PAN can cope with such incompleteness. On the other hand, the symmetric construction of quorum systems in [HL99a], i.e., the same size for all quorums, is not suitable for all replication systems (for instance when the arrival rates of queries and updates, respectively, diverge strongly). PAN, on the contrary, can adapt to a given situation by appropriately adjusting parameters. As far as the analytical methodology is concerned, the model in [HL99a] is more application oriented than the one of PAN. It provides an insight into the probabilistic quorum systems from a different perspective.

2.5 Analysis

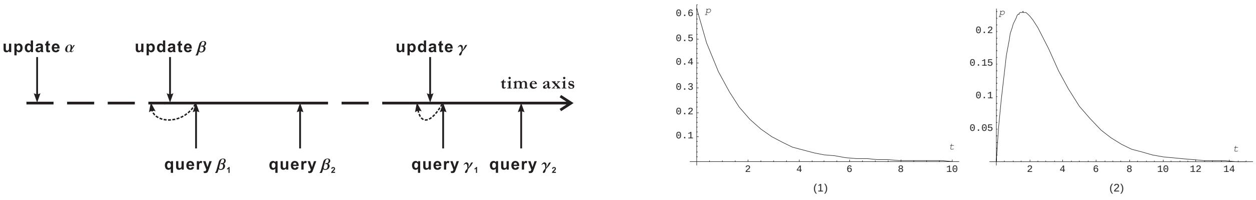
In this section, we show that the two metrics, $\mathcal{R}_d$ and $\mathcal{N}_l$ (defined in Section 2.3.2), are predictable given certain protocol parameters and information about the network. These analytical results are confirmed by simulations in the next section. Since the behavior of R^{2}DG pull depends on far more factors than that of RDG gossip, we will not consider this part of the protocol in the analysis. However, we will show the enhanced reliability by simulations.

2.5.1 Model

For the multicast protocol, we consider a single group $\mathbf{G}$ composed of $|\mathbf{G}| = n$ members and observe its behavior in terms of the dissemination of a *single* packet ("one run"), but also a *continuous* stream of packets (which is more realistic than related research proposals considering only the "one run" part). Each gossiping operation is modelled as a uniform random selection of F members out of n, i.e., without considering the topology-awareness, in order to simplify the tractability. According to the terminology of epidemiology [Mur93], a member that has received a certain packet is termed *infected*, otherwise *susceptible*. An infected member attempting to share the packet with others (i.e., a member who keeps gossiping the packet) is called *infectious*. We analyze our protocol in a network composed of a static set of nodes running closely "synchronized". More precisely, nodes gossip in synchronous rounds (T ms, identical for all nodes), and there is an upper bound on the network latency which is smaller than T. Note that this synchronization assumption is made just for the sake of the analysis; neither RDG nor PAN needs synchronization to operate correctly.

The probability of packet loss is closely related to the movement and traffic pattern, as well as to the length of the considered routing path. By assuming an identical and independent probability of failure p_f for each hop along a routing path in a certain network environment, the probability of losing a certain gossip message can be expressed as a function of the number of hops, H, of that routing path. We further assume that the lengths H of all routing paths between any two members follow the same distribution $f(h)$. On the other hand, p_f can be split into two parts: 1) p_{f_c} represents the probability of packet loss due to node crash and 2) $p_{f_{mo}}$ reflects the effects of node mobility and buffer overflow. Since $p_{f_c} \ll p_{f_{mo}}$ in general for mobile wireless networks, we directly use $p_{f_{mo}}$ to approximate p_f.

As for the data sharing service, we consider only the server protocol (including both update and query protocols) for analysis. The StS is assumed to consist of n servers. We also assume that query and update accesses arrive randomly at an arbitrary server, following Poisson processes with intensities of λ_q and λ_u, respectively. By further assuming that these two processes are independent, the overall access rate is given by $\lambda_o = \lambda_q + \lambda_u$.



(a) (b)

Figure 2.9: Time intervals between events and their distributions. (a) The occurrence of events in terms of absolute time. (b) The distributions of time interval between two events: (1) exponential distribution for consecutive events, (2) *Erlang* distribution for non-consecutive events.

The dissemination process of the server update is performed by RDG. As this process finishes, all infected servers form a write quorum with real size $\hat{\xi}_W$ following a certain probability distribution. We consider only the second query to a data object that was modified by the most recent update, while considering the first query as happening before the update.[11] For example, as shown in Fig. 2.9(a), only the pairs of (update β, query β_2) and (update γ, query γ_2) are considered, whereas queries β_1 or γ_1 are supposed to request previous updates (i.e., updates α and β, respectively). This assumption makes sense when we consider the time with respect to a server where updates and queries arrive, and also the property of a Poisson process shown in Fig. 2.9(b). Since there is always some delay for the message dissemination, the probability that the actual occurrence of events will follow the order of our assumption at that server is very high, according to different distributions of the time interval between two events within a Poisson process (see Fig. 2.9 (b)). This makes the present analysis a "viable" lower bound.

We continue using p_f to represent the network condition, but an empirical value p_e is also used in the case of queries to represent the server *unavailability* due to failure, at any time instant. One

---

[11] The time of an event is when it happens at an agent.

might argue that the server failure should be treated as a Poisson process [HL99a], but this is not justifiable with a failure recovery model, which is usually the case in ad hoc networks (e.g., nodes switching off for the purpose of battery replacement or operating system rebooting).

2.5.2 Stochastic Behavior of RDG

Considering a packet multicast by a member, we use $S_r \in \{0, \cdots, n\}$ to denote the number of members infected with the packet **after** round r. With the convention that $\Pr\{S_r = 0\} = 1$ for $r < 0$, it is easy to show that the sequence of random vectors $\mathbf{S}_r = [S_r, S_{r-1}, \cdots, S_{r-\tau_q}]^T_{r\geq 0}$ forms a Markov chain with values taken from the *state space* $\mathcal{E} = \overbrace{\{0, \cdots, n\} \times \cdots \times \{0, \cdots, n\}}^{\tau_q+1}$.

Recurrence Relation Given the probability p that a certain member is infected by a specific gossip message in one round, $q = 1 - p$ represents the probability of non-infection. Let $S_r = i$ (the number of infected members) and $S_r - S_{r-\tau_q} = k$ (the number of infectious members) in the current round; we introduce a binary random variable, X_l, for each of the remaining $n-i$ susceptible members, where $\Pr\{X_l = 0\} = q^k$, i.e., the probability that a certain susceptible member is not infected in the next round is the probability that it is not infected by any of the k infectious members. It is clear that $S_{r+1} - S_r = \sum_{l=1}^{n-i} X_l$ follows a binomial distribution. Let j be the number of infected members in the next round; the transition probability is expressed as:

$$\begin{aligned}
& \Pr\{S_{r+1} = j \mid S_r = i, S_r - S_{r-\tau_q} = k\} \\
= \; & \Pr\{\sum_{l=1}^{n-i} X_l = j - i \mid S_r - S_{r-\tau_q} = k\} \\
= \; & \begin{cases} \binom{n-i}{j-i}(1-q^k)^{j-i} q^{k(n-j)} & j \geq i \\ 0 & j < i \end{cases}
\end{aligned} \tag{2.1}$$

which leads to the following global balance equation of the chain:

$$\Pr\{\mathbf{S}_{r+1} = \mathbf{s}_{r+1}\} = \sum_{i_{\tau_q}=0}^{i_{\tau_q-1}} \binom{n-i}{j-i} (1 - q^{i-i_{\tau_q}})^{j-i} q^{(i-i_{\tau_q})(n-j)} \Pr\{\mathbf{S}_r = \mathbf{s}_r\} \tag{2.2}$$

where $\mathbf{s}_r = [i, i_1, \cdots, i_{\tau_q}]^T$, $\mathbf{s}_{r+1} = [j, i, i_1, \cdots, i_{\tau_q-1}]^T$, and $i = i_0$. Let the column vector ν_r, with $\nu_r(i) = \Pr\{S_r = i\}$ as its ith element, be the marginal distribution of $\mathbf{S}_r$. Given the initial distribution $\nu_0 = [0, 1, 0, \cdots, 0]^T$ and (2.2), ν_r is then computed as:

$$\nu_r(i) = \sum_{i_1=0}^{i} \sum_{i_2=0}^{i_1} \cdots \sum_{i_{\tau_q}=0}^{i_{\tau_q-1}} \Pr\{\mathbf{S}_r = \mathbf{s}_r\} \tag{2.3}$$

Computation of p According to our assumptions, the probability of infection p can be estimated by taking two conditions into account: (i) the considered node is chosen as a gossip destination and (ii) the gossip message is successfully received. This results in the following expression (remember that F is the protocol parameter fanout):

$$p = \overbrace{P_{gossip}}^{\text{(i)}} \overbrace{P_{succ}}^{\text{(ii)}} = \left(\frac{F}{n-1}\right) P_{succ} \tag{2.4}$$

Given a certain length (in hops) h of a routing path, the probability of a successful delivery is expressed as $P_{succ} = (1-p_f)^h$, i.e., there is no failure in each of the h hops. So we have:

$$P_{succ} = \sum_h (1-p_f)^h \Pr\{H = h\} = \mathbf{E}_H[(1-p_f)^H] \tag{2.5}$$

Therefore, p is expressed as:

$$p = \left(\frac{F}{n-1}\right) \mathbf{E}_H[(1-p_f)^H] \tag{2.6}$$

The distribution of H and the value of p_f are the network information we need. We refer to Appendix A for discussions about their estimations.

Reliability Degree $\mathcal{R}_{ds}$ and $\mathcal{R}_{dc}$ With the recurrence relation (2.3) of the single packet dissemination, the reliability degree can be expressed[12] in terms of $\nu(i)$ as follows. Note that the distribution of $\mathcal{R}_{ds}$ is always related to the group size n, while the distribution of $\mathcal{R}_{dc}$ is related to the number of packets in a stream, denoted by M in the formula.

$$\text{cdf of } \mathcal{R}_{ds}: \qquad \mathcal{F}_n(x) = \sum_{i=1}^{\lfloor nx \rfloor} \nu(i) \tag{2.7}$$

$$\text{cdf of } \mathcal{R}_{dc}: \qquad \mathcal{F}_M(x) = \sum_{i=0}^{\lfloor Mx \rfloor} \binom{M}{i} p_1^i (1-p_1)^{M-i} \tag{2.8}$$

where $p_1 = \sum i \cdot \nu(i)/n$ is the probability that a certain group member receives a single packet in a stream. Here we assume that the receptions of two distinct packets are independent events.

Network Load $\mathcal{N}_l$ The $\mathcal{N}_l$ for single packet dissemination is estimated straightforwardly by counting the number of unicast packets sent and the number of hops traveled by each of them:

$$\mathcal{N}_l = \mathbf{E}[S_{\tau_a}] \cdot F \cdot \tau_q \cdot \mathbf{E}[H] \tag{2.9}$$

---

[12] The subscript r is omitted hereafter, because we always consider the final distribution (i.e., after the last round).

Recall that τ_a limits the number of gossip rounds and τ_q defines how many times a packet is repeatedly relayed by a certain group member. The expression for $\mathcal{N}_l$ in the case of continuous packet dissemination is omitted as it becomes trivial with (2.9) and a given λ_o. This prediction is relatively rough because it is hard to find a way to precisely estimate the distribution of H as the distribution depends on several factors.

2.5.3 Stochastic Behavior of PAN

Since PAN directly uses RDG to diffuse an update, the distribution of $\hat{\xi}_W^r$ can be estimated with (2.3). The distribution of $\hat{\xi}_R$ can be expressed in a similar but more precise way, since τ_a is set to 1 (see Section 2.4.4). $\mathcal{N}_l$ is computable given the two distributions, but information about the time interval between a query and an update is necessary to compute $\mathcal{R}_{da}$.

Reliability Degree $\mathcal{R}_{da}$ According to the definition and the protocol description, this value is in fact the probability that a read quorum intersects the most recent corresponding write quorum. More precisely, we are looking for the probability that two subsets with sizes $\hat{\xi}_W$ and $\hat{\xi}_R$, taken from a set of n servers, intersect. Note that $\hat{\xi}_R$ is defined as the number of servers that effectively reply to the query back to its forwarding agent.

There exists an $\tilde{r}$ for which the dissemination process is finished, i.e., no new server is infected when $r \geq \tilde{r}$. Based on the assumption of synchronization (Section 2.5.1), we divide the time axis after a given update event β into $\tilde{r}+1$ intervals, as shown in Fig. 2.10. A read quorum, resulting from a query happening in-between two consecutive gossip rounds r and $r+1$, would have to intersect a write quorum of size $\hat{\xi}_W^r$ with a distribution ν_r. In order to find the probability of intersection, we need to calculate the read quorum size $\hat{\xi}_R$ (with a distribution μ) and p_r, the probability that the query event occurs in-between rounds r and $r+1$.

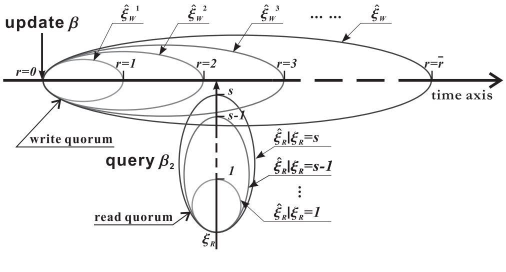


Figure 2.10: Incremental processes of read and write quorum size: $\hat{\xi}_W$ increases round by round, while $\hat{\xi}_R$ increases with the number of queries sent by an agent.

The distribution of $\hat{\xi}_R$, conditioned on $\xi_R = s$, is calculated as follows, with an initial value[13]

---

[13] Because the agent, one of the servers, has already received the query, it is sure that $\hat{\xi}_R = 1$, if ξ_R is set to 1.

$\Pr\{\hat{\xi}_R = 1|\xi_R = 1\} = 1$ and the convention $\Pr\{\hat{\xi}_R = k|\xi_R = s\} = 0$ if $s < 1, k < 1$ or $k > s$:

$$\begin{aligned} \mu_s(k) &= \Pr\{\hat{\xi}_R = k|\xi_R = s\} \ = \ \mu_{s-1}(k-1)p + \mu_{s-1}(k)(1-p) \\ &= \Pr\{\hat{\xi}_R = k-1|\xi_R = s-1\} \cdot p \\ &+ \Pr\{\hat{\xi}_R = k|\xi_R = s-1\} \cdot (1-p) \qquad k = 1, \cdots, s \text{ and } \ s \geq 2 \end{aligned} \tag{2.10}$$

where $p = \mathbf{E}_H[(1-p_f)^{2H}](1-p_e)$ is the probability that the agent forwarding a query receives the reply from a server belonging to the corresponding read quorum. The estimation of μ is somewhat conservative because servers with a relatively old data version do not reply to a query.

The time interval between an update and the second query to it is characterized by an Erlang distribution $\lambda_q^2 t e^{-\lambda_q t}$, with the assumption of a Poisson arrival process. Therefore, we have

$$p_r = \begin{cases} \int_{t_r}^{t_{r+1}} \lambda_q^2 t e^{-\lambda_q t} dt & r < \tilde{r} \\ \int_{t_r}^{\infty} \lambda_q^2 t e^{-\lambda_q t} dt & r = \tilde{r} \end{cases} \tag{2.11}$$

where t_r is when the round r starts. Now, the probability of intersection, i.e., $\mathcal{R}_{da}$, is expressed by taking an average over all possible cases:

$$\mathcal{R}_{da} = \sum_{r=0}^{\tilde{r}} \sum_{i=1}^{n} \sum_{j=1}^{s} \left(1 - \frac{\binom{n-\hat{\xi}_W^r}{\hat{\xi}_R}}{\binom{n}{\hat{\xi}_R}} \right) \mu_s(j)\nu_r(i)p_r \tag{2.12}$$

Network Load $\mathcal{N}_l$ For a certain $\mathcal{R}_{da}$ with its parameter pair F and ξ_R, we evaluate the corresponding $\mathcal{N}_l$ by averaging the load over a certain time unit (e.g., 1s), taking into account the arrival rate of updates and queries. Therefore, the loads generated by a single update and query are calculated separately, and then $\mathcal{N}_l$ is obtained by summing the products of the loads of the individual operations and their corresponding arrival rates.

$$\mathcal{L}_W = \mathbf{E}[\hat{\xi}_W] \cdot F \cdot \tau_q \cdot \mathbf{E}[H] \tag{2.13}$$

$$\mathcal{L}_R = 2 \cdot \xi_R \cdot \mathbf{E}[H] \tag{2.14}$$

$$\mathcal{N}_l = \lambda_u \mathcal{L}_W + \lambda_q \mathcal{L}_R \tag{2.15}$$

This estimation is conservative in the same sense as we mentioned before. Again, it is relatively rough compared with the one for $\mathcal{R}_{da}$, because we do not take into account the following two facts: 1) many packets get dropped before reaching their destinations, and 2) packets, especially those eventually dropped, may travel quite a long way due to stale routing information. We will show with simulations that the former fact has a dominating effect in most cases, but these facts tend to offset each other in some cases.

2.6 Simulations

This section presents the simulation results of our PILOT system in seven parts. Three subsections, 2.6.2 to 2.6.4, are dedicated to RDG/R^{2}DG (including a comparison between Anonymous Gossip and RDG) and the other four, 2.6.5 to 2.6.8, are devoted to PAN. The main goal is to confirm our claim that both the reliability degree $\mathcal{R}_d$ and the network load $\mathcal{N}_l$ of PILOT are predictable. The impact of the message arrival rate λ_o and of the server failures p_e on PAN, as well as the tunability of PAN, are also investigated by simulations in different settings.

2.6.1 Model and Parameters

The simulator we use is *ns-2* [FV02] with the Monarch Project wireless and mobile extensions. It provides both implementations of DSR and wireless MAC, based on the Lucent WaveLAN IEEE 802.11 product, with a 2Mbps transmission rate and a nominal range of 250m. We adopt the two-ray ground reflection model [Rap02] as the radio propagation model.

We simulate ad hoc networks in a square area of 1km^2. The movement pattern is defined by the "random waypoint" model [JM96]. The simulation parameters such as network size and maximum node speed are specified for each simulation. The STS for PAN always contains half of the network nodes. We do not justify this number[14], but only use it as an example. The servers in the STS are assumed to be predefined in order to simplify the simulation[15]. The client protocol is omitted to reduce side effects.

The gossip period is set to 200ms. For RDG/R^{2}DG, a CBR traffic generator produces 64 byte packets at regular intervals of 200ms, which gives a λ_o = 5pkt/s. The effect of the sending rate is not investigated in our current work. The arrival of queries or updates in PAN is emulated by a Poisson traffic source attached to each server, generating packets of 128 bytes with rate λ_o. We first investigate the impact of the overall access rate λ_o on the performance of PAN, then we take an appropriate value for all simulations. We set $\lambda_o = a\lambda_u$, and use $a = 8$ for most simulations. Only in Section 2.6.7, we change a to 4 in order to show the tunability of PAN. With certain simulation parameters (network size, maximum speed, pause time, and arrival rate), we vary the protocol parameters F and ξ_R in order to show the trade-off between the two metrics $\mathcal{R}_{da}$ and $\mathcal{N}_l$. As the last simulation parameter, p_e is first set to 1%, and then varied to show the sensitivity of PAN to server failures.

Both RDG/R^{2}DG and PAN are operated over 400 seconds of simulated time. The first 50 seconds of the simulation are used for system initialization. Then each traffic source continues generating traffic according to the predefined intensity until the end. Each simulation is carried out 10 times with different scenario files created by *ns-2*.

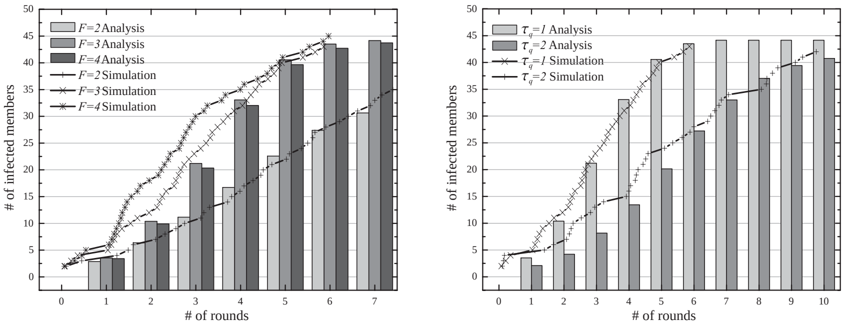
---

[14] It is not the goal of this thesis to find the optimal size for an STS, but we note that generally, the larger the size, the heavier the network is loaded, whereas the load on individual servers becomes smaller.

[15] Although the clustering algorithm is a popular way to elect representatives of the network, introducing such an algorithm into our simulation may only bring more overhead, without any help to show the essence of our system.

2.6.2 Single Packet Dissemination Reliability $\mathcal{R}_{ds}$

Fig. 2.11 shows the analytical and simulation results (i.e., the evolution of the infection processes) of the basic RDG protocol. The figure contrasts one process with another instead of providing the value of $\mathcal{R}_{ds}$ explicitly. These comparisons basically confirm that the theoretical prediction of the relationship between the reliability and the latency is valid.



(a) τ_q=*1* with different values for *F*

(b) *F*=*3* with different values for τ_q

Figure 2.11: Average number of infected members (simulation) and expected number of infected members (analysis) in time (expressed in rounds) with $n = 50$ in a network of 100 nodes. Each node has a maximum speed of 2m/s and an average pause time of 40s.

It is easy to observe that the reliability of the protocol with $F = 3$ is better than the one with $F = 2$, because the fanout has a significant effect on the reliability. However, when we further increase the fanout, the reliability decreases instead of increasing (analysis) or only marginally increases (simulation). The reason is that increasing the fanout has the same effect as increasing the number of connections, and p_f increases dramatically because of the network congestion. A similar reason accounts for what happens when τ_q changes from 1 to 2.

In fact, there is always a trade-off between certain requirements on reliability and the introduced overhead, characterized by the values of F and τ_q. Considering the network capacity imposes a further limitation not considered in other research proposals (considerably large F [KMG03] or unbounded τ_q [EGH$^+$03]). Therefore, for all simulations later in this chapter, we always take $F \leq 3$ and $\tau_q = 1$ for RDG.

2.6.3 Comparing Anonymous Gossip (AG) and RDG

A systematic comparison between RDG and AG [CRB01] (discussed in Section 2.2.3) is hard, due to their different design goals. We compare them in the context of small groups, which should actually favor AG since RDG is designed for larger groups. The comparison (Fig. 2.12) is done by superimposing a figure from [CRB01] with corresponding simulation results for RDG (for the same scenario). The figure shows that RDG is more reliable than AG in most cases. Furthermore, AG

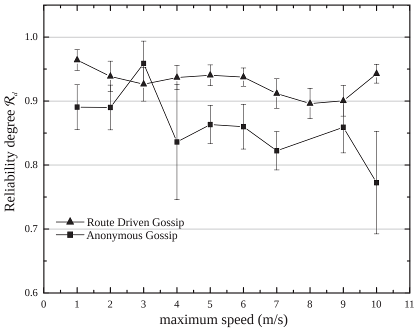
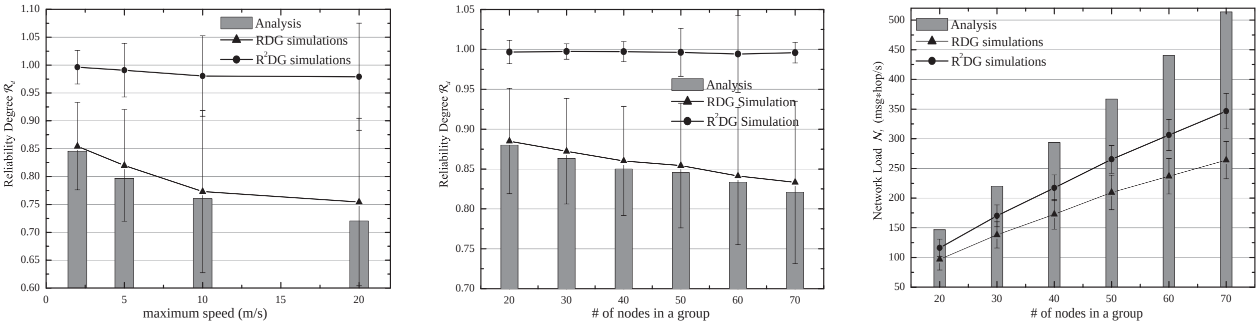


Figure 2.12: Reliability of the AG and RDG protocols in a network of 40 nodes with approximately one-third of them in a group, located within a square of 200m×200m. The maximum node speed varies between 1 ∼ 10m/s and the average pause time is 40ms. The transmission range is 75m.

cannot compete with RDG in terms of scalability because it is based on the underlying multicast protocol whose overhead is much larger than the one of the unicast protocol that RDG is based on. Finally, the reliability of the AG protocol is not as predictable as RDG's is, since it relies on the existence of an unpredictable multicast tree.

2.6.4 Continuous Packet Dissemination Reliability $\mathcal{R}_{dc}$ and Network Load $\mathcal{N}_l$

Fig. 2.13 shows $\mathcal{R}_{dc}$ and $\mathcal{N}_l$ of both RDG and R^{2}DG with different mobility patterns and group sizes. We provide here the mean value of $\mathcal{R}_{dc}$ and its standard deviation, which characterize the distribution function $\mathcal{F}$. The figures again exhibit the similarity between the simulation and analytical results with respect to RDG (see Section 2.5.2 for the explanation of the rough prediction



(a) 50 nodes in a group

(b) $Speed_{max}$=2m/s; $Time_{pause}$=10s

(c) $Speed_{max}$=2m/s; $Time_{pause}$=10s

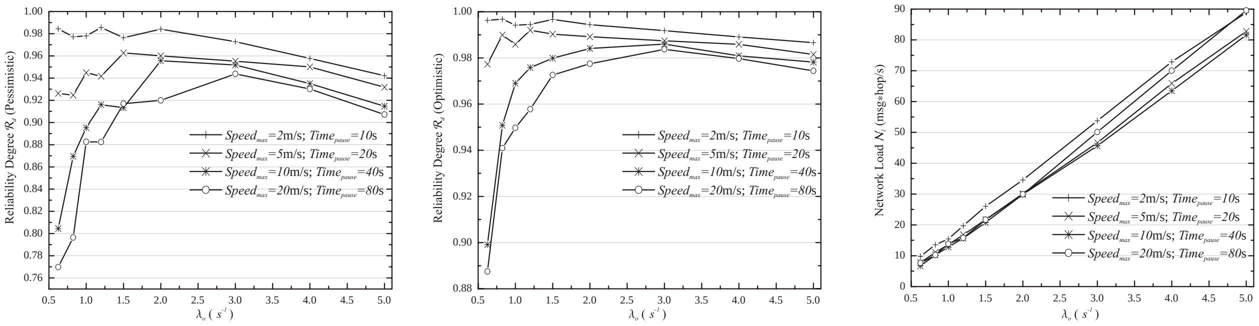
Figure 2.13: Reliability degree $\mathcal{R}_{dc}$ and network load $\mathcal{N}_l$ vs. mobility and group size in 100 nodes networks

on $\mathcal{N}_l$). As expected, R^{2}DG always performs better than RDG in terms of reliability, while the improvement is significant in high mobility and large group scenarios, thanks to the gossiper-pull mechanism. We also note that only a slight reliability degradation is observed (especially in the case of R^{2}DG) when the mobility or group size is increased (with a sub-linear increment of $\mathcal{N}_l$ in the case of increasing group size), illustrating the scalability of our protocols.

Note that two simulation parameters are paired to represent the mobility pattern such that each node has a maximum speed of 2m/s, 5m/s, 10m/s, and 20m/s, and a corresponding average pause time of 10s, 20s, 40s, and 80s, respectively (maximum speed is used as a symbol of the mobility pair in this case). This concept of mobility pattern will be used throughout the rest of this section.

2.6.5 Impact of λ_o on Pan Performance

Fig. 2.14 shows the performance of Pan (assuming $F = 2$ and $\xi_R = 4$) with respect to λ_o, the overall access rate. We observe that Pan performs in a relatively stable way for $1.5\text{s}^{-1} \leq \lambda_o < 3\text{s}^{-1}$, and $\mathcal{R}_{da}$ begins to degrade if we further increase λ_o, since the request arrival rate becomes larger than the service rate that Pan can provide. It is also natural to see that $\mathcal{N}_l$ increases linearly with λ_o by (2.15). However, it may seem somewhat odd to observe that $\mathcal{R}_{da}$ is very low in high mobility scenarios, when $\lambda_o < 1.5\text{s}^{-1}$. The main reason for this is the increased amount of stale routing information. In practice, this effect does not appear in the presence of background traffic. This problem can also be solved actively by requiring each StS server to send control packets during idle time in order to keep routing information fresh. Based on these observations, we apply $\lambda_o = 2\text{s}^{-1}$ for all other simulations.



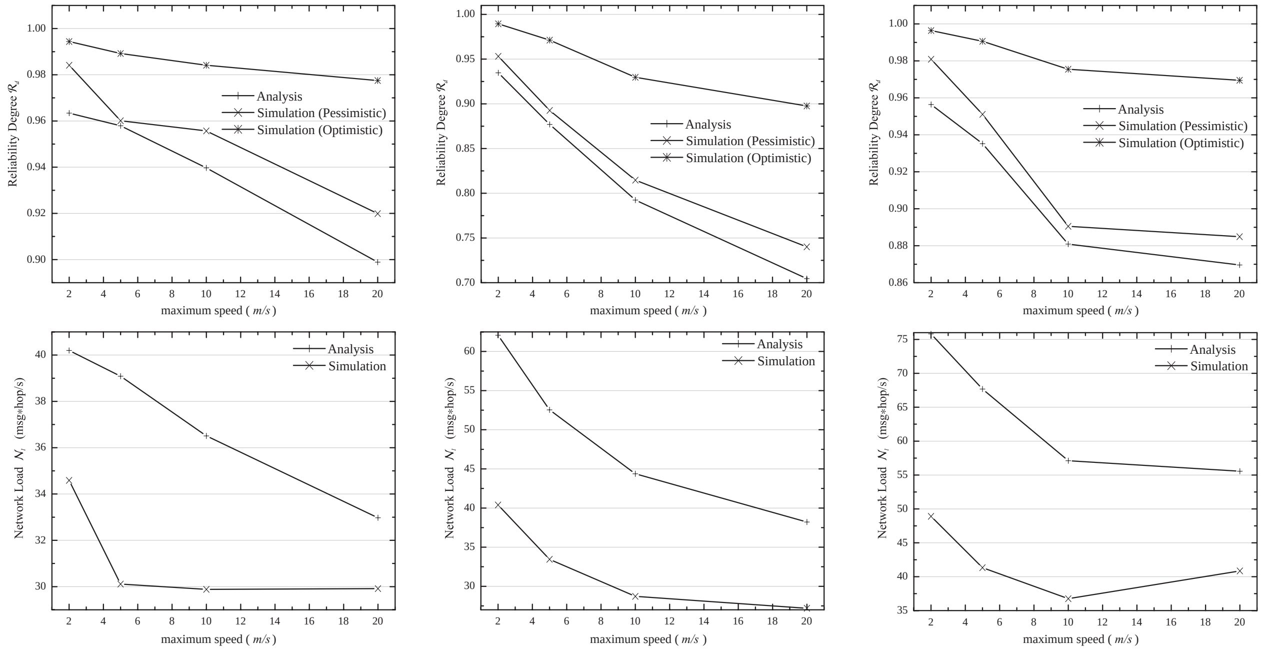
(a) Pessimistic reliability degree (b) Optimistic reliability degree (c) Network Load

Figure 2.14: Reliability degree $\mathcal{R}_{da}$ and network load $\mathcal{N}_l$ vs. overall access rate λ_o for 50 nodes networks.

The evaluations of $\mathcal{R}_{da}$ are presented in two ways. The "pessimistic" $\mathcal{R}_{da}$ refers to the probability that a query reaches the most recent update (with the same assumption as in Section 2.5.1 about the event order), whereas for the "optimistic" one, we consider a query to be successful even if it only retrieves the result of an update that occurred right before the most recent update. This second evaluation makes sense because, in practice, there are different data objects stored in an StS, and the probability that a queried data object has been modified by the most recent update is quite low. We will use these notations for all graph illustrations in the rest of this section.

2.6.6 Access Reliability $\mathcal{R}_{da}$ and Network Load $\mathcal{N}_l$

Fig. 2.15 shows comparisons between simulation and analytical results for networks of "normal" density, i.e., 50 nodes in an area of 1km^2, and "high" density, i.e., 100 nodes in an area of 1km^2. We vary the maximum speed and pause time to test the impact of mobility on the performance of Pan. The protocol parameters F and ξ_R are adjusted to cope with the increased network size. We note that a real number $x.y$ for the value of F means that each server, when gossiping an update, takes $F = x$ with probability $1 - y/10$ and $F = x + 1$ with probability $y/10$.



(a) Normal density network, *F=2* and ξ_R*=4*
(b) High density network, *F=2* and ξ_R*=4*
(c) High density network, *F=2.2* and ξ_R*=5*

Figure 2.15: Analytical and simulation results for reliability degree $\mathcal{R}_d$ and network load $\mathcal{N}_l$ vs. mobility pattern.

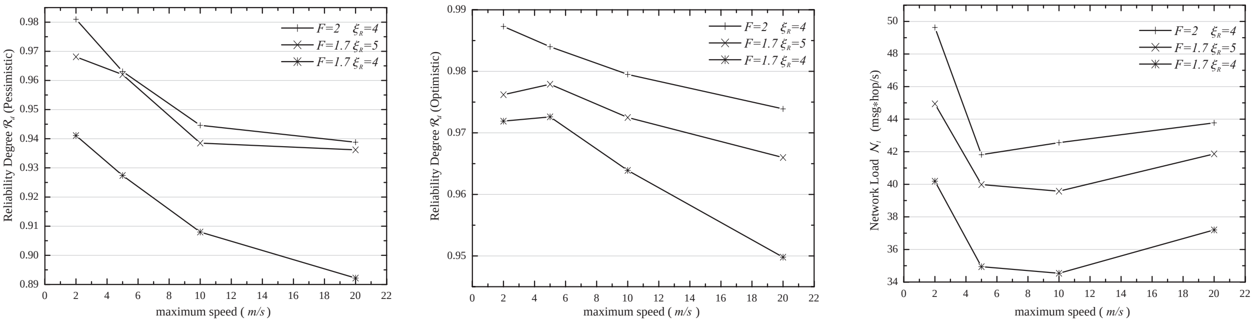
We make the following observations:

1. The simulation and analytical results of $\mathcal{R}_{da}$ match very well; this confirms the predictability on $\mathcal{R}_{da}$.

2. The analytical results of $\mathcal{N}_l$ provide certain information about the system overhead, such as the trend of its changes in different situations.
3. The optimistic $\mathcal{R}_{da}$ is always much higher than the pessimistic one; this basically means that the potential of PAN is much higher than what could be expected from the analytical results.
4. As the network size and the maximum node speed grow, protocol parameters have to be adjusted to maintain a good performance of $\mathcal{R}_{da}$, at the cost of an increased system overhead.

2.6.7 Adapting PAN to a New Environment

We simulate a new application environment by setting $a = 4$, i.e., increasing the update rate and decreasing the query rate. Intuitively, one would expect that shrinking the write quorum (decreasing F) and enlarging the read quorum (increasing ξ_R) would reduce $\mathcal{N}_l$, while still maintaining $\mathcal{R}_d$. We investigate this hypothesis with simulations shown in Fig. 2.16.



(a) Pessimistic reliability degree (b) Optimistic reliability degree (c) Network load

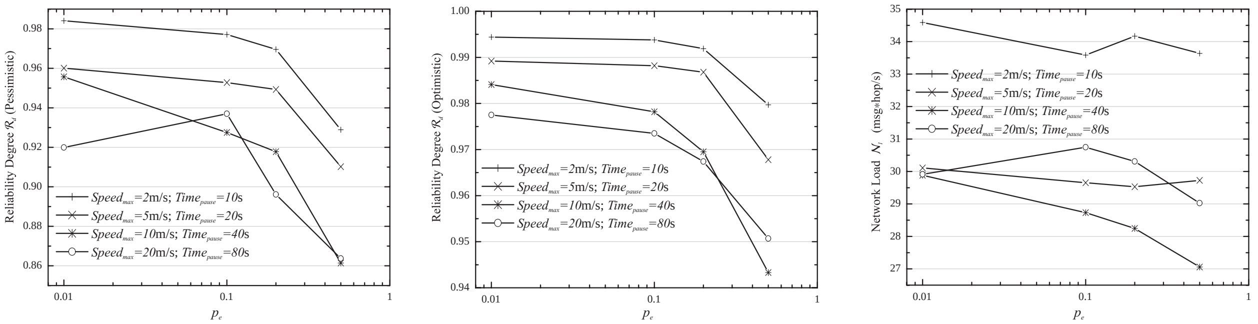
Figure 2.16: Reliability degree $\mathcal{R}_d$ and network load $\mathcal{N}_l$ vs. maximum speed with $a = 4$, for 50 nodes networks in a 1km^2 square.

The first set of design parameter adjustments (changing F from 2 to 1.7 and ξ_R from 4 to 5) results in a reduction of $\mathcal{N}_l$ (more than 5%) with virtually no expense of $\mathcal{R}_d$. Further parameter adjustments (reducing ξ_R to 4) yield significant reduction on $\mathcal{N}_l$ (up to 25%), but with a modest degradation of $\mathcal{R}_d$ (less than 5%). These results confirm what our intuition suggested.

2.6.8 Sensitivity to Server Unavailability p_e

According to the simulation results shown in Fig. 2.17, the sensitivity of PAN (assuming $F = 2$ and $\xi_R = 4$) to p_e increases as the node mobility grows. In addition, the sensitivity of PAN considering optimistic $\mathcal{R}_{da}$ is lower than the sensitivity considering pessimistic $\mathcal{R}_{da}$.

We also observe that the increase of p_e leads to an improvement of $\mathcal{R}_{da}$ in some cases. This paradox indeed suggests a way to optimize our system, i.e., a server belonging to a certain read



(a) Pessimistic reliability degree (b) Optimistic reliability degree (c) Network Load

Figure 2.17: Reliability degree $\mathcal{R}_{da}$ and network load $\mathcal{N}_l$ vs. server unavailability p_e for 50 nodes networks.

quorum would not always try to reply to a query back to its agent, even if the server is "alive" and has a new version of the queried data object. With such a behavior, Pan could avoid the case where more than one server replies to an agent with the same data object, thereby reducing the probability of packet collisions and, in turn, improving $\mathcal{R}_{da}$. However, we do not actually apply this optimization to Pan, because it is not as stable as the topology-awareness optimization in dynamic environments.

2.7 Conclusion

In this chapter, we are concerned with probabilistic reliable group communication in mobile ad hoc networks. We have studied two fundamental aspects, i.e., multicast and data sharing, within this framework and specified performance metrics that take the peculiarities of mobile ad hoc networks into account. We have proposed our Pilot system as a solution, based on the principle of gossip mechanisms and probabilistic quorum systems. The performance of Pilot has been analyzed by making use of, notably, epidemic theory. The evaluation and investigation of Pilot have also been carried out by simulations in *ns-2*.

As the first step towards building a probabilistic group communication toolkit, our Pilot system consists of two layers: RDG, at the bottom layer, is a gossip-based probabilistic reliable multicast protocol. At the upper layer, two dedicated services, R^{2}DG and Pan, provide continuous reliable packet dissemination and reliable data sharing, respectively. Our main contributions are:

1. an ad hoc adapted gossip mechanism,
2. a hybrid gossip including both push and pull,
3. gossip-based quorum access protocols, and

4. an asymmetric quorum construction.

We have proposed an analytical model to predict the performance of both RDG and PAN. The validity of these predictions has been evaluated by simulations. The results show that our analytical model provides predictions that are adequate for tuning the tradeoff between reliability degree $\mathcal{R}_d$ and network load $\mathcal{N}_l$. Our simulation results also show that, even under frequent topology changes, the reliability degrees of RDG/R^{2}DG and PAN are fairly high in practice. Finally, we have investigated also other aspects of PAN with intensive simulations, which confirm its robustness, in the sense that it can sustain a large access rate λ_o, different network sizes, and up to 50% server failures.

Chapter 3

DICTATE: Distributed Certification Authority with Probabilistic Freshness

3.1 Introduction

Ad hoc networks are collections of peered mobile nodes that communicate through wireless links. Such networks require stringent security protocols to protect their nodes from various attacks [ZH99, LKZ$^+$04]. However, implementing these protocols is difficult because these networks are constructed without using an on-line infrastructure and because the wireless links are particularly vulnerable. In this chapter, we design a secure and efficient *public-key infrastructure* (PKI) for ad hoc networks.

Public-key cryptography supports mechanisms that achieve security objectives such as confidentiality, authentication, and nonrepudiation. It can also pave the way for applying symmetric-key cryptography by bootstrapping a secured channel through mutual authentication and the establishment of a shared secret. However, a carefully planned PKI[1] is necessary to implement these security mechanisms. In the Internet, PKIs (e.g., [X50]) usually involve a *certification authority* (CA), which is a *trusted third party* (TTP) that certifies the authenticity of the binding between a public key and its subject entity. Whereas a CA can be implemented in a centralized server for a certain authority domain, a distributed implementation [RFLW96] could be preferable for improving availability. As an alternative, PGP [Zim95] is a more flexible PKI that enables users to enjoy public-key cryptography without any support from a CA.

In ad hoc networks, centralized CAs can work only for small authority domains, since the availability of such a CA would be problematic in a large domain due to the highly dynamic network topology. Based on this consideration and on previous results for wired networks, existing proposals for building a PKI in ad hoc networks can be classified into two main trends: 1) A single authority domain across the whole network with a distributed implementation of CA [ZH99, LKZ$^+$04] and 2)

---

[1] In this chapter, the term PKI is used in a broad sense. It refers to public-key management systems offering basic functions of certificate (or key) issuance, validation and revocation.

multiple authority domains of small sizes with a “centralized” authority for each [CBrH03].

In addition to distributing the certification authority, applying the joint authority approach [LABW92, Stu95] can further increase the security of a CA in large distributed systems. This approach advocates the combination of an off-line *identification authority* (IA) and an on-line *revocation authority* (RA). The IA authenticates the **initial** binding between a public key and its subject entity, and the RA keeps track of the status of certificates issued by the IA. Thanks to this separation, compromising the on-line authority (which is usually more vulnerable than an off-line authority) does not enable the adversary to issue certificates to new users, which limits the consequent damages. In spite of the apparent advantage, no known proposal for ad hoc networks has adopted this approach.

Our proposal improves the joint authority approach to build a PKI for ad hoc networks. Our approach uses an off-line IA to issue initial certificates and also to assign special nodes (or servers) to constitute an on-line **distributed** RA. We propose a DIstributed CerTification Authority with probabilisTic frEshness (DICTATE) to manage the on-line RA. DICTATE applies threshold cryptography to distribute trust among the RA servers. It also makes use of the services provided by PILOT, the group communication system we proposed in Chapter 2. To issue a (public-key) certificate, DICTATE requires a defined number of RA servers to sign the certificate and then replicates it in a quorum of RA servers. In response to a certificate query, DICTATE forwards it again to a quorum. Our underlying PILOT system guarantees that one quorum forms a probabilistic intersection with another, so that in practice a certificate query acquires, with a high probability, the most recent status of the targeted certificate. Our solution is network-friendly: DICTATE can tune the protocol performance **on-line** to a desired tradeoff between the freshness (of a certificate status) and overhead (to achieve the freshness), according to the level of the required freshness and network resource consumption.

The remainder of this chapter is structured as follows. Section 3.2 surveys the existing solutions and summarizes the motivations for our work. Section 3.3 details the problem to be solved and the system model. Section 3.4 presents our joint authority design and DICTATE protocols. Section 3.5 analyzes DICTATE against different attacks. Section 3.6 simulates DICTATE and compares the results with analytical results. Section 3.7 concludes the chapter.

3.2 Related Work

This section surveys PKIs in both wired networks and ad hoc networks. It shows that certain principles can be inherited from wired networks for implementing a PKI in ad hoc networks, but a direct migration would not perform well. We further summarize the rationale that leads to the design of our protocols at the end.

3.2.1 PKIs in Wired Networks

The X.509-based public-key infrastructure [X50], a representative of PKIs in wired networks, implicitly assumes centralized CAs. While the availability of such a centralized authority may become a problem in large distributed systems, individual CAs, whose compromise would paralyze the certification function of their domains, also appear to be single points of failure for security. The distributed implementation of a CA (e.g., Ω [RFLW96] and COCA [ZSvR02]) improves the availability of the certification function by organizing different certification servers into a peer-based structure. It also enhances the robustness of the authority against a certain amount of server failures through the use of threshold cryptography. However, all these benefits are obtained at the cost of additional protocol complexity. Particularly, maintaining a reliable group communication system [RFLW96] or Byzantine quorum systems [ZSvR02] is not a trivial task.

PGP [Zim95], an alternative to the PKI based on trusted authorities, provides practical security to protect low-value communications, such as e-mails. PGP is based on referral certification, which allows multiple users to "recommend" a certain user by signing certificates of its public key. This scheme is not perfectly secure because, for example, dishonest users may issue false certificates to cheat other users. The third solution to implement a PKI, SPKI/SDSI [SPK], has an egalitarian design similar to PGP. It circumvents the dependency on global name spaces, to which both X.509 and PGP are subject, with the concept of *linked local name spaces*.

3.2.2 PKIs in Ad Hoc Networks

The need for a PKI in an ad hoc network is due to the security requirements of various mechanisms, especially routing (e.g., [HPJ02, ZA02, HPJ03]). However, the distinctive features of ad hoc networks lead to designs of PKIs that are different from those in wired networks.

Zhou and Haas [ZH99] explore the issue of distributed CA in ad hoc networks, with the assumption of a single authority domain across the network. Their solution achieves availability by replicating certificates in multiple servers and employs threshold cryptography to thwart various attacks. However, [ZH99] does not contain a full description of protocols to maintain and control the access to the distributed CA. Luo et al. [LKZ$^+$04, LZK$^+$02] extend the work of Zhou and Haas by distributing the authority throughout the network. Their proposal focuses on performance: only localized protocols are used to access the certification function. Unfortunately, the on-line identification service provided by [LKZ$^+$04] seems to be vulnerable to the Sybil attack [Dou02], where an attacker can take enough shares within the CA by claiming several identities and can thus reconstruct the system's private key.

Hubaux et al. [HBv01, CBrH03] follow another approach by assuming each node to be its own authority domain. As a counterpart of PGP in ad hoc networks, the self-organized public-key management in [CBrH03] allows nodes to certify each other. With the assumption of transitive trust among nodes, appropriate certificate chains are found to verify the certificate of a public key. The disadvantage with such a scheme is that the assumption of a transitive trust could be too strong for mobile networks.

Recently, several proposals have extended the aforementioned approaches in different ways. Extensions to Zhou and Haas's work usually try to solve two problems left open in [ZH99]: 1) how to select the CA servers and 2) how to maintain the CA. Yi and Kravets [YK03] suggest selectively assigning powerful nodes as CA servers and apply multiple unicastings for accessing CA. They do not explicitly explain how the initial identification is performed when a node joins a network for the first time. Bechler et al. [BHK$^+$04] introduce a cluster-based architecture for supporting a distributed CA. However, the on-line identification service relies on referral certifications from existing network members, which could weaken the security level of the system. Khalili et al. [KKA03] apply an identity-based approach instead of the certificate-based one [YK03, BHK$^+$04]. They use a set of pre-configured nodes to form the distributed authority and apply localized protocols, similar to those of [LKZ$^+$04], for the key generation service. Unfortunately, key revocations appear to be difficult because the key generation service refuses to issue keys for a particular identity more than once in order to thwart identity spoofing.

An extension to the trust model in [CBrH03] is described in [WT02]. The trust chain and recommendation protocol used in [CBrH03] are again applied but supplemented by a reference protocol. This solution does not commit to perfect security since it is dedicated to low-value communications. A downside of such protocols is that they do not address the security issues with a network-oriented point of view (for example, the topology of a trust graph does not match the changing network topology), which could impair their viability in ad hoc networks. Montenegro and Castelluccia [MC02] describe a way of binding an identity to its public key without the need of a certificate: the hash of the public key is used as part of the IP address. Unfortunately, a node would have to change its "name" (or identity) upon a key revocation.

A common weakness of PKIs in ad hoc networks is the lack of proper revocation mechanisms. While proposals in [KKA03, BHK$^+$04, WT02] do not address the certificate revocation, the solutions in [LKZ$^+$04, CBrH03, YK03] rely on proactive mechanisms to **push** a certificate revocation list (CRL) to other nodes. Although no quantitative evaluation is provided in [LKZ$^+$04, YK03], there is no doubt that proactive pushing, by flooding the network, consumes network resources constantly. Therefore, an on-demand scheme that queries the status of a certificate in question would be more suitable for ad hoc networks.

3.2.3 Lessons from the Past

According to the experiences from the previous work mentioned above, we summarize our design rationale for the PKI in ad hoc networks as follows:

- Authentication techniques differ in the level of protections that they provide for the targeted communications. Relying on a TTP as an authority yields a high-level protection to secure high-value communications in large-scale networks.

- Performing initial identifications with a full on-line certification service is questionable. The joint authority approach that integrates an off-line IA and an on-line distributed RA can achieve both security and availability of the CA.

- The proactive share refreshing, which is already expensive enough in wired networks, is not suitable for ad hoc networks (except an extreme case [LKZ$^+$04] where localized protocols apply). Certain out-of-band mechanisms have to be used for refreshing key shares.
- The network performance should be kept in mind when addressing security issues. Security protection can be somewhat sacrificed to spare network resources in some cases.
- Certificate revocation is an important but often neglected CA service. In wireless networks, it becomes even more significant because (suspected) key compromises can happen more often.

3.3 Goal and Model

This section states the problem to be solved and models the considered environment.

3.3.1 Problem Statement

We consider a relatively large-scale ad hoc network (consisting of tens or even hundreds of nodes) with a random mobility pattern, i.e., nodes moving independently within a given field. The network (real life examples include networks temporarily built for rescue or exploration operations) has intermittent connections to a *mother certification authority* (or mCA), which is a trusted authority connected to the backbone with its public data (including its public key) known to all wireless nodes. When the network is disconnected from the mCA (e.g., to perform a rescue operation), it requires a CA that serves requests from nodes to update their public key or to query public key certificates of other nodes. We first give the three properties of a secure CA in such a network as follows:

- **Liveness**: The CA always processes a request in a finite amount of time.
- **Safety**: An adversary is never able to forge a certificate.
- **Freshness**: A query to the CA returns the most recent status of a targeted certificate.

Then we specify our *CA with probabilistic freshness*: it meets the liveness and safety properties, and it ensures the freshness property with a probability that is termed $\mathcal{F}_d$, or *freshness degree.*

For any correct nodes, our goals are to:

1. make the CA compliant with the specifications above,
2. provide a certain flexibility for the CA to tune the protocol performance (with respect to $\mathcal{F}_d$ and the overhead) **on-line** to the desired tradeoff,

3. maintain an adequate efficiency, i.e., incur reasonable overhead (defined as the *network load* in Section 2.3.2) even in the case of a high freshness requirement, and
4. keep all protocols transparent to a human user.

3.3.2 System and Adversary Model

We assume that each node owns a unique identity. We also assume that, in the network, there is a subset (typically 10% to 20% of the network)[2] $\mathbf{N}$ ($|\mathbf{N}| = n$) of securely protected and computationally powerful nodes (which the mCA can identify and will use to constitute a distributed CA). A subset $\mathbf{T} \subset \mathbf{N}$ can be compromised during a certain period of time, where $|\mathbf{T}| = t < n/3$.[3] The remaining part of $\mathbf{N}$ consists of a set $\mathbf{C}$ of correct nodes.

For the adversary model, we consider the following attacks that could be mounted by a malicious node:

- *Impersonation*: A node pretends to be someone else to submit a certificate update.
- *Key Compromise*: A node tries to compromise the private keys of other nodes.
- *Denial of Service (DoS)*: A node tries to slow down the CA by clogging the resources (especially communication resources).
- *Misc*: A node launches eavesdropping, message insertion, corruption, deletion, and replay attacks.

In addition to these attacks, a compromised node may exhibit Byzantine failure, i.e., deviate arbitrarily from the (DICTATE) protocol specification.

3.4 Our Solution: Joint Authority and DICTATE

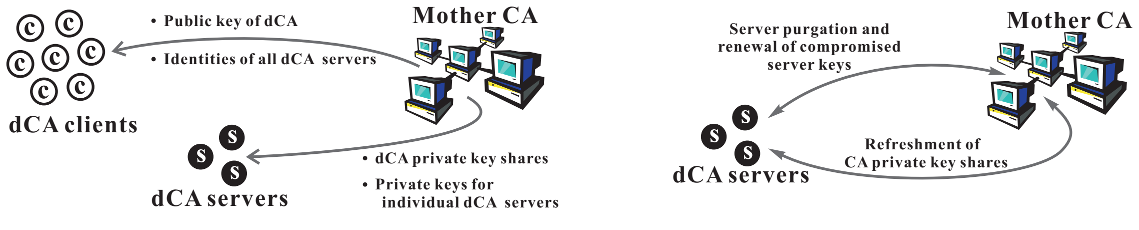
Our solution takes the joint authority approach [LABW92, Stu95]: the mCA (which is connected to the backbone) acts as the off-line IA and it assigns the set $\mathbf{N}$ of special nodes to constitute a distributed CA (dCA hereafter) that performs the role of the on-line RA (Fig. 3.1). The dCA nodes are named *server*s, and other nodes are named *client*s. The mCA controls the admission of a node (either a server or a client) to the network at its command, through the issuance of a certificate that asserts the binding between the identity and initial public key of the node. When the network is disconnected from the mCA, clients submit their requests to the dCA. On one hand, a *query*

---

[2] It is not our goal in this chapter to find the optimal size for this set, but we note that generally the larger the size is, the more heavily the network is loaded, whereas the load on individual nodes becomes smaller.

[3] Although probabilistic quorum systems [MRW01] (unlike Byzantine quorum systems [MR98]) still work if $t \geq n/3$, and a threshold cryptosystem requires only $t < n/2$, the system performance degrades dramatically when t goes beyond $n/3$. Therefore, we require $t < n/3$ even though our CA is based on the principle of probabilistic quorum systems.

request returns the public key certificate of another client. On the other hand, an *update* request updates a client's public key certificate stored in the dCA. Our solution prevents key compromises through the following revocation[4] mechanism: a certificate should be periodically updated or it will become invalid. The dCA guarantees the legitimacy of an update only if the private key of the client who submits the request has not been stolen or been compromised. Otherwise the client has to re-identify itself with the mCA through out-of-band mechanisms. A client does not always query a certificate in the case of secure communication; it performs a query only if it suspects the validity of the certificate (e.g., due to a relatively small version number, which we will explain later in this section) that it obtains directly from another client.


• Public key of dCA
• Identities of all dCA servers
Mother CA
dCA clients
• dCA private key shares
• Private keys for individual dCA servers
dCA servers
Server purgation and renewal of compromised server keys
Mother CA
Refreshment of CA private key shares
dCA servers

(a) Initialization phase (b) Checktime

Figure 3.1: Operating principles of the joint CA: (a) initialization and (b) checktime.

Our dCA has a public/private key pair. This public-key pair is based on RSA.[5] The public key, which bears a certificate issued by the mCA, is known to the whole network. The private key of the dCA is shared among the dCA servers by a $(t+1, n)$ robust threshold cryptosystem [GRJK00]. A credential generated with the private key testifies the authority of the whole dCA. Each dCA server, like a usual node, has its own public/private key pair. We apply an identity-based scheme [BF01], and a corresponding signature scheme [CC03], for this public-key pair and make the identities of all dCA servers publicly known, such that any network node knows the public key of a dCA server. The private key of a server is generated by the mCA; it only represents the authority of an individual server. Applying an identity-based cryptosystem incurs much less communication overhead (we elaborate on this issue when explaining the key revocation in the next paragraph, and we also describe the usage of this cryptosystem in Sections 3.4.2 and 3.4.3). The system initialization is illustrated in Fig. 3.1 (a).

Periodically,[6] there is a *checktime*, as shown in Fig. 3.1 (b), at which the dCA servers (physically)

---

[4] Strictly speaking, our solution revokes only (possible) compromised keys rather than compromised nodes [Sta03]. Since achieving the latter involves intrusion/misbehavior detections whose existing solutions (e.g., [ZLH03, LKZ$^+$04]) are vulnerable to various attacks, we provide interfaces (described in Section 3.4.3) in our protocols to accommodate future solutions.

[5] The threshold cryptosystem based on RSA allows a non-interactive signing protocol, which would not be the case, for example, of an ElGamal-like threshold cryptosystem.

[6] The length of this period depends on the hostility of the environment that a given remote operation of the network

go back to the mCA for a “purgation” (only dCA servers should go through this procedure; clients can still perform their remote operations). During the checktime, the mCA, through out-of-band mechanisms, detects compromised servers and has them re-initiated or substituted by new ones; it also refreshes the secret shared among the dCA. Since on-line detection of compromised servers is hard and cooperative detection schemes [ZLH03] may suffer from blackmail attacks, off-line detection seems to be the only way to guarantee security. Note that server identities are fixed, and the key revocation of a server is done by using as public key the combination of the identity and the time stamp corresponding to a certain *checktime interval* [BF01] (i.e., no information such as revocation lists or certificates needs to be distributed). A client can verify the validity of a message from any server with the knowledge of its identity and a local clock loosely synchronized with the checktime. The above descriptions show that, unlike the traditional joint authority approach, the mCA not only provides identification service but also manages the dCA. Therefore, we use the terms mCA and dCA instead of IA and RA in this chapter.

The interactions between nodes (including servers and clients) and the mCA (e.g, identification, key generation, and refreshing shared secrets) are well-defined in existing proposals [MvOV97, GRJK00, BF01], so we do not discuss them in detail; in the remainder of the chapter, we rather focus on DICTATE, which maintains the on-line part of our certification service (i.e., the dCA). We first introduce the principles of PILOT [LEH04], the basis of DICTATE, then we describe operations of DICTATE in detail.

3.4.1 Overview of PILOT

PILOT (Chapter 2) is a group communication system that provides services for multicast and data replication. The parts of our PILOT system used to implement DICTATE are illustrated by the grey part in Fig. 3.2. PILOT is a two-layer system. It has a probabilistic multicast protocol, Route Drive Gossip (RDG), as its basis. The protocol is gossip-based in nature: it proceeds **round**[7] by round and the receivers in each round are chosen randomly (weighted according to the length of the routing path); they **relay** packets to the receivers of the later round(s) (we allow a packet to be relayed once **only** to the receivers of the **next** round in this chapter). This protocol guarantees that the reliability degree, defined as the fraction of a multicast group receiving a given packet, is predictable in a probabilistic sense. The *fanout*, F, is a very important parameter related to RDG; it refers to the number of receivers chosen by a certain sender in a round and thus strongly influences the protocol reliability.

Upon this layer, the Probabilistic quorum system for AD hoc NETWORKS (PAN) provides reliable

---

involves and the amount of time spent for the operation; it should guarantee that no more than t dCA servers could be compromised during such a period.

[7] The duration of a round is a parameter of RDG: it should be short enough to maintain a low propagation delay but be long enough to avoid network congestion. Although we make a synchrony assumption (in terms of rounds) in Section 3.5.2 to facilitate our analysis, it does **not** mean that PILOT is a **synchronous** system. In fact, nodes are not synchronized in our simulations.

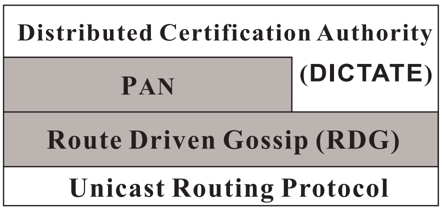

Distributed Certification Authority
(DICTATE)
PAN
Route Driven Gossip (RDG)
Unicast Routing Protocol

Figure 3.2: The protocol structure of DICTATE: it is built upon PILOT (the grey part).

data sharing. It assumes a special group of nodes to store the shared data in a replicated manner. Any node belonging to the group is termed *server*, and the rest of the nodes are termed *clients*. A data query or update is directed to an arbitrary server, and its dissemination within the group is performed by the PAN server query protocol or update protocol (the two protocols differ in that a query requires a reply and an update does not). Since the PAN server query and update protocols rely on RDG, the probability that a query acquires the most recent update of the corresponding data object is again predictable and rather high in practice. Certain parameters, including F, have to be set when a primitive in PILOT is invoked. These parameters determine the protocol performance, in terms of reliability and overhead (see Section 2.4.2).

Previous approaches for guaranteeing the reliability of queries and updates in a certification service apply either reliable group communication systems [RFLW96] or Byzantine quorum systems [ZSvR02]. These approaches incur too much overhead and are thus not practical in ad hoc networks. Our approach, on one hand, builds a certification service upon probabilistic quorum systems (PAN) and thus provides a way of flexibly trading reliability for efficiency. On the other hand, the use of probabilistic multicast (RDG) in our approach allows us to fulfill the threshold signature scheme with $(t+1)$anycasts.

3.4.2 Protocol Overview

In this section, we summarize the main operations of DICTATE. These operations are classified into external and internal (with respect to the dCA server group) protocols, according to the entities that are involved. The rationale behind these operations is explained in Section 3.4.3.

The notations used throughout all subsequent sections are as follows:

s, sid: DICTATE (or dCA) server and its identity.

c, cid: DICTATE (or dCA) client and its identity.

$\langle m\rangle_k$: message m with signature signed by key k (identity-based signature) of a server.

$[m]_k$: message m with signature signed by key k (RSA signature).

K_D: public key of DICTATE (or dCA).

k_D, k_{Ds}: private key of DICTATE (or dCA) and key share, respectively.

$K_?$, $k_?$: public key and private key of node "?", respectively. "?" can be c, s, or a.

$[Ct_K]_k$: certificate of public key K signed by private key k. The format of its data part is $[cid, K, v]$, where *cid* identifies the owner of K and v is the version number. Note that the certificate is not timestamped because it is not trivial to agree on a common timestamp among a set of servers who sign the certificate. Instead, the version number serves as a timestamp.

DICTATE External Protocol

This part of DICTATE runs between clients and servers.

Access Control When a client c wants to access DICTATE, it tries to contact an arbitrary DICTATE server s, and they exchange the following messages:

$$\begin{array}{rcl} c \rightarrow s & : & \mathcal{A}_c = [cid, [Ct_{K_c}]_{k_D}, rtype, seqno]_{k_c} \\ c \leftarrow s & : & \mathcal{A}_s = \langle sid, seqno \rangle_{k_s} \end{array}$$

where *rtype* refers to the request type (update or query) and *seqno* is a sequence number. The server s admits the access of the client c with $\mathcal{A}_s$ only if $\mathcal{A}_c$ is proved to be valid. After successfully finishing this interaction, this server becomes the *agent* (later referred to as a with identity *aid*) of the client for this specific request.

Client Request and Server Reply A client sends certificate update and replication requests to update its public key and related certificate, and it sends a certificate query to get the current public key certificate of another client.

To **register a new public key**, a client c first generates a new public/private key pair K'_c/k'_c and then sends an update request to its agent a:

$$c \rightarrow a \;\;:\;\; \mathcal{U}_c = [cid, [K'_c]_{k_c}]_{k'_c} \quad (\longrightarrow \text{ in Fig. 3.3})$$

where k_c is the current private key whose corresponding public key K_c is to be updated. Upon completing the task of certificate update, the agent responds with the following message:

$$c \leftarrow a \;\;:\;\; \mathcal{U}_a = [Ct_{K'_c}]_{k_D} \quad (\dashleftarrow \text{ in Fig. 3.3})$$

The client c verifies $[Ct_{K'_c}]_{k_D}$ using K_D. If the certificate is valid, c sends a replication request to a, so that the certificate will be stored in the dCA for future queries:

$$c \rightarrow a \;\;:\;\; \mathcal{R}_c = [cid, [Ct_{K'_c}]_{k_D}, \mathcal{F}_d]_{k_c} \quad (\longrightarrow \text{ in Fig. 3.4})$$

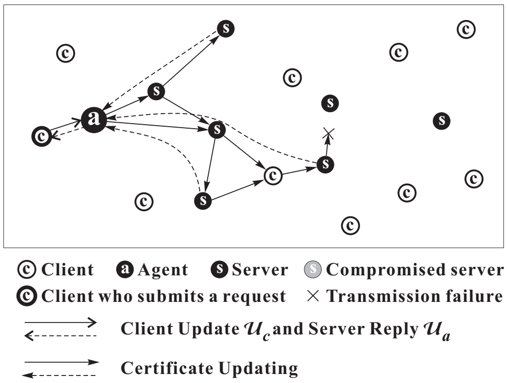

a
Client
Agent
Server
Compromised server
Client who submits a request
× Transmission failure
Client Update $\mathcal{U}_c$ and Server Reply $\mathcal{U}_a$
Certificate Updating

Figure 3.3: DICTATE operations: Certificate Update. Note that, since DICTATE uses the service provided by a multihop routing protocol, any node (either a server or a client) can serve as relay to forward a message.

where $\mathcal{F}_d$ indicates the required freshness degree (defined in Section 3.3.1). Finally, the agent a provides its client c with evidence that it has faithfully respected the protocol:

$$c \leftarrow a \quad : \quad \mathcal{R}_a = \langle sid_1, [Ct_{K'_c}]_{k_D} \rangle_{ks1}, \langle sid_2, [Ct_{K'_c}]_{k_D} \rangle_{ks2}, \cdots \qquad (\dashleftarrow \text{ in Fig. 3.4})$$

where $\langle sid_1, [Ct_{K'_c}]_{k_D} \rangle_{ks1}, \langle sid_2, [Ct_{K'_c}]_{k_D} \rangle_{ks2}, \cdots$ is a list of signatures generated by servers that receive $\mathcal{R}_c$; it proves that those servers have indeed received the new certificate. Though an abuse of concept, here we refer to $\langle sid, [Ct_{K'_c}]_{k_D} \rangle_{k_s}$ only as the signature (no plaintext included) for the concatenation of a server id sid and the certificate $[Ct_{K'_c}]_{k_D}$.

To **obtain the public key certificate of a client**, a client c sends a query request to its agent

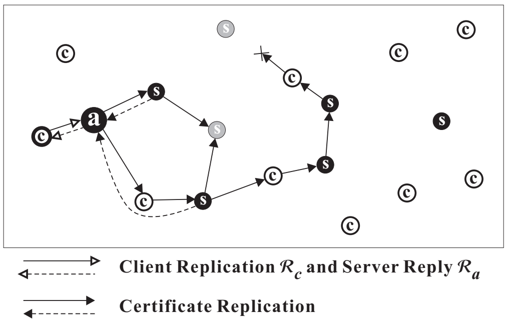

a
Client Replication $\mathcal{R}_c$ and Server Reply $\mathcal{R}_a$
Certificate Replication

Figure 3.4: DICTATE operations: Certificate Replication. Further legends can be found in Fig. 3.3

a:

$$c \rightarrow a \quad : \quad \mathcal{Q}_c = [cid, \hat{cid}]_{k_c} \quad (\longrightarrow \text{ in Fig. 3.5})$$

where $\hat{cid}$ is the identity of the owner of the queried public key certificate. A server s that receives the request replies **directly** to c with the following message:

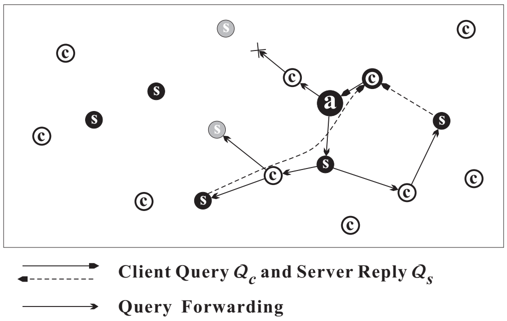


Figure 3.5: DICTATE operations: Certificate Query. Further legends can be found in Fig. 3.3

$$c \leftarrow s \quad : \quad \mathcal{Q}_s = \langle sid, [Ct_{K_{\hat{cid}}}]_{k_D} \rangle_{k_s} \quad (\dashleftarrow \text{ in Fig. 3.5})$$

Clients involved in the aforementioned protocols should be able to perform signing (with a RSA key), signature verifications (especially pairing computations [CC03]), and generating public/private key pairs. While a laptop does have this capability, small mobile devices (e.g., PDA) would need more powerful processing units to perform these computations.

DICTATE Internal Protocol

This part of DICTATE runs among servers.

Certificate Update An agent a forwards a valid client update to several servers via the PAN server query[8] protocol with a message $\langle aid, \mathcal{U}_c, [Ct_{K_c}]_{k_D} \rangle_{k_a}$ ($\longrightarrow$ in Fig. 3.3) and waits for enough copies of partially signed certificates $[Ct_{K'_c}]_{k_{D_s}}$ to come back ($\dashleftarrow$ in Fig. 3.3). The agent then tries to combine all these certificates to create a valid one, $[Ct_{K'_c}]_{k_D}$, signed by the private key of DICTATE, and returns this new certificate back to the client via the message $\mathcal{U}_a$.

---

[8] Although this protocol is a part of the DICTATE server update protocol, it invokes the underlying query protocol of PAN because it expects a reply from each receiver.

Certificate Replication The agent, upon receiving a replication request $\mathcal{R}_c$ from its client, replicates the certificate via the PAN server protocols with a message $\langle aid, \mathcal{R}_c \rangle_{k_a}$ (⟶ in Fig. 3.4). As a consequence, a quorum Θ_U ($|\Theta_U| = 5$ in Fig. 3.4)[9] of servers receives $\langle aid, \mathcal{R}_c \rangle_{k_a}$. The agent then expects acknowledgements $\{\langle sid, [Ct_{K'_c}]_{k_D} \rangle_{k_s}\} : s \in \Theta_{\bar{U}}$ from a set $\Theta_{\bar{U}}$ ($|\Theta_{\bar{U}}| = 2$ in Fig. 3.4) of servers (⇠ in Fig. 3.4), where $\Theta_{\bar{U}} \subseteq \Theta_U$. The size of Θ_U and $\Theta_{\bar{U}}$ is determined by the required freshness degree $\mathcal{F}_d$. Finally, the agent replies to the client with $\mathcal{R}_a$.

Query Forwarding An agent a forwards a valid client query to several servers via the PAN server update protocol with a message $\langle aid, \mathcal{Q}_c \rangle_{k_a}$ (⟶ in Fig. 3.5), but it does not expect replies from other servers. The replies $\mathcal{Q}_s$ from a quorum $\Theta_{\bar{Q}}$ ($|\Theta_{\bar{Q}}| = 2$ in Fig. 3.5) of servers are directly sent back to the client.

Detailed proofs of the protocol compliance with our specification are provided in Section 3.5 (where we also explain how to determine the quorum size). Here we only give some intuitive ideas on the relationship between the freshness degree $\mathcal{F}_d$ and the quorum sizes:

$$\mathcal{F}_d = \Pr\{\Theta_U \cap \Theta_{\bar{Q}} \nsubseteq \mathbf{T}\} \tag{3.1}$$

while $\mathcal{F}_d = 1$ iff

$$|\Theta_{\bar{U}}| + |\Theta_{\bar{Q}}| \geq n + t + 1 \tag{3.2}$$

Note that $\exists$ server $s : s \in \Theta_U \vee s \notin \Theta_{\bar{U}}$ could happen, because a reply from s to a might not get through due to the unreliable routing protocol. It is enough that $\Theta_U \cap \Theta_{\bar{Q}} \nsubseteq \mathbf{T}$ for probabilistic guarantee (3.1), but a deterministic guarantee (3.2) requires $\Theta_{\bar{U}} \cap \Theta_{\bar{Q}} \nsubseteq \mathbf{T}$.

3.4.3 Protocol Rationale

This section explains the underlying principles of the DICTATE protocols.

DICTATE External Protocol

Access Control A server verifies the validity of $\mathcal{A}_c$ by checking whether:

1. the nonce $\{cid, seqno\}$ is unique,
2. the certificate $[Ct_{K_c}]_{k_D}$ is valid and matches cid,
3. the signature by k_c is verifiable with K_c, and
4. the client is authorized, by the access control policy, to perform *rtype*.

---

[9] We deliberately use different symbols to represent a quorum in PAN (ξ_W and ξ_R) and in DICTATE (Θ_U and Θ_Q); the goal is to show that, although it is the same nodes involved in a given quorum, the protocols concerned are running at different layers.

We do not detail the access control policy in this chapter. This policy can be specified according to different application requirements and, in particular, provides an interface to accept inputs from certain intrusion/misbehavior detection algorithms (whose accusations against certain nodes disallow the access of those nodes and hence revoke their certificates). The access control phase is important because it allows a request (especially for certificate update) to be checked against certain policies before the client actually performs the computationally intensive key generation. The server maintains a state for each valid request until the request is fulfilled.

A client updates the nonce if it needs to retransmit $\mathcal{A}_c$ (e.g., due to the loss of $\mathcal{A}_s$). A server replies to every $\mathcal{A}_c$ that has a unique nonce (but only keeps one request state for the client if the time interval between two requests is too short). As a result, no reply $\mathcal{A}_s$ would be sent to an adversary that launches a replay attack.

Client Request and Corresponding Reply The role of the update request $\mathcal{U}_c$ is to prove to the agent that the client c owns the private keys k_c and k'_c corresponding to the current public key K_c and the newly proposed public key K'_c, respectively. The agent is able to check the validity of $\mathcal{U}_c$ given the $[Ct_{K_c}]_{k_D}$ transferred in the access control phase.

If an agent were trustworthy, it could be asked to directly perform the replication after obtaining $[Ct_{K'_c}]_{k_D}$, without notifying its client. However, we want to protect the client against possible compromise of the agent. The server reply $\mathcal{U}_a$ allows the client to check the validity of the new certificate before asking the agent to replicate it. Also, each server s that receives the replication request $\mathcal{R}_c$ is required to provide the "receipt" $\langle sid, [Ct_{K'_c}]_{k_D}\rangle_{k_s}$ to the client. This scheme defends DICTATE against a compromised agent who pretends to be correct by finishing the certificate update task but fails to replicate the certificate to other DICTATE servers afterward. In addition, the client mentions its required freshness degree[10] $\mathcal{F}_d$, so that the agent can set proper parameters to invoke the underlying protocols according to the requested degree. There is no way for both a replication and a query to request a freshness degree, otherwise the system performance becomes unpredictable. Although greedy clients might always request the highest freshness degree (i.e., $\mathcal{F}_d = 1$), which incurs a large protocol overhead, DICTATE discourages greed by means of an implicit "self-castigation": the higher $\mathcal{F}_d$, the longer the delay of the response. Note that a lower freshness degree does **not** mean a lower level of security; we refer to Section 3.5.2 for detailed explanations.

A query request $\mathcal{Q}_c$ is propagated through the agent, as it needs to go through certain sanity checks. But the reply should not go through the agent, otherwise a compromised agent could return bogus information to the client and the client has no way of knowing whether the agent is telling the truth or not. Therefore, the client should collect all replies and select the most recent version of the required certificate by itself.

---

[10] This value is set by an application without involving a human user, based on the value of the communication that will use the key later.

DICTATE Internal Protocol

In previous literature (e.g., [ZSvR02]), it is considered very important that a CA be independent from other network nodes. This requirement improves the scalability of a network because, for example, a client does not need to be informed about the revocation of a server public key. DICTATE is also divided into two parts, i.e., external and internal, for the same purpose. However, since we do not want to trust an agent (the bridge between the two parts), we would like a client to be able to verify the response from any server. This explains why we apply an identity-based public-key system for DICTATE servers: it grants the ability of verifying servers' responses to a client without jeopardizing the network scalability. Alternatively, a solution to our problem can be based on the concept of *Byzantine Fault Tolerance* [ZSvR02]. Such a solution would let a client access a set of agents ($t + 1$ in the case of t possible compromised servers) to "mask" the failure. Unfortunately, a resulting scheme would leave the certification service open to easily launched resource-clogging DoS attacks in wireless networks, and cause a well-behaved client to experience a very long delay for a request.

No underlying secure group communication scheme (e.g., [STW00]) is needed to support DICTATE. The system is built upon Pilot, which is in turn based directly on unicast routing; therefore it can secure itself with the prerequisite that the public key of each server is known to the whole network.

Certificate Update An agent a signs the update request $\mathcal{U}_c$ and the previous certificate $[Ct_{K_c}]_{k_D}$ from its client c and disseminates this message within DICTATE via the Pan server query protocol.[11] Each server again verifies the validity of $\mathcal{U}_c$, in the same way the agent did. A server s, upon validating the request from a, updates the certificate Ct_{K_c} by increasing the version number v as $v = v + 1$ and substituting K_c with K'_c. Then it generates a partially signed certificate $[Ct_{K'_c}]_{k_{Ds}}$ and sends it to a.

Determining the parameters (including the fanout F, see Section 3.4.1) to invoke Pan is a key issue to ensuring a successful completion of the threshold signing procedure, because an agent should collect at least $t + 1$ partially signed certificates to obtain a valid certificate signed by the private key of DICTATE. According to the analysis provided in Section 3.5.2, the agent can decide on values of certain parameters by which the Pan server query protocol reaches more than $t + 1$ servers. The agent might need to invoke this procedure more than once before finishing.

The other issue is how to transfer the responses to a query (with respect to Pan) back to an agent. This problem is common to all internal protocols. In [LEH04], we suggest a direct call to the unicast routing primitive. The strengths of this approach are: (i) the complexity of the protocols is low and (ii) a compromised server cannot block a transmission. Unfortunately, several parallel transmissions to a common destination may congest the network. *Reverse path forwarding*

---

[11] This protocol will be used for several purposes in DICTATE. Such flexibility is granted by a callback procedure [LEH04] included in Pan, which can be defined to retrieve any information (e.g., $[Ct_{K'_c}]_{k_{Ds}}$) from the upper layer (DICTATE in this case).

is an alternative. As illustrated in Fig. 3.3, a reply follows the tree created in the dissemination phase back to the agent and there are packet-level aggregations at internal vertices of the tree. This approach improves the protocol efficiency if the percentage of compromised servers is low, but suffers from a high protocol complexity. Even worse, a compromised server s launching a DoS attack by blocking the transmission in the reply phase can greatly reduce the efficiency of the protocol, especially when s is close to the agent. We suggest that an agent switches between unicast routing and reverse path forwarding based on the estimation of the percentage of compromised servers.

Certificate Replication This protocol is built upon both the Pan server update protocol and the Pan server query protocol. The parameters for invoking Pan are set to meet the freshness requirement $\mathcal{F}_d$ of the client. How to estimate the size of the resulting quorum Θ_U under certain parameters is described in Section 3.5.2. Each server, upon receiving the message $\langle aid, \mathcal{R}_c\rangle_{k_a}$, verifies its validity and then stores the certificate $[Ct_{K'_c}]_{k_D}$. If the client requests the highest freshness degree, the Pan server query protocol is invoked to require a reply $\langle sid, [Ct_{K'_c}]_{k_D}\rangle_{k_s}$ from each server $s \in \Theta_U$, and the resulting reply quorum $\Theta_{\bar{U}}$ should respect (3.2); otherwise the Pan server update protocol is invoked. In the latter case, replies are only required from those servers receiving a replication request directly from the agent, which is already enough to prove the agent's compliance with the protocol; the resulting freshness degree is estimated by (3.1).

Query Forwarding The agent's task is trivial in this protocol. It simply forwards a valid query request to other servers via the Pan server update protocol. Each server receiving the query request replies directly to the client with its own copy of the certificate, if the query is valid (proved by checking the signature of the agent). The client waits for a certain period of time until it collects the replies from a quorum $\Theta_{\bar{Q}}$ of servers. The minimum size expected for $\Theta_{\bar{Q}}$ is $|\Theta_{\bar{Q}}|_{min}$, which is set by DICTATE (based on the network size and the query rate [LEH04]) and known to all nodes; it cannot be modified by a client (in contrast to the situation of $|\Theta_{\bar{U}}|$ whose value can be modified by a client through $\mathcal{F}_d$). Within this collection, the certificate with the highest version number is chosen as the response to this query.

3.5 Security Analysis

In this section, we verify the security of DICTATE against the properties specified in Section 3.3.1 by considering malicious/compromised clients and compromised servers separately.

3.5.1 Protection from Malicious/Compromised Clients

Malicious clients have no way to impersonate other nodes, because a server can verify the identity of a client by checking the ownership of the private key corresponding to a certified public key. Hence, a malicious client cannot impair the safety property.[12] If the certificate update is performed frequently, an attacker is given virtually no chance to compromise a private key before the key expires. All exchanged messages are authenticated (signed by their senders) but do not require confidentiality, so attacks such as eavesdropping and message corruption cannot degrade the security of DICTATE. However, DICTATE can suffer from DoS attacks launched by clients replaying certain requests to cram the service queue. Defending against such attacks is difficult, because an agent should anyway verify a request before knowing its legitimacy. Actually, DICTATE already decreases the risk of such a DoS attack by involving only one agent for each request. Another kind of DoS attack is routing disruption [HPJ02]. Fortunately, the liveness property of DICTATE can be guaranteed, provided that, between a sender and a receiver, there exists at least one routing path that contains no malicious nodes.

Compromised clients, when behaving as described above, cannot compromise DICTATE. However, owning the identities and certificates of once legitimate nodes, they pose potential threats to other nodes that still believe in their legitimacy. DICTATE does not directly thwart these threats; it rather relies on inputs from misbehavior detection algorithms (e.g., reputation systems [BB02, CD03]) to identify and thus evict these nodes (we refer to Section 3.4.3 for details).

3.5.2 Defense against Compromised Servers

The attacks performed by a given compromised server vary with the role of that server.

Compromised Agents

A compromised agent may decline to serve a request for a client. A client can detect such an attack by setting a timer for each request and can change the agent should the timer expire. If an agent fulfilled the task of certificate update but tried to cheat its client without following up the replication, it would fail to provide evidences $\langle sid_1, [Ct_{K'_c}]_{k_D}\rangle_{k_{s1}}, \langle sid_2, [Ct_{K'_c}]_{k_D}\rangle_{k_{s2}}, \cdots$. In both cases, the liveness property of DICTATE is ensured, provided that the client eventually finds some correct server as its agent. A compromised agent may also provide fictitious requests (or replies) to other servers (or its clients). Such behavior does not compromise the safety property of DICTATE, because a message receiver can always verify the validity of the message signed by its original sender. Although fictitious requests disseminated by a compromised agent do reduce the service capacity of DICTATE, this kind of DoS attack can be easily detected, so that correct servers may convoke an emergency checktime in order to purge the compromised server.

---

[12] A bogus certificate update could be disseminated within DICTATE if this update request were admitted by a compromised agent. Since this attack has the same effect as an agent propagating a fictitious update or replica, we refer to Section 3.5.2 for a detailed analysis.

Compromised Common Servers

We call a *common server* a server that does not have the role of the agent for a given request. A compromised common server can 1) issue partially signed certificates of any bitstring, 2) behave as a malicious node, and 3) deviate from the protocol requirements by omitting the verification of all replicas (which leads to a later reply to a query with an obsolete certificate) or by simply not storing or forwarding a replica. DICTATE defends itself against 1) by using a $(t+1, n)$ threshold cryptosystem, such that the liveness and safety properties are guaranteed as long as the total number of compromised servers is no more than t. For the second type of attacks, we refer to Section 3.5.1 for related discussions. The third type consists in attacks of our particular interests; we consider them to be DoS attacks that we term *inaction* DoS (or iDoS) and analyze them in detail.

Since DICTATE is built upon a probabilistic quorum system, the degree $\mathcal{F}_d$ that iDoS attacks will not compromise the freshness property of DICTATE is the probability that $\exists s : s \in \mathbf{C} \cap \Theta_U \cap \Theta_{\bar{Q}}$ (i.e., there exists at least one correct server that receives both a replica and a later query to the replica), as expressed in (3.1). In this sense, DICTATE can be considered as a special instance of (b, ε) dissemination quorum systems [MRW01] ($b = t$ and $\varepsilon = 1 - \mathcal{F}_d$ in our case), innovating on the system design with an asymmetric quorum construction and a randomized quorum size. To compute $\mathcal{F}_d$, we rewrite (3.1) by using the concept of combination and also by taking expectations over probability distributions (of corresponding random variables) as follows:

$$
\begin{aligned}
\mathcal{F}_d &\geq \sum_{r=0}^{\bar{r}} \sum_{i=0}^{n} \left(1 - \frac{A}{B}\right) \nu_r(i) p_r \qquad (3.3) \\
A &= \binom{n - |\Theta_U^r|}{|\Theta_{\bar{Q}}|_{min}} \quad \begin{array}{l}\text{is the number of events} \\ \text{where } \Theta_U \cap \Theta_{\bar{Q}} = \emptyset\end{array} \\
B &= \binom{n}{|\Theta_{\bar{Q}}|_{min}} \quad \begin{array}{l}\text{is the total number of events} \\ \text{of taking } |\Theta_{\bar{Q}}|_{min} \text{ out of } n\end{array}
\end{aligned}
$$

where ν_r is the probability distribution of $|\Theta_U^r|$ (i.e., $\nu_r(i) = \Pr\{|\Theta_U^r| = i\}$), and p_r is the probability that a query occurs $r+1$ rounds[13] later than when the considered certificate was replicated. Note that both $|\Theta_U^r|$ and ν_r are functions of r. In order to simplify the case, we compute only the lower bound of $\mathcal{F}_d$ by plugging in the minimum value of $|\Theta_{\bar{Q}}|$ set by DICTATE. Now we show how to calculate ν_r with a recurrence relation defined by a Markov chain. Note that the Markov chain is 2-dimensional because the number of servers that will receive a message in the next round depends on the **increase** in the number of servers that have received the message in the current round (see Section 3.4.1 for details). Let $S_r = |\Theta_U^r|$ and $\mathbf{S}_r$ be a vector $[S_r, S_{r-1}]_{r\geq 0}^T$ for brevity; the distribution of S_r is estimated with the following recurrence relation, with the initial condition

---

[13] We assume that nodes gossip in synchronous rounds for the analysis, but our protocols do not rely on synchronization in practice.

$\Pr\{\mathbf{S}_0 = [1, 0]^T\} = 1$:

$$\begin{aligned}
\Pr\{\mathbf{S}_{r+1} = \left[\begin{smallmatrix} j \\ i \end{smallmatrix}\right]\} &= \sum_{i_1=0}^{i} \binom{n-i}{j-i} (1 - q^{i-i_1})^{j-i} q^{(i-i_1)(n-j)} \Pr\{\mathbf{S}_r = \left[\begin{smallmatrix} i \\ i_1 \end{smallmatrix}\right]\} \\
\nu_r(i) &= \sum_{i_1=0}^{i} \Pr\{\mathbf{S}_r = \left[\begin{smallmatrix} i \\ i_1 \end{smallmatrix}\right]\} \qquad (3.4)
\end{aligned}$$

where

$$q = 1 - \left(\frac{F}{n-1}\right)\left(\frac{n-t}{n-1}\right)\mathbf{E}_H[(1-p_f)^H] \qquad (3.5)$$

is the probability that a certain server does not receive the propagated message in a given gossip round (recall that F is the fanout, see Section 3.4.1). This expression takes into account the probability that either (i) the server is not chosen as a gossip destination, (ii) the server is compromised (we assume the worst case iDoS attack where a compromised server drops any message it receives), or (iii) the message fails to reach its destination: as p_f is an identical and independent probability of failure for each node along a routing path in a certain network environment and H is a random variable representing the length of an arbitrarily chosen routing path, the expectation (or $\mathbf{E}_H$) of $(1-p_f)^H$ characterizes the end-to-end failure probability. We refer to Section 2.5.2 for detailed discussions.

DICTATE is not limited to the probabilistic semantics of the freshness. A deterministic assurance can be provided by meeting the requirement of (3.2) such that $|\Theta_{\bar{U}} \cap \Theta_{\bar{Q}}| \geq t+1$ and thus $\Theta_{\bar{U}} \cap \Theta_{\bar{Q}}$ includes at least one correct server. In this case, DICTATE becomes an instance of strict dissemination quorum systems [MR98]. Of course, such a protocol setting incurs much higher overhead than that of a probabilistic protocol. As we mentioned in Section 3.4.3, a lower freshness degree does not mean a lower level of security. Since DICTATE provides an on-line revocation service, the goal of a client that queries the status of a given certificate is to check the validity of the corresponding public key obtained from the other client. Although there is a rare chance of obtaining an incorrect status due to the probabilistic nature of DICTATE, it does not compromise the security of the revocation service but only reduces its efficiency (because an incorrect status prevents the client from using a correct public key before the client completes a successful query).

3.6 Simulations

In this section, we verify the performance of DICTATE with respect to the freshness degree $\mathcal{F}_d$, under the iDoS attacks launched by compromised servers.

3.6.1 Parameters and Assumptions

We use *ns-2* [FV02] with the Monarch Project wireless and mobile extensions. This simulator provides both implementations of ad hoc routing protocols (we use DSR as an example) and wireless MAC, based on the Lucent WaveLAN IEEE 802.11 product, with a 2Mbps transmission rate and a nominal range of 250m.

We simulate an ad hoc network with 100 nodes in a square area of 1km^2. The movement pattern is defined by the "random waypoint" model [JM96], and we update it by setting a positive minimum speed (as suggested by [YLN03]). We pair the mobility parameters, such that each node has a maximum speed of 2m/s, 5m/s, 10m/s, and 20m/s, and a corresponding average pause time of 10s, 20s, 40s, and 80s, respectively.

In our simulations, the dCA contains 19 servers, which means that the system can sustain up to $6 = (19 - 1)/3$ compromised servers. The dCA servers are assumed to be predefined. The external protocols and the internal certificate update protocol of our DICTATE are omitted to simplify the interpretation of the results. Certificate replications and queries are assumed to be independent Poisson arrival processes with intensities λ_R and λ_Q, respectively. They are emulated by Poisson traffic sources attached to each server, generating packets of 512 bytes[14]. The overall access rate $\lambda_O = \lambda_R + \lambda_Q$ is set at 2s^{-1}, and we also assume that $\lambda_O = 8\lambda_R$. This allows each client to update its certificate about every 5 minutes and to query certificates of other clients every 45 seconds, which is more than enough to thwart key compromises and impersonations. The duration of a gossip round is set to 200ms, and the value of $|\Theta_{\bar{Q}}|_{min}$ is set to 3 for all simulations.[15]

We assume that all replication and query requests are targeted at the same certificate. The assumption might seem to be exaggeratedly pessimistic, because the chance that a queried certificate has just been updated is negligible (recall that there are in total $100 - 19 = 81$ certificates to be updated and queried). However, the goal is to force a query to always return the result of the latest replication and thus to illustrate the lower bound of the freshness degree that is provided by DICTATE. We term the resulting freshness degree *pessimistic* $\mathcal{F}_d$. In addition, we also evaluate *optimistic* $\mathcal{F}_d$: a query is considered to be successful even if it only returns the result of the penultimate replication.

We first investigate the impact of t (the number of compromised servers) on the performance of DICTATE, then we show how the analytical results in Section 3.5.2 can be used to tune the freshness degree to a required value. DICTATE is operated over 400 seconds of simulated time. The first 30 seconds of the simulation are used for system initialization. Then each traffic source continues generating traffic according to the predefined intensity until the end. Each simulation was carried out 10 times with different scenario files created by *ns-2*.

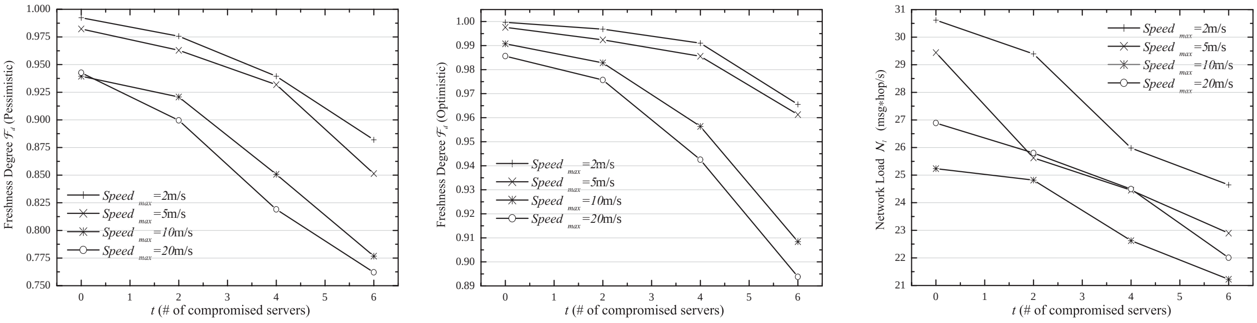
---

[14] This is approximately the size of $\langle aid, \mathcal{R}_c \rangle_{k_a}$, the largest message involved in the simulated part of DICTATE, if the RSA key is 1024 bits and the identity-based ECC key is 163 bits.

[15] Although DICTATE may not always guarantee $\mathcal{F}_d = 1$ with $|\Theta_{\bar{Q}}|_{min} = 3$ when $t > 1$, the simulation results show that $\mathcal{F}_d$ can be made as close to 1 as possible even when $t > 1$.

3.6.2 Impact of t on DICTATE Performance

We first set the fanout used for certificate replication at $F = 2$ and vary the number of compromised servers t, as well as node mobility parameters. Note that we always designate the mobility pair by the maximum node speed. As shown in Fig. 3.6 (a) and (b), the freshness degree $\mathcal{F}_d$ degrades modestly even in the worst case (i.e., $t = 6$) for low mobility scenarios (i.e., $Speed_{max} = 2$ or 5m/s). In these cases, we can claim that $\mathcal{F}_d \geq 0.95$ in practice, considering that $t = 6$ is a rare case and the pessimistic $\mathcal{F}_d$ shows the lower bound of the DICTATE performance. However, the effect of t is significant for high mobility scenarios (i.e., $Speed_{max} = 10$ or 20m/s). In both cases where $t = 4$ and 6, the pessimistic $\mathcal{F}_d$ drops to values around 0.80. The system has to adjust itself if, in such situations, a freshness degree higher than 0.90 is requested by a client. We show the tunability of DICTATE in Section 3.6.3.



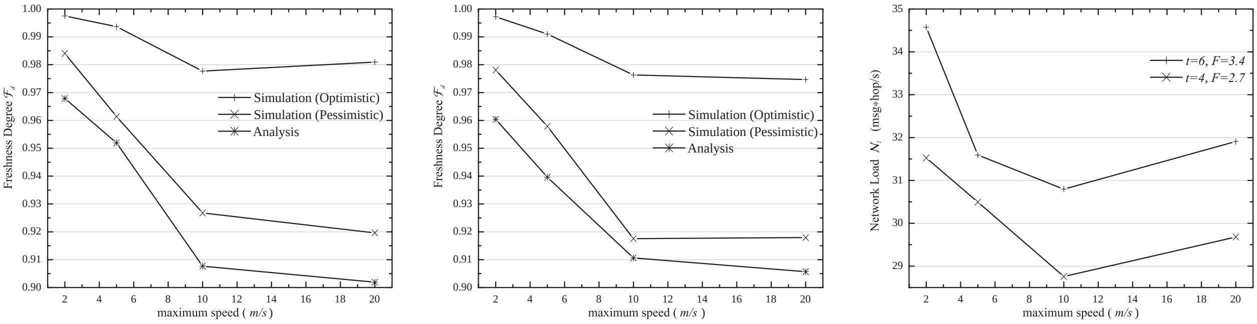
(a) Pessimistic freshness degree (b) Optimistic freshness degree (c) Network Load

Figure 3.6: Freshness degree $\mathcal{F}_d$ and network load vs. the number of compromised servers t, under four mobility scenarios. The fanout for certificate replication is set at $F = 2$.

The network load incurred by DICTATE is shown in Fig. 3.6 (c). This load is reasonably low since each request costs less than 8 unicastings on average, given that there are 2 requests per second and the expected length of a routing path is about 2 hops (for our simulation scenarios). If a traditional access protocol were used for these requests, which would mean accessing DICTATE servers by multiple unicastings, the resulting load would be much higher. It can also be observed that the larger the number of compromised servers, the lower the load (because a compromised server drops any message instead of relaying it).

3.6.3 Tuning the Freshness Degree

As we have already mentioned, the parameter settings used in Section 3.6.2 are not appropriate to cope with a large proportion of compromised servers in high mobility scenarios. So if an agent is aware of a significant increase in compromised servers, it will adjust the parameter settings for the internal protocols. The simulations described in this subsection illustrate how such adjustments are performed based on the analysis in Section 3.5.2, given a required freshness degree of 0.90.



(a) Freshness degree with *t=4*, *F=2.7* (b) Freshness degree with *t=6*, *F=3.4* (c) Network load

Figure 3.7: Freshness degree $\mathcal{F}_d$ and network load vs. the node mobility. The fanout for certificate replication is adjusted according to analytical results, in order to cope with a large proportion of compromised servers and high node speed.

A fanout F that ensures $\mathcal{F}_d \geq 0.90$ can be deduced from expressions (3.3) to (3.5), given a particular value of t and certain network conditions. The analytical results for $t = 4$ and 6 are shown by the bottom curves in Fig. 3.7 (a) and (b), respectively. The simulation results also provided in these two figures further prove that the fanouts resulting from the analysis ($F = 2.7$ for $t = 4$ and $F = 3.4$ for $t = 6$)[16] indeed lead to satisfactory freshness degrees, i.e., the experimental values of $\mathcal{F}_d$ are always higher than the predicted values. Of course, the system actually trades its overhead for the freshness degree in these cases. The comparison between Fig. 3.6 and Fig. 3.7 shows that an increase of network load up to 50% is traded for an improvement of about 300% of the freshness degree (i.e., $1-\mathcal{F}_d$ is divided by 3) in the extreme situation: $t = 6$ and $\mathit{Speed}_{max} = 20$m/s.

3.7 Conclusion

In this chapter, we have focused on the design of a certification authority in ad hoc networks. We take a joint authority approach that combines an off-line identification authority and an on-line distributed revocation authority. We have then proposed DICTATE, based on our previous work on reliable group communication systems, to control the on-line authority. The originality of DICTATE includes 1) flexible certificate management protocols with tunable freshness to trade overhead for robustness, and 2) provable robustness against various attacks, especially Byzantine failures of the DICTATE servers.

---

[16] A real number $x.y$ for F means that each server, when propagating a message, takes $F = x$ with probability $1 - y/10$ and $F = x + 1$ with probability $y/10$.

Our proposed specification of distributed CA with probabilistic freshness takes the peculiarities of ad hoc networks into account. As a consequence, the freshness property can be sacrificed to some extent, in the case that the required freshness degree is low and the network resources are scarce. In order to meet the specification, DICTATE is implemented in a “closed” server group backed by an identity-based cryptosystem, so that the Sybil attack is thwarted and messages from individual servers are universally verifiable. Also, the authority employs threshold cryptography to distribute trust in order to tolerate a certain number of compromised servers. Finally, DICTATE relies on a probabilistic group communication system, PILOT, to propagate certificate updates, replications, and queries to its server group, which guarantees a certain probability, rather high in practice, for a query to obtain the latest status of a certificate. Despite the seeming complexity, DICTATE executes automatically without the need of any human involvement.

We have identified potential attacks against DICTATE and evaluated the robustness of DICTATE through detailed security analysis. We notice that one of these attacks, called inaction DoS (iDoS) attack, is a serious threat to DICTATE; we have thus verified the system performance under such attacks by simulations. The results show that (i) iDoS attacks can only have a marginal effect on the performance of DICTATE in low node speed scenarios and (ii) increasing node speed weakens the tolerance of DICTATE to such attacks. In the latter case, the system can be tuned on-line to trade its overhead for a higher degree of freshness. We have improved the analytical model proposed for PILOT to predict the freshness degree of DICTATE. The validity of predictions is evaluated by simulations, in a way that the analytical results are used as the basis to adjust system parameters for tuning the DICTATE performance.

Part II

Turning Harm into Good

— Optimizing Network Performance with Controllable Mobility

Chapter 4

Joint Routing and Sink Mobility for Network Lifetime Elongation

4.1 Introduction

Apart from their great advantages, *wireless sensor network*s (WSNs) are subject to many limitations, among which the energy constraint has become an increasing concern [ASSC02]. In WSNs, the size of a node should be sufficiently small to avoid altering phenomena of interest, and nodes, in many cases, should operate for a long period without human attendance. These requirements imply a capacity-limited and (possibly) non-renewable power source for each node. As a result, the longevity of WSNs under energy reserve limitations is a major problem that should be addressed before making use of such networks. Recently, many communication protocols for energy conservation in WSNs have been proposed. These include, among others, energy conserving routing (e.g., [CT00, KKLT03, SL04, GZ04]), topology control (e.g., [LHB$^+$01, PHC$^+$03, LH04, LWW$^+$04]) and clustering (e.g., [HCB02, KK03, BGLA03, YF04]). Although all these protocols achieve their optimization goals under certain conditions, they always focus on the **sensor node**s.[1] We show in this chapter that further improvements on the lifetime of WSNs can be achieved if we shift our focus to the behavior of **sink**s.[2]

We observe that, as data traffic must be concentrated towards a small number (typically one) of sinks, the nodes around a sink have to forward data for other nodes whose number can be very large; this problem always exists, regardless of what energy conserving protocol is used for data transmission. In other words, applying energy conserving protocols does not directly lead to load balancing within the whole network. Using a simple analytical model, we will show how unevenly the load is distributed within a network. As a result, those bottleneck nodes around sinks deplete their batteries much faster than other nodes and, therefore, their lifetime upper bounds the lifetime

---

[1] In this and the next chapter, the words *sensor*, *sensor node* and *node* are used interchangeably.

[2] These are the devices that collect data from WSNs; sometimes they are also termed **base station**s.

of the whole network.

Intuitively speaking, the load of sensor nodes can be more balanced if a sink changes its position from time to time, because such a sink can distribute over time the role of bottleneck nodes and thus even out the load. Although a sink is usually assumed to be static, we argue that **mobile sink**s are practical for many realistic applications of WSNs. For example, suppose that a WSN is equipped with batteries that cannot be replaced, e.g., because the sensor nodes are not accessible, or because changing batteries would be hazardous or costly. This may be the case in sensors in smart buildings, where batteries might be designed to last for decades, and in environmental or military sensing under hostile or dangerous conditions (e.g., avalanche monitoring). In this case, it may be desirable and comparatively simple to move a sink very infrequently (e.g., once a day or a week) by a human or by a robot [But03, LBK$^+$02]. For example, in the avalanche monitoring scenario, a sink may be deployed at the periphery[3] of the monitored area, and moved by helicopter once in a while. In the building scenario, the sink can be "virtually" moved: computers in different offices serve as the sink in shifts. In the military monitoring scenario, moving the sink may require some effort, but it can be acceptable if done infrequently.

In this chapter, taking mobile sinks into account, we investigate the problem of load-balanced data collection in WSNs. The idea is to make use of existing multi-hop routing protocols and to achieve further improvements in terms of network lifetime by exploiting the *sink mobility*. Since multi-hop routing is used, our solution does not significantly affect latency and is thus different from the *mobile relay* approach [SRJB03, CSA03]. We discuss the joint sink mobility and routing problem under two different models, namely a graph model and a continuum model. We show that, under the graph model, the problem is very hard in general, but that certain induced sub-problems that have a potential to direct protocol designs in practice are tractable. Further investigations also allow us to have efficient approximation algorithms, as well as duality theory that proves the superiority of moving sinks over keeping them static. Using our continuum models, we are able to quantify the benefits that can be obtained when using a mobile sink in an asymptotic sense. These analytical results suggest that, in addition to routing, sink mobility does help to optimize the network lifetime. Considering jointly the mobility and routing strategies, we propose the optimal sink mobility trace and an improved data collection protocols that further balance the load. For both models, we perform numerical simulations to validate the analytical model and to quantify the lifetime elongation compared with WSNs with a static sink. Finally, we discuss the implementation issues concerned with sink mobility.

The rest of this chapter is organized as follows: Section 4.2 surveys related work. Section 4.3 investigates our joint sink mobility and routing optimization problem under the graph model. Section 4.4 presents both analytical and simulation results under a continuum model; it also describes our optimal sink mobility and improved routing strategies for data collection. Section 4.5 discusses related practical issues. Finally, Section 4.6 concludes the chapter.

---

[3] As we will show later, the optimum trace (in terms of network lifetime) for a mobile sink is the network periphery.

4.2 Related Work

We first briefly discuss a few topics related to energy conservation in WSNs, which do not consider exploiting sink mobility. We then elaborate recent proposals that are closely related to the problem of improving network lifetime with mobile sinks. Finally, we also mention other ways of using mobility in both sensor and ad hoc networks.

4.2.1 Energy Conservation Protocols

The existing work concerned with lifetime (or energy conservation) issues is so vast that it would deserve several comprehensive surveys. Hence, we only discuss a few related topics here. Energy conserving routing (e.g., [CT00, KKLT03, SL04, GZ04]) aims at balancing the energy consumption among all nodes, instead of minimizing the absolute consumed power. Topology control (e.g., [LHB$^+$01, PHC$^+$03, LH04, LWW$^+$04]) and clustering[4] (e.g., [KK03, BGLA03, YF04]) are usually closely correlated [BGLA03]. They both manipulate the network topology by exploiting the nodes' ability to adjust their transmission power dynamically; the goal is to maintain network connectivity while reducing energy consumption and improving network capacity. Although these proposals are somewhat orthogonal to the idea we present in this chapter, they are all potentially complementary to our idea.

4.2.2 Moving Sinks to Improve Network Lifetime

Exploiting sink mobility to improve network lifetime usually takes two different methods, depending on the relationship between the *mobility time-scale* and the *delay time-scale*. The mobility time-scale refers to the time over which the movement of the mobile sink covers a significant portion of the entire network; the delay time-scale refers to the tolerable delay that sensor data is allowed to incur between its origin and the sink.

In the *fast mobility* regime, the mobility time-scale is of the order of the delay time-scale. The WSNs may then take advantage of *mobility capacity* [GT02], i.e., the ability to transport information in part by physically carrying it in mobile nodes, rather than transmitting it through wireless links. In this *mobile relay* approach [SRJB03, CSA03], the mobile sink "picks up" data from nodes and transports the data back with mechanical movements. By "picking up" we mean that the sink should move as close to a node as possible before asking the node to transmit its data. This approach trades data delivery latency for the reduction of energy consumption of nodes. While both [SRJB03] and [CSA03] leverage only on uncontrollable (although predictable for [CSA03]) mobility of the sink,

---

[4] LEACH [HCB02] represents a special type of clustering that is close to the sink mobility problem investigated in this chapter. LEACH applies a randomized rotation of cluster heads and requires a cluster head to directly transmit data out of a network. Although mobile sinks superficially do seem to be analogous to the rotated cluster heads, the assumption made by LEACH that every node has the ability to perform long-range transmission might not be very realistic. In addition, our analysis in this chapter aims at identify the optimal positions for mobile sinks, which is bound to achieve a longer lifetime than that can be obtained by a randomized rotation.

Kansal et al. [KSJ^+04][5] investigate the controllable mobility. Their proposal is a compromise between the mobile relay approach and ours: the sink still relays data with its movements, but a node that does not lie in the vicinity of the mobility trace transmits data through a multi-hop routing when the sink moves to the closest point to the node. A field study in this fast mobility regime is reported in [KSJ^+04], but it would also be interesting to have a theoretical analysis on this hybrid approach. In this chapter, we briefly investigate the tractability of the lifetime optimization problem using a controlled mobile relay.

In the *slow mobility* regime, the mobility time-scale is longer than the required delay time-scale. In this case, the network cannot benefit from mobility capacity, as the delay bounds of most data packets would be exceeded. Therefore, data packets have to be carried from their origin to the sink through multihop transmissions. However, it has recently been observed [GDPV03] that sink mobility can still offer benefits in terms of network lifetime. This is because a load-balancing effect arises due to sink mobility, which averages the role of relay over many nodes. Therefore, in the slow mobility scenario, the network lifetime can be longer than in an equivalent static scenario, with no need to sacrifice latency. Gandham et al. [GDPV03] present a *integer linear programming* (ILP) formulation that takes sink mobility into account. Unfortunately, their formulation cannot lead to a global lifetime maximization, because the time spans of all rounds are artificially set to be equal and only local optimums are obtained for each pause time. In addition, the hardness of the problem is not evaluated in [GDPV03]. In several later contributions [WBMP05, PG05], the ILP formulations are upgraded by taking into account variable pause times, but the problem of non-global optimality persists. In this chapter, we propose a general framework for analyzing the problem of joint sink mobility and routing for lifetime optimization, which encompasses the ILP formulations in [GDPV03, WBMP05, PG05].

In their very recent work [WSC05], Wang et al. propose a *mobile node*[6] approach to prolong network lifetime. The basic idea there is that a few powerful mobile nodes can be deployed to replace different (heavily loaded) static nodes. In their approach, a mobile node inherits the responsibilities of a static node with which it is co-located, such that the static node can shut down for energy saving. Whereas our framework will not cover this problem due to the fundamental difference between moving nodes and sinks, we note that this approach can achieve the same order of lifetime as the mobile sink approach only if a sufficient number of mobile nodes ($O(\sqrt{n})$ for an n-nodes network [WSC05]) are deployed.

4.2.3 Mobility Helps in Other Aspects

It is also important to note that the mobility of network nodes can help to improve the performance of wireless ad hoc and sensor networks in other aspects (rather than network lifetime). First,

---

[5] This work is the representative of a set of subsequent proposals, including its journal version (available at http://www.ee.ucla.edu/%7kansal/publications.html) and [JSS05].

[6] The authors also term their approach "mobile relay". In order to be consistent with the terminology used in [LH05] (where "mobile relay" was given to the first approach), we give another name to this approach.

node mobility facilitates sensor deployment [WCP04]; movement-assisted protocols can re-shape a WSN to improve the network coverage. Secondly, moving sensor nodes reduces sensing uncertainty [BRY$^+$04]; for achieving the same level of sensing fidelity, it can be much cheaper to deploy a few mobility capable nodes instead of a large number of static nodes. Thirdly, moving sinks can prevent the data buffer in sensor nodes from overflow [SRS04]; the basic idea is that nodes that sample at higher rates than others need to be visited more frequently. Fourthly, mobility is a potential network control primitive to improve communication performance in ad hoc networks [GLM$^+$04] or in *delay-tolerant network*s (DTNs) [ZAZ05]. Last but not least, using mobile beacons can facilitate the positioning of network nodes in their deployment areas [PBDT05, LSH06].

4.3 Optimization under a Graph Model

In this section, we investigate the lifetime optimization problem in WSNs under a graph model, assuming multiple mobile sinks. We show that the problem, in general, is very hard, but certain induced sub-problems that still have a practical significance are tractable. Moreover, the algorithm that solves the sub-problem that involves only a single sink can be generalized to approximate the general problem. Finally, we illustrate the benefit of using mobile sinks by applying our algorithm to a set of typical topological graphs.

4.3.1 System Model

We model a WSN as a digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ represents $|\mathcal{V}| = n$ sensor nodes. There is a cost assignment $c\colon \mathcal{V} \times \mathcal{V} \to \mathbb{R}^+$, such that $\exists (i,j) \in \mathcal{E}$ if and only if the transmission energy (defined later) of $i \in \mathcal{V}$ is no less than $c(i,j)$. Apart from the sensor nodes, there is a set $\mathcal{S}$ ($|\mathcal{S}| = m < n$) of sinks that harvest data from the WSN. The following properties are specified for the network:

[a]. Sensor nodes are stationary except the sinks, which change their positions from time to time with a negligible travelling time between two positions.

[b]. We only consider the case where the location of each sink coincides with the location of one of the nodes, such that the network topology (thus the graph $\mathcal{G}$) does not change with different locations of the sinks.

[c]. Each sink inherits the data collection function of the co-located node. It behaves like a node for receiving data, but it has long-range (wireless) communication facilities to transmit data out of the considered WSN.

[d]. The data traffic flows from each node $i \in \mathcal{V}$ (i.e., all nodes are *source*s) to one of the *sink*s $s \in \mathcal{S}$ (through multi-hop relaying if no direct connection exists between i and s exists), and the control traffic involved (e.g., in a routing protocol) is not considered since it has the same effect to all nodes.

[e]. Data transmission and reception are the dominating factors for the energy consumption of a node.

[f]. The transceiver of a node is not duplex, so the (wireless) channel capacity of a node has to be divided for both transmission and reception.

In addition, there are attributes associated with a node $i \in \mathcal{V}$:

[h]. a value E_i (Joules) representing the initial energy reserve of the node,

[i]. two values e_i^t and e^r representing the energies for the node to transmit and receive a unit of data (e.g., Joules/byte),

[j]. a quantity R_i that upper bounds the node's transmission rate (e.g., bytes/second), and

[k]. a rate λ_i of the information generation.

As shown in [SHC$^+$04], most sensor radios have a constant e^r regardless of the transmission power of a sender. Fig 4.1 shows the graph representation of a WSN.

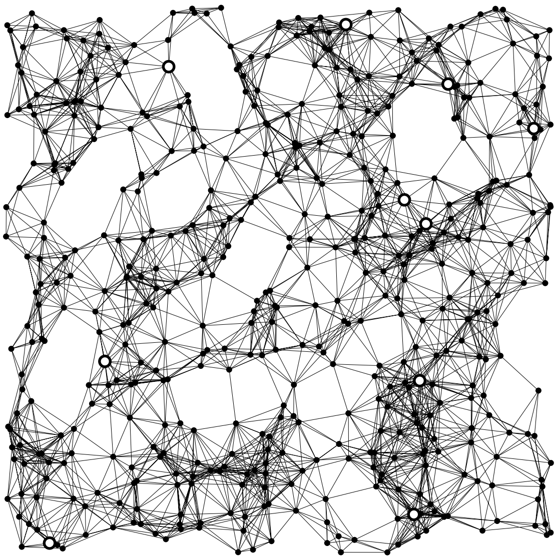


Figure 4.1: Network graph model. The WSN has 500 nodes (black points) and 10 sinks (white points). Nodes are uniformly distributed within a square area. Here we take a cost assignment based on the Euclidean distance and a uniform transmission energy assignment.

We have several remarks on our model. First, the assumption of a negligible travelling time for a sink comes from the fact that the pause time can be long enough to amortize the routing overhead introduced by the sink mobility. Second, one might expect that relaxing the location limitation of sinks could further improve the lifetime. However, as we show in Section 4.3.4, the degree of freedom introduced by sink mobility is confined by the network size; it will not increase unboundedly by adding more potential sink locations. Finally, we note that one significant difference between our model and those in [CT00, SL04] is that, in our model, the energy consumption of data transmission e_i^t is associated with a vertex (a node) instead of an edge (a link). We consider this model to be more realistic, because, although nodes have the flexibility to tune their own transmission power, it is not cost-effective to dynamically tune the power for destinations at different distances. In addition, tuning transmission power according to transmission distances is not always feasible either since a node might not know the distances. Therefore, a reasonable scenario, in our opinion, is that each node sets up a transmission power according to certain topology control mechanisms [LHB⁺01, LH04, LWW⁺04] at the network initialization phase and this power is fixed until some topology changes happen.

4.3.2 Problem Formulation

We begin with the definition of network lifetime for WSNs, and then formulate the problem of *maximizing network lifetime* (MNL) as a convex optimization.

Network Lifetime

The *network lifetime* can be defined in various ways; these definitions focus on either individual [CT00] or collective [BS02] behaviors of nodes. Because the individuality has an implication on the collectiveness (e.g., the death of a node is soon followed by the death of all nodes one-hop away [GDPV03]), we define the network lifetime as the time period for the first node to run out of its energy reserve [CT00].

Convex Optimization

We denote the lifetime by T and use t_k to indicate the time span for the kth *epoch*: a new epoch begins when some sinks change their locations. We also define q_{ij}^k as the total information flow from node i to node j during t_k. Given the assumption [i], the total energy consumed by node i during t_k is given by:

$$\sum_j q_{ij}^k e_i^t + e^r \sum_l q_{li}^k$$

where the sum is over all adjacent vertices of i (the adjacency is implied by the cost assignment c and the transmission energies of node i and its neighbors). In order to indicate the location of a sink during t_k, we use a binary variable δ_{is}^k to represent the relation between the location of a sink $s \in \mathcal{S}$ and that of node $i \in \mathcal{V}$, such that $\delta_{is}^k = 1$ if the location of sink s coincides with the one of the node i and $\delta_{is}^k = 0$ otherwise. We also associate with node i an outgoing flow F_i^k for epoch t_k; it becomes positive only if $\exists s \in \mathcal{S}\colon \delta_{is}^k = 1$. Hence the convex optimization for MNL is as follows:

$$
\begin{aligned}
&& \text{Maximize} \quad T &&&& (4.1)\\
s.t. \quad & \sum_j q_{ij}^k - \sum_l q_{li}^k - \lambda_i t_k + F_i^k &= 0 && \forall i,k && (4.2)\\
& F_i^k - \sum_j \lambda_j \cdot t_k \sum_s \delta_{is}^k &\le 0 && \forall i,k && (4.3)\\
& \sum_s \delta_{is}^k \le 1, \quad \sum_s \sum_i \delta_{is}^k - m &= 0 && \forall k && (4.4)\\
& \sum_j q_{ij}^k + \sum_l q_{li}^k - R_i t_k &\le 0 && \forall i,k && (4.5)\\
& \sum_k \left(\sum_j q_{ij}^k e_i^t + e^r \sum_l q_{li}^k \right) - E_i &\le 0 && \forall i && (4.6)\\
& \sum_k t_k - T &= 0 &&&& (4.7)\\
& q_{ij}^k,\ t_k,\ F_i^k \ge 0 \quad & \delta_{is}^k \in \{0,1\} && \forall i,j,k,s && (4.8)
\end{aligned}
$$

where $\sum_i$ means a sum over all possible i (this **notation** is used throughout this chapter). We explain each constraint in the following:

- Flow Conservation (4.2–4.4): The outgoing flow should exceed the incoming flow by an amount of $\lambda_i t_k$ (see assumption [k]) if $\sum_s \delta_{is}^k = 0$; otherwise the flow F_i^k should be taken into account. Note that the information flows out of the network depends on the sink locations (4.3), no sink co-locates with another sink, and the number of sinks is exactly m (4.4).
- Rate Constraint (4.5): The average rate of data transmission and reception by node i should not exceed the channel capacity, due to our assumption [f].
- Energy Constraint (4.6): The energy spent by node i to transmit and to receive data during the whole network lifetime is upper bounded by the initial energy reserve E_i, according our assumptions [h] and [i].
- Time Constraint (4.7): The sum of all epoches t_k is bounded by the lifetime T.

In this chapter, we ask the question of what is the maximum lifetime, for which constraint (4.5) is inactive if the rate set $\{\lambda_i\}$ is chosen properly. In addition, we could also ask what is the maximum amount of data that can be collected, considering $\lambda_i, i = 1, 2, \cdots, n$ as variables. In such a case, the objective function becomes $T \sum_i \lambda_i$ and (4.5) limits the potential choices of λ_i.

4.3.3 Hardness of the Problem

The potential number of sink layouts for the MNL problem is $\binom{n}{m}$, which, by Stirling's approximation, is exponential in n for an arbitrary m. Given the exponential (in n) number of columns in the programming, it is not difficult to believe that the MNL problem is "very hard". In order to formally evaluate its hardness, we consider the following decision problem that is derived as a restricted case for the original MNL, which we term MSP (standing for Mobile Sink Positioning):

> INSTANCE: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a cost assignment c, a set $\mathcal{S}$ of sinks with $|\mathcal{S}| < |\mathcal{V}|$, and for each $i \in \mathcal{V}$, a transmission energy e^t, a energy reserve E, a rate λ, and a positive real number t.
>
> QUESTION: Is there a *sink layout schedule* $\{(sl_k, t_k)\}$ (sl_k is a vector of $[\delta_{is}^k]$ where $\delta_{is}^k : \mathcal{V} \to \{0, 1\}$ and $\sum_i \sum_s \delta_{is}^k = |\mathcal{S}|$) such that the lifetime $T = \sum_k t_k$ is at least t?

Claim 1 *The MSP problem is NP-hard.*

Proof: The NP-hardness of the MSP problem can be shown by giving a polynomial-time reduction from DOMINATING SET [GJ79] to a special case of MSP where a schedule consists of only one element (sl, t). Given an undirected graph $\mathcal{G}' = (\mathcal{V}, \mathcal{E}')$ (an instance of DOMINATING SET), we construct a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, let $e^t = 1$, $E = 1$, and $\lambda = 1$, and define the cost assigments as:

$$c(e) = \begin{cases} 1 & e \in \mathcal{E}' \\ 2 & e \notin \mathcal{E}' \end{cases} \tag{4.9}$$

The reduction is straightforward and can be done in polynomial time. Now, $\mathcal{G}$ admits a lifetime $T = 1$ with a sink set $\mathcal{S}$ if and only if $\mathcal{G}'$ has a dominating set of size $|\mathcal{S}|$.[7] ■

However, MSP is not in NP, because a schedule $\{(sl_k, t_k)\}$ may include all possible layouts of an arbitrary number of sinks and hence it is not "easy" to verify its feasibility. Given such a schedule, one should verify if the time constraint (4.7) is met and if the flow λ injected at each node is admissible given the energy constraint (4.6) and the flow conservation (4.2) and (4.3). The verification of the latter conditions leads to the following linear program:

$$\text{Maximize} \quad \Theta \tag{4.10}$$

$$s.t. \quad \sum_j q_{ij}^k - \sum_l q_{li}^k - \lambda t_k \cdot \Theta + F_i^k = 0 \quad \forall i, k \tag{4.11}$$

$$F_i^k - n\lambda t_k {\textstyle\sum_s} \delta_{is}^k \cdot \Theta \leq 0 \quad \forall i, k \tag{4.12}$$

$$\sum_k \sum_j q_{ij}^k - E/e^t \leq 0 \quad \forall i \tag{4.13}$$

$$q_{ij}^k,\ F_i^k \geq 0 \quad \forall i, j, k \tag{4.14}$$

---

[7] A similar reduction has been shown by Bogdanov et al. [BMR04] for proving the hardness of the BSP (Base Station Positioning) problem. While their problem intends to find the maximum rate admitted by a WSN, we try to maximize the network lifetime for a given rate.

If $\Theta < 1$, the answer is no, because there is no flow that admits the total injection $n\lambda(\sum_k t_k)$ under the energy constraint; otherwise it is yes. We can re-formulate the above Arc-Flow form into the Path-Flow form and get its dual problem. The time complexity of solving the dual problem of this verification depends on the complexity of the related separation problem[8] [NW88]. The separation problem (which we are not going to elaborate here) requires the computation of sum of the minimum costs of shipping a unit flow from all nodes to all sink layouts (whose number could potentially be exponential in n). Therefore, we have the following corollary:

Corollary 1 *The MSP problem is NP-complete if the length of a schedule* $\{(sl_k, t_k)\}$ *is restricted to be polynomial in* n.

The above proof leaves several problems open. First, the proof is not constructive because it does not suggest a solution to the problem provided that there was a solution to the DOMINATING SET problem (a common weakness of applying *restriction* [GJ79] for proving NP-hardness). Second, it fails to expose the structure of the problem: for example, whether the complexity lies in the selection of sl_k or of t_k. Finally, it does not show if the problem is tractable with a single sink; the DOMINATING SET problem is tractable if the set contains only one element. Therefore, we provide a constructive proof in Appendix B, and we answer other questions in the next sections.

At this point, it is also worth noting the lifetime optimization problem related to the joint mobility and routing strategy proposed in [KSJ+04] is also intractable, because the corresponding decision problem, which we term SMRP (standing for Single Mobile Relay Positioning):

> INSTANCE: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a cost assignment c, a set $\mathcal{S}$ of **sink positions** with $|\mathcal{S}| < |\mathcal{V}|$, and for each $i \in \mathcal{V}$, a transmission energy e^t, a energy reserve E, a rate λ, a constraint that i sends data only to a certain $s \in \mathcal{S}$, and a positive real number t.
>
> QUESTION: Is there a sink layout schedule $\{(\delta_{is}, \tau)\}$, where $\delta_{is} : \mathcal{V} \to \{0, 1\}$, $\sum_i \delta_{is} = 1$ (only one sink at a given moment) and $\sum_i \sum_s \delta_{is} = |\mathcal{S}|$, such that the lifetime $T = \tau$ is at least t?

is also NP-Complete. A simple proof can be achieved by again applying the reduction from the DOMINATING SET.

4.3.4 Induced Sub-Problems

As we have shown in Section 4.3.3, the MSP problem, although a simplified version of the original MNL by considering homogenous (in terms of, e.g., energy reserve and consumption) nodes, is still hard. Therefore, we turn to investigate several relaxed versions of the MNL problem, as well as the related decision problem, by decomposing the two variable sl_k and t_k in the schedule. We note that a common approach of relaxing the integer constrains, i.e., (4.8), does not work for the MNL problem, because this relaxation renders every node a sink and thus leads to a trivial solution.

---

[8] We intensively use this concept in this chapter. A more detailed explanation is provided in Appendix B.

Uniform Epoch

If we fix the time period t_k of a epoch to τ, the question asked by the resulting MSP-UE (Mobile Sink Positioning with Uniform Epoch) problem becomes:

> QUESTION: Is there a sink layout schedule $\{(sl_k, \tau)\}$ of length l such that the lifetime $T = l\tau$ is at least t?

Claim 2 *The MSP-UE problem is NP-complete if l is polynomial in n.*

Proof: Similar to the proof for **Claim 1**, The NP-hardness of MSP-UE can be shown by the reduction from the DOMINATING SET to the MSP-UE with single element schedule. The proof that MSP-UE is in NP follows straightforwardly from **Corollary 1**. ■

The above result shows that relaxing time schedule does not significant simplify the MSP (and thus the MNL) problem. It hints that the complexity lies in the selection of sink layouts instead of the time schedule. In fact, a further relaxation was described in [GDPV03]; it consists in solving a sequence of MSP-UEs with $l = 1$. Such a problem is still NP-complete due to the hardness of MSP-UE problem itself.

Pre-defined Flow Schedule

Given all possible sink layouts sl_k: $k \leq \binom{n}{m}$, it is always possible to come up with a flow schedule (feasible but not necessarily optimal[9]) fs_k (where a fs_k is a vector of $[\tilde{q}^k_{ij}] : \mathcal{E} \to \mathbb{R}^+_0$) for each sl_k. Note that here a flow $\tilde{q}^k_{ij}$ on an edge (i, j) is defined as the information **rate** (as opposed to the total information q^k_{ij} defined in Section 4.3.2). Now the MNL problem becomes a search for a time schedule $\{t_k\}$ such that $T = \sum_k t_k$ is maximized; we formulate the problem as the following linear program:

$$\begin{aligned} & \text{Maximize} \quad \textstyle\sum_k t_k && && (4.15)\\ s.t. \quad & \sum_{fs_k : i \in fs_k} p^k_i t_k \quad \leq \quad E_i \quad \forall i && && (4.16)\\ & p^k_i,\ t_k \quad \geq \quad 0 \quad \forall i, j, k && && (4.17) \end{aligned}$$

where $p^k_i = \sum_j \tilde{q}^k_{ij} e^t_i + e^r \sum_l \tilde{q}^k_{li}$ is the **power** consumption of node i during the kth epoch.

Although the above linear program could involve an (potentially) exponential number of variables t_k, the vector $[t_k]$ for the optimal solution contains no more than n non-zero elements. Since the matrix $\mathbf{P} = [p^k_i]$ has a row rank of at most n, it cannot have more than n linearly independent columns. Consequently, each basic solution $[\hat{t}_k]$ contains at most n non-zero elements. The problem

---

[9] It becomes a GREEDY approximation if optimal flows are required. However, GREEDY does not necessarily perform better than an arbitrary (feasible) flow, because what matters is the complementariness among flows that leads to the overall optimality rather than the individual optimality.

involving only a single sink leads to n potential layouts, for which the solution becomes straightforward. If multiple sinks are involved, the problem might have a very high complexity due to the possibly exponential number of variables. We can apply an approximation algorithm in this case.

It is easy to see that the above linear program is a packing LP (a linear program in the form $\max\{c^T x | Ax \leq b, x \geq 0\}$); its $(1+\epsilon)$-approximation algorithm is proposed by Garg and Könemann [GK97]. The idea is as follows: we first consider the rows of A (i.e., $[p_i^1, \cdots, p_i^k, \cdots]$) as "edge"s and columns (i.e., $[p_1^k, \cdots, p_i^k, \cdots]^T$) as "path"s. Now, the problem becomes a *maximum multicommodity flow* problem where the objective is to find a multicommodity "flow" $\{t_k\}$ whose sum is maximized under the constraint of edge capacity E_i. The dual problem is to find an edge weight assignment $\{w_i\}$ such that $\sum_i E_i w_i$ is minimized under the constraint that the "length" of a path k ($l_k = \sum_i p_i^k w_i$) is larger than 1. Therefore, the dual problem is equivalent to finding $\{w_i\}$ such that $\frac{\sum_i E_i w_i}{\min_k l_k}$ is minimized. The problem can be solved (approximately) by an iterative procedure; it keeps updating the primal variables t_k and dual variables w_i in such a way that the increased flow goes through the shortest path and the edge that a flow goes through is "penalized" by a heavier weight. This procedure essentially tries to balance the flow (time in our case) on all edges (nodes in our case).

We note that the formulation of (4.15)–(4.17) has also a significance in practice. In reality, the flow in a network is shaped by a certain routing protocol instead of being defined by solutions of some optimization problem. Therefore, once we introduce a set of sink layouts schedule sl_k into a WSN, the network itself will figure out a flow schedule fs_k. This "pre-defined" schedule can then be taken as the input to the above program to obtain an optimal time schedule. We will apply this idea in the Section 4.4, and we will also show in Section 5.5 how this relaxation can be applied when designing a practical routing protocol for supporting mobile sinks.

4.3.5 Maximizing Network Lifetime for a Single Mobile Sink (MNL-SMS)

Although the MNL-SMS problem is again a sub-problem of MNL, we discuss it in a separate section due to its relevance. The problem is polynomially solvable in its original form, because it can be formulated as a linear program with only $O(n^2)$ constraints. In addition, we propose an approximation algorithm that solves the problem efficiently, which is then generalized to solve MNL along with other approximation algorithms. Most importantly, we are able to solve the following crucial decision problem:

> TO MOVE OR NOT TO MOVE (TMNTM): Is there a sink layout schedule $\{(sl_k, t_k)\}$ such that the lifetime $T = \sum_k t_k$ is longer than what is achieved by any fixed layout sl?

which was never theoretically answered for general MNL.

The MNL-SMS Problem

Based on (4.1)–(4.8), the Arc-Flow form of MNL-SMS problem can be formulated as the following linear program:

$$\begin{aligned}
&\text{Maximize} \quad \textstyle\sum_k t_k && && (4.18)\\
s.t. \quad &\sum_j q_{ij}^k - \sum_l q_{li}^k - \lambda_i t_k &=& \; 0 \qquad \forall k, i \neq k && (4.19)\\
&\sum_k \left(\sum_j q_{ij}^k e_i^t + e^r \sum_l q_{li}^k \right) - E_i &\leq& \; 0 \qquad \forall i && (4.20)\\
&q_{ij}^k, \; t_k &\geq& \; 0 \qquad \forall i, j, k && (4.21)
\end{aligned}$$

We simplify the formulation by assuming that the sink co-locates with node k during the kth epoch. We also deliberately drop the rate constraint (4.5) because we can anyway scale the set $\{\lambda_i\}$ to meet this constraint. It can be easily seen that the number of constraints is bounded by $|\mathcal{V}|^2 + |\mathcal{V}|$ (with (4.19) introducing the first term and (4.20) accounting for the second)and is thus polynomial in n. Therefore, this problem is polynomially solvable [Meg84]. However, directly solving the linear program is practically ineffective on all but very small scale problems (a n nodes WSN leads to a linear program of complexity $O(n^8)$). In addition, we have to solve the MNL-SMS problem and n maximum concurrent flow problems for all sink layouts in order to answer the question of TMNTM. In the rest of this section, we will discuss an approximation algorithm that solve the problem efficiently. Moreover, the elegance of this algorithm in the understanding of the problem structure by the interpretation of the dual problem also help us to build the related duality theory; it allows us to easily address the TMNTM decision problem.

Approximation Algorithm

Let us re-formulate the MNL-SMS problem into a Path-Flow form:

$$\begin{aligned}
&\text{Maximize} \quad \textstyle\sum_k t_k && && (4.22)\\
s.t. \quad &\sum_{p \in P_{ik}} f(p) - \lambda_i t_k &=& \; 0 \qquad \forall i, k && (4.23)\\
&\sum_k \sum_{p \in P_i^k} f(p)(e_i^t + e^r) - E_i &\leq& \; 0 \qquad \forall i && (4.24)\\
&f(p), \; t_k &\geq& \; 0 \qquad \forall p, k && (4.25)
\end{aligned}$$

where p refers to a certain path and $f(p)$ is the flow that goes through p. Furthermore, P_{ik} stands for the set of paths between a node i and the kth sink location and P_i^k represents the set of paths that go through node i in the kth epoch.

The dual problem is given by:

$$\text{Minimize} \quad \textstyle\sum_i E_i w(i) \tag{4.26}$$

$$s.t. \quad \sum_i \lambda_i W(i,k) \geq 1 \qquad \forall k \tag{4.27}$$

$$\sum_{j\in p,\ p\in P_{ik}} w(j)(e_j^t + e^r) - W(i,k) \geq 0 \qquad \forall i,k \tag{4.28}$$

$$W(i,k),\ w(j) \geq 0 \qquad \forall i,j,k \tag{4.29}$$

where $W(i,k)$ is the weight assigned to a "commodity" (data flow injected at a node in our case) from node i to the kth sink location and $w(j)$ is the weight assigned to a node j. The weight of a node $w(j)$ represents the marginal cost of using an additional unit of energy of the node, and the weight of a commodity $W(i,k)$ represents the marginal cost of rejecting a unit of demand of the commodity. Provided that the maximum lifetime is achieved, (4.27) says that the sum of λ_i multiplied by weights $W(i,k)$ for all n commodities in any epoch k is a least 1 (i.e., increasing the lifetime by one time unit without admitting yet another $\sum_i \lambda_i$ unit of demands is not beneficial), and (4.28) states that the shortest path between an arbitrary node pair i and k (the cost of routing a unit of demand) is no less than $W(i,k)$ (the cost of rejecting a unit of demand from i to k); otherwise we could have a longer lifetime either by rejecting or by admitting (thus routing) more demands. Here the length of a path is computed as the sum (over all nodes along the path) of the product of node weight $w(i)$ and the node energy consumption $e_i^t + e^r$.

Usually, the flow maximization problem involving multiple s-t flows can be solved by one of the algorithms proposed by Garg and Könemann [GK97]. However, MNL-SMS is a combination of two problems, namely a *maximum concurrent flow* problem and a *maximum multicommodity flow* problem. NML-SMS is, on one hand, a maximum concurrent flow problem because each node has a demand λ_i and the objective is to find a maximum multiple T for all nodes. On the other hand, if the time schedule $\{t_k\}$ is considered as a set of "commodities", then the objective is to maximize $\sum_k t_k$ without caring about any demand (note that some "commodities" can be zero); so this is a maximum multicommodity problem. Therefore, we need to develop new algorithms to solve MNL-SMS.

Let us denote the objective of the dual problem by $G(w) = \sum_i E_i w(i)$. In order to minimize $G(w)$, $w(i)$ should be as small as possible, but it is bounded from below by $W(i,k)$ through (4.27) and (4.28). Taking an arbitrary assignment w and $W(i,k) = \sum_{j\in\min\{p|p\in P_{ik}\}} w(j)(e_j^t + e^r)$ (i.e., the length of the shortest path from i to k), we meet (4.28). Then (4.27) becomes the following constraints:

$$\sum_i \lambda_i \left(\sum_{j\in p,\ p\in P_{ik}} w(j)(e_j^t + e^r) \right) \geq 1 \qquad \forall k \tag{4.30}$$

This assignment is not necessarily feasible because it might violate the above constraints. However, it can be made feasible by finding the most violated constraint and scale the assignment accordingly.

In other words, if there is an *oracle* that identifies $\min_k \rho_k(w)$: $\rho_k(w) = \sum_i \lambda_i W(i,k) < 1$, we can scale all assignments $w(j), W(k,i), \forall i,j,k$ by $[\min_k \rho_k(w)]^{-1}$ and thus make a feasible assignment. Therefore, the dual problem is equivalent to finding a weight assignment w: $\mathcal{V} \to \mathbb{R}_0^+$ such that $G(w)/\rho(w)$: $\rho(w) \equiv \min_k \rho_k(w)$ is minimized. We denote $\min_w G(w)/\rho(w)$ by β. Note that this interpretation of the dual problem already suggests a duality theorem analogous to the *max-flow min-distance ratio* theorem given in [SM90] (which is in turn analogous to the *max-flow min-cut* theorem of Ford and Fulkerson [FF62] for single s-t flow). We will discuss it more in detail later.

The approximation algorithm proceeds in iterations. Let w_{i-1}, W_{i-1} be the weight assignment at the beginning of the ith iteration and $\{t_{k,i-1}\}$ be the time schedule after iterations $1, \cdots, i-1$. In the ith iteration, we route $\sum_l \lambda_l$ units of commodity along the paths (and thus to the corresponding sink location) given by an oracle (we will specify it later) that computes $\min_k \rho_k(w)$ and let $t_{k,i} = t_{k,i-1} + 1$. Let $f_i(l)$ be the flow through node l and $p_{l,k}, \forall l$ be the paths suggested by $\min_k \rho_k(w)$ in this iteration. The new weight assignment to a node l is given by $w_i(l) = w_{i-1}(l)(1 + \epsilon f_i(l)(e_l^t + e^r)/E_l)$, and the new weight assignments to a commodity are computed as $W_i(l,k) = \sum_{j \in p_{l,k}} w_i(j)(e_j^t + e^r)$. Note that $p_{l,k}$ is indeed the shortest path between l and k because it is suggested by the oracle that computes $\min_k \rho_k(w)$. Now the dual objective is updated as:

$$\begin{aligned} G(w_i) &= \sum_l E_l w_i(l) \\ &= G(w_{i-1}) + \epsilon \sum_l w_{i-1}(l) f_i(l)(e_l^t + e^r) \\ &= G(w_{i-1}) + \epsilon \cdot \rho(w_{i-1}) \end{aligned} \tag{4.31}$$

Initially, the weight assignment to a node l is $w_0(l) = \delta/E_l$. The iteration stops when $G(w_i) \geq 1$ for the first time. We refer to Appendix C for details of setting parameters ϵ and δ.

The oracle that computes $\min_k \rho_k(w)$ is simply an extension of the Floyd-Warshall algorithm [Flo62] that computes all-pairs shortest path with a time complexity of $\Theta(n^3)$. We organize the results of the Floyd-Warshall algorithm into "clusters"; each cluster includes paths that have a common end. Then we run a search algorithm in order to find the best "median" k that achieves $\min_k \rho_k(w)$. This oracle has a time complexity of $\Theta(n^3)$ (because the later clustering and searching both have a time complexity at least one order lower than that of the Floyd-Warshall algorithm). Combining the oracle with the iteration procedure, we have the following claim:

Claim 3 *Given* $\sum_i \lambda_i \leq E_i/(e_i^t + e^r), \forall i,$[10] *there is an algorithm that computes a* $(1-\epsilon)^{-2}$*-approximation to the MNL-SMS problem in time* $\Theta(n \log n) \cdot T_{\text{oracle}}$*, where* $T_{\text{oracle}} = \Theta(n^3)$ *is the time complexity for the oracle to compute* $\min_k \rho_k(w)$*.*

---

[10] This assumption is reasonable because each sensor node should be equipped with an energy source that is at least enough to forward data for all nodes in one time unit. Otherwise if a node l: $E_l/(e_l^t + e^r) < \sum_i \lambda_i$ is deployed close to a static sink (assuming a randomly deployed WSN), the network lifetime can be even less than one time unit. In addition, it can be proved that an approximation ratio of $(1-\epsilon)^{-3}$ is still achievable without this assumption.

We provide the proof of this claim in Appendix C.

Another significance of this algorithm is that, if we have an oracle that is able to solve the following *p-median* problem:

> INSTANCE: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a weight assignment ω: $\mathcal{V} \rightarrow \mathbb{R}_0^+$, a length assignment[11] l: $\mathcal{V} \rightarrow \mathbb{R}_0^+$, positive integer $K \leq |\mathcal{V}|$, and positive rational number B.
>
> QUESTION: Is there a set $\mathcal{P}$ of K "points on $\mathcal{G}$" such that if $d(v)$ is the length of the shortest path (the sum of all lengths along the path) from v to the closest point in $\mathcal{P}$, then $\sum_{v \in \mathcal{V}} \omega(v) \cdot d(v) \leq B$?

then we are able to solve the original MNL problem (we refer to Appendix B for a formal proof of this point). However, the p-median problem is NP-complete [GJ79]. Yet, a reasoning procedure similar to what is shown in [CaAOZ03] suggests that if the oracle has an α-approximation, then the above algorithm provides an $\alpha \cdot (1-\epsilon)^{-2}$-approximation. In fact, Arya et al. [AGK$^+$04] gave a $(3+\omega)$-approximation algorithm for the p-median problem. Therefore, we have an algorithm to approximate the original MNL problem with a factor of $(3+\omega)(1-\epsilon)^{-2}$.

Duality Theory for MNL-SMS

We recapitulate the observation that we make on the dual problem of MNL-SMS in the following theorem:

***Theorem* 1 (Max-lifetime Min-potential ratio Theorem)** *Given the maximization lifetime problem formulated in (4.22)–(4.25), the optimal lifetime T is such that*

$$T = \min_w \left[\frac{G(w)}{\rho(w)}\right]$$

where $G(w) = \sum_i E_i w(i)$ is a linear combination of the energy reserves of all nodes with coefficients $w(i)$, $\rho(w) \equiv \min_k \rho_k(w) = \min_k \left(\sum_i \lambda_i \sum_{j \in \min\{p | p \in P_{ik}\}} w(j)(e_j^t + e^r)\right)$ is the minimum "potential" (computed as the sum of the minimum cost, given $w(i)$, to route λ_i from node i to a center k) achieved among all possible centers (i.e., sink locations).

We omit the detailed proof of this theorem; a sketch of the proof has been provided when explaining our approximation algorithm.

We also quote the theorem given in [SM90] and improved in [GK97] as follows:

---

[11] Usually, a length assignment is associated with edges. However, we can always convert our node-capacity based problem to a link-capacity based version by replacing a node with two nodes and a link having the same capacity.

***Theorem* 2 (Max-Flow Min-distance ratio Theorem)** *Given the maximization lifetime problem formulated in (4.22)–(4.25) but with a schedule consisting of only one element* (sl, t)*, the optimal lifetime* T' *is such that*

$$T' = \min_{w}[\frac{G(w)}{\rho_k(w)}]$$

where $G(w)$ *and* $\rho_k(w)$ *are exactly the same as what are defined in the previous theorem, and the center* k *is the sink location defined by* sl*.*

Claim 4 $T > T'$*, i.e., the answer to the TMNTM decision problem is positive.*

Proof: Assume that $\hat{T}$ is the maximal one among all possible T's, and $\{\hat{w}\}$ is the corresponding weight assignment. By plugging $\hat{w}$ into the dual problem of MNL-SMS (4.26)–(4.29), we can always identify a violated constraint with the oracle that computes $\min_k \rho_k(w)$. For instance, assume that the current sink location is i and its most loaded neighbor is j (for which (4.24) is active). Applying complementary slackness, we have $\rho_i(\hat{w}) = 1$ (by $\hat{T} > 0$), $\hat{w}(i) = 0$ (by the fact that (4.24) is inactive for i), and $\hat{w}(j) > 0$ (by the fact that (4.24) is active for j). The potential $\rho_j(\hat{w})$ is bound to be less than 1, because, by moving the sink from i to j, we shorten the length of some paths by $\hat{w}(j)$ without increasing the length of other paths going through i. Therefore, we identify that $\{\hat{w}\}$, as the dual solution, is infeasible. Consequently, according to the principle of *certificate of optimality*, we know that $\hat{T}$, as the primal solution, is not optimal and thus $T > \hat{T}$. Note that the above procedure works also for the case of multiple sinks, but, as we show in Appendix B, the oracle that computes the minimum "potential" is NP-complete. ■

Our positive answer to TMNTM basically says that, if we only take data traffic into account (see our assumption [d]), using a mobile sink always achieves a lifetime longer than what is achievable by a static sink at any location. However, as we will show in Chapter 5, moving the sink might not always improve the lifetime when control traffic is also considered. Therefore, theoretical computations should be complemented by practical tests in order to justify the benefit of applying mobile sinks.

4.3.6 Numerical Experiments

In this section, we test our approximation algorithm by positioning a single mobile sink in several WSNs of typical topologies. We always assign the same energy reserve E and transmission energy e^t to all nodes in order to facilitate the interpretation of the results. As a consequence, we can use $e = e^t + e^r$ to represent the energy spent by each node to forward one byte of data. Without loss of generality, we also assume $\lambda = 1$, $e = 1$, and $E = n$. We set $\epsilon = 0.01$, i.e., the lifetime computed by our algorithm will be at least 98% of the optimal solution.

Line Network

For the line network shown in Fig. 4.2, it is easy to see that the best (static) sink location is at node 0. The reason is that any deviation from this center point to the left (right) will increase the load taken by the right (left) neighbor of the sink and thus shorten the lifetime. Note that, according to assumption [c],the sink inherits the data collection function of the co-located node. Otherwise, if the sink used the co-located node as its gateway to the network, the lifetime would not change with different sink locations because the co-located node would always take the forwarding load from all the $2m+1$ nodes and would thus always "die" first. Now we run our algorithm to show how a mobile sink should be positioned.

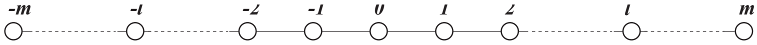


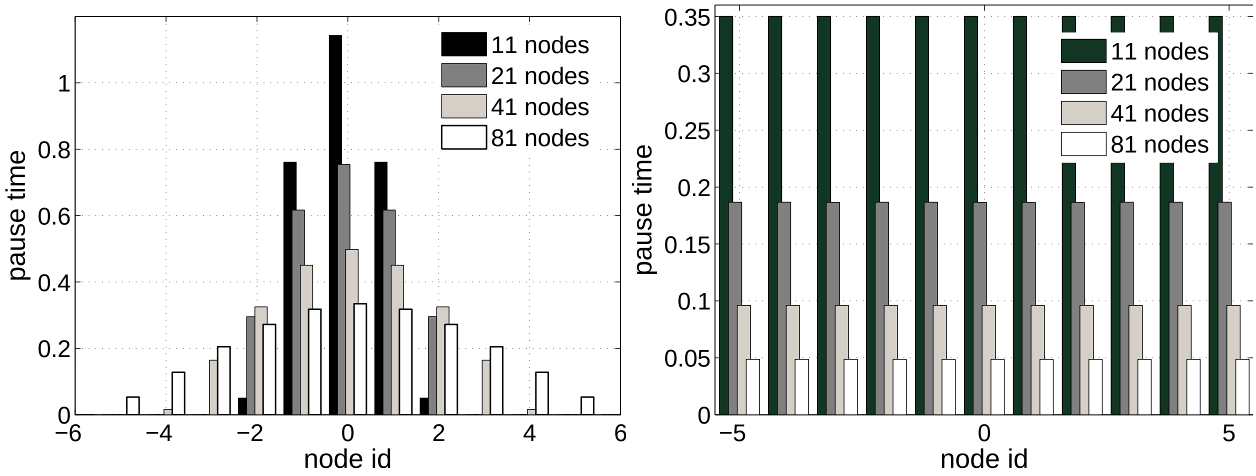
Figure 4.2: A line network with $2m+1$ nodes. Each node (except nodes m and $-m$ that only have one link) has two links with its left and right neighbors.

As we show in Table 4.1, using a mobile sink can always achieve a longer lifetime than using a static one. However, the relative improvement decreases with the size of a network. The reason is as follows: besides the load-balancing effect that we discussed in Section 4.1, there is another "hidden" benefit from moving the sink; it inherits the data forwarding load from the node that it co-located with and thus saves the energy consumption of that node (we call it *substitution effect*). The mobile node approach [WSC05] has exactly exploited this effect to improve lifetime. In a line network, moving a sink does not lead to load balancing, because it can be easily seen that moving the sink only results in an increase of load for some nodes without lightening others'. Therefore, the lifetime improvement is only brought in by the substitution effect, whose absolute quantity grows only sub-linearly with the network size.

$\lvert\mathcal{V}\rvert$	**Network Lifetime T**		
	mobile sink	static sink (optimal)	improvement (%)
11	2.765	2.200	25.67
21	2.578	2.100	22.77
41	2.408	2.050	17.47
81	2.285	2.025	12.82

Table 4.1: Comparing the achievable lifetime between using a mobile sink and a static sink (at its optimal position) in line networks.

In Fig. 4.3 (a), we also show the trace of the mobile sink. It can be immediately seen that the larger the network size is, the shorter the sink pauses at node 0 (and thus the longer it pauses at other nodes). The reason is that, when the network grows in size (and thus length), it appears (to



(a) Line networks

(b) Ring networks

Figure 4.3: Pause times of a mobile sink in line and ring networks. For the line networks, only non-zero values are shown. For the ring networks, we zoom to the section between nodes -5 and 5.

a centered node) more and more like a ring. For ring networks (Section 4.3.6), the sink pauses at every node for the same amount of time. Therefore, a larger line network tends to have a more "spread" pause time distribution.

Ring Network

For the ring network shown in Fig. 4.4, the achievable lifetime by a static sink is again $2 + 1/m$, but it can be obtained by putting the sink at any node, due to the symmetry of such a network. The relative improvement is converging to 100% with an increasing network size (Table 4.2). There is no surprise here because the traffic load is fully averaged among all nodes. This averaging effect can be also seen in Fig. 4.3 (b) (where the pause time is illustrated); the sink pauses at every node for the same amount of time $t = T/(2m + 1)$.

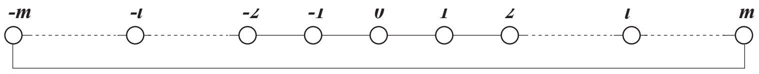


Figure 4.4: A ring network with $n = 2m + 1$ nodes. Each node has two links with its left and right neighbors.

$\|\mathcal{V}\|$	**Network Lifetime** T		
	mobile sink	static sink	improvement (%)
11	3.851	2.200	75.05
21	3.920	2.100	86.66
41	3.940	2.050	92.18
81	3.945	2.025	94.81

Table 4.2: Comparing the achievable lifetime between using a mobile sink and a static sink in ring networks.

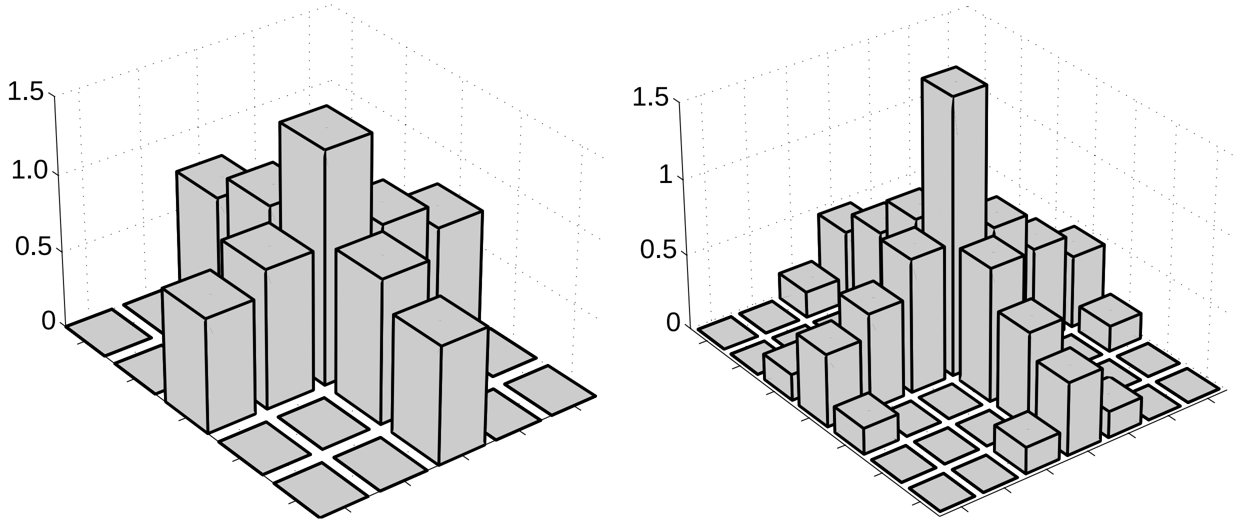
Grid Network

For grid networks on $\sqrt{n} \times \sqrt{n}$ lattices, the maximum achievable lifetime by a static sink is $n/(\lceil (n-5)/4 \rceil + 1)$, because the lifetime is maximized if the forwarding load is balanced among the 4 neighbors of the sink. This lifetime can be obtained by putting the sink at the network center (if $\sqrt{n}$ is odd) or at any of the four nodes close to the center (if $\sqrt{n}$ is even). While this lifetime is converging to 4 when $n \to \infty$, the lifetime achieved by a mobile sink increases dramatically with the network size (Table 4.3). For small-size networks (e.g., $|\mathcal{V}| = 9$ in Table 4.3), the substitution effect dominates the load balancing effect, so the relative improvement is small. With an increasing network size, the number of alternative paths between an s-t pair is also increasing. Consequently, the load balancing effect becomes increasingly remarkable and thus produces significant improvements on the lifetime.

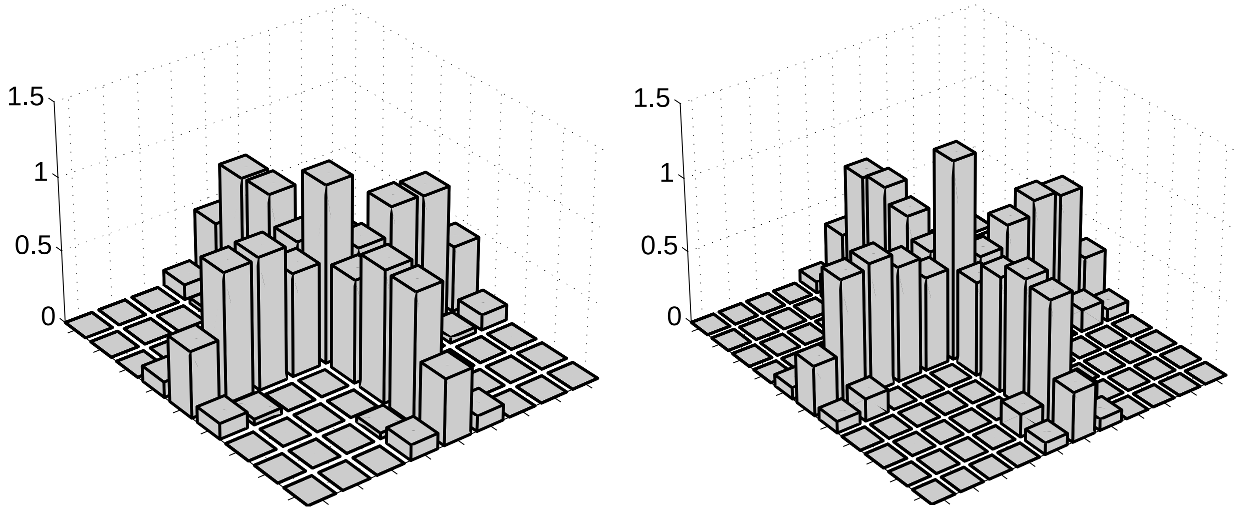
$\|\mathcal{V}\|$	**Network Lifetime** T		
	mobile sink	static sink (optimal)	improvement (%)
9	5.331	4.500	18.47
16	6.509	4.000	62.72
25	8.146	4.167	95.51
49	11.09	4.084	171.7
81	14.08	4.050	247.6
121	17.07	4.033	323.2

Table 4.3: Comparing the achievable lifetime between using a mobile sink and a static sink in grid networks.

We illustrate the pause time in four networks in Fig. 4.5. Our observation is that the sink tends to move toward the periphery of a network with an increasing n. The intuition is that, for a 3D grid on a sphere, the sink should pause everywhere with same time period (analogous to a ring in 2D). Therefore, the pause times spread out when the network grows in size and thus appears to a centered node more and more like a sphere grid (analogous to a line in 1D). This observation also corroborates the result in Section 4.4: the network periphery, as a sink moving



(a) 25 nodes (b) 49 nodes



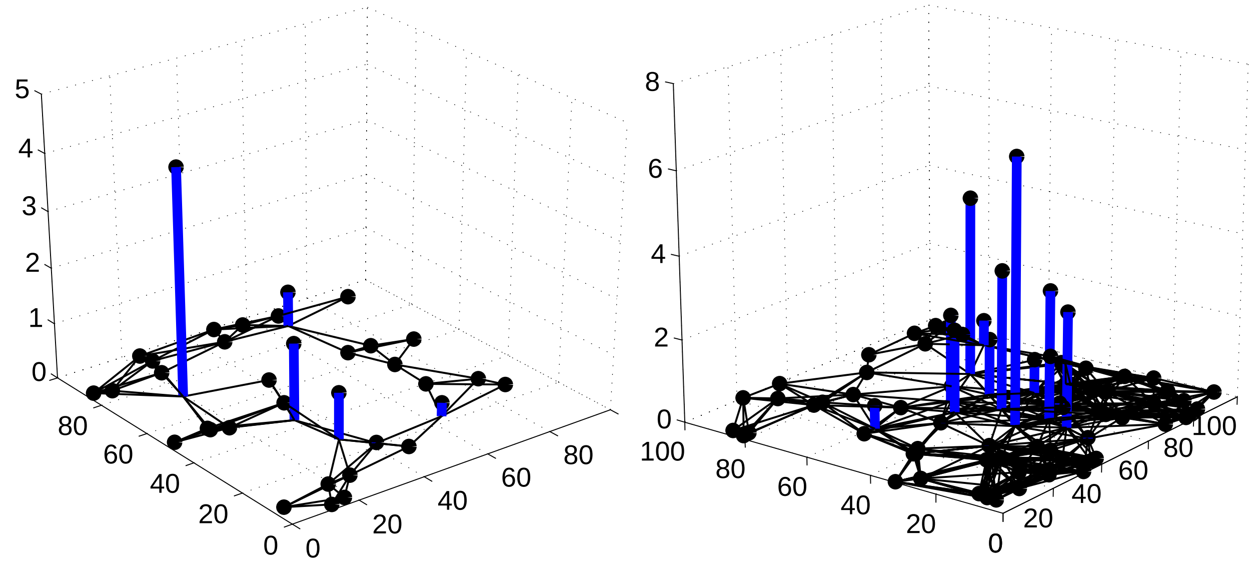
(c) 81 nodes (c) 121 nodes

Figure 4.5: Pause times of a mobile sink in grid networks. The z-axis represents the pause time.

trace, is asymptotically optimal.

Random Network

We also perform experiments on random networks (nodes uniformly distributed within a square). We illustrate the pause time in two networks in Fig. 4.6. A direct observation is that the sink tends to pause at the node whose degree is high. This is intuitive because the more the neighbors the more balanced load can be achieved. The relative improvements are 30.52% and 24.83% in networks of 35 and 80 nodes, respectively. Since we fixed the area of the square, increasing network size leads to higher connectivity. Therefore, these experiments also show that higher connectivity disfavors sink mobility (in particular, the network with a fully connected topology needs no sink mobility).



(a) 35 nodes (b) 80 nodes

Figure 4.6: Pause times of a mobile sink in random networks. The z-axis represents the pause time.

4.3.7 Summary

Our investigations in this section show that the problem of maximizing lifetime using multiple mobile sinks is generally hard. Although some relaxations are tractable, the complexity of solving these problems, though polynomial in n, is still high. For example, solving the MNL-SMS problem in a 200-node network with MATLAB® needs several days. In addition, a general picture of what the sink moving trace looks like in networks of very large size is still missing; which is mainly due to the graph model that does not allow such a characterization. All these observations motivate us to consider a continuum model, under which the lifetime optimization problem is readily solvable and the moving trace can be mathematically figured.

4.4 Optimization under a Continuum Model

In this section, we switch to a model where the whole WSN is considered to be a liquid mass and the traffic load can be modelled as the intensity of pressure. We first show that moving a sink (even arbitrarily) significantly improves the network lifetime compared with a static sink. We then describe both the optimal moving trace of a sink and an improved routing strategy. We also confirm our analytical results with simulations.

4.4.1 System Model

We assume a WSN consisting of a set N of static sensor nodes that harvest data from the area covered by the network and one sink that collects data from all nodes. We focus only on the communications between the nodes and the sink, whereas the communications between the sink and devices outside the network are out of the scope of this chapter. We assume that nodes are distributed uniformly with density ρ within a circle $\mathcal{C}_{OR}$ of center O and radius R. The density ρ is so large that the network is strongly connected and one can treat the whole network as a liquid mass. Each node sends data to the sink with a constant rate λ. The overall energy for a node to receive and transmit a unit of data is ε (it is the sum of e^t and e^r). The transmission and sensing ranges of all nodes are identical and fixed at r ($r \ll R$). For simplicity, we assume an ideal load-balanced short path routing protocol[12] (we refer to [GZ04] for a profile of such protocols), and we do not consider data aggregation (e.g., [IGE$^+$03, PSRR03]) when data are collected[13]. Fig. 4.7 illustrates this model. The model can be extended to cases where multiple sinks exist by dividing the network into several sections with one sink assigned to each.

4.4.2 Problem Formulation

Let us first define the relevant concepts before specifying our problem.

---

[12] We actually apply the second relaxation mentioned in Section 4.3.4, i.e., we “pre-define” the flow schedule by choosing a specific routing protocol.

[13] On one hand, it can be easily seen that applying these strategies can only improve the performance of our protocols. On the other hand, assuming redundancy in order to use data aggregation becomes impractical if the sensed areas are much smaller than the overall areas under monitoring (for instance, collecting soil moisture over large agricultural areas [CSense]).

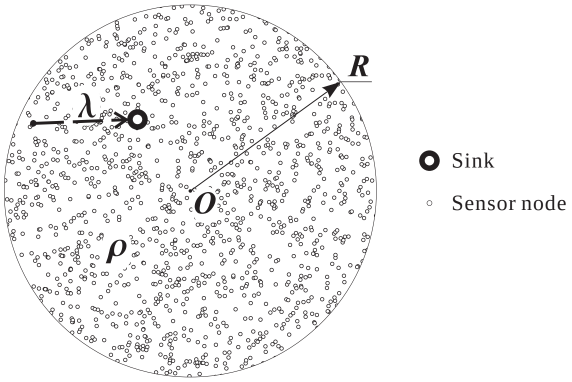


Figure 4.7: Network continuum model.

Network Lifetime

We define the *network lifetime* as the time span from the sensor deployment to the first *loss of coverage* [BC02] (i.e., the time when some area initially covered by the network is not sensed by any active node any more). Although this definition seems different from the one we have in Section 4.3.2, it is just an adapted version under the continuum model, for which the death of one molecule (i.e., a node) affects nothing.

Traffic Load

The *traffic load* (or *load* in brief) of node n, $load_n$, is the power that n consumes to transmit and receive data. It is obvious that the higher $load_n$ is, the shorter the lifetime of n is. The *average load* of n, $\overline{load}_n$, is an average over both a time period and a subset of nodes that are within the sensing range of n. The time average is necessary if the sink mobility (*mobility* hereafter) is introduced, since $load_n$ becomes time-variant. And the geographical average makes sense due to our definition of network lifetime; whereas the quick loss of individual nodes bearing a high load may not lead to the loss of coverage, a subset of nodes with a high average load do leave a coverage "hole" in the network after depleting their batteries all together. We also provide our definition of *energy efficiency* in order to clarify its relationship with the network lifetime: A protocol is energy efficient if it minimizes the accumulative energy consumption for fulfilling its task (e.g., minimum-energy broadcast [CHE02]). It has long been recognized that an energy efficient protocol does not necessarily maximize the network lifetime [CT00].

Min-Max Problem

According to the relation between network lifetime and the average load taken by a node, we convert the problem of maximizing network lifetime to a problem of load balancing, and formulate it as a min-max problem in terms of the average load of a certain node n:

$$\begin{aligned} &\text{Minimize} \quad load_N \equiv \max_{\forall n \in N} \overline{load}_n(\text{strategies}) \qquad (4.32) \\ &\text{Constraints : specific to given strategies.} \end{aligned}$$

The intuition behind this formulation is that the network lifetime is roughly in inverse proportion to $load_N$, or the *network load*. Existing solutions, when specifying the problem *decision variables* (or *strategies*) and *constraints*, only take routing strategies into account. In this chapter, we intend to show that, by considering both mobility and routing strategies, one can yield a better solution to the problem. But before doing this, we first demonstrate in the next section that mobility is indeed a strategy that deserves to be considered. Note that, in our proposal, the sink **does not** rely **only** on mobility to retrieve data from sensors (in opposition to the proposals in [SRJB03, CSA03]); the data collection procedure continues through multi-hop routing wherever the sink stays.

4.4.3 To Move or Not To Move

In this section, we compare two cases. In one case, we locate the sink in a place where the most energy efficient data collection is achieved. In the other case, we require the sink to move in an arbitrary way. We analytically quantify the benefit in terms of network lifetime due to mobility, and thus show that mobility deserves to be a strategy for solving the problem specified by (4.32).

Networks with a Static Sink

We first place the sink at its optimum location in terms of energy efficiency, then we show that the network lifetime is quite limited with this optimum sink position.

Claim 5 *The center of the circle $\mathcal{C}_{OR}$ is the optimum position for a sink in terms of energy efficient data collection.*

Proof: Let the sink be at B (x_B, y_B) and consider an infinitesimal area S that measures $\mathrm{d}x \times \mathrm{d}y$ and is centered on (x, y), as shown in Fig. 4.8. Given the Euclidean distance $d = \sqrt{(x - x_B)^2 + (y - y_B)^2}$ from the center of S to B, the routing path length (in hops) l from S to B is approximately linear in d (i.e., $l \approx kd$) due to the assumption of a short path routing protocol. As a result, the energy consumed to transmit data from S to B is $kd\varepsilon \times \lambda t\rho \times \mathrm{d}x\mathrm{d}y$, where $\lambda t\rho \times \mathrm{d}x\mathrm{d}y$ is the amount of data produced within time t and $kd\varepsilon$ is the energy spent to transmit a unit of data from S to B. The total energy consumption is $E = \int_y \int_x kd\varepsilon\lambda t\rho \, \mathrm{d}x\mathrm{d}y$. It is easy to

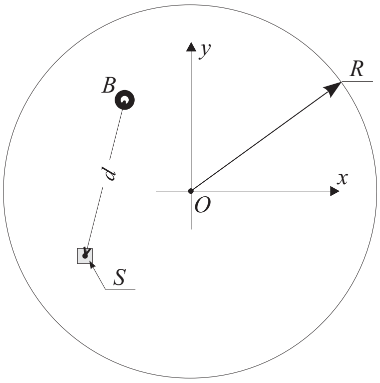


Figure 4.8: Proof of optimum sink position.

see that minimizing E is equivalent to minimizing

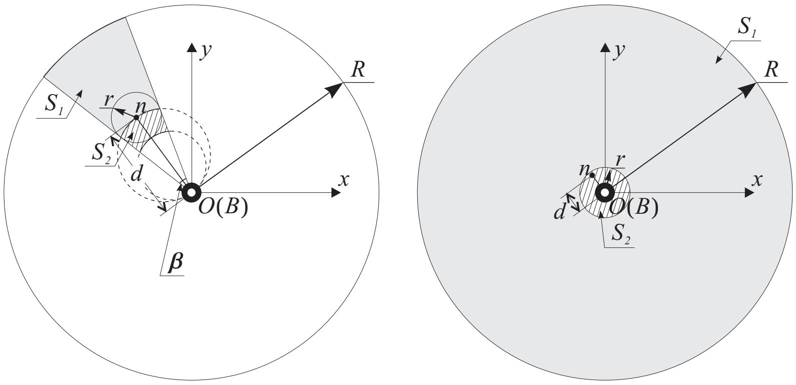
$$\begin{aligned}\int_{-R}^{+R}\int_{-\sqrt{R^2-y^2}}^{+\sqrt{R^2-y^2}} &[(x-x_B)^2+(y-y_B)^2]\mathrm{d}x\mathrm{d}y \\ &= \frac{1}{2}\pi R^2(2x_B^2+2y_B^2+R^2)\end{aligned}$$

which achieves the minimum value if $x_B = y_B = 0$, i.e., the sink is placed at the center O. ■

We will show that, even with this optimum position, the load is poorly distributed among the nodes. Given a node n that is at distance d from the sink B (also the center O), as shown Fig. 4.9, the geographical average load taken by this node is in proportion to $(S_1 + S_2)/S_2$. Here we adopt and extend the model proposed in [GK04]. In Appendix D.1, we give detailed explanations about how this model is built. The intuition is that all the traffic flows from both areas S_1 and S_2 have to go through nodes in area S_2, which forms a "pressure" on those nodes. The (geographical) average power that a node in S_2 spends to forward the data traffic can be calculated as the intensity of the pressure:

$$\begin{aligned}\overline{load_n} &= \frac{(S_1+S_2)\rho\lambda\varepsilon}{S_2\rho} \\ &\begin{cases} \approx \frac{\frac{\beta}{2}(R^2-d^2)\lambda\varepsilon}{\frac{\pi}{2}r^2} + \lambda\varepsilon & d \geq r \\ = \frac{R^2\lambda\varepsilon}{r^2} & d < r \end{cases}\end{aligned} \tag{4.33}$$

where $\beta = 2\arcsin(r/d)$. As shown in Fig. 4.10, the average load of a sensor node increases dramatically with the decreasing distance between the node and the sink. This means that the nodes around the sink use up their energy much faster than other nodes, because they have to



(a) $d \geq r$ (b) $d < r$

Figure 4.9: Calculation of load distribution with a centered static sink. S_1 and S_2 in (a) are two disjoint parts of a sector whose two sides are tangent to a circle of center n and radius r.

forward a great amount of traffic flows even though their number is limited. Therefore, the network lifetime is upper bounded by the lifetime of these nodes. Also, when these nodes run out of batteries, the sink has no way to collect data any more (even though a large part of the network is still "alive"), because the network is partitioned. Finally, the $load_N$ does not vary with different sink position, according to (4.33) with $d < r$. However, the centered position is optimum in terms of energy efficiency, as shown by **Claim 5**.

Networks with a Moving sink

Intuitively speaking, a moving sink can distribute over time the role of "hot spots" (i.e., the nodes around the sink), such that the load can be evened out. In this section, we prove that this intuition is indeed correct. Since the data collection procedure continues wherever the sink stays, 1) any departure of the sink from the center increases the worst-case latency (whose maximum value doubles compared with that in the case of centered static sink) and 2) the moving speed is **not** essential to a mobility strategy.

We assume that the sink moves in such a way that it appears everywhere with the same frequency in the long run. We could continue using the model in Section 4.4.3, but it would result in an extremely complex integral, which can only be computed through numerical methods and does not provide enough insight into the system performance. Therefore, we simplify the model in order to obtain a closed form expression. Let us consider the power consumption of an arbitrary node n that is at distance d from the center, with respect to a random sink position B. As shown in Fig. 4.11, we consider that node n is charged with the (geographical average) forwarding load from a small sector S_3, when the sink stays at B on segment nC; A and C are intersections of line nB and circle $\mathcal{C}_{OR}$,

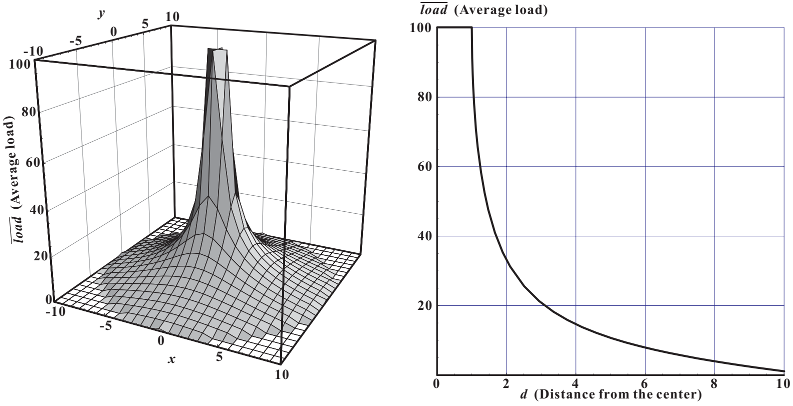


Figure 4.10: Load distribution with a centered static sink. We assume $R = 10$, $r = 1$, $\rho = 8/\pi$, $\lambda = 1$, and $\varepsilon = 1$.

and S_3 is centered around line nA with an angle of θ. This model is not physically as explainable as the model in Section 4.4.3. However, in Appendix D.2, we show that they are equivalent (in a sense that $(S_1 + S_2)/S_2 = S_3\rho$) and θ is a decreasing function of $|nB|$. For simplification, we apply an average value $\bar{\theta} = 0.2$ that is estimated in Appendix D.2 for any positions of B and n within $\mathcal{C}_{OR}$. In order to facilitate further calculation, we also consider that B is located in another sector S_4 centered around line nC, with an angle of $\Delta\gamma$. When $\Delta\gamma \to 0$, it goes back to the situation where B is on segment nC. Since B visits everywhere within $\mathcal{C}_{OR}$, we divide $disk_{OR}$ (the area within $\mathcal{C}_{OR}$) into disjoint sectors $\{S_4^\gamma\}$ such that $\bigcup_\gamma S_4^\gamma = disk_{OR}$. Now we calculate the average load of node n by further taking a time average:

$$\begin{aligned}
\overline{load}_n &= \sum_{\gamma=0}^{2\pi} \overline{load}_n|_{\{B \text{ in } S_4^\gamma\}} \times \mathrm{Fr}\{B \text{ in } S_4^\gamma\} \\
&= \sum_{\gamma=0}^{2\pi} S_3^\gamma \rho\lambda\varepsilon \times \frac{S_4^\gamma}{\pi R^2} \\
&\approx \sum_{\gamma=0}^{2\pi} \frac{1}{2}|nA|^2\bar{\theta}\rho\lambda\varepsilon \times \frac{\frac{1}{2}|nC|^2\Delta\gamma}{\pi R^2} \\
&= \sum_{\gamma=0}^{2\pi} \frac{(R^2 - d^2)^2\bar{\theta}\rho\lambda\varepsilon\Delta\gamma}{4\pi R^2}
\end{aligned} \tag{4.34}$$

where γ takes only a discrete value of $k\Delta\gamma, k \in \mathbb{Z}^+$. The calculation of the *occupying frequency* $\mathrm{Fr}\{B \text{ in } S_4^\gamma\}$ is based on the assumption that the sink visits everywhere with the same frequency.

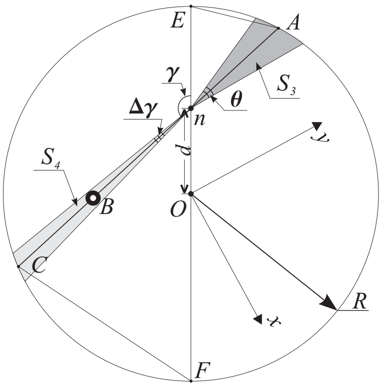


Figure 4.11: Calculation of load distribution with a mobile sink.

The reason $|nA|^2 \times |nC|^2 = (R^2 - d^2)^2$ is that triangles ΔnAE and ΔnFC are similar. Let $\Delta\gamma \to 0$, the sum over $[0, 2\pi]$ in (4.34) becomes an integral over $[0, 2\pi]$:

$$\begin{aligned} \overline{load}_n &= \int_0^{2\pi} \frac{(R^2 - d^2)^2 \bar{\theta}\rho\lambda\varepsilon}{4\pi R^2} \mathrm{d}\gamma \\ &= \frac{1}{2}\left(\frac{R^2 - d^2}{R}\right)^2 \bar{\theta}\rho\lambda\varepsilon \qquad (4.35) \end{aligned}$$

The average load is plotted in Fig. 4.12. A comparison between Fig. 4.12 and Fig. 4.10 shows

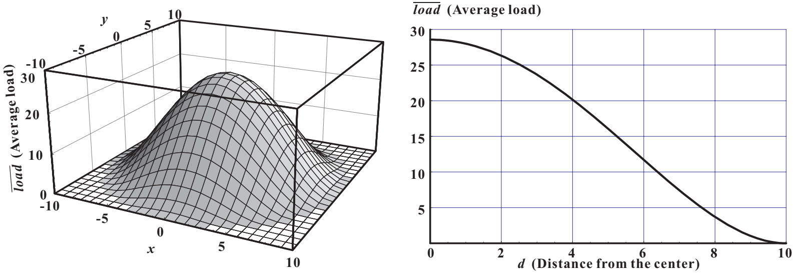


Figure 4.12: Load distribution with a mobile sink. We assume $R = 10$, $r = 1$, $\bar{\theta} = 0.2$, $\rho = 8/\pi$, $\lambda = 1$ and $\varepsilon = 1$.

that the maximum average load is much lower in the case of a mobile sink, indicating an extended network lifetime (more than 3 times). We also provide simulation results for these two cases in Section 4.4.5, which corroborate our analysis. In addition, the closed form expression (4.35) actually suggests that, given certain values for λ and ε, there are two possibilities for further improvements:

- Reducing θ for the hot spot (the center),
- Reducing the network size characterized by R.

Note that reducing node density ρ does not help, because, according to Appendix D.2, θ increases when decreasing ρ.

4.4.4 Optimal Mobility and Improved Routing Strategy

In the previous section, we have shown that mobility helps to balance the load and prolong the network lifetime. This suggests that mobility is indeed a promising strategy to optimize the network lifetime. Now, we refine our problem definition of maximizing network lifetime as:

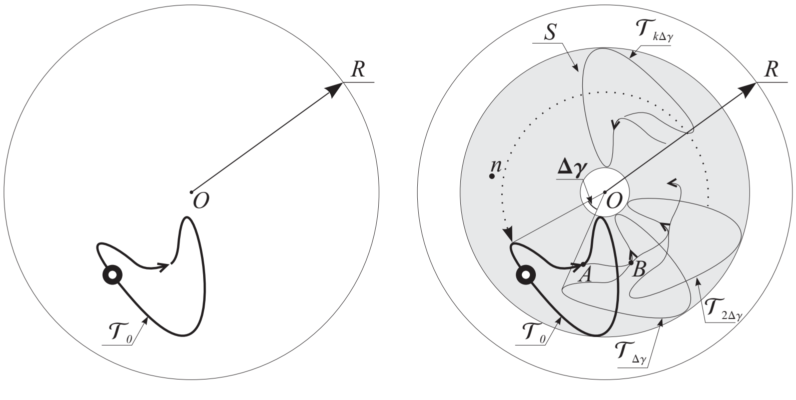
$$\begin{aligned} \text{Minimize} \qquad & load_N \equiv \max_{\forall n \in N} \overline{load}_n(\mathcal{M}, \mathcal{R}) \\ \text{Constraints :} \quad & \mathcal{M} \text{ constraints; } \mathcal{R} \text{ constraints} \end{aligned} \tag{4.36}$$

where $\mathcal{M}$ and $\mathcal{R}$ refer to mobility strategies and routing strategies, respectively. A traditional way of solving the problem with only routing taken into account is linear programming [CT00], in which the routing constraints are actually flow conservation. However, since we add the mobility strategy, the size of the strategy space increases dramatically. Therefore, we only rely on the following heuristics to achieve a "better" (but not necessarily the "best") solution to the problem: we first fix the routing strategy to short path routing and search for the optimum mobility strategy, then based on the optimum mobility strategy, we search for a routing strategy that performs better than short path routing. The constraints are discussed separately for each strategy.

Optimum Mobility Strategy

We first fix the routing strategy and search for the optimum mobility strategy under the constraint that the sink should not move out of the network region (otherwise it cannot collect data any more). At first glance, the strategy space of mobility is enormous because the number of traces that can be chosen is infinite. Fortunately, by defining *periodic* mobility as *recurrent movements with a constant period*, we can first reduce the size of the strategy space by removing all *aperiodic* mobility strategies (i.e., mobility strategies that are not periodic within the network lifetime). In fact, this category of mobility strategies can always be considered as a periodic mobility whose period is the same as the network lifetime. In addition, the following claim further limits our searching to periodic mobility strategies whose traces have *rotation symmetry of all degrees around the network center* (*symmetric strategies* hereafter for brevity and *non-symmetric strategies* otherwise):

Claim 6 *For each non-symmetric strategy that achieves a network load $load_N$, there exists one corresponding symmetric strategy that achieves a network load no larger than $load_N$.*


R
O
$\mathcal{T}_0$
S
$\mathcal{T}_{k\Delta\gamma}$
R
n
$\Delta\gamma$
O
A
B
$\mathcal{T}_{2\Delta\gamma}$
$\mathcal{T}_0$
$\mathcal{T}_{\Delta\gamma}$

(a) Non-symmetric strategy (b) Symmetric strategy

Figure 4.13: Transformation from a non-symmetric mobility strategy to a symmetric one.

Proof: Let $\mathcal{T}_0$ in Fig. 4.13(a) be the trace of an arbitrary non-symmetric mobility strategy, and let $load_N$ be the network load achieved with $\mathcal{T}_0$. In order to obtain the symmetric strategy corresponding to $\mathcal{T}_0$, we modify the trace in such a way that, after completing one period, the sink changes to another trace that rotates the previous one about the center with a $\Delta\gamma$ angle. For example, upon completing $\mathcal{T}_0$, the sink moves from point A to point B and then starts to move along $\mathcal{T}_{\Delta\gamma}$, as shown in Fig. 4.13(b). When $\Delta\gamma \to 0$, the mobility strategy becomes an *identical frequency movement*[14] in the annulus S (i.e., the sink appears everywhere with the same frequency on any concentric circle $\mathcal{C}_O \subset S$), which is a symmetric strategy. Note that the mobility strategy we considered in Section 4.4.3 is a special case of this category of strategies. Now, let us consider the average load of an arbitrary node n in the network. Let $\overline{load}_{n_{k\Delta\gamma}}$ be the average load of n when only the strategy $\mathcal{T}_{k\Delta\gamma}$ is considered, we have $\overline{load}_n = M^{-1}\sum_{k=0}^{k=M}\overline{load}_{n_{k\Delta\gamma}}$, where $M = 2\pi/\Delta\gamma$. Let $\Delta\gamma \to 0$, we have $\overline{load}_n = (2\pi)^{-1}\int_0^{2\pi}\overline{load}_{n_\gamma}\mathrm{d}\gamma \leq (2\pi)^{-1}\int_0^{2\pi} load_N \mathrm{d}\gamma = load_N$, since $load_N$ remains constant with rotation because of the embedded symmetry. Therefore, the network load achieved by the symmetric strategy ($\max_{\forall n\in N}\overline{load}_n$) is no larger than $load_N$. ■

This claim actually limits our search to two categories of mobility traces: movements on concentric circles and identical frequency movements in annuli, because they are the only symmetric strategies we can have within the network region. Finally, the following claim gives us the best mobility strategy we can have, under the condition of short path routing:

Claim 7 *The optimum symmetric strategy is the one whose trace is circle $\mathcal{C}_{OR}$ (i.e., the periphery of the network).*

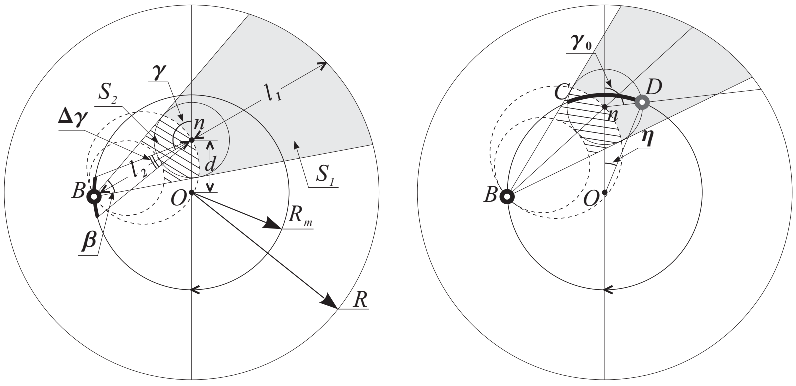
---

[14] This movement does not necessarily follow the trace we suggest here or a random walk (they both have an infinite period); it can always be achieved by a regular moving trace (e.g., a *space-filling curve* [Sag94]) with a short period.

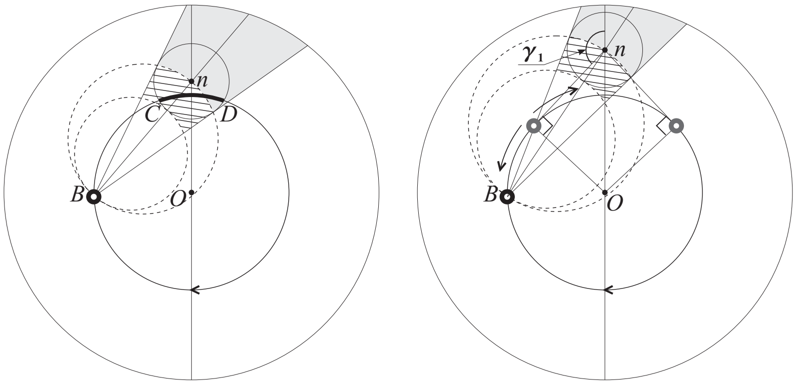
A formal proof is hard to achieve in this case, so we instead confirm the claim by analytically comparing mobility traces on concentric circles of radius R_m. With a circle trace, we cannot obtain a closed form expression, even with the simplified model of Section 4.4.3. As shown in Fig. 4.14, we turn back to use the model applied in Section 4.4.3. Similar to (4.33), we can calculate the average load as follows:

$$\overline{load}_n = \int_{\gamma} \frac{(S_1 + S_2)\rho\lambda\varepsilon}{S_2\rho} \times \mathrm{Fr}\{B \text{ on arc } l_2 \mathrm{d}\gamma\} \tag{4.37}$$

where $S_1 \approx \frac{\beta}{2}((l_1+l_2)^2 - l_2^2)$ and $S_2 \approx \pi r^2/2$. While $l_1 = \sqrt{R^2 - d^2\sin^2(\gamma)} + d\cos(\gamma)$, the calculation of l_2 depends on $d = |nO|$. There are four cases that should be distinguished:



(a) $0 \le d < R_m$-r (b) R_m-$r \le d < R_m$



(c) $R_m \le d < R_m+r$ (d) $R_m+r \le d < R$

Figure 4.14: Calculation of load distribution with the mobility strategy on a concentric circle.

[a]. $0 \le d < R_m - r$ in Fig. 4.14(a),

[b]. $R_m - r \le d < R_m$ in Fig. 4.14(b),

[c]. $R_m \le d < R_m + r$ in Fig. 4.14(c),

[d]. $R_m + r \le d < R$ in Fig. 4.14(d).

In the case of ([a]), $l_2 = \sqrt{R_m^2 - d^2 \sin^2(\gamma)} - d\cos(\gamma)$. Possible situations where $|nB| < r$ have to be taken into account for ([b]). The frequency of such events is η/π, and the average load in that case can be calculated by (4.33) with $d < r$; otherwise the integral is over $[\gamma_0, 2\pi - \gamma_0]$. The cases of ([c]) and ([d]) are different because γ only varies within $[\gamma_1, 2\pi - \gamma_1]$ and the integration has two parts: a clockwise one and a counter-clockwise one, as shown in Fig. 4.14(d). The calculation of the counter-clockwise integration is the same as ([a]). For the clockwise part, $l_2 = -\sqrt{R_m^2 - d^2 \sin^2(\gamma)} - d\cos(\gamma)$ and, in the case of ([c]), possible situations where $|nB| < r$ are treated in the same way as for ([b]).

Assuming $R = 10$, $r = 1$, $\rho = 8/\pi$, $\lambda = 1$, and $\varepsilon = 1$, we compute $\overline{load}_n$ with a numerical method. The results for $R_m = 3, 5, 7, 10$ are plotted in Fig. 4.15, and the result in Fig. 4.12 is also re-plotted (denoted by $R_m \in [0,\ 10]$). It is easy to see that (i) the trace with $R_m = R$ is the best

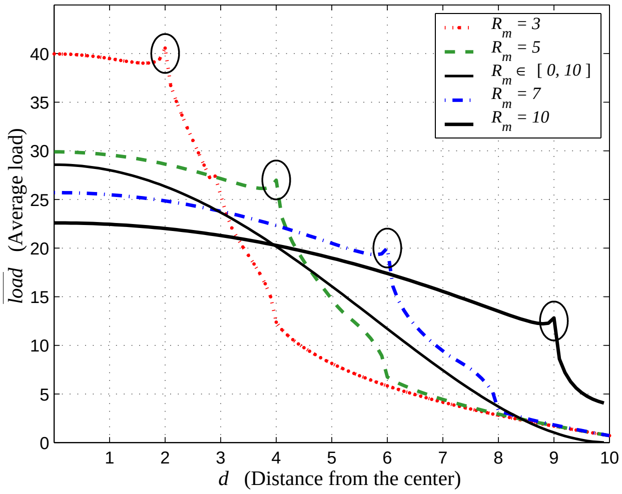


Figure 4.15: Comparison between mobility traces with different R_m. The singular regions (circled in the figure) are due to the imperfections of our approximations.

among all circle traces and (ii) the maximum average load is always achieved at the network center. According to these observations and considering the fact that moving in an annulus is a weighted average over movements on a set of concentric circles (e.g., the movement with $R_m \in [0,\ 10]$ is roughly equivalent to a trace of $R_m = 6$), we know that circle $\mathcal{C}_{OR}$ is the optimum mobility strategy under the condition of short path routing. These results prove the correctness of **Claim 7**. They also match the intuition we get from (4.35); the trace with $R_m = R$ maximizes the distance from the sink to the network center (which is always the hot spot), and thus minimizes the angle θ for the center, which in turn minimizes $load_N$. We validate this analysis by simulations in Section 4.4.5.

Improved Routing Strategy

According to (4.35) in Section 4.4.3, there are two ways to reduce the network load. In the previous section, we have already applied one of them (i.e., reducing θ) to achieve the optimum mobility strategy. So the only way to further reduce the network load is to decrease R. This implies a sacrifice of the network size, if only mobility strategy is considered. Fortunately, we still have another dimension of design strategy, i.e., routing. By investigating the load distribution of networks with a mobile sink (e.g., Fig. 4.15), we find that the nodes that are near to the border of the network always take a lighter load than the nodes near the center. A "better" routing strategy should exploit the energy capacity of these nodes to compensate the energy consumption of the hot spots. Our heuristic on joint routing and mobility strategy is shown in Fig. 4.16. The network is divided into two parts: an area within a concentric circle of radius $R_m < R$ and an area between that circle and the periphery of the network (the grey annulus). The sink only moves on the circle of radius R_m. The routing constraints[15] are such that the nodes within the inner circle still take the short routing path when transmitting data, whereas nodes in the annulus perform a two-step routing: the path first circles around the center O until it reaches OB (*round routing* hereafter), then it follows a short path to the sink. The direction of the round routing depends on the location of a node: clockwise on one side of the diameter OB and counterclockwise on the other side. The rationale

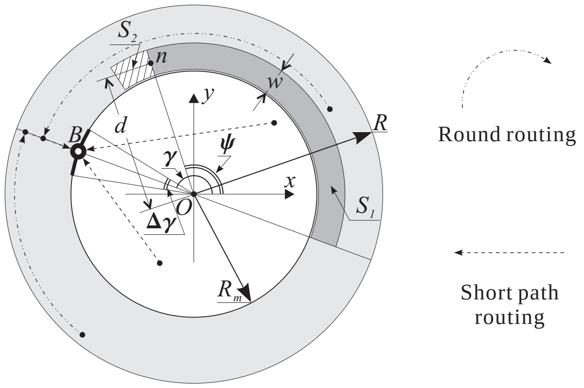


Figure 4.16: Joint mobility and routing strategy.

---

[15] Detailed constraints such as flow conservation could be applied for further optimization, but we use only this "high level" specification, because the chapter aims at demonstrating the benefit of mobility.

behind this heuristic is that this joint strategy tends to achieve a better performance (i.e., a lower network load) by reducing the radius of the network section that applies short path routing (but not of the whole network) from R to R_m.

The same analysis in Section 4.4.4 can apply for nodes located within the trace circle. Since the radius of this part has been reduced from R to R_m, the maximum load of this part is approximately $(R_m/R)^2$ of the $load_N$ achieved by the optimum mobility strategy (i.e., $R_m = R$), according to (4.35). For nodes outside the trace circle, a different analysis should be made because those nodes do not apply short path routing. If we could characterize the load distribution for this part of the network, we would choose R_m in order to balance the load between these two network sections.

It is easy to model the load distribution if we omit the load incurred by the second routing step. The S_1 and S_2 for a node n in this case are shown in Fig. 4.16. They have the same (radial) width w that is centered on n. The length of S_1 is $d|\psi + \pi - \gamma|$, and the length of S_2 is always r. Note that we are using rectangles to approximate the areas of S_1 and S_2, because w can be arbitrarily small. We can then estimate the average load of n with respect to the round routing as follows:

$$\begin{aligned}
\overline{load_n} &= \int_{\gamma=0}^{2\pi} \frac{(S_1+S_2)\lambda\varepsilon}{S_2} \times \mathrm{Fr}\{B \text{ on arc } R_m \mathrm{d}\gamma\} \\
&\approx \int_{0}^{2\pi} \left(\frac{wd|\psi+\pi-\gamma|\lambda\varepsilon}{rw} + \lambda\varepsilon\right) \times \frac{R_m \mathrm{d}\gamma}{2\pi R_m} \\
&= \left(\frac{d\lambda\varepsilon}{\pi r} \int_{\psi}^{\pi+\psi} (\psi+\pi-\gamma)\mathrm{d}\gamma\right) + \lambda\varepsilon \\
&= \frac{(\pi d + 2r)\lambda\varepsilon}{2r} \qquad (4.38)
\end{aligned}$$

However, this model works only for $d - R_m > r$. As we will see in Section 4.4.5, the load distribution in the annulus $[R_m - r, R_m + r]$ is hard to characterize, simply because, in reality, there is no clear demarcation between the two areas that apply different routing strategies. As a result, we have to rely on simulations to determine the radius R_m of a trace circle that performs better than the optimum mobility strategy. Nevertheless, the analytical model still provides instructive information. We can compute by (4.38) that the load taken by nodes near the network periphery is about 17, given the assumption that $R = 10$, $r = 1$, $\rho = 8/\pi$, $\lambda = 1$, and $\varepsilon = 1$. We also know that the maximum load of the inner network section is approximately $(R_m/R)^2$ of the $load_N$ achieved by the optimum mobility strategy (which is about 23 from Fig. 4.15). Therefore, R_m should be within [8, 10]: smaller values of R_m would not lead to lower $load_N$, because $load_N \geq 17$.

4.4.5 Simulations

In this section, we provide simulation results for the strategies presented in Section 4.4.3 and 4.4.4, including the static sink, the mobile sink, and the joint mobility and routing. We also compare these results with their corresponding analytical results. We perform simulations with a high level simulator programmed in MATLAB®, which ignores the MAC effects.

Our simulation set-ups consist of about 800 nodes deployed within a circle of $R = 10$ units. The nodes are randomly scattered as a Poisson process with density $\rho = 8/\pi$. Each node has a transmission range of $r = 1$ unit. We also normalize λ (data rate), ε (energy consumption for a data unit), and T (simulation time) to 1. For strategies with a mobile sink, we always consider a *discrete* (in the sense of both space and time) mobility trace for the sink. So if a mobility trace consists of m steps, the sink spends $1/m$ time for each step. We also assume that the sink only stays at sensor locations in order to keep the network connectivity independent of the positions of the sink; this results in an actual mobility trace that does not exactly follow the defined trace (e.g., a circle). We emulate the effect of load-balanced routing by randomizing all link weights before searching a routing path; this scheme distributes the traffic forwarding load from other nodes among a set of neighboring nodes. For each strategy, we perform 10 simulations with different node deployments.

Static vs. Mobile

We perform simulations for the two strategies analyzed in Section 4.4.3. A static sink is located at the node whose distance to the network center is the smallest, and a mobile sink stays at each node for a time period of $1/|N|$ (remember that we have $|N| \approx 800$ nodes in a network).

Fig. 4.17 compares the simulation results with the analytical ones. The figure represents the average load of nodes versus their distance d from the center. The analytical results are simply

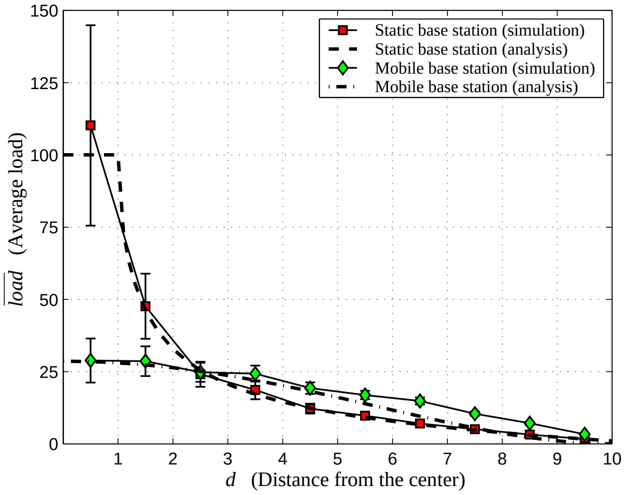
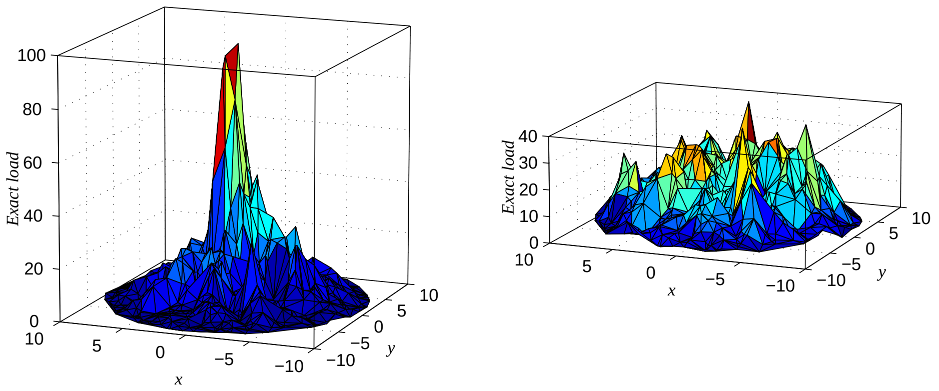


Figure 4.17: Comparison between analytical and simulation results on the average load (confidence intervals 95%).

a re-plot of the curves in Fig. 4.10 and 4.12. Simulation results match the analytical ones very well in the case of a static sink. For the mobile sink strategy, analytical results appear to be a bit over-optimistic, but they do not differ too much from the simulation results when d varies between 0 and 5. The difference becomes larger when d goes beyond 5; the reason is that we apply the $\bar{\theta}$ estimation of the center for every other nodes. In spite of these errors, the overall approximation is valid. Most importantly, it is observed that a mobile sink reduces $load_N$ ($\equiv \max_{\forall n \in N} \overline{load_n}$) by about 75%; this implies a 400% increase of the network lifetime.

We also evaluate the network lifetime defined as the time for the first node to die [CT00, DMLM04], which is often pessimistic [BS02]. Fig. 4.18 shows the simulation results of the exact (not average) load distribution for one of the network configurations. One can easily see the similarity



(a) (b)

Figure 4.18: Illustration of the exact load distribution: (a) static sink and (b) mobile sink.

between these plots and Fig. 4.10 and 4.12. The unusual spikes appearing in Fig. 4.18(b) (compared with Fig. 4.12) are due to three facts: (i) the network topology is not regular, (ii) the movement of the sink is not continuous, and (iii) the emulation of load-balanced routing is not perfect. However, the gain in lifetime is still quite obvious even with these spikes. Since the maximum load taken by a node is reduced by about 60%, the lifetime is increased by about 250%.

Optimum Mobility with Short Path Routing

We evaluate different mobility strategies (under the condition of short path routing) in this section. We make a comparison between three strategies: (i) movement on a concentric circle of radius $R_m = 5$, (ii) identical frequency movement in the network (the one described in Section 4.4.3, equivalent to $R_m \in [0,\ 10]$), and (iii) peripheral movement with $R_m = 10$. Each discrete movement on a circle consists of 72 steps, so a mobile sink stays at each step for a time period of $1/72$. Fig. 4.19 shows the comparison results. It actually confirms our **Claim 7** that peripheral movement is the best strategy. Furthermore, the comparison between analytical and simulation results also prove the

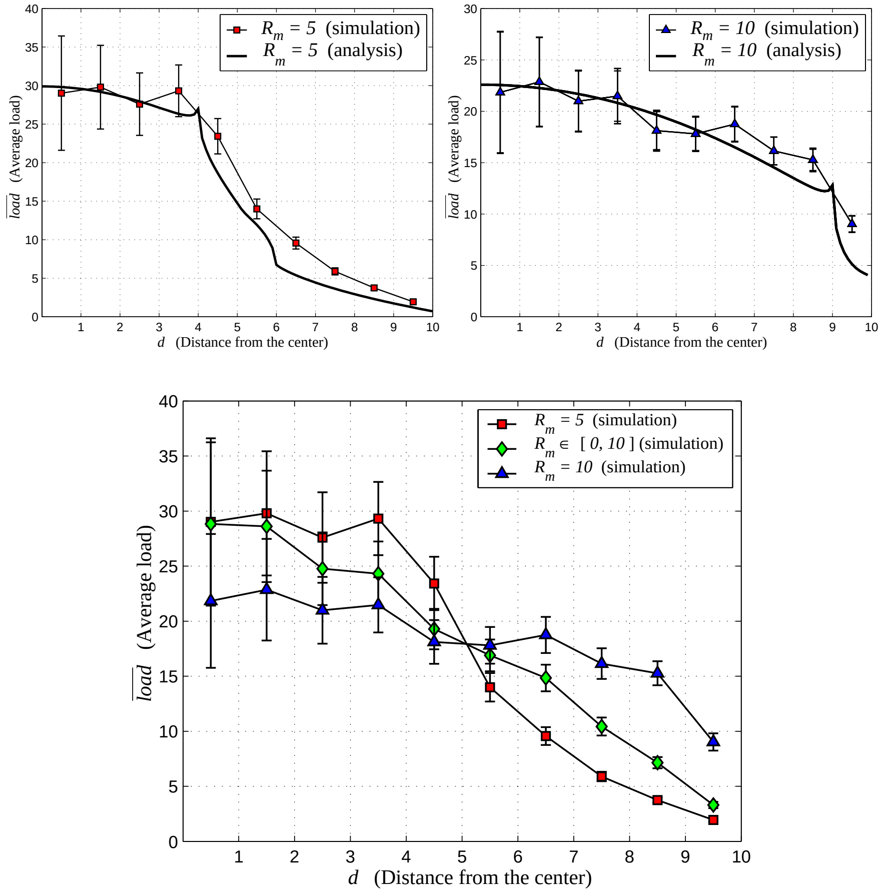


Figure 4.19: Comparison between different mobility strategies (confidence intervals 95%).

validity of our models. Finally, we observe that the identical frequency movement is better than the circle movement of $R_m = 5$ and is worse than the peripheral movement. This result supports our argument that the identical frequency movement is a weighted average over all the circle movements it covers.

Joint Mobility and Routing

In this section, we provide simulation results for the joint strategy proposed in Section 4.4.4. The results plotted in Fig. 4.20 show that the joint strategy indeed reduces the maximum load in the network section within the mobility trace, at the cost of an increased load in the network section outside the mobility trace. The maximum load of the two network sections becomes equal when

$R_m = 9$, which gives us the best choice of the joint strategy. The joint strategy further reduces the network load by about 10% compared with the optimum mobility strategy ($R_m = 10$). As a result, the overall improvement of the network lifetime, compared with the case of static sink (Fig. 4.17), reaches about 500%.

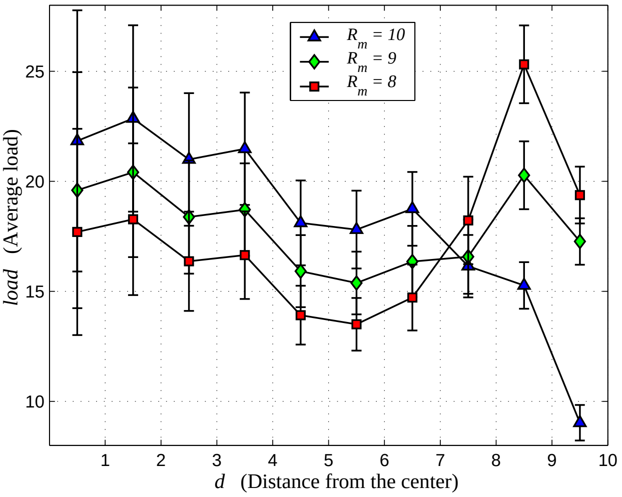


Figure 4.20: Comparison between different joint mobility and routing strategies (confidence intervals 95%).

4.5 Implementation Issues

We now explain the implementation aspects of the mobility and routing strategies described in Sections 4.3 and 4.4. The explanations are expressed as the answers to five questions.

1) How to make a sink move? There are basically three methods to move a sink: manually by human (or man-driven vehicles such as helicopter), automatically by relying on unmanned vehicles (e.g., robots [But03, KSJ$^+$04]), and virtually by requiring a set of fixed devices to act as a sink in shifts. The first and second methods usually work for outdoor WSNs (with the second one being more suitable for human-hostile environments), and the third one applies to indoor deployment (a good example is a smart building where wireless sensors are embedded in the structure). It is important to note that, in order to achieve a balanced load distribution, the mobility period (i.e., the time to complete one round along a trace) should divide the network lifetime. This implies, in practice, a period much shorter than the lifetime.

2) How to route data to the sink if the sink changes its location form time to time? We need to consider two cases:

a) The quality of links between nodes are very stable: The routing is easy to achieve in this case, if there is a (loose) time synchronization among nodes. The sink layout schedule and the corresponding routing paths can be determined after the deployment but before the operation of the network. As a result, each node knows the location of the sink at a given time and which routing path it should use to transmit data. Since the routing computation is performed offline, we can apply the algorithm described in Section 4.3.5 to find the optimal solution.

b) The quality of links between nodes are volatile: The sink has to announce whatever network topology changes caused by the mobility, in order to refresh the routing information of the nodes. This scheme does not necessarily bring too much additional overhead, because sensor networks with a static sink also need periodic query flooding [IGE$^+$03] and because practical routing protocols (e.g., MintRoute [WTC03]) apply a table-driven scheme. If the sink and the sensor nodes are location-aware (using GPS or other localization methods [SHS01, CHH02, HHB$^+$03, RRP$^+$03]) and the sink announces its location to all nodes, building load-balanced routing paths online is possible [GZ04]. We refer to Chapter 5 for our implementation of a routing protocol that supports sink mobility.

3) How to implement the round routing (which is needed in Section 4.4.4)? The trace based forwarding, proposed in [NN03, YPK04], provides a way to shape a routing path into a predefined curve. In our case, the curve is simply an arc parameterized by a radius and the coordination of its center. The location-awareness is necessary for sensor nodes that apply such a routing, which brings extra overhead and could potentially offset the benefit of our joint strategy. Therefore, a careful design based on field testing is needed.

4) What if the network region is not circular? We can always apply our LP-based approach to obtain the optimal (or approximately optimal) solutions. If the size of a network is too large to be handled by the LP-based approach, we can still take the periphery mobility strategy, which can be conjectured as being (at least) nearly optimum, because it could be the best way to disperse the traffic flows. Note that this strategy also has a practical significance: certain applications (e.g., habitat monitoring [MPS$^+$02]) of sensor networks do prefer less human intervention in the inner part of the networks in order to reduce the disturbance effects.

5) How to adapt sink mobility if the information generation rate λ is time variant? This would require an **on-line** joint sink mobility and routing. Although extending our current proposals to an on-line mechanism is not trivial, certain principles described in this chapter can be helpful. In Section 5.5, we propose an algorithm for adaptive sink mobility, based on the formulation in Section 4.3.4.

4.6 Conclusion

In this chapter, we have investigated the problem of lifetime maximization under sink mobility. The problem was treated under two different models: a graph model and a continuum model.

Our investigations under the graph model jointly consider sink mobility and routing for lifetime maximization. We have formally proved the NP-hardness of maximizing network lifetime (MNL) problem involving multiple mobile sinks. We have then identified the sub-problem that has a potential to direct routing protocol designs in practice. In particular, we have developed an efficient algorithm to solve the MNL problem involving only a single mobile sink; we have further generalized the algorithm to approximate the general MNL problem. In addition, using the duality theory, we have proved that, theoretically, moving the sinks is always better than keeping them static. Finally, we have illustrated the benefit of using a mobile sink by applying our algorithm to a set of typical topological graphs.

For the continuum model, we have first shown that, with a static sink, the sensor nodes located close to it suffer premature battery depletion, leading to an early disconnection of the network. In order to better balance the load among the nodes, we have moved the sink in an arbitrary way and demonstrated that the traffic experienced by the most heavily loaded nodes is reduced by a factor of 3. In order to further exhibit the benefit of sink mobility, we have compared different mobility strategies and obtained the optimum one under the assumption of certain routing strategies. We have also explained how routing can be fine-tuned to leverage on the trace of sink mobility; in particular, we have shown how to better exploit the transmission capabilities of the nodes located at the periphery of the network. Our joint mobility and routing strategy achieves a 500% improvement of the network lifetime. For all these scenarios, we have developed an analytical model, corroborated by a set of simulations.

Chapter 5

MobiRoute: Routing Towards a Mobile Sink for Improving Lifetime

5.1 Introduction

In Chapter 4, we investigate the problem of maximizing network lifetime in wireless sensor networks (WSNs) by taking sink mobility into account. We theoretically prove that, compared with static sinks, mobile sinks can significantly improve the network lifetime. However, there is a common doubt in the research community that moving sinks is a practical approach. One of the major concerns behind this doubt is that mobility inevitably incurs extra overhead in data communication protocols and the overhead may potentially offset the benefit brought by mobility. The doubt is not answered in Chapter 4, because control overhead is not considered there. In this chapter, we deal with implementation issues related to mobile sinks in WSNs. With the routing protocol that we engineer to support sink mobility, we intend to show that the results obtained in Chapter 4 are valid under realistic circumstances.

Numerous routing protocols have been proposed in the last decade to support data communications in either mobile ad hoc networks (MANETs) or WSNs (in most cases with static nodes). While the protocols for MANETs (e.g., [JMH04, PBRD03]) are definitely an overkill for supporting mobile sinks because the basic assumption in MANETs is that every node moves in an unpredictable way, the protocols for WSNs usually take static sinks for granted (e.g., [IGE$^+$03, WTC03], although some exceptions exist [YLC$^+$05, KAK03, BUK04, KSJ$^+$04]). It is true that a routing protocol that supports mobile sinks to collect data in WSNs, compared with existing protocols for static WSNs, will have higher protocol complexity and overhead. Fortunately, the following favorable features of our sink mobility strategy and of the existing routing protocols help to limit the side effects of sink mobility:

- The mobility is controllable and thus predictable.

- The pause time of a sink along its moving trace is much longer than the actual moving time.
- Existing routing protocols usually possess some proactive features to cope with the dynamics in link quality; this can be exploited to support sink mobility.

In the spirit of Chapter 4, we further investigate in this chapter the performance, with respect to both lifetime and reliability (measured by packet delivery ratio), of WSNs with a mobile sink. We consider a scenario where nodes of a WSN periodically sample data and transfer these data through multi-hop routes towards the sink, while the sink intermittently changes its position according to certain predefined traces. We propose a routing protocol, MobiRoute, dedicated to support sink mobility. MobiRoute is developed based on MintRoute, a routing protocol already put into use (we refer to [TinyOS] for details). It takes into account the favorable features we mentioned above and thus only marginally increases the protocol complexity and overhead. By performing intensive simulations with this protocol, our investigations take into account realistic conditions such as control overhead with a routing protocol and collision/overhearing [YHE02, PHC04] at the MAC layer. We use TOSSIM [LLWC03] as the simulator. Our simulation results demonstrate the efficiency of MobiRoute, in terms of both an improved network lifetime and an undegraded reliability, in several deployment scenarios.

The rest of this chapter is organized as follows: Section 5.2 surveys related work. Section 5.3 clarifies the metrics and methodologies we apply. Section 5.4 presents our MobiRoute protocol. Section 5.5 describes the algorithm that control the sink mobility in an adaptive way. Simulation results are reported in Section 5.6. Finally, Section 5.7 concludes the chapter.

5.2 Related Work

There are a few proposals for data collection with mobile sinks [YLC$^+$05, KAK03, BUK04, KSJ$^+$04]. While most of these proposals [YLC$^+$05, KAK03, BUK04] consider sink mobility as an inherent behavior of WSNs and try to cope with it, only Kansal et al. [KSJ$^+$04] share the same opinion of exploiting controllable sink mobility as ours.

The *two-tier data dissemination* (TTDD) approach presented in [YLC$^+$05] is actually based on the existing idea of virtual backbone. The backbone (or grid in TTDD's terminology) is formed proactively upon detection of a stimulus. The mobile sinks send queries to the nearest grid points with a flooding. Queries are routed along the grid and data trace the reverse path back to the sinks. As a consequence, the control overhead introduced by sink mobility is limited to the grid cell where a sink is located. Aiming at further limiting the widespread diffusion of control messages introduced by sink mobility, the *scalable energy-efficient asynchronous dissemination* (SEAD) protocol [KAK03] assigns particular nodes as the *access node*s and relies on such fixed anchors to limit the control traffic. Mobile sinks only need to select one of their neighbors as an access node and maintain the link with it. SEAR constructs an energy-efficient dissemination tree from a source to difference access nodes. Data are routed along the tree and then unicast from the access nodes to their sinks.

To handle mobility, a sink may handoff from one access node to another if the tradeoff between the energy consumed to build another tree and the data delivery latency exceeds a given threshold. While TTDD and SEAD both heavily rely on the assumption of location-aware sensor nodes and are thus not suitable in our case, the data generation model is also different from ours (see our assumption [k] in Chapter 4). Under our data generation model (where every node produces data with the same rate), the virtual backbone used in TTDD to carry most traffic would become fixed and the delayed handoff in SEAD could lead to suboptimal routing trees for a substantial amount of time; they both offset the load balancing effect resulted from sink mobility.

The *hybrid learning-enforced time domain routing* (HLETDR) proposed in [BUK04] comes a bit closer to our situation. They assume a somewhat fixed sink trace and the same data generation model as ours. However, instead of requiring the sink to inform other nodes about its location changes, HLETDR lets sensor nodes figure out the sink trace through a learning-based approach. This learning process is done through positive and negative reinforcements according to the probability density function indicating how far the sink is from the nodes close to the sink trace. A tour period of the sink is divided into m domains. When a node has data to be forwarded, the probability of selecting the next node is determined according to its possibility being on a shortest path to the sink in that time domain. HLETDR has a high complexity of the routing table, i.e., $O(m)$, when a fine time granularity is required. In addition, the relatively slow learning procedure limits the adaptability of the sink mobility (which is very crucial for load balancing in our case).

The routing protocol described in [KSJ$^+$04] is based on *directed diffusion* [IGE$^+$03], a routing protocol dedicated to *data-centric*[1] communications. Kansal et al. [KSJ$^+$04] extend directed diffusion by adding mainly three components: 1) a pre-move phase for the nodes to learn the sink trace, 2) an acknowledgement-retransmit scheme to handle packet loss during handoff, and 3) a pre-fetch mechanism to improve the data delivery. The data generation model assumed in [KSJ$^+$04] is quite different from ours: nodes send data only when the data are queried whereas nodes proactively push data to the sink in our model. In addition, the sink moves continuously in [KSJ$^+$04] because their protocols work in the fast mobility regime (see Section 4.2 for details). Therefore, the control objective is the moving speed for [KSJ$^+$04], whereas we consider the adjustment of pause times. In this chapter, we take an approach similar to that of [KSJ$^+$04] in that, instead of developing a routing protocol from scratch, we extend an existing routing protocol with the ability of handling sink mobility.

5.3 Problem, Metrics and Methodology

We again take the definition that we make in Section 4.3.2 for network lifetime, i.e., the time period for the first node to run out of its energy reserve. When evaluating this quantity, we convert the problem of maximizing network lifetime to a min-max problem in terms of the **radio** energy

---

[1] Data generated by sensor nodes are named by attribute-value pairs. A node requests data by sending interests for named data rather than for named nodes.

consumption of individual nodes. Another performance index we want to evaluate is the packet delivery ratio (or *reliability*). In fact, a possible side-effect brought by sink mobility could be an increase in packet loss due to occasional topology changes; the lifetime elongation resulting from sink mobility is justifiable only if the increase in packet loss is tolerable.

We assume that nodes generate data and send them to the sink with the same rate. In our approach, the mobility pattern of a sink takes a *discrete* form: the moving trace consists of several *anchor point*s between which the sink moves and at which the sink pauses. We require each *epoch* (the time during which the sink pauses) to be much longer than the moving time, such that the routing overhead introduced by sink mobility becomes negligible due to its amortization across a long epoch. Imposing these anchor points simplifies the design of the mobile sink[2] and limits the extra overhead introduced to the routing protocol (see Section 5.4 for details). In addition, a continuous movement is not necessary, as a granularity of (sink) displacement smaller than the magnitude of the effective radio range may not lead to any topological change (whereas topological changes are what we expect from the sink mobility). In order to better adapt to the topology and dynamics of a given network, we also intend to control the sink mobility *on-line* (based on the *off-line* optimization described in Chapter 4).

Our experiment methodology involves simulations with TOSSIM [LLWC03]. The main benefit of using the TOSSIM simulator is that the protocol used for simulations can be directly adopted by real sensor nodes. We simulate a set of networks with nodes on 4×4, 5×5, and 7×7 point lattices; these scenarios represent outdoor WSNs in general. We also simulate a network that we intend to deploy as an in-building testbed.

5.4 MobiRoute: Routing Data Towards a Mobile Sink

According to the definition of the discrete mobility pattern described in Section 5.3, the sink changes its location from time to time. A routing protocol that transfers data towards such a sink should perform the following operations that are not needed for traditional WSNs:

1. Notify a node when its link with the sink gets broken due to mobility.
2. Inform the whole network of the topological changes incurred by mobility.
3. Minimize the packet loss during the sink moving period.

Operation 1 seems to be encompassed by 2, but the level of urgency is different. Packets forwarded by a last-hop node will get lost if the node does not detect the link breakage, while a remote node can still send its data to the sink successfully without knowing the topological changes. However, the routing optimality is compromised without operation 2. It is not possible to avoid packet loss, because a realistic failure detector (which usually relies on a timer) always has some delay.

---

[2] The mobile sink can simply be a laptop (moved occasionally by a human), rather than a sophisticated robot as used in [KSJ+04].

Therefore, the goal of operation 3 is to minimize rather than eliminate packet loss. Possible scenarios related to these operations are illustrated in Fig. 5.1.

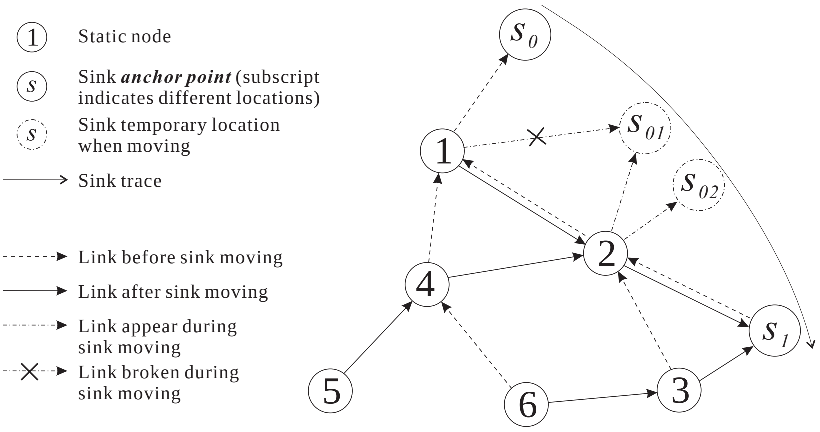

Static node
Sink *anchor point* (subscript indicates different locations)
Sink temporary location when moving
Sink trace
Link before sink moving
Link after sink moving
Link appear during sink moving
Link broken during sink moving
s_0
s_{01}
s_{02}
s_1
1
2
3
4
5
6

Figure 5.1: This example illustrates possible scenarios where additional operations are necessary. Assuming the sink, after its (long) pause at s_0, moves to s_1, (1) the link breakage happening when the sink reaches intermediate location s_{01} (where it loses connectivity with node 1) should be notified to node 1, otherwise the node will have to drop packets sent from other nodes, (2) nodes 3, 4, and 6 should be informed about the topological changes at a proper time, otherwise, for example, 6 might take the following sub-optimal routing path: $6 \rightarrow 4 \rightarrow 1 \rightarrow 2 \rightarrow s$.

Our routing protocol, MobiRoute, is a superset of the Berkeley MintRoute [WTC03]. MobiRoute extends MintRoute by adding functions that perform the aforementioned operations. We first introduce MintRoute briefly in Section 5.4.1, then we describe the extended functions of MobiRoute separately in Section 5.4.2 – 5.4.4. The state diagram shown in Fig. 5.2 is used when we present MobiRoute.

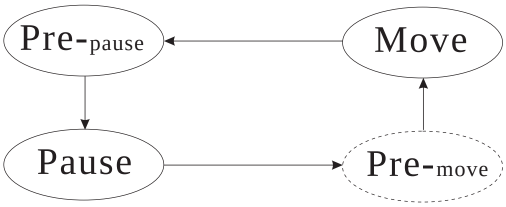


Figure 5.2: States and transitions involved in MobiRoute. Note that only the protocol running at the sink side has the *pre-move* state.

5.4.1 MintRoute

Berkeley MintRoute [WTC03] is a routing protocol designed specifically for the all-to-one data transmission style of WSNs. It takes a distributed distance-vector based approach: route messages (i.e., control packets) are exchanged periodically among neighbor nodes, and the next hop nodes (or *parent*s in MintRoute nomenclature) are chosen by evaluating the costs of routing data through different neighbors. The exchanged route messages not only help to measure the distance (in terms of the number of possible transmissions) from the sink but also provide a way to evaluate the link qualities (from both directions) between nodes. As a result, MintRoute applies a **M**inimum **T**ransmission (MT) metric, where the goal is to minimize the total number of transmissions (including retransmissions). Since the data rate in WSNs is low, route messages do not need to be exchanged frequently (the rate is actually a multiple of the data rate in MintRoute). This helps MintRoute to greatly reduce its energy consumption. Although MintRoute does not explicitly apply a metric that considers load balancing, the protocol, according to our experience, does balance the traffic load with occasional switches of nodes' parents (which is a direct consequence of the MT metric). This feature makes MintRoute a leading candidate for supporting a mobile sink. Finally, we note that, in order to detect packet loss and thus evaluate link quality, MintRoute applies a sequence number for each packet; this sequence is shared by both control and data packets.

5.4.2 Detecting Link Breakage

In order to inform the nodes located close to the sink trace about the state of their links with the sink, MobiRoute applies a beacon mechanism. The sink, during the entire moving period, periodically broadcasts a beacon message (*s-beacon* hereafter). A node, upon receiving a s-beacon, sets (or resets) its **detecting** timer. If the timer times out before receiving the next s-beacon, the failure detector at this node indicates a link breakage and a new parent is chosen. We now discuss several crucial points of this seemingly simple mechanism.

First, we require the sink to transit from the **pause** state to the **pre-move** state before physically beginning to move. The sink begins to broadcast s-beacons under the pre-move state and evolves to the **move** state after a while. The sink moves while broadcasting s-beacons under the move state. A node, after receiving the first s-beacon under its current **pause** state, transits to the **move** state directly. The importance of the pre-move state can be straightforwardly seen: it guarantees the reception of s-beacons at the nodes' side before the link quality changes due to the sink mobility.

Secondly, although only the sink (who is assumed to have sufficient energy reserve) spends energy to send s-beacons, nodes also spend energy to receive these beacons. Therefore, the frequency of s-beacons should not be too high. Nevertheless, low frequency sending retards failure detection, which in turn increases packet loss. We apply a simple heuristic: the frequency is set in the same order as the accumulative packet sending rate. For example, if the sending rate of each node is 1 packet/minute and there are 60 nodes in the network, the accumulative rate at a last-hop node is at most 1 packet/second, and the beacon frequency is set to 1Hz. A related parameter is the timeout value for the detecting timer. Fortunately, the value can be relatively small, because a node will detect a false-positive when receiving another s-beacon.

Finally, the beacon mechanism is a costly procedure, regardless of which beacon frequency is chosen. Fortunately, since the moving period accounts only for a small fraction of the network lifetime, its costs will be amortized across the lifetime. A continuous sink movement, on the contrary, would incur such costs permanently.

5.4.3 Conveying Topological Changes

MobiRoute could have relied on MintRoute to propagate the topological changes resulting from sink mobility. However, the rate of route message exchanges in MintRoute is very low. Therefore, it takes a long time to convey the topological changes to the whole network; during this period, many packets are routed through sub-optimal paths, which consumes additional energy and thus offsets the benefit of sink mobility. As a result, MobiRoute needs a speed-up (route message exchange) rate for propagating the topological changes.

Propagating information throughout a network is a costly procedure (message complexity $O(n)$); it cannot be performed frequently. So MobiRoute only performs a propagation upon the sink reaching an anchor, and it tolerates a limited number of sub-optimal routing during the moving period. The sink enters the **pre-pause** state (see Fig. 5.2) when it stops moving; it then sends route messages with a speed-up rate, which causes their receivers to enter the same state. Nodes that receive messages **directly** from the sink also send speed-up route messages; they re-evaluate the quality of their links with the sink using these exchanges. A node receiving speed-up messages **indirectly** also enters the pre-pause state; it forwards the message only if its distance towards the sink changes significantly (e.g., node 6 in Fig. 5.1 might not forward messages received from node 3). The energy consumption of the propagation procedure is effectively reduced, because there are nodes that are not affected for a given move of the sink. Every node (including the sink) in the pre-pause state transits to the **pause** state after a short time-span controlled by a timer.

5.4.4 Minimizing Packet Losses

Although packet loss cannot be avoided during the sink moving period due to the lag of the failure detector, there are ways to mitigate the losses. Taking advantage of having a very short moving period (which we would not have if the sink moved continuously), the protocol tries to reduce the sending rate of the last-hop nodes, by asking them to buffer data packets using the interface queue (`QueuedSend` module) in MintRoute. We also add the following command to `QueueControl` interface, such that the routing module can access the interface queue to change the next-hop address of the buffered packets upon detecting a link failure.

Nodes can only buffer data packets; control packets should still be sent. However, if we simply picked up control packets from the interface queue and sent them, there would be gaps among the sequence numbers (remember that MintRoute applies the same sequence for both control and data

```
command void QueueControl.setAddrInQueue(uint16_t parent) {
    uint16_t i;
    if (!fQueueIdle)
        for (i = dequeue_next; i != enqueue_next;
             i = (i + 1) % MESSAGE_QUEUE_SIZE)
            msgqueue[i].address = parent;
}
```

packets). These gaps would mislead a neighbor node about a degradation in the link quality. Two solutions can be applied: 1) using separate sequences and queues for data and control packets or 2) changing the sequence number within the queue, such that control packets sent have consecutive sequence numbers. In the short term, we adopt the second solution because it is easy to implement, but the first could be desirable in a long-run perspective.

5.5 Algorithm for Adaptively Controlled Mobility

According to our simulation results in Section 5.6, a mobility strategy that adapts to the network topology (for which no a priori knowledge exists) performs better than a static schedule. In this section, we describe the adaptive algorithm to control sink mobility. Our algorithm adaptively changes the epoch of the sink at each anchor point, according to the power consumption profile of the network. We derive the algorithm from the following linear program:

$$\text{Maximize} \quad \text{lifetime } T = \sum_k t_k \tag{5.1}$$

$$\text{Constraints}: \quad \sum_k t_k \mathbb{P}_k \leq \mathbb{E} \tag{5.2}$$

where $\mathbb{P}_k$ and $\mathbb{E}$ are vectors that represent the power consumptions of each node (referred to as *P-profile* hereafter) when the sink pauses at a certain anchor point k and the initial energy reserves of all nodes, respectively. This formulation basically means that we weigh, through the epoch t_k, the anchor points based on the corresponding P-profile $\mathbb{P}_k$s, in such a way that the $\mathbb{P}_k$s that complement each other are favored. It is not difficult to see that we are actually applying the second relaxation of the joint sink mobility and routing problem described in Section 4.3.4.

In practice, we propose the following 2-phase algorithm to approximate the above programming problem:

- **Phase I–Initialization**: The mobile sink visits the anchor points one by one and pauses at each point for a short *sampling period*. During each sampling period, the sink collects the power consumption records from all nodes and builds a P-profile for that anchor point. At the end of this phase, the sink performs the programming (5.1) and drops an anchor point if its weight T_i is extremely low. It is not worth keeping such a point because its corresponding epoch is not long enough to amortize the routing overhead introduced by the sink mobility.

- **Phase II–Operation**: The mobile sink goes through the trajectory repetitively but only pauses at those chosen anchor points. At a given point k, the sink again collects power consumption information and builds a profile $\mathbb{P}_k$. Based on the new profile and previous profiles for other chosen points, the programming (5.1) is re-solved to deduce t_k. The actual epoch is computed as $\hat{t}_k = t_k/\delta$, where $\delta > 1$ is an integer. Applying the δ makes it possible for the sink to repeat the movement pattern several turns, which allows the sink to be more adaptive to the network dynamics.

We have the following remarks on the algorithm:

- If we make a discrete search over the whole surface covered by the network to obtain those anchor points, the time to finish Phase I could become comparable to the network lifetime; the algorithm would thus lose its adaptability (e.g., the sink might not even get a chance to enter Phase II). Alternatively, we can search over a “good” trace. A candidate of such a trace could be the periphery of the network, according to the theory presented in Chapter 4 .
- The sink could have directly applied the results (i.e., t_ks) of the first phase to the second phase if the routing topology were fixed. However, according to our experiences with real WSNs, the routing topology keeps evolving even though the nodes are static. As a result, the P-profiles obtained from the first phase can only be considered as estimations and should be updated if new profiles are available.

5.6 Simulations

We report two sets of simulations in this section. In one set, network nodes are located on point lattices; simulation results of this set represent outdoor WSNs in general. In another set, nodes form a ring; the simulations emulate our future field tests with an in-building WSN. All simulations are performed in TOSSIM.

5.6.1 Grid Networks

We arrange nodes on a point lattice of size 4×4, 5×5, and 7×7. For each network, we either 1) put the sink (node 0) at the network border (the midpoint of one side), or 2) at the center, or 3) let the sink move around the network periphery. There is a constant distance between any two consecutive anchor points; the sink pauses on an anchor point and moves in between two anchors according to the instruction from a Tython [DL] code. The connectivity[3] of a node with other nodes is shown in Fig. 5.3. The transmission range is set to 1.2 times longer than the distance between

---

[3] We make use of the *fixed radius* model, although it is less realistic than the *empirical* one (we refer to [LLWC03] for the definitions of these models). The reason is that, given a set of geo-distributed nodes, applying the empirical

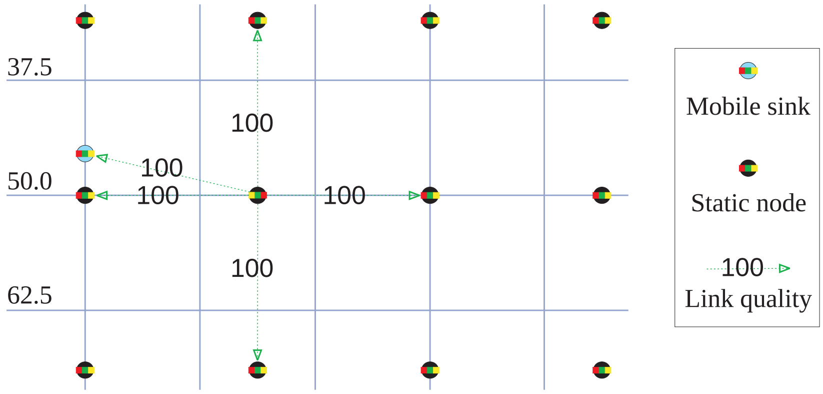


Figure 5.3: Neighborhood graph in TinyViz [LLWC03]. The number beside a link states the link quality: 100 stands for a perfect link. The numbers on grid lines represent coordinates.

two neighbor nodes. Each node generates a data packet every 60 seconds. A control packet (route message) is sent every 120 seconds in the pause state and every 2 seconds (speed-up rate) in the pre-pause state. The s-beacon rate is one per second. The retransmission is disabled for all nodes if not stated otherwise. The epoch of non-adaptive mobility allows each node to send 10 data packets (i.e., 600 seconds)[4], and the moving time is 10 seconds for the 49 nodes network and 20 seconds for the other two. The sink moves at a speed of 1 ft/s in the move state. The full simulation time is just long enough to let the sink go through one round of its trip; the simulation for a given network is repeated 10 times. For the measurement of energy consumptions[5], we use the number of (both control and data) packets that a node is involved in to characterize the energy consumption. By doing this, we implicitly assume that 1) radio communication is the dominating energy consumer, 2) sending and receiving a packet consumes the same amount of energy, and 3) control and data packets are of the same size.

---

model usually leads to small network diameter due to the occasional existence of *shortcuts*. A relatively large network diameter (up to 10 hops) is essential to fully exhibit the benefit of using a mobile sink, but increasing the network size to achieve larger diameter results in a simulation time of unreasonable duration (e.g., 100 hours). By using the fixed radius model, we simply assume that, for a certain node, only nodes within its *effective region* [WTC03] are considered as its neighbors.

[4] This duration is much shorter than what it could be in a real deployment, where it might last for days or even weeks. Therefore, the performance of MobiRoute is expected to be better in practice, thanks to a longer amortization period.

[5] Since TOSSIM uses a MAC that never switches off its radio, tools such as Power-TOSSIM [SHC$^+$04] always report a flat energy consumption pattern of a network no matter where the sink is located. In reality, motes equipped with B-MAC [PHC04] do switch off their radio when there is no transmission going on.

Non-Adaptive Mobility

The spatial distributions of energy consumptions for networks with 25 and 49 nodes are shown in Fig. 5.4. According to the lifetime definition in Section 5.3, the smaller the maximum energy consumption in a network, the longer the network lifetime will be. By comparing the two cases with a static sink and the case with a mobile sink, we make the following observations:

- The load-balancing effect of using a mobile sink is evident. The network with a mobile sink always lives longer than the network with a static sink at its border and no shorter than the network with a static sink at the center.
- In the network of 49 nodes, using a mobile sink is the best choice, irrespective of whether the overhearing at the MAC layer exists or not. However, overhearing does offset the benefits of using a mobile sink: the 100% improvement on the lifetime (comparing the network having a mobile sink with the one having a centered static center) is reduced to 50% if overhearing exists.
- In smaller networks of 16 and 25 nodes (only the latter case is shown in Fig. 5.4 due to their similarity), using a mobile sink is not necessarily helpful, because it does not improve the lifetime compared with using a centered static sink while increasing the accumulative energy consumption of the network.

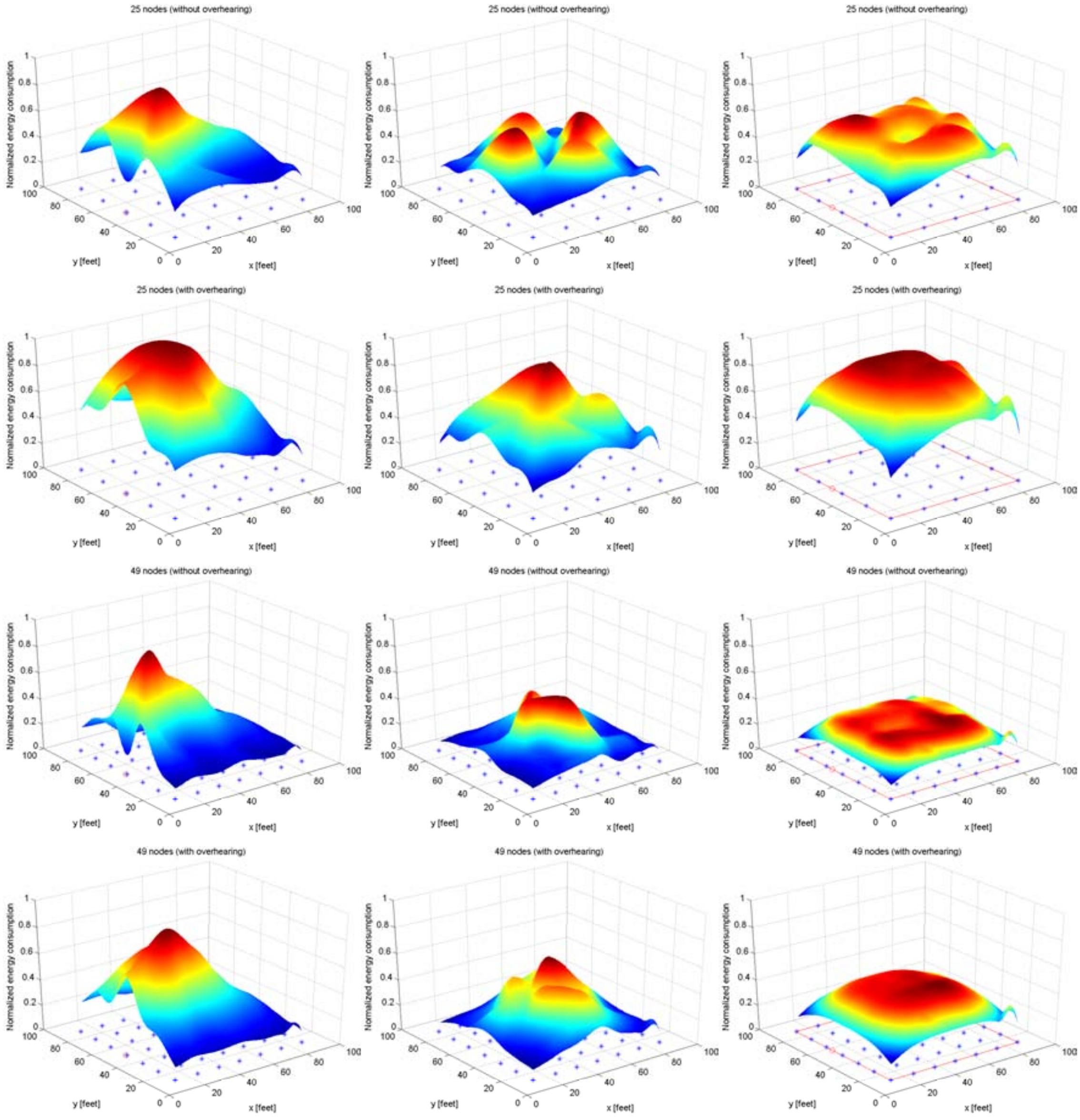
A straightforward conclusion is that using a mobile sink is more beneficial in large networks. Since the function of the mobile sink is to disperse the traffic flows, the network should be large enough to provide nodes with a sufficient number of alternative routing paths. However, since locating a sink at the network center is not always practical[6], using a mobile sink does help to improve the lifetime in most networks.

Another implication of our observations is that a MAC protocol free of overhearing is very important to improve the effectiveness of using a mobile sink. Unfortunately, the current MAC of motes (i.e., B-MAC [PHC04]) suffers much from overhearing [SMP$^+$04], and protocols with the potential to avoid overhearing (e.g., S-MAC [YLC$^+$05]) do not necessarily have an overall performance better than B-MAC due to their burdensome synchronization schemes. So, we expect future technology to provide sensor nodes with overhearing-free MACs.

We plot the cumulative distribution functions of the packet delivery ratio in these two networks in Fig. 5.5. The comparisons are only made between a centered static sink and a mobile sink, because the ratios are quite similar for both networks with a static sink. The figures show that, without retransmission, the packet delivery ratio is always lower in the case of a mobile sink, which is intuitive (see the reasons that we described in Section 5.4). Also, the difference between the two ratios increases with the network size. The reason is that using a mobile sink increases the worst-case

---

[6] For habitat and environment monitoring, unobtrusive observation is key for studying natural phenomena [SMP$^+$04]. Although nodes are small enough for this purpose, a sink (especially when it has to transmit the collected data out of the network area) can hardly makes itself invisible in the environment.



Static sink at the border Static sink at the center Mobile sink

Figure 5.4: Energy consumption of WSNs. Two networks with 25 and 49 nodes, respectively, are simulated. For each network, we either put the sink at the network border (the midpoint of one side), or at the center, or let the sink move around the network periphery. For each comparative case (i.e., one row in the figure), the energy consumptions are normalized to a common scale factor.

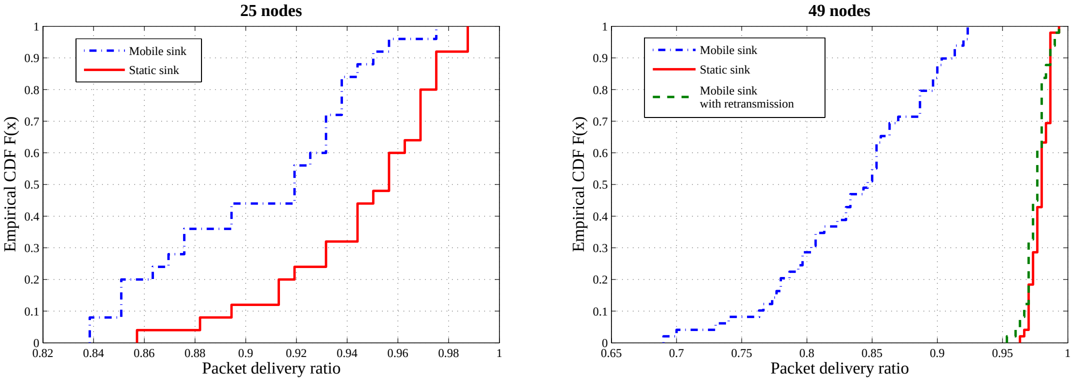


Figure 5.5: Comparisons of packet delivery ratio

routing path length (actually, a static sink located at one vertex of the network periphery achieves the same ratio). This is not a major problem, because we would expect a much higher reliability

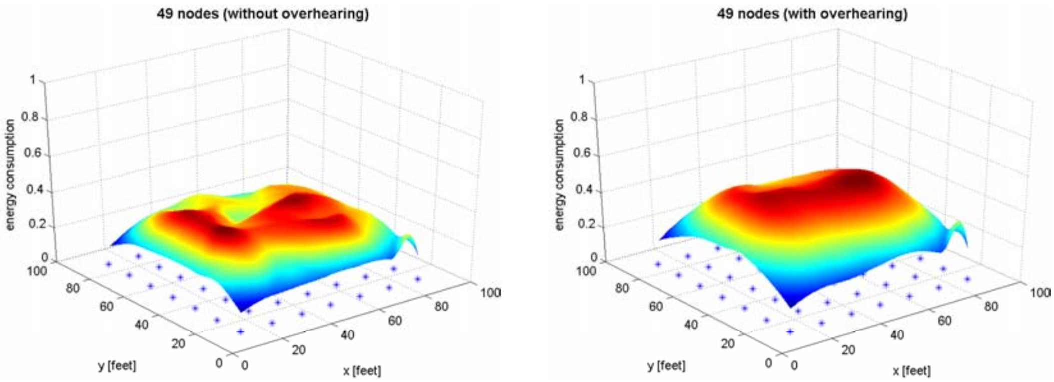
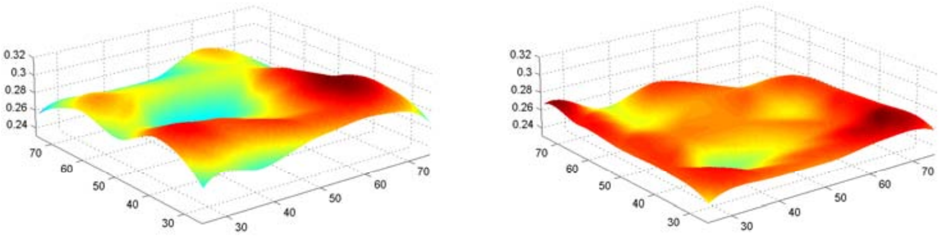


Figure 5.6: Energy consumption of a WSN with a mobile sink and retransmission enabled. The scale factors take the same value as used for Fig. 5.4.

in reality, where a node typically sends data only every tens of minutes [SMP$^+$04]. Actually, if we enable the retransmission, the packet delivery ratio in the case of a mobile sink can be as high as that in the case of a static sink, but at the cost of increased energy consumption (Fig. 5.6), whose maximum value is still low enough to justify the benefit of using a mobile sink.

Adaptive Mobility

Zooming into the spatial distribution of energy consumption in the network with a mobile sink (as shown in Fig. 5.7 (a)), we observe that the load taken by nodes near the corner is heavier than that of other nodes. Applying the algorithm described in Section 5.5, we actually find that the sink should pause less time at those anchors near the corner. The resulting load, shown in Fig. 5.7 (b), is further balanced; which improves the network lifetime by about 10%. Note that the sink, in our simulations, only circles around the network twice: one in phase I and another in phase II (see Section 5.5); the network lifetime can be further improved with more rounds in phase II.



(a) Non-adaptive mobility (b) Adaptive mobility

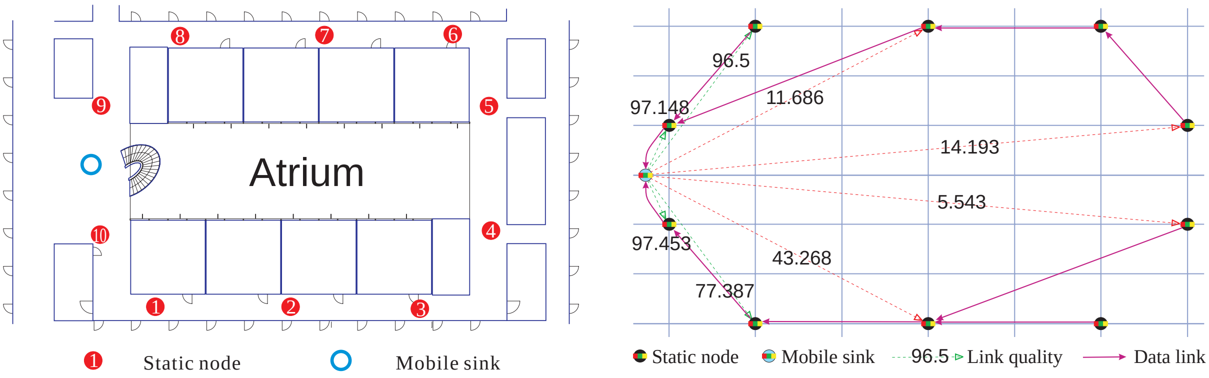
Figure 5.7: Zooming in the distribution of energy consumption with a mobile sink.

5.6.2 Ring Network

This section presents the simulation with a ring network. We use this simulation scenario to emulate a network deployed in our building, as shown in Fig. 5.8 (a). While a static sink[7] is located in-between nodes 9 and 10, a mobile sink moves around the circle and pauses in between two consecutive nodes. We use the *empirical* model [LLWC03] to characterize the connectivity in this set of simulations. As an example, the connectivity graph for the sink (node 0) is shown in Fig. 5.3 (b). Each node generates a data packet every 30 seconds. A control packet (route message) is sent every 60 seconds in the pause state and every 2 seconds (speed-up rate) in the pre-pause state. The s-beacon rate is one per second. The retransmission is disabled for all nodes. Each of the 10 simulations lasts for 17600 seconds and the epoch of non-adaptive mobility is 1760 seconds. The sink moves at a speed of 1 ft/s in the move state, and the moving time is 25 seconds. The

---

[7] The atrium inside of our building prevents us from locating the sink at its optimum position (i.e. the center of the network). This indeed corroborates our claim in Section 5.6.1 that locating a sink at the network center is not always practical.



(a) The experimental network

(b) Simulation scenario

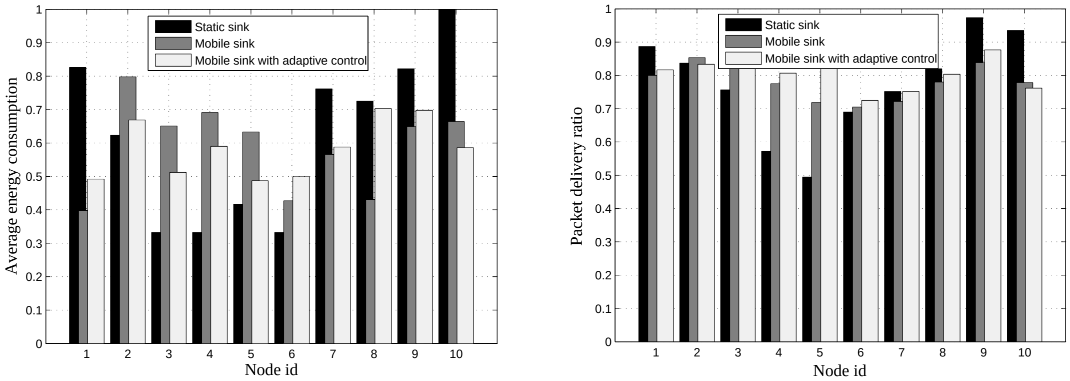
Figure 5.8: The plan of our network deployment (a) and the simulation scenario (b). Nodes are numbered the same way in (b) as in (a).

measurement of energy consumptions is the same as for Section 5.6.1, and the overhearing is not taken into account.

We illustrate the simulation results with bar graphs in Fig. 5.9. As shown in Fig. 5.9 (a), the load balancing effect is already very evident by simply moving an uncontrolled sink, which improves the lifetime by 20%. Further improvement is achieved (an additional 15% of improvement on lifetime compared to the non-adaptive mobility) by controlling the mobile sink adaptively. The behavior in packet delivery, plotted in Fig. 5.9 (b), differs from that shown in Fig. 5.5; the averaging effect also arises due to the special network topology. In this specific scenario, the averaging effect makes a mobile sink beneficial not only to the network lifetime but also to the reliability, because nodes that are far away from the static sink perform poorly in terms of the reliability of packet delivery.

5.7 Conclusion

In this chapter, we have presented a routing protocol, MobiRoute, to support wireless sensor networks (WSNs) with a mobile sink. This is a follow-up of chapter 4 where we theoretically proved that moving the sink can improve network lifetime without sacrificing data delivery latency. By intensively simulating MobiRoute with TOSSIM (in which real implementation codes are running), we have demonstrated the benefit of using a mobile sink rather than a static one. We have simulated both general networks with nodes located in point lattices and a special in-building network with nodes forming a ring. The results are very promising: a mobile sink, in most cases, improves the



(a) (b)

Figure 5.9: Simulation results. (a) The energy consumptions are normalized by the largest energy consumption observed (i.e., node 10 in the case of a static sink). (b) The averaging effect arises also for the packet deliver ratio.

network lifetime with only a modestly degraded reliability in packet delivery.

Conclusion

> *Things which are put together are both whole and not whole, brought together and taken apart, in harmony and out of harmony; one thing arises from all things, and all things arise from one thing.*
>
> Heraclitus (Quote in Aristotle, *On the World*)

> *As a single, unified thing there exists in us both life and death, waking and sleeping, youth and old age, because the former things having changed are now the latter, and when those latter things change, they become the former.*
>
> Heraclitus (Quote in pseudo-Plutarch, *Consolation to Apollo*)

With the flourishing evolution of mobile computing (on the side of applications) and wireless networking (on the side of enabling technologies), mobility is bound to become a component of the network design and control. In this thesis, we bring out our insights on the issues of **tackling** and **exploiting** mobility in the process of networking protocol design. Although we mention tackling and exploiting as two aspects of handling mobility, it does not necessarily mean that they are mutually exclusive; they are actually the **unity of opposites**. On one hand, when coping with the dynamics introduced by node mobility, our probabilistic protocols are actually taking advantage of the underlying randomness resulting from node mobility (see Chapters 2 and 3). On the other hand, controllable mobility does have the potential to improve network performance (see Chapter 4), but we have to confront the side effects brought by mobility (see Chapter 5).

According to our experience, introducing mobility anyway affects the performance of networking protocols. As a result, the way to face mobility is always to identify certain **tradeoff**s and corresponding application scenarios where these tradeoffs are **meaningful**. Here a tradeoff implies a compromise of certain performance aspect(s) in order to improve the other(s), and by meaningful we mean that those enhanced aspects are really important and those sacrificed aspects do have minor impacts in the given application scenario. In addition, a controllable (or tunable) tradeoff would be more useful because a minor aspect at one time might become major at another time and vice versa. The above principles are fully carried out in Chapters 2 and 3, where reliable protocols are developed for mobile ad hoc networks.

In Chapter 2, we have explored the tradeoff between protocol efficiency and reliability for wireless group communications. It is a common observation that efficiency and reliability are at odds because a best-effort (or opportunistic) protocol (e.g., UDP) is much lighter weight than a protocol that provides certain guarantees (e.g., TCP). The group communication protocols (which we name PILOT) proposed in this chapter take a step forward: by involving probabilistic features in a protocol, we are able to fine tune the tradeoff between two extremes. The protocols we have developed include an underlying many-to-many multicast protocol, a multicast streaming service and a data storage service. They are all equipped with probabilistic features and thus have the ability of trading efficiency for reliability or the other way around. We have performed detailed analysis and simulations to verify the tunability of PILOT.

In Chapter 3, the tradeoff is again between protocol efficiency and reliability, for a certification service this time. However, in terms of a security protocol, the reliability can be the security protection itself, which should not be sacrificed; otherwise the protocol is not secure anymore. Therefore, the protocols we have proposed for supporting a distributed certification service (DICTATE) trades the "freshness" of certificates (sent to clients in response to their queries) for efficiency. Sacrificing freshness does not immediately compromise the protocol security; it only happens in very rare cases that an attacker is able to exploit a "stale" certificate. Since DICTATE uses the service provided by PILOT, it naturally has the ability of fine tuning the tradeoff between protocol efficiency and certificate freshness. The security of DICTATE as well as the validity of its tunability are both verified by security analysis and simulations.

The common way of exploiting node mobility to improving protocol performance lies in the fact that mobility increases the **dimension** (thus the degree of freedom) of a problem. This follows the principle that optimizing an objective in a high-dimension space always leads to a result no worse than what can be achieved in a space with reduced dimension. However, mobility does come with a cost and solving problems in high-dimension space becomes much more difficult; so if the improvement obtained by applying mobility is not overwhelming, there is no point in exploiting the mobility.

In Chapter 4, we have focused on the theoretical analysis on the problem of exploiting sink mobility to improve the lifetime of wireless sensor networks. We have investigated the problem under both a graph model and a continuum model. Although the problem, in its general form, has a provable hardness under the graph model, we have shown that there exists an efficient algorithm to approximate the optimal solution. Further insight on the problem has been obtained by the analysis under a continuum model; the analysis depicts the shape of the moving trace that a mobile sink should take. Both analytical and simulation results show that the benefits (in terms of prolonged lifetime) brought by moving sinks are very significant. Whereas this chapter basically says that solving the lifetime optimization problem that involves sink mobility is not prohibitively harder (though with higher complexity) than solving traditional problems without sink mobility, the cost incurred by sink mobility has not been taken into account, which naturally leads to the last chapter.

In Chapter 5, we have further demonstrated the benefit of sink mobility by taking many related practical issues into consideration. We have devised a protocol, MobiRoute, to support sink mobility and run it in a simulation environment, TOSSIM, to test the performance in terms of both network lifetime and data delivery reliability. Although there are many proposals for supporting node mobility in mobile ad hoc networks, MobiRoute is distinct because it needs only to work with the mobility of sinks (a minority in a sensor network) in a network with a large number of static nodes. By taking this peculiarity into account, MobiRoute is able to limit the side effects (e.g., packet loss and protocol overhead) resulting from sink mobility to their least scale. The simulation results with MobiRoute on various network deployment scenarios show that the network lifetime is indeed improved by sink mobility in most cases.

Summary of Contributions

We summarize the original contributions of this thesis in the following:

Tackling Mobility with Probabilistic Protocols

- an ad hoc adapted gossip mechanism to perform probabilistic reliable multicasting,
- quorum systems applying gossip-based quorum access protocols and an asymmetric quorum construction for reliable data storage in ad hoc networks,
- a distributed certification service for supporting public-key cryptography in ad hoc networks,
- a new concept of probabilistic freshness concerned with a public-key certificate, and
- a mechanism for fine tuning the tradeoff between reliability/freshness and protocol overhead, along with a detailed analytical and simulation study.

Exploiting Mobility for Improving Network Lifetime

- a constructive proof of the NP-hardness of maximizing network lifetime (MNL) problem involving multiple mobile sinks,
- an efficient algorithm for the sub-problem involving a single sink, which can be generalized to approximate the general MNL problem,
- a formal proof on the superiority of moving the sinks over keeping them static,
- a closed-form expression of the optimal sink moving trace under a continuum model, and
- a routing protocol dedicated to support sink mobility.

Future Research Directions

The probabilistic protocols we have proposed in Chapters 2 and 3 work fine only if the underlying randomness (e.g., node distribution or traffic pattern) exists; we are actually relying on node mobility to provide this randomness. In reality, we have to adapt our protocols to the cases where the underlying randomness is not perfect, for example, nodes move only according to specific patterns.

Regarding PILOT, we are considering the possibility of improving the analytical model by taking more realistic conditions into account. In addition, we are also considering other models [KMG03, ZH99] in order to further understand the benefits of gossip-based protocols and to provide numerical comparisons between PILOT and similar systems for ad hoc networks, which would better justify the deployment of PILOT.

As for DICTATE, we need to further study its performance with a complete implementation. It is also a part of our future work to consider the integration of DICTATE with potential applications such as secure routing and secure group communications.

In terms of improving lifetime with sink mobility, we are in the process of performing full-scale field tests with the in-building network. We will also improve MobiRoute based on the experience obtained from our field tests. On the theoretical side, we intend to study how data aggregation protocols can be efficiently integrated into the proposed scheme.

More generally, an intriguing research area would consist in further exploiting the node and sink mobility for optimizing various aspects of wireless sensor networks.

Bibliography

[ADS00] Y. Amir, C. Danilov, and J. Stanton. A low latency, loss tolerant architecture and protocol for wide area group communication. In *Proc. of DSN*, pages 327 –336, 2000.

[AG00] N. Asokan and Philip Ginzboorg. Key-agreement in ad-hoc networks. *Computer Communications*, 23(17):1627–1637, 2000.

[AGK$^+$04] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local Search Heuristics for k-Median and Facility Location Problems. *SIAM J. on Computing*, 33(3):544–562, 2004.

[ASSC02] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless Sensor Networks: A Survey. *Elsevier Computer Networks Journal*, 38(4):393–422, 2002.

[AW00] L.-G. Alberto and I. Widjaja. *Communications Networks.* McGraw Hill Higher Education, 2000.

[BB02] S. Buchegger and J.-Y. Le Boudec. Performance Analysis of the CONFIDANT Protocol (Cooperation Of Nodes - Fairness In Dynamic Ad-hoc NeTworks). In *Proc. of ACM MobiHoc'02*, 2002.

[BC02] M. Bhardwaj and A.P. Chandrakasan. Bounding the Lifetime of Sensor Networks Via Optimal Role Assignments. In *Proc. of the 21st IEEE INFOCOM*, 2002.

[BF01] D. Boneh and M. Franklin. Identity-based Encryption from the Weil Pairing. In J. Kilian, editor, *Lecture Notes in Computer Science*, volume 2139, pages 13–229. Springer-Verlag, 2001.

[BGLA03] L. Bao and J.J. Garcia-Luna-Aceves. Topology Management in Ad Hoc Networks. In *Proc. of the 4th ACM MobiHoc*, 2003.

[BGM87] D. Barbara and H. Garcia-Molina. The reliability of vote mechanisms. *IEEE Trans. on Computers*, 36(10):1197–1208, 1987.

[BHK+04] M. Bechler, H.-J. Hof, D. Kraft, F. Pahlke, and L. Wolf. A Cluster-Based Security Architecture for Ad Hoc Networks. In *Proc. of IEEE INFOCOM'04*, 2004.

[BHO+99] K.P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM Trans. on Computer Systems*, 17(2):41–88, 1999.

[BMR04] A. Bogdanov, E. Maneva, and S. Riesenfeld. Power-aware Base Station Positioning for Sensor Networks. In *Proc. of the 23rd IEEE INFOCOM*, 2004.

[BrH02] L. Buttyán and J.-P. Hubaux. Report on a working session on security in wireless ad hoc networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 6(4), 2002.

[BRY+04] M.A. Batalin, M. Rahimi, Y. Yu, D. Liu, A. Kansal, G.S. Sukhatme, W.J. Kaiser, M.Hansen, G.J. Pottie, M.Srivastava, and D. Estrin. Call and Response: Experiments in Sampling the Environment. In *Proc. of the 2nd ACM SenSys*, 2004.

[BS02] D.M. Blough and P. Santi. Investigating Upper Bounds on Network Lifetime Extenstion for Cell-based Energy Conservation Techniques in Stationary Ad Hoc Networks. In *Proc. of the 8th ACM MobiCom*, 2002.

[BUK04] P. Baruah, R. Urgaonkar, and B. Krishnamachari. Learning Enforced Time Domain Routing to Mobile Sinks in Wireless Sensor Fields. In *Proc. of the 1st IEEE EMNETS*, 2004.

[But03] J. Butler. Robotics and Microelectronics: Mobile Robots as Gateways into Wireless Sensor Networks. *Technology@Intel Magazine*, May 2003.

[CaAOZ03] G. Calinescu, S. Kapoor adn A. Olshevsky, and A. Zelikovsky. Network Lifetime and Power Assignment in Ad-Hoc Wireless Networks. In *Proc. of the 11th EATCS ESA*, 2003.

[CBrH03] S. Capkun, L. Buttyán, and J.-P. Hubaux. Self-organized public-key management for mobile ad hoc networks. *IEEE Trans. on Mobile Computing*, 2(1):52–64, 2003.

[CC03] J.C. Cha and J.H. Cheon. An Identity-based Signature from Gap Diffie-Hellman Groups. In Y.G. Desmedt, editor, *Lecture Notes in Computer Science*, volume 2567, pages 18–30. Springer-Verlag, 2003.

[CD03] C. Crepeau and C.R. Davis. A Certificate Revocation Scheme for Wireless Ad Hoc Networks. In *Proc. of ACM SASN'03*, 2003.

[CHE02] M. Cagalj, J.-P. Hubaux, and C. Enz. Minimum-energy Broadcast in All-wireless Networks: NP-completeness and Distribution Issues. In *Proc. of the 8th ACM MobiCom*, 2002.

[CHH02] S. Capkun, M. Hamdi, and J.-P. Hubaux. GPS-free Positioning in Mobile Ad-Hoc Networks. *Cluster Computing Journal*, 5(2):118–124, 2002.

[CJD03] T. Clausen, P. Jacquet, and S.R. Das. *Optimized Link State Routing Protocol*, October 2003. IETF RFC 3626, Network Working Group.

[CN02] K. Chen and K. Nahrstedt. Effective location-guided tree construction algorithms for small group multicast in MANET. In *Proc. of INFOCOM*, pages 1192–1201, 2002.

[CRB01] R. Chandra, V. Ramasubramanian, and K. Birman. Anonymous gossip: Improving multicast reliability in mobile ad-hoc networks. In *Proc. 21st International Conference on Distributed Computing Systems (ICDCS)*, pages 275–283, 2001.

[CSA03] A. Chakrabarti, A. Sabharwal, and B. Aazhang. Using Predictable Observer Mobility for Power Efficient Design of Sensor Networks. In *Proc. of the 2nd IEEE IPSN*, 2003.

[CSense] http://commonsense.epfl.ch/.

[CT00] J.-H. Chang and L. Tassiulas. Energy Conserving Routing in Wireless Ad-hoc Networks. In *Proc. of the 19th IEEE INFOCOM*, 2000.

[DGH$^+$87] A. Demers, D. Greene, C. Hauser, W. Irish, and J. Larson. Epidemic algorithms for replicated database maintenance. In *Proc. of 6th Annual ACM Symposium on Principles of Distributed Computing (PODC'87)*, pages 1–12, 1987.

[DL] M. Demmer and P. Levis. *Tython: A Dynamic Simulation Environment for Sensor Networks.* http://www.tinyos.net/tinyos-1.x/doc/tython/tython.html.

[DMLM04] E.J. Duarte-Melo, M. Liu, and A. Misra. A Modeling Framework for Computing Lifetime and Information Capacity in Wireless Sensor Networks. In *Proc. of the 2nd WiOpt*, 2004.

[Dou02] J. Douceur. The Sybil Attack. In *Proc. of IPTPS'02*, 2002.

[DSW02] S. Dolev, E. Schiller, and J.L. Welch. Random walk for self-stabilizing group communication in ad hoc networks. In *Proc. of SRDS*, pages 70–79, 2002.

[Edd04] W. Eddy. At What Layer Does Mobility Belong? *IEEE Communications Magazine*, 42(10):155–159, 2004.

[EGH$^+$03] P. Eugster, R. Guerraoui, S. Handurukande, A.M. Kermarrec, and P. Kouznetsov. Lightweight probabilistic broadcast. *ACM Transactions on Computer Systems*, 21(4):341–374, 2003.

[EIA04] W. Eddy, J. Ishac, and M. Atiquzzaman. *An Architecture for Transport Layer Mobility*, August 2004. Internet-Draft, draft-eddy-tlmarch-00.txt. Work in progress.

[FC94] A.W. Fu and D.W. Cheung. A transaction replication scheme for a replicated database with node autonomy. In *Proc. of VLDB*, pages 214–225, 1994.

[FF62] L.R. Ford and D.R. Fulkerson. *Flows in Networks.* Princeton University Press, Princeton, N.J., 1962.

[FJL+97] S. Floyd, V. Jacobson, C-G. Liu, S. McCanne, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Trans. on Networking*, 5(6):784–893, 1997.

[Flo62] R.W. Floyd. Algorithm 97: Shortest Path. *Commun. ACM*, 5(6):345, 1962.

[FV02] K. Fall and K. Varadhan, editors. *The ns Manual.* The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC, Apr. 2002. Availiable from http://www.isi.edu/nsnam/ns/.

[GDPV03] S.R. Gandham, M. Dawande, R. Prakash, and S. Venkatesan. Energy Efficient Schemes for Wireless Sensor Networks with Multiple Mobile Base Stations. In *Proc. of IEEE Globecom*, 2003.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, New York, 1979.

[GK97] N. Garg and J. Könemann. Faster and Simpler Algorithms for Multicommodity Flow and other Fractional Packing Problems. In *Proc. of the 38th IEEE FOCS*, 1997.

[GK04] Y. Ganjali and A. Keshavarzian. Load Balancing in Ad Hoc Networks: Single-path Routing vs. Multi-path Routing. In *Proc. of the 23rd IEEE INFOCOM*, 2004.

[GLM+04] D. Goldenberg, J. Lin, A.S. Morse, B. Rosen, and Y.R. Yang. Towards Mobility as a Network Control Primitive. In *Proc. of the 5th ACM MobiHoc*, 2004.

[GRJK00] R. Gennaro, T. Rabin, S. Jarecki, and H. Krawczyk. Robust and Efficient Sharing of RSA Functions. *Journal of Cryptology*, 13(2):273–300, 2000.

[GS99] S.K.S. Gupta and P.K. Srimani. An adaptive protocol for reliable multicast in mobile multi-hop radio networks. In *Proc. of the 1st IEEE WMCSA*, 1999.

[GT02] M. Grossglauser and D. Tse. Mobility increases the capacity of ad hoc wireless networks. *IEEE/ACM Trans. on Networking*, 10(4):477–486, 2002.

[GZ04] J. Gao and L. Zhang. Load Balanced Short Path Routing in Wireless Networks. In *Proc. of the 23rd IEEE INFOCOM*, 2004.

[Har01] T. Hara. Effective replica allocation in ad hoc networks for improving data accessibility. In *Proc. of INFOCOM*, pages 1568–1576, 2001.

[Hay97] M.G. Hayden. *The Ensemble System.* PhD thesis, Department of Computer Science, Cornell University, 1997.

[HBv01] J.-P. Hubaux, L. Buttyán, and S. Čapkun. The quest for security in mobile ad hoc networks. In *Proc. of MobiHoc'01*, 2001.

[HC04] J.W. Hui and D. Culler. The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale. In *Proc. of the 2nd ACM SenSys*, 2004.

[HCB02] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. An Application-specific Protocol Architecture for Wireless Microsensor Networks. *IEEE Trans. on Wireless Communications*, 1(4):660–670, 2002.

[HGBV01] J.-P. Hubaux, T. Gross, J.-Y. Le Boudec, and M. Vetterli. Toward self-organized mobile ad hoc networks: the terminodes project. *IEEE Communications Magazine*, 39(1):118–124, 2001.

[HHB$^+$03] T. He, C. Huang, B.M. Blum, J.A. Stankovic, and T. Abdelzaher. Range-free Localization Schemes for Large Scale Sensor Networks. In *Proc. of the 9th ACM MobiCom*, 2003.

[HHL02] Z.J. Haas, J.Y. Halpern, and L. Li. Gossip-based ad hoc routing. In *Proc. of INFOCOM*, pages 1707–1716, 2002.

[HKB99] W. Heinzelmann, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proc. of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 174–185, 1999.

[HL99a] Z.J. Haas and B. Liang. Ad hoc mobility management with randomized database groups. In *Proc. of IEEE ICC*, 1999.

[HL99b] Z.J. Haas and B. Liang. Ad hoc mobility management with uniform quorum systems. *IEEE/ACM Trans. on Networking*, 7(2):228–240, 1999.

[HPJ02] Y. Hu, A. Perrig, and D.B. Johnson. Ariadne: A Secure On-demand Routing Protocol for Ad Hoc Networks. In *Proc. of ACM MobiCom'02*, 2002.

[HPJ03] Y. Hu, A. Perrig, and D.B. Johnson. Packet Leashes: A Defense Against Wormhole Attacks in Wireless Networks. In *Proc. of IEEE INFOCOM'03*, 2003.

[HT93] V. Hadzilacos and S. Toueg. Fault-tolerant broadcasts and related problems. In *Distributed Systems*, chapter 5, pages 97–145. Addison-Wesley, 2 edition, 1993.

[IGE$^+$03] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. on Networking*, 11(1):2–16, 2003.

[JC01] L. Ji and M.S. Corson. Differential destination multicast - a MANET multicast routing protocol for small groups. In *Proc. of INFOCOM 2001*, pages 1192–1201, 2001.

[JM96] D.B. Johnson and D.A. Maltz. Dynamic source routing in ad hoc wireless networks. In Tomasz Imielinski and Hank korth, editors, *Mobile Computing*, chapter 5, pages 153–181. Kluwer Academic Publishers, 1996.

[JMH04] D.B. Johnson, D.A. Maltz, and Y-C. Hu. *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)*, July 2004. Internet-Draft, draft-ietf-manet-dsr-10.txt. Work in progress.

[JSS05] D. Jea, A. Somasundara, and M.B. Srivastava. Multiple Controlled Mobile Elements (Data Mules) for Data Collection in Sensor Networks. In *Proc. of the 1st IEEE/ACM DCOSS*, 2005.

[KAK03] H.S. Kim, T.F. Abdelzaher, and W.H. Kwon. Minimum Energy Asynchronous Dissemination to Mobile Sinks in Wireless Sensor Networks. In *Proc. of the 1st ACM SenSys*, 2003.

[KK03] V. Kawadia and P.R. Kumar. Power Control and Clustering in Ad Hoc Networks. In *Proc. of the 22nd IEEE INFOCOM*, 2003.

[KKA03] A. Khalili, J. Katz, and W.A. Arbaugh. Toward Secure Key Distribution in Truly Ad-Hoc Networks. In *Proc. of IEEE WSAAN'03*, 2003.

[KKLT03] K. Kar, M. Kodialam, T.V. Lakshman, and L. Tassiulas. Routing for Network Capacity Maximization in Energy-constrained Ad-hoc Networks. In *Proc. of the 22rd IEEE INFOCOM*, 2003.

[KMG03] A.-M. Kermarrec, L. Massoulie, and A. Ganesh. Probabilistic reliable dissmination in large-scale systems. *IEEE Trans. on Parallel and Distributed Systems*, 14(3):248–258, 2003.

[KSJ+04] A. Kansal, A. Somasundara, D.D. Jea, M.B. Srivastava, and D. Estrin. Intelligent Fluid Infrastructure for Embedded Networks. In *Proc. of the 2nd ACM/USENIX MobiSys*, 2004.

[LABW92] B. Lampson, M. Abadi, M. Burrows, and E. Wobber. Authentication in Distributed Systems: Theory and Practice. *ACM Trans. on Computer Systems*, 10(4):265–310, 1992.

[LBK+02] A. LaMarca, W. Brunette, D. Koizumi, M. Lease, S. Sigurdsson, K. Sikorski, D. Fox, and G. Borriello. Making Sensor Networks Practical with Robots. In F. Mattern and M. Naghshineh, editors, *LNCS*, pages 152–166. Springer-Verlag, 2002.

[LEH03] J. Luo, P.Th. Eugster, and J.-P. Hubaux. Route Driven Gossip: Probabilistic Reliable Multicast in Ad Hoc Networks. In *Proc. of the 22nd IEEE INFOCOM*, pages 2229–2239, 2003.

[LEH04] J. Luo, P.Th. Eugster, and J.-P. Hubaux. PILOT: ProbabilistIc Lightweight grOup communication sysTem for Mobile Ad Hoc Networks. *IEEE Trans. on Mobile Computing*, 3(2):164–179, 2004.

[LH04] N. Li and J. Hou. Topology Control in Heterogeneous Wireless Networks: Problems and Solutions. In *Proc. of the 23rd IEEE INFOCOM*, 2004.

[LH05] J. Luo and J.-P. Hubaux. Joint Mobility and Routing for Lifetime Elongation in Wireless Sensor Networks. In *Proc. of the 24th IEEE INFOCOM*, 2005.

[LHB+01] L. Li, J.Y. Halpern, P. Bahl, Y.-M. Wang, and R. Wattenhofer. Analysis of A Cone-based Distributed Topology Control Algorithm for Wireless Multi-hop Networks. In *Proc. of the 20th ACM PODC*, 2001.

[LHE03] J. Luo, J.-P. Hubaux, and P.Th. Eugster. PAN: Providing Reliable Storage in Mobile Ad Hoc Networks with Probabilistic Quorum Systems. In *Proc. of the 4th ACM MobiHoc*, 2003.

[LKZ+04] H. Luo, J. Kong, P. Zerfos, S. Lu, and L. Zhang. URSA: Ubiquitous and Robust Access Control for Mobile Ad-Hoc Networks. *IEEE/ACM Trans. on Networking*, 12(6), 2004.

[LLWC03] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In *Proc. of the 1st ACM SenSys*, 2003.

[LM00] M-J. Lin and K. Marzullo. Directional gossip: gossip in a wide area network. In *Proc. of European Dependable Computing Conference (EDCC-3)*, 2000.

[LPCS04] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A Self-regulating Algorithm for Code Maintenance and Propagation in Wireless Sensor Networks. In *Proc. of the 1st USENIX/ACM NSDI*, 2004.

[LSG02] S.J. Lee, William Su, and M. Gerla. On-Demand Multicast Routing Protocol in Multihop Wireless Mobile Networks. *ACM/Kluwer Mobile Networks and Applications*, 7:441–453, 2002.

[LSH06] J. Luo, H.V. Shukla, and J.-P. Hubaux. Non-Interactive Location Surveying for Sensor Networks with Mobility-Differentiated ToA . In *Proc. of the 25th IEEE INFOCOM (to appear)*, 2006.

[LWW+04] X.-Y. Li, Y. Wang, P.-J. Wan, W.-Z. Song, and O. Frieder. Localized Low-Weight Graph and Its Applications in Wireless Ad Hoc Networks. In *Proc. of the 23rd IEEE INFOCOM*, 2004.

[LZK+02] H. Luo, P. Zerfos, J. Kong, S. Lu, and L. Zhang. Self-securing Ad Hoc Wireless Networks. In *Proc. of IEEE ISCC'02*, 2002.

[MC02] G. Montenegro and C. Castelluccia. Statistically Unique and Cryptographically Verifiable (SUCV) Identifiers and Adresses. In *Proc. of ISOC NDSS'02*, 2002.

[Meg84] N. Megiddo. Linear Programming in Linear Time When the Dimension Is Fixed. *J. ACM*, 31(1):114–127, 1984.

[MPS+02] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless Sensor Networks for Habitat Monitoring. In *Proc. of the 1st ACM WSNA*, 2002.

[MR98] D. Malkhi and M.K. Reiter. Byzantine Quorum Systems. *Distributed Computing*, 11(4):203–213, 1998.

[MRW01] D. Malkhi, M.K. Reiter, and A. Wool. Probabilistic quorum systems. *Information and Computation*, 170(2):184–206, 2001.

[Mur93] J.D. Murray. *Mathematical Biology.* Springer, Berlin, 2nd edition, 1993.

[MvOV97] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.

[MVW02] N. Malpani, N.H. Vaidya, and J.L. Welch. Distributed token circulation in mobile ad hoc networks. In *Proc. of ICNP*, pages 4–13, 2002.

[NN03] D. Niculescu and B. Nath. Trajectory Based Forwarding and Its Applications. In *Proc. of the 9th ACM MobiCom*, 2003.

[NP02] S. Nesargi and R. Prakash. MANETconf: configuration of hosts in a mobile ad hoc network. In *Proc. of INFOCOM*, pages 1059–1068, 2002.

[NTCS99] S.Y. Ni, Y.C. Tseng, Y.S. Chen, and J.P. Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proc. of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 151–162, 1999.

[NW88] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley & Sons, New York, 1988.

[PB94] C.E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proc. of ACM SIGCOMM*, 1994.

[PBDT05] N.B. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller. Mobile-Assisted Localization in Wireless Sensor Networks. In *Proc. of the 24th IEEE INFOCOM*, 2005.

[PBRD03] C.E. Perkins, E.M. Belding-Royer, and S.R. Das. *Ad hoc On-Demand Distance Vector (AODV) Routing*, July 2003. IETF RFC 3561, Network Working Group.

[PC00] C.E. Perkins and P. Calhoun. *Mobile IPv4 Challenge/Response Extensions*, November 2000. IETF RFC 3012, Network Working Group.

[Per96] C.E. Perkins. *IP Mobility Support*, October 1996. IETF RFC 2002, Network Working Group.

[PG01] G. Pei and Mario Gerla. Mobility management for hierarchical wireless networks. *ACM/Kluwer Mobile Networks and Applications (MONET)*, 6(4):331–337, 2001.

[PG05] I. Papadimitriou and L. Georgiadis. Maximum Lifetime Routing to Mobile Sink in Wireless Sensor Networks. In *Proc. of the 13th IEEE SoftCom*, 2005.

[PH99] M.R. Pearlman and Z.J. Haas. Determining the Optimal Configuration for the Zone Routing Protocol. *IEEE J. Select. Areas Commun.*, 17(8):1395–1414, 1999.

[PHC$^+$03] J. Pan, Y.T. Hou, L. Cai, Y. Shi, and S.X. Shen. Topology Control for Wireless Sensor Networks. In *Proc. of the 9th ACM MobiCom*, 2003.

[PHC04] J. Polastre, J. Hill, and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *Proc. of the 2st ACM SenSys*, 2004.

[Pow96] D. Powell. Group communication (special issue). *Communications of the ACM*, 39(4):50–97, 1996.

[PR99] E. Pagani and G. P. Rossi. Providing Reliable and Fault Tolerant Broadcast Delivery in Mobile Ad-Hoc Networks. *ACM/Kluwer Mobile Networks and Applications*, 5(4):175–192, 1999.

[PS01] M. Papadopouli and H. Schulzrinne. Effects of power conservation, wireless coverage and cooperation on data dissemination among mobile devices. In *Proc. of MobiHoc*, pages 117–127, 2001.

[PSLB97] S. Paul, K.K. Sabnani, J.C. Lin, and S. Bhattacharyya. Reliable multicast transport protocol. *IEEE J. Sel. Areas Commun.*, 15(3):784–893, 1997.

[PSRR03] D. Petrovic, R.C. Shah, K. Ramchandran, and J. Rabaey. Data funneling: Routing with aggregation and compression for wireless sensor networks. In *Proc. of the 1st IEEE SNPA*, 2003.

[Rap02] T.S. Rappaport. *Wireless Communications: Principles and Practice.* Prentice Hall, Upper Saddle River, New Jersey, 2nd edition, 2002.

[RFLW96] M.K. Reiter, M.K. Franklin, J.B. Lacy, and R.N. Wright. The Ω Key Management Service. *Journal of Computer Security*, 4(4):267–287, 1996.

[RHH01] G.-C. Roman, Q. Huang, and A. Hazemi. Consistent group membership in ad hoc networks. In *Proc. of ISCE*, 2001.

[RP99] E.M. Royer and C.E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *Proc. of the 5th ACM MobiCom*, 1999.

[RRP$^+$03] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic Routing without Location Information. In *Proc. of the 9th ACM MobiCom*, 2003.

[Sag94] H. Sagan. *Space-Filling Curves.* Springer-Verlag, New York, 1994.

[Sch93] F.B. Schneider. Replication Management Using the State-Machine Approach. In *Distributed Systems*, chapter 6, pages 169–197. Addison-Wesley, 2 edition, 1993.

[SHC$^+$04] V. Shnayder, M. Hempstead, B. Chen, G.W. Allen, and M. Welsh. Simulating the Power Consumption of LargeScale Sensor Network Applications. In *Proc. of the 2nd ACM SenSys*, 2004.

[SHS01] A. Savvides, C. Han, and M.B. Srivastava. Dynamic FineGrained Localization in Ad-Hoc Networks of Sensors. In *Proc. of the 7th ACM MobiCom*, 2001.

[SL04] A. Sankar and Z. Liu. Maximum Lifetime Routing in Wireless Ad-hoc Networks. In *Proc. of the 23rd IEEE INFOCOM*, 2004.

[SM90] F. Shahrokhi and D.W. Matula. The Maximum Concurrent Flow Problem. *J. ACM*, 37(2):318–334, 1990.

[SMP^{+}04] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler. An Analysis of a Large Scale Habitat Monitoring Application. In *Proc. of the 2nd ACM SenSys*, 2004.

[SPK] *Simple Public Key Infrastructure (SPKI).* SPKI Working Group, The Internet Engineering Task Force (IETF). http://www.ietf.org/html.charters/spki-charter.html.

[SRJB03] R.C. Shah, S. Roy, S. Jain, and W. Brunette. Data MULEs: Mobeling a Three-tier Architecutre for Sparse Sensor Networks. In *Proc. of the 1st IEEE SNPA*, 2003.

[SRS04] A.A. Somasundara, A. Ramamoorthy, and M.B. Srivastava. Mobile Element Scheduling for Efficient Data Collection in Wireless Sensor Networks with Dynamic Deadlines. In *Proc. of the 25th IEEE RTSS*, 2004.

[SSB99] R. Sivakumar, P. Sinha, and V. Bharghavan. CEDAR: a core-extraction distributed ad hoc routing algorithm. *IEEE Journal on Selected Areas in Communications (Special Issue on Ad-hoc Routing)*, 17(8):1454–1465, 1999.

[Sta03] W. Stallings. *Cryptpgraphy and Network Security: Principle and Practices.* Prentice Hall, Upper Saddle River, New Jersey, 2003.

[Stu95] S. Stubblebine. Recent-Secure Authentication: Enforcing Revocation in Distributed System. In *Proc. of IEEE S&P 1995*, 1995.

[STW00] M. Steiner, G. Tsudik, and M. Waidner. Key Agreement in Dynamic Peer Groups. *IEEE Trans. on Parallel and Distributed Systems*, 11(8):769–780, 2000.

[TinyOS] http://www.tinyos.net/.

[VB00] A. Vahdat and D. Becker. Epidemic routing for partially-connected ad hoc networks. Technical Report CS-2000-06, Duke University, 2000.

[Vol03] E. Vollset. The autograph protocol - a preliminary report on a reliable broadcast protocol for mobile ad-hoc networks. In *Proc. of DSN - Student Forum*, 2003.

[WBMP05] Z.M. Wang, S. Basagni, E. Melachrinoudis, and C. Petrioli. Exploiting Sink Mobility for Maximizing Sensor Networks Lifetime. In *Proc. of the 38th HICSS*, 2005.

[WCP04] G. Wang, G. Cao, and T. La Porta. Movement-Assisted Sensor Deployment. In *Proc. of the 23rd IEEE INFOCOM*, 2004.

[WL02] K.H. Wang and B. Li. Efficient and guaranteed service coverage in partitionable mobile ad-hoc networks. In *Proc. of INFOCOM*, pages 1089–1098, 2002.

[WSC05] W. Wang, V. Srinivasan, and K.-C. Chua. Using Mobile Relays to Prolong the Lifetime of Wireless Sensor Networks. In *Proc. of the 11th ACM MobiCom*, 2005.

[WT02] A. Weimerskirch and G. Thonet. A Distributed Light-weight Authentication Model for Ad-Hoc Networks. In K. Kim, editor, *Lecture Notes in Computer Science*, volume 2288, pages 314–354. Springer-Verlag, 2002.

[WTC03] A. Woo, T. Tong, and D. Culler. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In *Proc. of the 1st ACM SenSys*, 2003.

[X50] *Public-Key Infrastructure (X.509).* PKIX Working Group, The Internet Engineering Task Force (IETF). http://www.ietf.org/html.charters/pkix-charter.html.

[XHG02] K. Xu, X. Hong, and M. Gerla. An ad hoc network with mobile backbones. In *Proc. of ICC*, volume 5, pages 3138–3143, 2002.

[YF04] O. Younis and S. Fahmy. Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-Efficient Approach. In *Proc. of the 23rd IEEE INFOCOM*, 2004.

[YHE02] W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proc. of the 21st IEEE INFOCOM*, 2002.

[YK03] S. Yi and R. Kravets. MOCA: Mobile Certificate Authority for Wireless Ad Hoc Networks. In *Proc. of PKI'03*, 2003.

[YLC+05] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang. A Two-tier Data Dissemination Model for Large Scale Wireless Sensor Networks. In *Proc. of the 8th ACM MobiCom*, 2005.

[YLN03] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *Proc. of INFOCOM*, pages 1312–1321, 2003.

[YPK04] M. Yuksel, R. Pradhan, and S. Kalyanaraman. An Implementation Framework for Trajectory-Based Forwarding in Ad-Hoc Networks. In *Proc. of IEEE ICC*, 2004.

[ZA02] M.G. Zapata and N. Asokan. Securing Ad Hoc Routing Protocols. In *Proc. of ACM WiSe'02*, 2002.

[ZAZ05] W. Zhao, M. Ammar, and E. Zegura. Controlling the Mobility of Multiple Data Transport Ferries in a Delay-Tolerant Network. In *Proc. of the 24rd IEEE INFOCOM*, 2005.

[ZH99] L. Zhou and Z.J. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, 1999.

[Zim95] P. Zimmermann. *The Official PGP User's Guide.* MIT Press, Cambridge, Massachusetts, 1995.

[ZLH03] Y. Zhang, W. Lee, and Y. Huang. Intrusion detection techniques for mobile wireless networks. *ACM/Kluwer Wireless Networks*, 9(5):545–556, 2003.

[ZSvR02] L. Zhou, F.B. Schneider, and R. van Renesse. COCA: A Secure Distributed On-line Certification Authority. *ACM Trans. on Computer Systems*, 20(4):329–368, 2002.

Appendix A

Estimations of the Network Parameters

In order to obtain the distribution of H, we assume that the network nodes are uniformly distributed within a circle of diameter equal to 10 hops[1]. Then, by repeating the procedure of randomly picking up two points within this circle and computing the distance between them, we obtain the distribution function of H in a numerical way. The distribution $f(h)$ is shown in Fig. A.1.

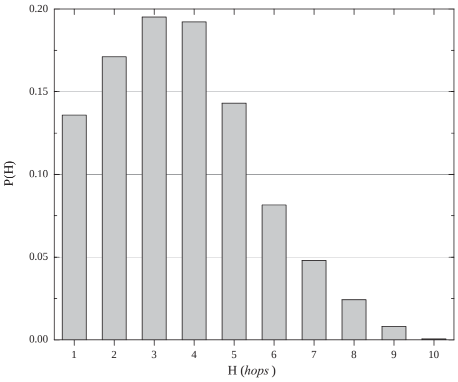


Figure A.1: Distribution of H. Here H is the random variable representing the distance between two randomly picked points within a circle. It can be considered as the length in hops of a routing path between two randomly picked network nodes, with the assumption that the nodes are uniformly distributed.

---

[1] This means that a node at the end of the diameter should take 10 hops to reach a node at the other end. The uniform distribution also implies that the path length between two nodes is approximately the same as the distance between them.

The other important step is to estimate p_f. We assume that $p_{f_{mo}} \gg p_{f_c}$, so $p_{f_{mo}}$ is directly used to approximate p_f. The estimation of $p_{f_{mo}}$ is done by simulation with *ns-2*. Since this parameter is determined by both movement and traffic pattern, we apply the same movement scenario as to the simulation for our protocol with the heaviest traffic load. The heaviest load of our protocol is when the network is loaded with about $F \times n$ connections and the sending rate is the basic rate imposed by the upper layer times the τ_q. For example, we simulate a scenario of 50 sources and 150 connections for a group of 50 members with $F = 3$. The results are average packet loss ratio p_l and the distribution of the number of hops H_l traveled by a packet before getting dropped (See Fig. A.2 for an example). It is easy to see that $\Pr\{H_l = 1\}p_l = \sum_h p_{f_{mo}} \Pr\{H = h\}$. In fact, both sides of the equation give the probability that a packet gets lost at the first hop. Therefore, we have $p_{f_{mo}} = \Pr\{H_l = 1\}p_l$. An example of the values used for the analysis is provided in Fig. A.3. It can be observed that $p_{f_{mo}}$ is an increasing function of both F and τ_q.

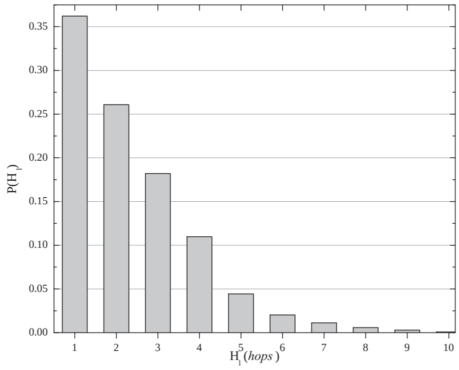


Figure A.2: Distribution of H_l when average packet loss ratio equals to 12.7%, assuming a group size of 50 and a network size of 100 with $F = 3$ and $\tau_q = 1$.

τ_q / F	1	2
2	0.0200	—
3	0.0460	0.2749
4	0.1686	—

Figure A.3: The $p_{f_{mo}}$ with respect to different values of F and τ_q, assuming a group size of 50 and a network size of 100.

Appendix B

Another Proof of Claim 1

Our proof of **Claim 1** is based on the principle of *equivalence of separation and optimization* given in [NW88]. **Theorem 3.3** of [NW88] states that a linear programming problem is solvable in polynomial time (by the *ellipsoid* algorithm) if and only if there exists a *separation oracle* that has a polynomial complexity. The function of the oracle is to identify a violated constraint or to verify that there is no such a constraint. The theorem also implies that the linear programming problem is NP-hard if the separation oracle is NP-complete, because there is a polynomial-time reduction of the separation oracle to the linear programming problem.

Let us re-formulate the MNL problem into a Path-Flow form by dropping the non-essential constraint (4.5):

$$\text{Maximize} \quad \textstyle\sum_k t_k \tag{B.1}$$

$$s.t. \quad \sum_{p \in P_{ik}} f(p) - \lambda_i t_k = 0 \qquad \forall i, k \tag{B.2}$$

$$\sum_k \sum_{p \in P_i^k} f(p)(e_i^t + e^r) - E_i \leq 0 \qquad \forall i \tag{B.3}$$

$$f(p),\ t_k \geq 0 \qquad \forall p, k \tag{B.4}$$

where p refers to a certain path and $f(p)$ is the flow that goes through p. Furthermore, P_{ik} stands for the set of paths between a node i and one of the sinks during the kth epoch and P_i^k represents the set of paths that go through node i in the kth epoch.

The dual problem is given by:

$$
\begin{array}{rcccr}
 & \text{Minimize} \quad \sum_i E_i w(i) & & & \text{(B.5)} \\
s.t. \quad & \displaystyle\sum_i \lambda_i W(i,k) \geq 1 & & \forall k & \text{(B.6)} \\
 & \displaystyle\sum_{j \in p,\ p \in P_{ik}} w(j)(e^t_j + e^r) - W(i,k) \geq 0 & & \forall i,k & \text{(B.7)} \\
 & W(i,k),\ w(j) \geq 0 & & \forall i,j,k & \text{(B.8)}
\end{array}
$$

where $W(i,k)$ is the weight assigned to a "commodity" (data flow injected at a node in our case) from node i to one of the sinks during the kth epoch and $w(j)$ is the weight assigned to a node j.

The separation oracle for the dual problem should check if one of the constraints in (B.6) and (B.7) is violated. It is equivalent to verify if one of the following constraints are violated:

$$\sum_i \lambda_i \sum_{j \in p,\ p \in P_{ik}} w(j)(e^t_j + e^r) \geq 1 \qquad \forall k \tag{B.9}$$

Actually, by taking the minimum value on the left-hand side of (B.9), the oracle only needs to work on one constraint:

$$\min_k \left(\sum_i \lambda_i \sum_{j \in \min\{p | p \in P_{ik}\}} w(j)(e^t_j + e^r) \right) \geq 1 \tag{B.10}$$

Now, by taking $\omega(i) = \lambda_i$, $l(i) = w(i)(e^t_i + e^r)$, $K = |\mathcal{S}|$, and $d(i) = \sum_{j \in \min\{p|p \in P_{ik}\}} w(j)(e^t_j + e^r)$, the minimization problem involved in the oracle leads to the following decision problem:

> Instance: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a weight assignment $\omega(i)$: $\mathcal{V} \to \mathbb{R}^+_0$, a length assignment[1] $l(i)$: $\mathcal{V} \to \mathbb{R}^+_0$, positive integer $K \leq |\mathcal{V}|$, and positive rational number B.
>
> Question: Is there a set $\mathcal{P}$ of K "points on $\mathcal{G}$" such that $\sum_{i \in \mathcal{V}} \omega(i) \cdot d(i) \leq B$?

The problem, which is known as the p-median problem [GJ79], is NP-complete. Therefore, according to our arguments at the beginning, the MNL problem is NP-hard. Q.E.D.

Since we explicitly show a polynomial reduction of the p-median problem to our MNL problem in the above proof, any solution (or approximation) to the p-median problem can be directly applied to MNL.

---

[1] Usually, a length assignment is associated with edges. However, we can always convert our node-capacity based problem to a link-capacity based version by replacing a node with two nodes and a link having the same capacity.

Appendix C

Proof of Claim 3

Our proof of **Claim 3** is based on the proof given by Garg and Könemann [GK97]. We have the following relation from (4.31):

$$G(w_i) = G(w_{i-1}) + \epsilon \cdot \rho(w_{i-1})$$

Since $\frac{G(w)}{\rho(w)} > \beta$ and $G(w_0) = n\delta$, we have for the ith iteration:

$$\begin{aligned} G(w_i) &\leq G(w_{i-1})(1 + \epsilon/\beta) \\ &\leq G(w_{i-1})e^{\epsilon/\beta} \\ &\leq n\delta e^{i\epsilon/\beta} \end{aligned}$$

Suppose that the procedure stops at tth iteration for which $G(w_t) \geq 1$, we have:

$$1 \leq G(w_t) \leq n\delta e^{t\epsilon/\beta}$$

which suggests that:

$$\frac{\beta}{t} \leq \frac{\epsilon}{\ln(n\delta)^{-1}} \tag{C.1}$$

Note that t is not necessarily the lifetime we achieve, because it is possible that the flow already violates the constraints, namely (4.24), of the primal problem.[1] Let us consider a certain node l. For every $E_l/(e_l^t + e^r)$ units of flow routed through l, the weight $w(l)$ is increased by at least $1 + \epsilon$. In addition, the weight $w(l)$ is increased by at most $1 + \epsilon$ each iteration due to the assumption that $\sum_i \lambda_i \leq E_l/(e_l^t + e^r)$. Since $w_0(l) = \delta/E_l$ and $w_{t-1}(l) < 1/E_l$ (due to the fact that $G(w_{t-1}) < 1$), the total amount of flow through l during the first t iteration is strictly less than $\frac{E_l}{e_l^t+e^r} \log_{1+\epsilon} \frac{1+\epsilon}{\delta}$. Therefore, (4.24) can be violated by at most a multiple of $\log_{1+\epsilon} \frac{1+\epsilon}{\delta}$, and thus $t \cdot \log_{1+\epsilon}^{-1} \frac{1+\epsilon}{\delta}$ gives a

---

[1] Constraints (4.23) are always satisfied because the iteration procedure increases t_k by one only if $\sum_i \lambda_i$ units of commodities from all nodes are admitted.

feasible primal solution. If we denote the ratio of the dual and the primal solutions by γ and apply the bound on β/t given in (C.1), we have:

$$\gamma = \frac{\beta}{t}\log_{1+\epsilon}\frac{1+\epsilon}{\delta} \leq \frac{\epsilon\log_{1+\epsilon}\frac{1+\epsilon}{\delta}}{\ln(n\delta)^{-1}} = \frac{\epsilon}{\ln(1+\epsilon)}\cdot\frac{\ln\frac{1+\epsilon}{\delta}}{\ln(n\delta)^{-1}}$$

For $\delta = (1+\epsilon)[(1+\epsilon)n]^{-1/\epsilon}$, we have:

$$\gamma \leq \frac{\epsilon}{(1-\epsilon)\ln(1+\epsilon)} \leq \frac{\epsilon}{(1-\epsilon)(\epsilon-\epsilon^2/2)} < (1-\epsilon)^{-2}$$

which means that, if the maximal lifetime is T ($= \beta$ according to strong duality), the algorithm achieves a lifetime $\hat{T} = t\cdot\log^{-1}_{1+\epsilon}\frac{1+\epsilon}{\delta} > (1-\epsilon)^2 T \geq (1-2\epsilon)T$.

The time complexity can be obtained by applying weak duality:

$$1 \leq \gamma = \frac{\beta}{t}\log_{1+\epsilon}\frac{1+\epsilon}{\delta}$$

and thus the number of iterations in our algorithm t is less than $\beta\log_{1+\epsilon}\frac{1+\epsilon}{\delta}$. Given $\delta = (1+\epsilon)[(1+\epsilon)n]^{-1/\epsilon}$, we have $t = \lceil\frac{\beta}{\epsilon}\log_{1+\epsilon}(1+\epsilon)n\rceil$. Since each iteration involves one call to the oracle, the actual time complexity is $\lceil\frac{\beta}{\epsilon}\log_{1+\epsilon}(1+\epsilon)n\rceil\cdot T_{\text{oracle}}$.

Notice that the run time depends on β, which can be very large if $E_l \gg e_l^t + e^r$ for each node l (since $\beta \leq \min_l[E_l/(e_l^t+e^r)]$). Fortunately, we do not need to solve a full-scale problem. If we scale down every E_l by a factor of ϕ, we actually scale down the dual objective $\sum_l E_l w_l$ (and thus the lifetime $\sum_k t_k$) by ϕ. Therefore, we choose the largest ϕ without violating $\sum_i \lambda_i \leq E_l/(e_l^t+e^r)$. So $\phi = E_{\hat{l}}/[(e_{\hat{l}}^t+e^r)\sum_i\lambda_i]$ where $\hat{l} = \arg\min_l E_l/[(e_l^t+e^r)]$. Now $\beta = \Theta(n)$ and thus the time complexity becomes $\Theta(n\log n)\cdot T_{\text{oracle}}$. Note that the solution of this reduced-scale problem should be scaled up by ϕ to obtain the real solution. Q.E.D.

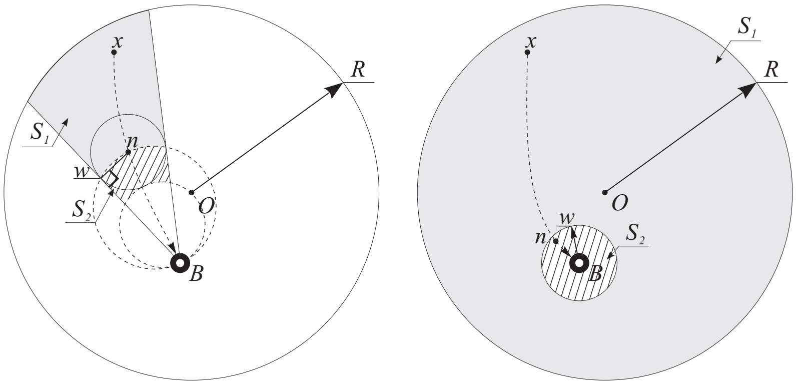
Appendix D

Load Model for Wireless Sensor Networks

D.1 Modeling the Sensor Load: The Case of a Static Sink

We sketch the analytical model proposed by Ganjali and Keshavarzian [GK04] and show how we extend their model for our analysis. Let us consider two fixed points B and n (which refer to the sink and an arbitrary node in our case). Ganjali and Keshavarzian observe that the set of nodes used by routing paths between any two nodes approximately lie in a rectangle of width $2w$. They suggest that the locus of a node x whose routing path towards B goes through n should meet two conditions if $|xB| \geq w$: (i) the distance from n to the line segment xB should be less than w, and (ii) the projection of point n on xB should lie between x and B. These two criteria indicate that the locus of x is the area S_1 in Fig. D.1(a). In our case, we only assume single-path routing, so it is intuitively correct to set $w = r$.

Actually, it is only in the worst case that n should forward all traffic flows from S_1. Since we apply an ideal load-balanced routing, the load will be shared among nodes in S_2, i.e., nodes within this one-hop belt have the same probability to be chosen as forwarding nodes. As a result, $\overline{load}_n$ is in proportion to $(S_1 + S_2)/S_2$ (remember that nodes in S_2 have to forward their own data traffic). Directly calculating the area of S_2 leads to a very complex expression, but the area can be approximated by $\pi r^2/2$, as suggested in Fig. D.1(a). When $|nB| < r$, [GK04] concludes that the locus of x consists of the intersection of a half plane and the network region. This result, however, does not apply to our average load computation. For average load evaluation, it is easy to see that nodes (including n) within the transmission range r of B share the load of forwarding traffic flows from all nodes in the networks, as shown in Fig. D.1(b), again due to our assumption of an ideal load-balanced routing.



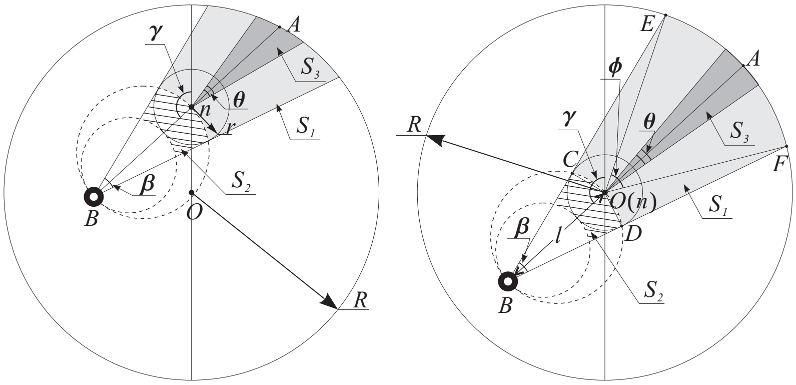
(a) $|nB| \geq w$ (b) $|nB| < w$

Figure D.1: Modeling sensor load in the case of a static sink.

D.2 Modeling the Sensor Load: The Case of a Mobile Sink

For the case of a mobile sink, it is very hard to obtain a closed form expression for $\overline{load}_n$ if we continue using the model in Appendix D.1. The reason is that the angle β in Fig. D.2(a), and thus the area of S_1, depends on the positions of both n and B. Let us consider a new model where node n has the duty of forwarding data traffic from S_3, a sector centered around nA with an angle of θ. This model is equivalent to the model in Appendix D.1 if $(S_1 + S_2)/S_2 = S_3\rho$. Given a position of n and an angle γ, since S_1 decreases with increasing $|nB|$ and S_2 is somewhat constant, S_3, and thus θ, is a decreasing function of $|nB|$. If we can find a sector $\bar{S}_3$ of angle $\bar{\theta}$ that is equivalent to $(S_1 + S_2)/S_2$ **on average** but is **decoupled** from $|nB|$, the estimations provided in Section 4.4.3 would be justified. However, obtaining a value equivalent to $(S_1+S_2)/S_2$ on average for an arbitrary n has the same complexity as calculating the $\overline{load}_n$. Fortunately, $\bar{\theta}$ is computable when n lies on the center, as shown in Fig. D.2(b). Also, it is observed that the area of S_3 depends mostly on its height, i.e., $|nA|$ in both Fig. D.2(a) and (b), instead of θ, since $S_3 \approx \theta|nA|^2/2$. So we can calculate $\bar{\theta}$ for the centered n and apply this value as an estimation for an arbitrary n. Let $\overline{load}_n$ be the value computed with $(S_1 + S_2)/S_2$ and $\widehat{load}_n$ be the value computed with $\bar{\theta}$ (thus $\bar{S}_3$), we want $\overline{load}_n = \widehat{load}_n$ for a centered n. These two values can be computed as follows:

$$\begin{aligned}
\widehat{load}_n &= \int_{\gamma=0}^{2\pi}\int_{l=0}^{R} S_3\rho\lambda\varepsilon \times \mathrm{Fr}\{\text{sink at } B\} \\
&= \int_0^{2\pi}\int_0^{R} \frac{1}{2}\bar{\theta}R^2\rho\lambda\varepsilon \times \frac{l \times \mathrm{d}l \times \mathrm{d}\gamma}{\pi R^2} \\
&= \frac{1}{2}\bar{\theta}R^2\rho\lambda\varepsilon \qquad \text{(D.1)}
\end{aligned}$$



(a) Arbitrary node *n* (b) Centered node *n*

Figure D.2: Modeling sensor load in the case of a mobile sink.

$$
\begin{aligned}
\overline{load}_n &= \int_{\gamma=0}^{2\pi}\int_{l=0}^{R} \frac{(S_1+S_2)\lambda\varepsilon}{S_2} \times \mathrm{Fr}\{\text{sink}^1\text{at } B\} \\
&\approx \left\{\int_{0}^{2\pi}\int_{r}^{R} \frac{(S_{OEF}+S_{OCE}+S_{ODF}+S_2)\lambda\varepsilon}{S_2}\right. \\
&\times \left.\frac{l \times \mathrm{d}l \times \mathrm{d}\gamma}{\pi R^2}\right\} \\
&+ \int_{0}^{2\pi}\int_{0}^{r} \frac{\pi R^2\lambda\varepsilon}{\pi r^2} \times \frac{l \times \mathrm{d}l \times \mathrm{d}\gamma}{\pi R^2} \\
&\approx \frac{4\lambda\varepsilon}{\pi r^2 R^2}\left\{\int_{r}^{R}\Big((\arcsin(\frac{r}{l}) + \arcsin(\frac{r}{R}))R^2\right. \\
&+ \left. r\sqrt{R^2-r^2} + \frac{\pi r^2}{2}\Big)\, l\mathrm{d}l\right\} + \lambda\varepsilon \qquad \text{(D.2)}
\end{aligned}
$$

Let $\overline{load}_n = \widehat{load}_n$ and assume $R = 10$, $r = 1$ $\rho = 8/\pi$, $\lambda = 1$, and $\varepsilon = 1$; we thus have $\bar{\theta} \approx 0.2$.

---

[1] By "sink at B", though an abuse of terminology, we mean that the sink is in an infinitesimal area (which measures $l \times \mathrm{d}l \times \mathrm{d}\gamma$ in *polar* coordinates) centered on B.

Index

Ω, 39
PAN, 13, 19, 44, 52
PILOT, 8, 13, 38, 44
StS, 12, 13, 19, 29
7DS, 11

ack implosion, 8
ad hoc network, 12, 37, 41
AG (Anonymous Gossip), 9, 30
agent, 19, 46, 48, 50–53
anycast, 45

Byzantine failure, 42
Byzantine fault tolerance, 51
Byzantine quorum systems, 39, 45

CA (Certification Authority), 37
CBR, 29
certificate-based, 40
client, 13, 19, 42, 52
clustering, 63, 65
COCA, 39
crashing, 12
CSMA/CA, 12

data replication, 11, 12
data sharing, 7, 12, 19
dCA (distributed CA), 42, 56
DDM (Differential Destination Multicast), 10
delay time-scale, 65
delay-tolerant network, 67
DG (Directional Gossip), 9
DICTATE, 38, 42
diffusion, 11
directed diffusion, 107
dissemination quorum systems, 54
DoS (Denial of Service), 42, 52, 53
DSR, 12, 14, 16, 29, 56

empirical model, 118
energy conserving routing, 63, 65
Ensemble, 8
epidemic model, 11
epidemiology, 9, 23
Erlang distribution, 24, 28

fanout, 9, 15, 26, 30, 44, 51, 55, 57, 58
fast mobility regime, 65
FIFO order, 12, 19
flooding, 7, 9, 10
freshness, 38, 41
freshness degree, 41, 50, 56, 57

gossip, 8, 9, 13, 15
group communication system, 7, 39, 45

HLETDR(hybrid learning-enforced time domain routing), 107

IA (Identification Authority), 38
identity-based, 40, 43, 51
iDoS (inaction DoS), 54
IEEE 802.11, 12, 29, 56
impersonation, 42
intrusion/misbehavior detection, 50
IP multicast, 8

joint authority, 38
joint authority approach, 42, 44

key compromise, 42

lifetime, 63, 65
liveness, 41, 53

MAODV, 9, 10
mCA (mother certification authority), 41, 42
membership management, 8
message overhead, 8
MintRoute, 110
mobile ad hoc networks, 7
mobile sink, 64
mobility management, 7
mobility model, 11
mobility time-scale, 65
MobiRoute, 106
multicast, 7, 12

network lifetime, 64, 69, 85, 107
network load $\mathcal{N}_l$, 13, 26, 28, 29
node mobility, 7

on-demand routing protocol, 14
overhearing, 106, 115

packet delivery ratio, 108
PGP, 37, 39
Poisson process, 24, 28, 56
probabilistic broadcast, 9
probabilistic freshness, 41
probabilistic multicast, 9, 45
probabilistic quorum system, 54
probabilistic quorum systems, 8, 10, 45
probabilistic reliability, 7, 12
public-key cryptography, 37
public-key infrastructure, 37

quorum systems, 10, 19, 23

R^{2}DG (Reliable RDG), 13, 18
RA (Revocation Authority), 38
random graph, 9
random walk agent, 8
random waypoint model, 29, 56
randomized database groups, 11, 23
RDG (Route Driven Gossip), 13, 15, 44
reliability, 7, 108
reliability degree $\mathcal{R}_d$, 8, 12, 13, 26, 27, 29
reliable broadcast, 8
reliable multicast, 8, 12
revocation, 40, 43, 55
robust threshold cryptosystem, 43
RSA, 43

safety, 41, 53
SEAD (scalable energy-efficient asynchronous dissemination), 106
self-organized public-key management, 39
sensor node, 63
server, 13, 19, 42, 52, 54
shared-private, 12
sink, 63
sink mobility, 64
slow mobility regime, 66
source routing, 10
SPKI/SDSI, 39
Spread, 8
state-machine, 10
stateless multicast, 10
strict quorum systems, 10
Sybil attack, 39

threshold cryptography, 38, 39
threshold cryptosystem, 54
token circulation, 8
topology control, 63, 65
TTDD (two-tier data dissemination), 106
TTP (Trusted Third Party), 37

unicast, 8, 10

WSN (wireless sensor network), 63, 105

X.509, 39

JUN LUO

Nationality:	P.R. China	Phone #:	+4121-6934668
Mail Address:	EPFL-IC-ISC-LCA1	Fax #	+4121-6936610
	BC 201 (Bâtiment BC), Station 14	E-mail:	jun.luo@epfl.ch
	CH-1015 Lausanne	Webpages:	http://people.epfl.ch/jun.luo

EDUCATION:

Aug. 2001 – Jan. 2006

PhD in the Doctoral School of Dept. of Communication Systems, EPFL, Lausanne, Switzerland.

Dissertation Title: MOBILITY IN WIRELESS NETWORKS: FRIEND OR FOE
— Network Design and Control in the Age of Mobile Computing
Dissertation Advisor: Prof. Jean-Pierre Hubaux

Oct. 2000 – July 2001

Graduated in the Pre-doctoral School of Dept. of Communication Systems, EPFL, Lausanne, Switzerland. Overall GPA: 5.5 / 6.0

Sept. 1997 – July 2000

Graduated and obtained M.E. (Electrical Engineering) in Dept. of Electrical Engineering, Tsinghua University, Beijing, China. Overall GPA: 88 / 100, Class Rank: 3rd / 32

Sept. 1992 – July 1997

Graduated and obtained B.E. (Electrical Engineering) in Dept. of Electrical Engineering, Tsinghua University, Beijing, China. Overall GPA: 85 / 100, Class Rank: 5th / 34

PROFESSIONAL EXPERIENCE:

July 2001 – Present

Research and Teaching Assistant, EPFL-IC-ISC-LCA1

Engaged in the **Mobile Information & Communication Systems** (**MICS**: http://www.mics.ch/) Project and two courses on mobile and self-organized networks

- **Research activities**
  - **Group Communications in Ad Hoc Networks** (http://gcomm.epfl.ch/): Reliable protocols for supporting many-to-many information disseminations in mobile ad hoc networks, as well as the related services for multicast streaming, data sharing, and public key infrastructure [3, 4, 6, 10, 11].
  - **Wireless Sensor Networks with Mobile Elements** (http://sensemob.epfl.ch/): Exploiting the mobile elements in wireless sensor networks to improving the performance of such networks, such as increasing lifetime and enhancing location-awareness [1, 2, 7, 8].
  - **Vehicular Networks** (http://ivc.epfl.ch/): Reliable information dissemination and node coordination in vehicular networks as well as the corresponding security issues [5, 9, 12].
  - **Testbed Prototyping** (http://wasal.epfl.ch/): Joint efforts on prototyping a sensor/actuator testbed for the MICS community.
- **Teaching experience**
  - **Student project supervision**: Advised six master students by guiding their study of the

literature, their ability of protocol design, and their experiment skills in using network simulators.

- **Teaching assistant**
    - Led a topic on sensor networks in a graduate course: **Self-Organized Wireless and Sensor Networks**, 2003 — 2005. Delivered lectures and supervised related projects.
    - Led exercise sessions in an undergraduate course: **Mobile Networks**, 2003 — 2005. Held office hours, as well as designed homework and examination problems.
    - Led a topic on ad hoc networks in a graduate course: **Cellular and Ad Hoc Networking**, 2002 — 2003. Produced slides and delivered lectures.

Sept. 1995 – July 2000

Research Assistant, System Identification and Security Monitoring Lab., Dept. of Electrical Engineering, Tsinghua University

Engaged in Frequency Characteristic Analysis and Artificial Neural Network Application and System Parameter Identification and Simulation of Control System Elements

Sept. 1998 – July 1999

Teaching Assistant, Dept. of Electrical Engineering, Tsinghua University

Engaged in the course of Principles of Micro-computer

Sept. 1996 – July 1998

Part-time Software Engineer, Beijing Omni-Solution Computer System Integration Institute

Engaged in Software Development and Localization

PROFESSIONAL SERVICE AND ACTIVITIES:

- **Organization co-chair of ACM Workshop on Applications of Mobile Embedded Systems (WAMES 2004), June 6, 2004, Boston, Massachusetts, USA**
- **Reviewer for the following journals and conferences:**
    - ☐ IEEE Journal on Selected Areas in Communications
    - ☐ IEEE Transactions on Parallel and Distributed Systems
    - ☐ IEEE Transactions on Mobile Computing
    - ☐ IEEE Transactions on Vehicular Technology
    - ☐ IEEE Communication Letters, Elsevier Ad Hoc Networks
    - ☐ ACM Transactions on Autonomous Adaptive Systems
    - ☐ ACM Mobile Computing and Communications Review
    - ☐ Wiley's Wireless Communications and Mobile Computing Journal
    - ☐ International Journal of Ad Hoc and Ubiquitous
    - ACM Conference on Mobile Computing and Networking (MobiCom)
    - ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)
    - ACM Conference on Embedded Networked Sensor Systems (SenSys)
    - ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)
    - ACM Workshop on Wireless Security (WiSe)
    - IEEE Conference on Computer Communications (INFOCOM)
    - IEEE Conference on Communications (ICC)
    - Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)
    - Workshop on Trust, Security and Privacy for Ubiquitous Computing (TSPUC)

PUBLICATIONS:

Submitted for publishing

1. Jun Luo and Jean-Pierre Hubaux, "**A Unified Framework for Joint Sink Mobility and Routing to Increase the Lifetime of Wireless Sensor Networks**", Submitted to *the 7th ACM International Symposium on Mobile Ad Hoc Networks and Computing (MobiHoc'06),* 2005
2. Jun Luo, Jacques Panchard, Michal Piorkowski, Matthias Grossglauser, and Jean-Pierre Hubaux, "**MobiRoute: Routing towards a Mobile Sink for Improving Lifetime in Sensor Networks**", Submitted to *the 2nd International Conference on Distributed Computing in Sensor Systems (DCOSS'06),* 2006

Journal Articles

3. Jun Luo, Jean-Pierre Hubaux and Patrick Th. Eugster, "**DICTATE: DIstributed CerTification Authority with probabilisTic frEshness for Ad Hoc Networks**", *IEEE Trans. on Dependable and Secure Computing,* Vol. 2, No. 4, pp311—323, October-December 2005
4. Jun Luo, Patrick Th. Eugster and Jean-Pierre Hubaux, "**Probabilistic Reliable Multicast in Ad Hoc Networks**", *Elsevier Ad Hoc Networks,* Vol. 2, No. 4, pp369—386, October, 2004
5. Jean-Pierre Hubaux, Srdjan Capkun, and Jun Luo, "**The Security and Privacy of Smart Vehicles** ", *IEEE Security & Privacy Magzine,* Vol. 2, No. 3, pp49—55, May-June 2004
6. Jun Luo, Patrick Th. Eugster and Jean-Pierre Hubaux, "**PILOT: ProbabilistIc Lightweight grOup communication sysTem for Mobile Ad Hoc Networks**", *IEEE Trans. on Mobile Computing,* Vol. 3, No. 2, pp164—179, April-June 2004

Conference Papers

7. Jun Luo, Hersh Vardhan and Jean-Pierre Hubaux, "**Non-Interactive Location Surveying for Sensor Networks with Mobility-Differentiated ToA**", In *Proceedings of the 25th Annual Conference of the IEEE Communications Societies (INFOCOM'06),* Barcelona, Spain, April 2006
8. Jun Luo and Jean-Pierre Hubaux, "**Joint Mobility and Routing for Lifetime Elongation in Wireless Sensor Networks**", In *Proceedings of the 24th Annual Conference of the IEEE Communications Societies (INFOCOM'05),* Miami, FL, USA, March 2005
9. Jun Luo and Jean-Pierre Hubaux, "**NASCENT: Network Layer Service for Vicinity Ad-hoc Groups**", In *Proceedings of the 1st IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON'04),* Santa Clara, CA, USA, October 2004
10. Jun Luo, Jean-Pierre Hubaux and Patrick Th. Eugster, "**PAN: Providing Reliable Storage in Mobile Ad Hoc Networks with Probabilistic Quorum Systems**", In *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networks and Computing (MobiHoc'03),* Annapolis, MD, USA, June 2003
11. Jun Luo, Patrick Th. Eugster and Jean-Pierre Hubaux, "**Route Driven Gossip: Probabilistic Reliable Multicast in Ad Hoc Networks**", In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'03),* San Francisco, CA, USA, March 2003

Book chapters

12. Jun Luo and Jean-Pierre Hubaux, "**A Survey of Research in Inter-Vehicle Communications**", In *Embedded Security in Cars – Securing Current and Future Automotive IT Applications*, Springer-Verlag, October 2005