Service Provisioning in Mobile Networks Through Distributed Coordinated Resource Management

Morcos, Hany

Boston University Computer Science Department

http://hdl.handle.net/2144/1717

Boston University
SERVICE PROVISIONING IN MOBILE NETWORKS THROUGH DISTRIBUTED COORDINATED RESOURCE MANAGEMENT

by

HANY MORCOS
B.Sc., Cairo University, 1998
MEE., Rice University, 2003

Submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
2009
Approved by

First Reader
Azer Bestavros, Ph.D.
Professor of Computer Science

Second Reader
Ibrahim Matta, Ph.D.
Associate Professor of Computer Science

Third Reader
Mark Crovella, Ph.D.
Professor of Computer Science
DEDICATION

I dedicate this thesis to my lovely wife Rasha, my beautiful daughter Julia, my beloved mom Samia, my very dear sister Hanaa, and my dear dad Samy who intercedes on my behalf before the throne of GOD.
Acknowledgments

The past five years have been a great learning experience for me. During this period, I received a lot of support from many individuals. First and foremost, my advisors: Azer Bestavros and Ibrahim Matta. I would like to thank them both for they have invested a great deal of time and effort in me to make it possible that I reach this stage in my academic career. They have been more than great teachers; they have been mentors, and dear friends to me. They were always ready to hear my ideas, discuss them, and always point out directions for improvements.

I would like also to thanks professors Mark Crovella, John Byers, Richard West and Steve Homer for serving in my thesis committee.

Over the years, and way before my PhD, I have been receiving immense support, motivation, and understanding from my family in every step of my life. I cannot possibly describe how much affection, love, and care my mother Samia has given me since my first days until now. Her love and care is truly the candle that lights my dark paths in life. My sister Hanaa is my twin, her prayers have supported me when prayers were all I had to lean against. My wife Rasha, is a gift from Heaven; a source of pure love and support. I praise her courage and patience to endure the “PhD experience” with me. My daughter Julia is a blessing from Heaven to Rasha and I. She is also the sweetest motivation that I had to complete my PhD. My father Samy, is my first teacher. From him, I learned how to be a man when I was ten years old.

I would like also to thank my extended family: my aunt Soad, my uncles Samy and Samir, and their families for their love, and support over the years.

My friends helped me get over the hard times, and made the good times even better.
I would like to thank my “Boston Brothers”: George Atia, Mina Guirguis, Karim Mattar and Raymond Sweeha. Also, I would like to thank my pre-Boston fiends: Mark Ayaad, and Amir Botros for their support.

Finally and most importantly, I would like to thank God the Most High, who is the owner of me and every good action that comes out of me. God has shown me time after time the truth of his promises: that “I can do all things in Christ who strengthens me” (Philippians 4:13)
SERVICE PROVISIONING IN MOBILE NETWORKS THROUGH DISTRIBUTED COORDINATED RESOURCE MANAGEMENT

(HANY MORCOS)

Boston University, Graduate School of Arts and Sciences, 2009

Major Advisor: Azer Bestavros, Professor of Computer Science

ABSTRACT

The pervasiveness of personal computing platforms offers an unprecedented opportunity to deploy large-scale services that are distributed over wide physical spaces. Two major challenges face the deployment of such services: the often resource-limited nature of these platforms, and the necessity of preserving the autonomy of the owner of these devices. These challenges preclude using centralized control and preclude considering services that are subject to performance guarantees. To that end, this thesis advances a number of new distributed resource management techniques that are shown to be effective in such settings, focusing on two application domains: distributed Field Monitoring Applications (FMAs), and Message Delivery Applications (MDAs).

In the context of FMA, this thesis presents two techniques that are well-suited to the fairly limited storage and power resources of autonomously mobile sensor nodes. The first technique relies on amorphous placement of sensory data through the use of novel storage management and sample diffusion techniques. The second approach relies on an information-theoretic framework to optimize local resource management decisions. Both approaches are proactive in that they aim to provide nodes with a view of the monitored field that reflects the characteristics of queries over that field, enabling them to handle more queries locally, and thus reduce communication overheads.

Then, this thesis recognizes node mobility as a resource to be leveraged, and in that respect proposes novel mobility coordination techniques for FMAs and MDAs. Assuming that node mobility is governed by a spatio-temporal schedule featuring some slack, this thesis presents novel algorithms of various computational complexities to orchestrate the
use of this slack to improve the performance of supported applications.

The findings in this thesis, which are supported by analysis and extensive simulations, highlight the importance of two general design principles for distributed systems. First, a-priori knowledge (e.g., about the target phenomena of FMAs and/or the workload of either FMAs or DMAs) could be used effectively for local resource management. Second, judicious leverage and coordination of node mobility could lead to significant performance gains for distributed applications deployed over resource-impoverished infrastructures.
## Contents

Acknowledgments \hspace{1cm} v  
Contents \hspace{1cm} ix  
List of Tables \hspace{1cm} xii  
List of Figures \hspace{1cm} xiii  
List of Abbreviations \hspace{1cm} xvii  

1 Introduction \hspace{1cm} 1  
1.1 Motivation \hspace{1cm} 1  
1.2 Controlled Resources \hspace{1cm} 3  
1.2.1 Conventional Resources \hspace{1cm} 4  
1.2.2 Mobility \hspace{1cm} 5  
1.3 Application Domains \hspace{1cm} 5  
1.3.1 Field Monitoring Applications (FMAs) \hspace{1cm} 6  
1.3.2 Message Delivery Applications (MDAs) \hspace{1cm} 7  
1.4 Dissertation Contributions \hspace{1cm} 7  
1.4.1 Conceptual Contributions \hspace{1cm} 8  
1.4.2 Technical Contributions \hspace{1cm} 9  
1.5 Dissertation Overview \hspace{1cm} 10  

2 Distributed Storage Management in FMAs \hspace{1cm} 14  
2.1 Definitions and Problem Statement \hspace{1cm} 15  
2.2 Related Work \hspace{1cm} 17  
2.3 Solution From The Literature: Directed Placement and Retrieval (DPR) \hspace{1cm} 19  
2.4 Shortcomings of DPR \hspace{1cm} 21
# Distributed Mobility Management in MDAs

## 5 Distributed Mobility Management in MDAs

### 5.1 Motivation

### 5.2 Related Work

### 5.3 Constrained Mobility Coordination for Message Delivery (CMC-MD)

### 5.4 Detour for optimized Message Delivery (DMD)

### 5.5 Detour for maximizing Node Encounters (DNE)

### 5.6 Performance Evaluation

#### 5.6.1 Evaluation Using Synthetic Workloads

#### 5.6.2 Trace-Driven Evaluation

### 5.7 Conclusion

## Conclusion

### 6 Conclusion

#### 6.1 Future Work

#### 6.1.1 Load Balancing

#### 6.1.2 Correctness/Security issues

#### 6.1.3 Pricing Models

#### 6.1.4 New Application Domains

## Bibliography

## Curriculum Vitae
List of Tables

4.1 Schedule of $n_1$. ................................................................. 79
4.2 KL distance of resulting distribution when only a percentage of nodes follow
the detours of TFM. ................................................................. 86
4.3 KL distance resulting from applying TFM on cab traces with different values
of maximum length of PFPs $h_{\text{max}}$. ....................................... 90
5.1 Example of a node schedule .................................................. 104
5.2 Costs of edges between vertices in $V_2$ ................................. 104
List of Figures

1.1 Dissertation structure. .......................................................... 10

2.1 Application of Algorithm 2.6 (a) calculating the probability between point location $d$ and the center of the distribution, (b) find $d'$, the distance between the image of the original sample and the center of the field. .................. 27

2.2 Idealized field coverage by four nodes applying QCCM. Notice that any area $A = S_{opt} \times S_{opt}$ will have exactly one sample from each node. .................. 30

2.3 Effect of storage management algorithm on query success ratio for $n$ mobile nodes. Notice RND2, RSM under mobility model 2 (Figure 2.4 right), is better than RND1, RSM under mobility model 1 (Figure 2.4 left), since the earlier is less skewed than the latter. .................. 31

2.4 $p_{ij}$ for the two mobility models used in Figure 2.3 .................. 31

2.5 Effect of APR mechanisms on QSR. ................................. 33

2.6 Effect of sample exchange size: Query success ratio (left) and Query success ratio per unit of energy (right). ................................. 34

2.7 Effect of exchange rate: Query success ratio (left) and Query success ratio per unit of energy (right). ................................. 35

2.8 Effect of communication range: Query success ratio (left) and Query success ratio per unit of energy (right). ................................. 36

2.9 Effect of query precision: Query success ratio (left) and Query success ratio per unit of energy (right). ................................. 37

2.10 Effect of storage size: Query success ratio (left) and Query success ratio per unit of energy (right). ................................. 38
2.11 Effect of sample TTL: Query success ratio (left) and Query success ratio per unit of energy (right). .................................................. 38
2.12 Effect of packet loss: Query success ratio (left) and Query success ratio per unit of energy (right). .................................................. 39
2.13 Effect of query precision in a static network: Query success ratio (left) and Query success ratio per unit of energy (right). ............... 40
2.14 Effects of non-uniform query distribution. Output of the stretching algorithm (left), QSR (Center), success per consumed power unit(Right). ........... 42
3.1 Performance as a function of the storage size for a Gaussian phenomenon (left), and a non-Gaussian phenomenon (right). ....................... 57
3.2 Performance as a function of the variance of the query distribution for a Gaussian phenomenon (left), and a non-Gaussian phenomenon (right). .... 57
3.3 Performance of multiple nodes as a function of the storage size for a random phenomenon (right), cooperation gain (ratio of MSE with a single node and with n nodes). .................................................. 59
3.4 Performance of multiple nodes as a function of the storage size for a random phenomenon (right), cooperation gain (ratio of MSE with a single node and with n nodes). .................................................. 59
3.5 Performance of multiple cooperative nodes as a function of the number of nodes n. ................................................................. 60
4.1 Visualization of a node schedule: Schedule entries are marked with circles. Rectangles mark the legitimate paths a node could take during each journey. 67
4.2 Location \(v_c\), and its neighboring locations (left), \(C_i\): local field view of node \(n_i\) (center), and range of PFP \(F(v_f, h)\), where \(h = 1\). .................. 74
4.3 Location \(v_c\), its neighboring locations \(N(v_c)\), the destination waypoint, and the utility of each location(left), \(P_{best(v_f,3)}\) (center) and \(P_{best(v_f,1)}\) (right). 80
4.4 KL distance of TFM and RND as a function of the number of nodes in the system for different slack values. ................................. 84
4.5 Query success ratio of TFM and RND as a function of the number of nodes in the system for different slack values ........................................... 85
4.6 KL distance of TFM and RND as a function of the maximum slack for different number of nodes. ......................................................... 85
4.7 Query success ratio of TFM and RND as a function of the Maximum slack for different TTL values ....................................................... 86
4.8 Effect of partially following hints on query success ratio for three different values of slack= 0, 20, and 100. ......................................................... 87
4.9 Paths followed by cabs in the SF Bay area, from traces used to evaluate mobility coordination approaches (left) An actual Map of the San Francisco Area (right). Notice the similarity between the original and constructed map. 88
4.10 Example of the distribution we used with the San Francisco cab traces. The lighter the area the higher the coverage percentages \(D(v)\). \(v_1\) in this case is marked by the red (dark) dot in the light area .................................. 90
4.11 QSR of TFM and WAD as a function of TTL of samples (TTL measured in minutes). The graph show results when using two values of the communication range: 600 and 1200 meters. ......................................................... 91
5.1 A three-node schedule (left) with potential encounters 1 through 6 (center) and PEG graph (right). ................................................................. 102
5.2 Using PEG in determining the route of a message \(g\) that originates at \(n_2\) to \(n_1\) at time = 18, (A) applying the five steps, (B) finding the shortest path . 105
5.3 Comparing DMD to OPT, the optimal mobility planning. relative delay (left), relative throughput (right) ....................................................... 107
5.4 Performance of mobility synthesis approaches. ................................. 111
5.5 Effect of following hints with a given probability on delivery ratio and delay. 112
5.6 Effect of a ratio of nodes following hints on throughput and delay. ......... 113
5.7 Comparison of throughput and delay of DMD, DNE, data mules, and ferries. Number of nodes = 5. ....................................................... 113
5.8 Comparison of throughput and delay of DMD, DNE, data mules, and ferries.

Number of nodes = 10. ............................................. 114

5.9 DNE vs WAD using Groovenet. ............................................. 117

5.10 DNE vs WAD using SF cab traces. ............................................. 117

5.11 Earliest and latest times that node $n_1$ could visit various locations during its first journey (start $(\tau_{11} = 1, l_{11} = 8)$, end $(\tau_{12} = 20, l_{12} = 13)$). UR stands for “unreachable” ............................................. 124
# List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2UA</td>
<td>Two-phase Utility Assignment</td>
</tr>
<tr>
<td>APR</td>
<td>Amorphous Placement and Retrieval</td>
</tr>
<tr>
<td>AR</td>
<td>Autoregressive</td>
</tr>
<tr>
<td>BLSE</td>
<td>Bayesian Least Square Error</td>
</tr>
<tr>
<td>CBT</td>
<td>Correlation Based Technique</td>
</tr>
<tr>
<td>CDF</td>
<td>Cumulative Distribution Function</td>
</tr>
<tr>
<td>CMC-FC</td>
<td>Constrained Mobility Coordination- Field Coverage</td>
</tr>
<tr>
<td>CMC-MD</td>
<td>Constrained Mobility Coordination- Message Delivery</td>
</tr>
<tr>
<td>DCR</td>
<td>Data-Centric Routing</td>
</tr>
<tr>
<td>DCS</td>
<td>Data-Centric Storage</td>
</tr>
<tr>
<td>DEBT</td>
<td>Distributed Entropy Based Technique</td>
</tr>
<tr>
<td>DMD</td>
<td>Detour for maximizing Message Delivery</td>
</tr>
<tr>
<td>DNE</td>
<td>Detour for maximizing Node Encounter</td>
</tr>
<tr>
<td>DPR</td>
<td>Directed Placement and Retrieval</td>
</tr>
<tr>
<td>DTN</td>
<td>Delay Tolerant Network</td>
</tr>
<tr>
<td>FIFO</td>
<td>First In First Out</td>
</tr>
<tr>
<td>FMA</td>
<td>Field Monitoring Application</td>
</tr>
</tbody>
</table>
GPSR . . . . . . . . . . . . . . . . . . . . . . . . . . Greedy Perimeter Stateless Routing

ISS . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . Inter-Sampling Spacing

KL . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . Kullback-Leibler

LLSE . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . Linear Least Square Error

LSE . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . Least Square Error

MDA . . . . . . . . . . . . . . . . . . . . . . . . . . . . . Message Delivery Application

MLT . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . Minimum Latency Tour

MSE . . . . . . . . . . . . . . . . . . . . . . . . . . . . Mean Square estimation Error

PEG . . . . . . . . . . . . . . . . . . . . . . . . . . . Potential Encounter Graph

PFP . . . . . . . . . . . . . . . . . . . . . . . . . . . . Potential Future Path

PLP . . . . . . . . . . . . . . . . . . . . . . . . . . . Packet Loss Probability

QCCM . . . . . . . . . . . . . . . . . . . . . . . . . . Query-Cognizant Cache Management

QSR . . . . . . . . . . . . . . . . . . . . . . . . . . Query Success Ratio

RAM . . . . . . . . . . . . . . . . . . . . . . . . . . Random Access Memory

RND . . . . . . . . . . . . . . . . . . . . . . . . . Random Mobility strategy

RR . . . . . . . . . . . . . . . . . . . . . . . . . . Responsibility Regions

RSM . . . . . . . . . . . . . . . . . . . . . . . . . Random Storage Management

SEAD . . . . . . . . . . . . . . . . Scalable Energy-efficient Asynchronous Dissemination

SNR . . . . . . . . . . . . . . . . . . . . . . . . . . . Signal to Noise Ratio

SPE . . . . . . . . . . . . . . . . . . . . . . . . . . . Success Per unit Energy

STC . . . . . . . . . . . . . . . . . . . . . . . . . . Spatio-Temporal Correlation
TFM .......................................................... Targeted Field Monitoring
TTDD .......................................................... Two-Tier Data Dissemination
TTL .......................................................... Time To Live
WAD .......................................................... Wait At Destination
WAS .......................................................... Wait At Source
WSN .......................................................... Wireless Sensor Network
Chapter 1

Introduction

1.1 Motivation

Traditionally, service provisioning to mobile users is done in a centralized fashion, in which, the service is presented to interested users through direct interaction with a single service provider. For example, each cell phone user has to directly interact with a wireless phone service provider to gain access to the service.

Resource management in such systems, although not trivial, is considerably facilitated by the fact that all resources are under control of a single party. In the cell phone example, to support this service, a service provider would have to setup the needed infrastructure: base stations, base station controllers, packet data serving nodes, home agents, foreign agents, etc. All of these resources are centrally controlled by this provider. Due to their centralized nature, scalability of services provided in such a model is always an important issue to address. Careful system dimensioning and resource planning is needed to make sure that the deployed infrastructure is enough to support the current number of users.

The need for laying down infrastructure (e.g., in new places) to help expanding services has always been a hurdle in building large-scale systems in this model. Generally, putting infrastructure in place to support service provisioning is costly for a number of reasons. One of the reasons is the cost of the infrastructure itself. Another reason is the associated
cost to put it in place (e.g., digging tunnels, building towers, etc.). Moreover, some other considerations (e.g., impossibility to put up wireless towers in a historic site) might also set back the process. All these issues call for a new model of service provisioning in mobile networks.

The pervasiveness of personal computing devices and embedded sensors offer an unprecedented opportunity for a new model of service provisioning in mobile networks. In the new model, provisioning of services is done in a distributed manner, rather than a centralized one. More specifically, the “service” is provided to users through cooperative sharing among them, where each user is willing to contribute a small fraction of her own resources to be a part of the distributed system. Coordinated management of fractions of resources to provide distributed services is at heart of the contributions of this thesis contributions.

To illustrate the idea, consider a traffic congestion monitoring application in a given section of a metropolitan area. In order to provide such a service according to the traditional model, some party would have to deploy a dense sensor network (of e.g., cameras) in the target city. This party would be also responsible for maintaining such a network, and making sure that its output (i.e., reports about traffic status) is accessible to answer queries of interested users. As we pointed out, this model has multiple limitations. First, there is the cost of network setup and maintenance, then, the cost of data management and accessibility. Finally, the need for a dense deployment in order to be able to cover all locations of interest (e.g., intersections) in the target city.

Providing the same service under our proposed model is radically different. Since density (of users) could be used as an indication of congestion, a density inference protocol could be used to detect congestion. Hence, to support this service, we need to deploy an application on each cell phone to detect the number of other cell phones in its vicinity. If this number is small, then, there is probably no congestion, and vice versa. A cell phone could keep “samples” representing road congestion statuses sampled in the past, along with the time and location of each sample. Such samples could be useful to handle future user queries
about these respective portions of the road. \(^1\) The beauty of this model is that, deploying such a system on devices that are in close proximity of users for prolonged periods of time (\textit{e.g.}, cell phones, or handheld devices) would be ideal, since users could use their own devices to interact with the system and access its services, \textit{e.g.}, to query the state of traffic in some street.

In this example, although the latter model rids the service provider of laying down an infrastructure and worrying about central data management and accessibility, it does introduce new issues that needs proper handling, if the provided service is to be of any value. For example, in the traditional model, query handling is done centrally by the system, assuming that a central facility is able to collect samples of all sensors. In our proposed model however, we need to design some distributed data management scheme that decides which specific traffic reports are stored at which cell phones. In this case, obviously, handling users’ queries is a system-wide effort, not a single node’s mission. The reason is that, a user is likely to be interested in finding out the state of a road that she has not already visited, hence, it is probable that there would be no sample in her local storage to answer such a query. Therefore, the entire system, composed of all participant cells phones, should cooperate to find answers to user queries. Another problem is the limited resources offered up by users. The system has to cleverly manage these resources to maximize their utility. The following subsection address this problem in more details.

### 1.2 Controlled Resources

In our model, providing distributed services would require allowing the distributed service to control some resources of the host mobile computing platform. Generally, it is conceivable that the performance of a distributed service is immensely affected by the nature and the capacity of resources that it controls. In the example of distributed traffic congestion monitoring, the service would require temporary control over the CPU to run the density

\(^1\)Clearly, we have to handle issues of freshness of samples, since, for example, a congestion-free road thirty minutes ago, is not necessarily clear now.
inference protocol. This protocol would probably need access to the antenna and the communication subsystem of the cell phone. Also, to enable storing past samples, the system would also need to control a limited fraction of the storage available on the cell phone. In some situations, if possible, the service might control the host mobility to sample some specific location.

In this thesis, we consider allowing the service to control the following resources:

1.2.1 Conventional Resources

Examples of conventional resources are storage, battery power, or the ability to use an attached sensor. We study controlling these resources in a setup where nodes participating in the distributed system are assumed to be embedded on thin devices. By thin devices, we mean devices that are (relatively) limited in their resources and capabilities. Cell phones, handheld devices, and wrist-watches are all examples of thin devices. In such a setting, energy conservation, smart storage management and judicious use of system resources in message routing are important factors in system design. A representative example of such a setting is embedded wireless sensor networks, and a representative service is distributed field monitoring. In such a service, different users would collect samples from different field locations and keep them (either locally or remotely). Then, these samples are later used to answer users queries about the target phenomenon. Clearly, the goal of the service is to maximize the number of answered queries. In order to achieve this goal, the host devices will have to grant control over some of their resources to the field monitoring service. For example, the service will have to have control over a fraction of the device storage in order to keep samples. It could also have control over the sensor, i.e., it gets to decide when to sample the current location of the user. Since, these resources are clearly limited, shrewd management is crucial to maximize their utility, and optimize performance of the system.
1.2.2 Mobility

One of the original contributions of this thesis is to recognize users’ mobility as a valuable resource to be managed. Commonly, it is assumed that the mobility model of users is controlled by an unknown, exogenous process, that is, at many times, unjustified (e.g., random waypoint mobility model). While this assumption may make sense in some settings, it its too restrictive in general. For example, consider the case where some node needs to go from location $A$ to location $B$, and there are multiple routes to do this, and all of these alternatives have, roughly, the same cost (e.g., path length). In such a case, mobility of nodes could be actively orchestrated in order to simultaneously satisfy external scheduling constraints of the mobile nodes (e.g., get to location $B$ by some given deadline), and optimize performance of the distributed service that depends on the fine details of the nodes’ mobility (i.e., which specific route the node takes to get to location $B$). We illustrate the idea with another field monitoring example.

In some setups, sensors are embedded into thick devices, more powerful platforms that are resource-rich (e.g., automobiles, war tanks, etc.). Here, the goal of the distributed service is to monitor a given field with specific monitoring distribution. This distribution defines the percentage of time different field locations should be monitored, by at least one node. In this setup, saving conventional system resources is not the prime goal (although a favorable behavior), however, achieving the required monitoring distribution of the field is. We assume that each node’s mobility is governed by a schedule that defines the general mobility behavior of this node while allowing some level of slack to these nodes. In this case, we show that it is possible to coordinate utilizing slacks of different nodes in order to achieve the required monitoring of the field, while satisfying schedules of all nodes.

1.3 Application Domains

We evaluate the premise of distributed service provisioning in two different application domains. Specifically, Field Monitoring Applications (FMAs), and Message Delivery Ap-
plications (MDAs).

1.3.1 Field Monitoring Applications (FMAs)

Traditional research efforts in field monitoring have focused on spraying a dense (static) wireless sensor network (WSN) over the target field. In these efforts, a unique party (referred to as the *sink*) in the network is designated as the originator of all queries and receiver of answers. Hence, to support monitoring the field, these efforts establish a forwarding structure from all sensors to the sink. Resorting to flooding to decide the best forwarding paths from each sensor to the sink could be justified, since the initial high price of discovering such forwarding paths is amortized over the lifetime of the network. On the other hand, power conservation was the primary goal in designing WSNs protocols. The motivation behind this is that, sensors are assumed to be static elements, hence they are unreachable once deployed. When a sensor battery dies, this sensor becomes nonfunctional and could affect the connectivity of the entire network. Partitioned networks are not expected to perform their intended missions efficiently.

In services that we envision, many assumptions are different. First, we assume that all nodes are mobile and autonomous. This commands a change in the solution approach as follows. First, since nodes are mobile, the routing problem could not be solved “once and for all”, since discovered paths could become invalid due to mobility. Second, a resource-wasteful routing scheme (*e.g.*, flooding) could not be employed to discover routes and forward queries to other nodes; since repeated applications of this scheme might be unavoidable due to the constantly changing topology. Third, although host mobility does not affect the rate of power consumption of a sensor, it makes sensors temporarily reachable/unreachable. Hence, power efficiency is a desirable feature, however, it is not the primary requirement in designing data management protocols in FMAs. Finally, assuming that any node could pose a query to the system (due to the autonomy assumption), mandates that the system be able to route data *to any* node, which is a harder problem compared to being able to route to the sink only.
Furthermore, we assume that nodes offer up a small fraction of their resources to be under the control of the FMAs. These resources include conventional resources (e.g., storage, sensor), and mobility of the host. The goal of the FMAs is to manage whatever resources under its control in order to be able to answer as many queries as possible (or as accurately as possible).

1.3.2 Message Delivery Applications (MDAs)

We study the problem of message delivery in delay-tolerant networks (DTNs). In such systems, it is assumed that, almost always, there is no end-to-end path between sources and destinations. Hence conventional ad hoc routing techniques will not work. Each node in this system has a number of messages to deliver to other nodes. Although these messages do not require immediate delivery (due to lack of end-to-end paths), the goal is to minimize the average delay of message delivery. Routing in DTNs means that a message could be hosted by multiple carriers before reaching its final destination. To achieve this goal, mobility of nodes is exploited to circumvent the lack of an end-to-end path. A store-carry-and-forward model is adapted to deliver messages to their destinations minimizing the delay of delivering each message [GGT06, SH03b, SRJB03, BLB02, LMZ+06, WW06, DFL01].

We assume that the MDAs has control over the host mobility, and that this mobility is externally controlled by a schedule. Then, we show that encounters between members of the network could be orchestrated in order to help optimize performance of the MDAs, while satisfying schedules of all nodes.

1.4 Dissertation Contributions

The contributions of this dissertation could be classified into two categories: conceptual and technical.
1.4.1 Conceptual Contributions

Identifying a new model of service provisioning in mobile networks

We propose new systems that are composed of multiple autonomous parties, each with an attached computing platform (e.g., cell phone, or embedded sensor). A fraction of resources of these platforms are coordinated, so that the entire system could provide its users with a useful service that is not attainable by a single node. Unlike, previous models, there is no “server” that could replace the effort of the entire distributed system. Systems that we propose are resilient to asynchronous addition and deletion of members.

These systems depend on autonomous mobility of their hosts as a conduit to achieve their goals (e.g., sampling different field locations, or encountering other nodes to help messages delivery). This model taps the enormous potential of the high penetration ratio of handheld and embedded computing and sensing devices.

Proposing a new model for distributed field monitoring

Conventional research in both static and mobile WSNs designates a specific party in the network as the sink, where all queries about the field originate and all answers are routed back. We propose a new model, where all nodes are peers; all of them monitor the field, and any one could pose queries to the system, and get answers from it. Handling queries in the new model poses a harder challenge, since the system should be able to route answers to any node in the system, rather than the sink only.

Identifying autonomous mobility of nodes’ as a system resource in DTNs and WSNs

Conventional research in DTNs deems mobility of nodes as a constant input. Research efforts in DTNs extract node encounters induced by the given mobility, and strive to develop routing protocols, which decide which messages should be forwarded to which neighbors upon an encounter. In this thesis, we study nodes’ mobility as a flexible parameter that could be fine-tuned in order to optimize performance of a MDA. We show that coordinating such mobility has enormous potential in improving performance of MDAs in DTNs.

In the context of FMAs, some research efforts targeted actively mobilizing some elements
(e.g., robots) to achieve a specific (e.g., uniform) coverage of the field. In these efforts total control over the nodes mobility is assumed. In FMAs that we consider, we address a more general problem in which mobility of nodes is externally constrained by a schedule featuring some slack. We show that in the context of FMAs, mobility could be tuned to achieve a given field monitoring, while satisfying external scheduling constraints.

1.4.2 Technical Contributions

*Developing a novel data management framework to support distributed field monitoring* 

In order to support FMAs, we develop data management techniques that plan storage and retrieval of samples in the system. We devise novel query-cognizant storage management algorithms to maximize utility of the available limited storage. We also develop an information-theoretic framework to handle the problem of sensor management. We show that techniques that we develop achieve superior performance compared to techniques proposed in the literature in terms of success in field monitoring, and saving system resources. Our solutions are proactive in that they aim to provide nodes with a view of the monitored field reflecting the characteristics of queries over that field, enabling them to handle more queries locally, and save communication overhead.

*Designing novel mobility strategies to support FMAs and MDAs*

We show that the problem of mobility planning is NP-complete in FMAs, and NP-hard in MDAs. As an approximation, we propose two types of mobility coordination strategies. The first is a distributed mobility strategy that handles mobility in FMAs to achieve a specific monitoring of a field. We show that performance of this model is superior in terms of achieving the required monitoring distribution, and in handling user queries compared to naïve mobility strategies. It is worth noting that, previous research efforts in mobility management to achieve field coverage develop models that move nodes from one static configuration to another static configuration. More specifically, the steady state of nodes, under these models, is static. This is due to the assumption that nodes’ density is very high that a static node configuration is enough to achieve the required coverage. Our model
is unique in that, the steady state of nodes is *mobile*. To our knowledge, this is the first model to do this. The power of this model is that, it leverages temporal correlation of the monitored phenomenon at any location, and moves nodes to cover new locations, achieving superior coverage over space and time.

The second type of mobility strategies that we develop aim to plan mobility of nodes in order to optimize performance of a MDAs. It is worth mentioning that, our problem definition is powerful enough to model previously proposed techniques for message delivery in the literature (*e.g.*, message ferries, and data mules). In that regard, we propose a centralized algorithm and a distributed algorithm to solve the problem. The mobility strategies that we propose do not depend on external helping elements, *e.g.*, message ferries, or data mules, to help in the MDA. However, we compare our models to these techniques. Interestingly, we show that our models, without help of external helper elements, achieve lower message delay, and higher delivery ratio, compared to message ferries and data mules. This underscores superiority of our models, and shows that cooperation between autonomous members of a distributed system has enormous untapped potential.

### 1.5 Dissertation Overview

Figure 1.1 visually depicts the structure of this Thesis. We progressively assume that the distributed service has control over more resources.
In Chapters 2 and 3, we discuss distributed service provisioning by managing conventional resources, while in Chapters 4 and 5 we address distributed service provisioning by managing nodes’ mobility.

In Chapter 2, we address supporting FMAs. In this chapter, we assume that the distributed service is allowed to control a limited fraction of the host storage. Neither the sensor scheduling nor the host mobility is controlled by the service. The goal of this system is to answer as many user queries as possible consuming the least amount of resources. To achieve this goal, we design APR [MBM08b], a novel data management technique that avoids multi-hop communication, and rather depends on direct communication between nodes. To achieve its goals, APR depends on two main mechanisms: query-cognizant cache management, and sample diffusion. We show through analysis and simulation that APR is superior to techniques in the literature in terms of query success ratio and consumed power.

In Chapter 3, we adopt similar setup with minor changes. We assume that the distributed service has control over when to use the sensor to sample the current host location. We also assume that there is some form of correlation between the values of the target phenomenon at different field locations at different times. The goal of the system in this case is to estimate answers to queries minimizing the estimation error. We design an information-theoretic framework [MABM08] to solve problems of: sensor management, storage management, and query handling. Based on this framework, we propose two techniques with different assumptions about the accessible information. We show that our techniques provide much more accurate query answers compared to random caching.

In Chapter 4, we address the problem of mobility management to support FMAs. We assume that there are no constraints on the storage nor sensor scheduling of the host. However, mobility of the host is constrained by an external schedule that specifies, for each node, a list of points in the space-time plan that must be visited by this node. We assume that the distributed service has limited control over mobility of each node, provided that schedules of each node are satisfied. The goal of the system, in this case, is to manage mobility of nodes in order to simultaneously achieve some given monitoring distribution of
the field, and satisfy schedules of all nodes. First, we show that this problem is NP-complete, then, we propose TFM, a distributed mobility coordination strategy to solve the problem. We show that TFM achieves very close distribution to the required one. We also show that, under TFM, nodes achieve, at least, 2-fold improvement in the query success ratio compared to that under the random mobility strategy. Furthermore, we confirm the premise of TFM using cab traces from the San Francisco area. results of the latter evaluation is very close to that of the first, underscoring potential of TFM.

In Chapter 5, we study mobility management to support MDAs in DTNs. We follow a similar setup to that of Chapter 4. In this setup, each node has a number of messages to be delivered to other nodes in the system. The goal is to plan mobility of the nodes in order to minimize the average delay of messages delivery. We show that this problem is NP-hard, then, propose two heuristic solutions [MBM08c]. The first is a centralized workload-aware solution that plans mobility to explicitly minimize delay of delivering each message. The second heuristic is distributed and workload-oblivious solution. It depends on increasing node encounters, hoping that it would create useful encounters to minimize delay of delivering the messages. We compare the two heuristics to the random mobility strategy, message ferries, and data mules. We show that these heuristics are superior to other mobility strategies. Similar to Chapter 4, we use the same cab traces to evaluate our heuristics. The results show that the distributed version is superior to random mobility.

Chapter 6 concludes the thesis by stressing the the two main messages of our work.

The first message is that, in managing the limited resources under their control, the distributed systems that we develop make a crucial use of different pieces of information. This knowledge helps to immensely improve performance of the provided distributed services.

In the context of FMAs, the system uses characteristics of the target phenomenon, as well as the workload. The workload in this case is represented by the specific pattern of users’ interest in the field to manage limited storage. The characteristics of the target phenomenon define the amount of information provided by samples collected at a certain location and certain time about the value of the phenomenon at the current time, at any field
location. While the workload defines which areas of the field are high-demand. Combining the two pieces of information, the system would give higher priority to samples which provide more information about high-demand areas compared to other samples which give less information about such areas. Likewise, in the context of MDAs, knowledge of the workload greatly improves performance of the system. The workload in this case is the source, destination and origination time of each message. Acquiring, and leveraging such knowledge enables distributed systems to greatly improve performance of MDAs.

The second message is about users autonomous mobility. First, we recognize this mobility as a valuable resource worthy of management and coordination. Furthermore, we show that careful coordination of such mobility has a huge untapped potential to improve distributed service provisioning in both FMA, and MDA. We show that such coordination is so powerful, that its effects surpass that of using external helper nodes without relying on mobility coordination.
Chapter 2

Distributed Storage Management in FMAs

Consider an infrastructure of embedded sensors (in e.g., wearable computers), that are only turned on at specific points in time (e.g., once every 5 minutes), for the sake of preserving battery power. Can we use this infrastructure to build a distributed system that enables users to query the target phenomenon (i.e., the phenomenon sampled by the sensors) at any nearby location? If yes, what are the main components of the system? How will the collected samples be managed? Also, it is perceivable that such sensors would only have access to a limited storage. In such a case, how should this storage be managed? When a user submits a query, how should the system react? What is the overhead of such a system in terms of consumed power? What are the different parameters in the system? And, how do they affect the its performance?

This Chapter answers these questions. We start by giving a detailed problem definition. Then, we survey research efforts related to this problem, and, discuss directed placement and retrieval (DPR), an approach based on the current literature. We point out shortcomings of DPR, then propose Amorphous Placement and Retrieval (APR). We discuss the main mechanisms embodied in APR, and their theoretical foundation. Then, we present analysis that confirms the premise of the storage management technique employed by APR. Finally, we evaluate the performance of APR and DPR, pointing out the strengths of each technique. We conclude this chapter by highlighting the salient features of APR.
We summarize contributions of this chapter as follows.

- We propose the problem of distributed data management in support of distributed field monitoring in mobile sensor networks.

- To solve this problem, we propose Amorphous Placement and Retrieval (APR), a distributed light-weight algorithm that avoids multi-hop communication and depends on mobility to save communication overhead. Moreover, APR uses the presented workload (i.e., distribution of user queries) in order to guide its storage management decisions.

- We adapt a greedy algorithm that solves the problem of mutually distance sampling to handle storage management in APR. We also propose the sample diffusion mechanism to diversify contents of storage of different nodes.

- Using an analytical model, we quantify the gain attained by adapting QCCM, our storage management mechanism over random cache management.

- We evaluate APR versus DPR (Directed Placement and Retrieval), and show that in most operational conditions, APR has higher query success ratio than DPR, at a less cost of communication overhead.

2.1 Definitions and Problem Statement

**Basic Assumptions:** We assume that $n$ nodes move independently and autonomously. In particular, node mobility is not driven by the need to effectively sample the field. We assume that nodes know their locations, either relative or absolute. We assume that storage devoted to the field monitoring service is of size $c$, and is limited, $^1$ necessitating the use of

$^1$RAM size in today’s hand-held devices is limited—typically around 64KB to 128KB. This RAM has to fit a lot of OS modules and programs, hence what is left for applications’ data is much less. Second, many sensor modalities require significant storage per sample (sensory reading). For example, in an imagery sampling application, if each image is 1.5KB, then only few tens or hundreds of samples could be kept in the cache given today’s standards. Lastly, memory chips of smaller sizes need less energy to refresh, which makes them more suitable for hand-held devices.
a storage management strategy. We also assume that nodes have unique known IDs.

**Data Sampling:** We assume that mobile nodes sample the field according to a Poisson process. Nodes collect spatio-temporal samples, *i.e.*, every collected sample is associated with an \((x, y)\) coordinates, along with a time-stamp to indicate both the sample’s location and “age”.

**Query Origin and Target:** While roaming the field, users may become interested in querying the value of the sampled phenomenon at a remote location in the field. Such queries are submitted to the system through the *query origin* (the node associated with the inquirer’s device). We denote the remote location in which the user is interested by the *query target*. Different applications may exhibit different distributions of query origins and query targets. While the distribution of query origins may reflect the mobility model of users,\(^2\) the same cannot be said about the distribution of query targets. In particular, *a priori* knowledge of the distribution of query targets could be used to *improve* the performance of the system (*e.g.*, by allowing nodes to give different weights to caching entries based on the spatial coordinates of the entries) [LXC\(^+\)05]. We, first, address the problem when nodes are equally interested in the entire field (*i.e.*, query targets follow a spatial uniform distribution over the field). Then, we consider the case when this interest is skewed (*e.g.*, more query targets in the center of the field).

**Query Precision:** One particularly important parameter of queries is the tolerable inaccuracy in the result. We assume that queries target a specific location in the field along with some desirable *precision* \((\ell)\), which constrains how far the samples used to answer the query could be from the query target. We can think of a circle whose center is the query target and whose radius is the value of the precision. In this case, the query answer is any sample that is collected from within this circle.

We assume that the value of the precision is phenomenon-specific, hence, all queries inquiring about the same target phenomenon share the same value of the precision requirement. Introducing query precision allows the support of applications in which queries might

\(^2\)For example, a mobility model that results in higher concentration of users in a particular part of the field will result in a higher number of queries originating from that part of the field.
target locations in the field where no readings were collected. The value of the precision constraint $\ell$ is defined as part of the query.

**Data Freshness:** In order to be useful, returned query answers should not be “stale”. Thus, we assume that a well-defined mechanism exists via which nodes are able to discard obsolete samples (e.g., a time-to-live (TTL) for each sample), or otherwise assign a marginal utility to keeping one sample versus another – i.e., an aging mechanism. Clearly, choosing the right parameters for aging depends on the stationarity (or time-scale of change) of the target phenomenon sampled by the sensors.

### 2.2 Related Work

There has been extensive research on data management and query resolution in sensor networks. Applications where sensors are mobile and produce large-size samples (e.g., cameras) make these problems more challenging. We broadly categorize research in this area based on whether the network or the users (sinks) are mobile.

**Static/mobile network, static sinks:** Data Centric Routing (DCR) and Data Centric Storage (DCS) fall into this category. DCR, such as Directed Diffusion [IGE00], employs flooding. The overhead of flooding is justified by assuming long-running queries. A sink floods its query/interest, and targeted sensors respond. DCS [SRK+03] attempts to avoid flooding altogether, hence is suitable for one-shot queries. DCS employs hashing to associate a data item with a specific location in the field. A geographic routing protocol, such as GPSR [KK00], is used to transport a query/answer for a data item. Directed Placement and Retrieval (DPR), a variant of DCS that handles mobility, will be described in the following section. We use DPR as representative of solutions based on the current literature.

Mobility challenges the design of both DCR and DCS—it continually changes the topology underneath the routing protocol.

Proposals such as data mules [SRJB03], smart tags [BLB02] and mobile relays [WSC05], employ mobile elements as relays among static sensors/sinks. Another delay-tolerant scheme
was proposed by Small et al. [SH03b] for a whale monitoring application.

**Static network, mobile sinks:** Both TTDD [YLC+02] and SEAD [KAK03] fall into this category. They target long-running queries from mobile users. Essentially, these schemes can be thought of as a hierarchical extension to Directed Diffusion, whereby the effect of sink mobility is localized.

**Mobile network, mobile sinks:** The work by Zaho et al. [ZAZ04] and Lee et al. [LMZ+06] fall into this category. The first proposal employs powerful message ferries to act as relays. In the latter proposal, each node keeps track of its recent contacts, along with their sensed events, and employs last-encounter routing to locate a target node. In a similar setting of delay-tolerant applications, Wang et al. [WW06] employs history-based forwarding and buffer management.

The above schemes target applications, in which all collected samples should be forwarded to sink(s). Hence, the notion of queries is different than the model we consider. Furthermore, in such systems, there is a clear separation in functionality between nodes, i.e., some nodes are for collecting samples (sensors), while others are for collecting results (sinks). This is different from the model we propose, in which nodes are peers that are equal in duties (collecting samples) and rights (getting query answers).

Moreover, Most of these research efforts do not address the problem of limited storage, rather, they only concentrate on the problem of data placement, assuming existence of enough storage. The two data management schemes discussed in this Chapter address the problem of limited storage.

Another related body of work, is data management in ad hoc networks [YC04, Har01]. The main difference between these efforts and ours is that: usually, in ad hoc networks, the set of data objects, in which users are interested, is limited with a known source for every object, which is not the case in mobile sensor networks, since any node can sample any field location. Moreover, correlation between different data objects are usually ignored. Hara et al. consider this correlation in [HMN04]. However, the correlation structure they consider is random. In our case, the correlation between samples is manifested in utilizing
samples to answer queries targeting close-by (i.e., within the precision constraints) field locations. Hence, the correlation has a physical interpretation and is not random. Finally, in these research efforts, it is assumed that there is a “server” that is able to satisfy queries of all nodes. The goal of the research is to alleviate the load on this server, by handling some requests using neighbors storage. In the service model that we consider, there is no such server, rather the “service” is provided to users through cooperation and distributed coordination of limited resources of these users. The problem we consider is clearly more challenging, due to lack of central service provider.

The sample diffusion mechanism used by Amorphous Placement and Retrieval (APR) (described in Section 2.6), the data management algorithm that we propose, resembles gos- siping and randomized routing [KMG03, Cho, VJvS03]. In these efforts multi-hop routing is avoided, and mobility is deployed instead. APR, also, borrows ideas from the summary cache [FCAB00] by Fan et al. to maximize its gain.

2.3 Solution From The Literature: Directed Placement and Retrieval (DPR)

Based on Data-Centric Storage (DCS), Directed Placement and Retrieval (DPR) plans storage of each collected sample. The identity of the node where a given sample is stored is independent of the sensor collecting this sample, hence, this approach mandates transporting samples from the collecting sensor(s) to the storage sensor(s). We call the latter the home of the sample. DPR has two main questions to resolve: (1) how to plan sample placement, and (2) how to transport samples from the collecting sensors to the home sensor.

Hashing is a widely used technique to answer questions like the first. In systems like [RKY+02] and [SH03a], it was proposed to hash samples (based on the sample name) to some location in the field. Nodes closest to that location are considered the home nodes for these samples. To account for mobility, sample replication was proposed to maintain the semantics of the approach (i.e., hashing any sample e to get a field location, sensors closest
to a hashed location are the home sensors for $e$). Queries are, likewise, hashed using the same function to get the location where answers to the query should be found.

To answer the second question, geographic routing techniques (e.g., GPSR [KK00]) were employed. Assuming nodes are aware of their own location and that of their neighbors, GPSR can be used to route packets to the node closest to a given location in the field.

DPR uses a slightly different approach from the one described above. Instead of hashing samples and queries to field locations, DPR hashes them to node ID's. Given a field location, the hash function returns an ID of a home node responsible for storing samples from this location, and answering queries about this location. Samples (queries) are then routed from the sampling (inquiring) node to the home node. When receiving a query, the home node replies with an answer to the query. The query answer is routed back to the inquirer node.

The specific hashing function that we used is based on the sample location. More specifically, we divide the field into Responsibility Regions (RR for short), each region is assigned to a node. All samples collected by any node at the RR of node $z$ are forwarded to $z$. $z$ manages its cache such that, it keeps samples collected only from its RR. Queries are likewise hashed, based on the query target to get the ID of the home node. Queries are forwarded to their home node, and answers are routed back to inquirer. Having in mind that nodes only keep samples form their respective RR, the cache management technique used to manage these samples does not have a huge impact on performance. The reason is that the area of RR is usually much smaller than that of the field, that the effect of the cache management is not really noticeable on performance.

It is worth noting that, DPR-like algorithms are not originally designed to handle mobile nodes. However, to give DPR the benefit of doubt, we assume that, under DPR, any two nodes $a$ and $b$ route packets (samples, queries, and query answers packets) between them on the optimum route found by applying Dijkstra’s algorithm. Dijkstra’s algorithm requires instant knowledge of the entire network topology — a piece of information that many realistic systems would lack. Therefore, results reported in this chapter should be viewed as providing an upper-bound on any realistic implementation of DPR algorithms.
2.4 Shortcomings of DPR

As described, DPR has the following characteristics/shortcomings:

- It depends on multihop communication in transporting samples, queries, and query answers. This has the following consequences:
  - It usually has higher communication overhead compared to direct communication.
  - It requires the underlying network to be very well-connected in order for the routing process to succeed. DPR is not expected to perform well in loosely-connected networks.
  - It makes DPR more sensitive to packet losses.
  - In order to figure multihop routes between nodes, DPR has to operate an ad hoc (or a geographic) routing protocol. This adds more storage overhead in terms of storing a routing state in each node. It also adds communication overhead; due to the exchange of routing packets in order to update the routing state at each node. Moreover, the performance of DPR will be upperbounded by that of the routing protocol it uses.

- To satisfy the semantics of DPR, each sample has to be stored in its home node. If the rate of sampling the field is high compared to that of querying it, DPR would have to transport many samples, not all of which are useful to query answering, to their home nodes. This behavior is, obviously, not the optimum in terms of conserving battery power.

- The notion of “planning” storage of samples and partitioning the field between nodes makes adding/deleting nodes to/from the system not seamless. More specifically, the system has to have a distributed mechanism to assign a responsibility region to each node upon joining. It also has to have a distributed process to re-assign a responsibility region to another node, when a node leaves the system.
These issues with DPR motivate our proposal of Amorphous Placement and Retrieval (APR). First, we briefly explain a theoretical background that APR depends on, then we give the details of APR.

2.5 Background: Mutually Distant Sampling

Teng [Ten99] discusses the NP-hard problem of mutually distant sampling over a metric space and provides an analysis of the performance of a greedy approximation thereof. Here, we present a brief description of the problem of mutually distant sampling.

Let $\Gamma = (D, |||)$ be a metric domain, where $|||$ is a non-negative measure of distance, which satisfies the triangle inequality, over the domain $\Gamma$. Given a positive integer $k$, then mutually distant $k$-sampling of the domain amounts to finding a set $S$ such that $|S| = k$, and $S$ maximizes the minimum distance of its points. The minimum distance of $S$ is defined as follows:

$$\min(S) = \min_{i \neq j} ||s_i, s_j||$$

(2.1)

i.e., $S$ maximizes the minimum mutual distance between its samples.

**Greedy Approximation:** Teng [Ten99] proves that the greedy algorithm sketched below provides a 0.5-approximation to the problem, i.e., $\min_{i \neq j} ||s_i, s_j|| \geq 0.5 \times \min_{i \neq j} ||t_i, t_j||$, where $s_i, s_j \in S$, $t_i, t_j \in T$, where $T$ is the optimal solution, and $S$ is the set returned by the greedy algorithm.

When $D$ is a set of points $\{p_1, p_2, ..., p_n\} \in \mathbb{R}^d$, the complexity of this greedy algorithm is $O(k^2n)$.

2.6 Amorphous Placement and Retrieval (APR)

APR is a simple scalable algorithm that employs mobility, as opposed to multi-hop communication, whenever possible, to diffuse field samples from field locations to nodes that have never visited these locations. Hence, it improves the local view of each node of the entire field and enables nodes to answer more queries locally saving communication overhead.
Input: \( \Gamma = (D, \|\|) \), an integer \( k \geq 2 \).

1. Start with a random point \( x \in D \). Let \( S = \{x\} \).

2. For \( j = 2 \) to \( k \), repeat the following:
   2.1) Select the point \( y \in D \) such that \( y \) maximizes the following function
   \[
   \psi(S, y) = \min_{q \in S} \|q, y\|
   \]
   \( \psi(S, y) \) defines the distance between a set \( S \) and a point \( y \) using the measure \( \|\| \), as the minimum distance between \( y \) and all points \( q \in S \).
   2.2) Set \( S = S \cup \{y\} \)

3. Return \( S \).

Avoiding multi-hop communication makes APR efficient in terms of energy consumption and more robust in case of node failures or packet losses, moreover, it saves the overhead of operating an ad-hoc routing protocol. We also show that, interestingly, under limited node mobility, APR results in an informed multi-hop diffusion of field readings (akin to a selective delay-tolerant multi-hop forwarding of these readings).

APR employs two main mechanisms: (1) sample diffusion, and (2) cache management. Both mechanisms work together to enable nodes to optimize the contents of their caches resulting in better matching between the distribution of query targets and locally/nearby kept samples.

**Sample Diffusion:** The set of samples that are locally cached by a node (i.e., the node’s view of the whole field) is a subset of the set of samples this node collects while moving in the field. The latter set is totally defined by the mobility model of nodes, since nodes sample the field along their movement trajectory. However, the workload presented to any node (i.e., targets of queries posed to this node) is independent of the node’s trajectory. Hence, we need a mechanism to “decouple” each node’s view of the field from its movement trajectory. This mechanism relies on sample diffusion, whereby upon encountering each other, nodes exchange a small number of samples. This amounts to diversifying the contents of each storage, allowing improved matching of the nodes’ local view of the whole field to the query distribution, for both nodes.
More specifically, a node $z$ declares its presence to its neighbors by broadcasting a short **Hello** packet every $\alpha$ seconds. **Hello** packets contain a compact summary of the cache of $z$ (using a Bloom filter). Upon receipt of such a packet, a neighbor $y$ replies with an **Exchange** packet: a packet containing (up to) $k$ of its samples that failed the Bloom filter test (i.e., node $z$ does not have similar samples and hence its local view of the field would improve by getting these samples). The **Exchange** packet, likewise, contains a compact summary of $y$'s cache. Upon receiving an **Exchange** packet, $z$ adds the received samples to its cache applying the QCCM cache management algorithm described below, if needed. Then it replies to $y$ with a similar packet containing $k$ of its own samples.

Some parameters decisively affect the performance of the sample diffusion process. The first parameter is $\alpha$, the rate of sample diffusion. Slow sample diffusion rates may not specifically help diversifying the contents of caches, resulting in poor performance. While, high diffusion rates may cost too much communication power. The other parameter is the diffusion size $k$. Too small of a value may not be enough to improve performance, while a very large value means more energy consumption. These parameters need to be carefully tuned to optimize the performance of APR. In Section 2.8, we will investigate the effect of these parameters on the performance of APR.

**Query-Cognizant Cache Management**: Since the queries originating at a node follow a certain target distribution that is independent of the movement trajectory of this node, it is best to manage the node's storage in a way that makes it store a representation that mirrors this distribution—e.g., when query targets are uniformly distributed, the storage management should strive to cover the entire field as uniformly as possible. To achieve this goal we propose Query-Cognizant Cache Management (QCCM) policy. QCCM is based on maximizing the mutual distance between samples, as explained in Section 2.5. Whenever the cache is full and there are more than one sample to be added to the cache (due to sample diffusion), Algorithm 2.5 is applied to determine which set of samples should be retained in the cache.

Notice that, in case the query distribution follows a nonuniform, but smooth and contin-
uous, distribution over the field, we can apply a linear transformation on samples to get the effect of “stretching” (or scaling) the field at the area of high interest. Such that applying the QCCM algorithm will not evict samples from an area of high interest in favor of other samples covering a less important area. The exact form of this transformation depends on the exact distribution of query targets over the field.

As an example, we consider the case when queries follow a bivariate normal distribution whose mean $\mu$ is the center of the field ($i.e., \mu = (L/2, L/2)$). Variance of this distribution defines the concentration of queries. Small variance means that queries are mostly concentrated in a small area in the center of the field, while larger values of variance makes the query access pattern approaches uniform. The transformation we suggest here is adequate for all 2D distributions that are symmetric in all directions ($e.g.$, bivariate normal with a symmetric covariance matrix). A listing of the Stretching Algorithm is given below.

**The Stretching Algorithm:** To be applied on samples’ coordinates before feeding them to QCCM, when query targets follow a nonuniform smooth symmetrical distribution over the field.

Figure 2.1 shows steps 1.2 through 1.4 in Algorithm 2.6. It shows how to find $d'$, the distance between $s'_i$ and the center of the distribution.

Notice that to find distribution $Q'$ in step 2.1, we can take any section in $Q$ passing through the mean $\mu$, since $Q$ is assumed to be symmetric in all directions. Hence $Q'$ could be calculated once off-line. Notice also, that, $p$ in step 2.2 can be calculated using the cumulative distribution function (CDF) of $Q'$.

The Stretching Algorithm uses a linear transformation to adjust the distance between every sample and the center of the distribution so that the mapped distribution of points matches, as much as possible, the uniform distribution. This transformation has a similar effect of “stretching” the bivariate normal distribution so that it matches a uniform distribution. The net effect on sample locations is that: samples that fall in areas of high-demand ($i.e.$, where the query distribution has high values) are moved further from the center of the distribution, that is, distances between points from such areas get *extended*. However,
Input: Two dimensional bivariate normal symmetric distribution \( Q \) with mean \( \mu \) and symmetric covariance matrix \( \Sigma \), such that the diagonal elements of \( \Sigma \) equal \( \sigma^2 \). Field dimensions \( L \times L \), and a set of samples \( S = \{ s_1, s_2, \ldots, s_m \} \) every sample \( s_i \) has locations \( (x_i, y_i) \).

Output: A set of “mapped” samples \( S' = \{ s'_1, s'_2, \ldots, s'_m \} \), where every sample \( s'_i \) is the image of sample \( s_i \), and the mapped location is \( (x'_i, y'_i) \).

1. Let \( h \) be the value of the probability density function of a uniform distribution over the field. \( h = 1/L^2 \)

2. For \( i = 1 \) to \( m \):
   1) Consider the section of distribution \( Q \) defined by the plane going through \( \mu \) and \( (x_i, y_i) \), the location of sample \( s_i \). Call this section \( Q' \). \( Q' \) will be a 1-D normal distribution with mean = \( L/2 \) and variance \( \sigma \).
   2) Sum the probability density in \( Q' \) between the mean of \( Q' \) and \( (x_i, y_i) \), call this amount \( p \).
   3) Sample \( s_i \) will be moved along the line connecting it to \( \mu \). Let \( d' \) denote the new distance between \( s' \) and \( \mu \).
   4) To calculate \( d' \) divide the probability mass \( p \) by \( h \) (i.e., \( d' = p/h \)).
   5) To calculate the coordinates of the mapped sample \( s'_i \), map \( (x_i, y_i) \) to the corresponding polar coordinates \( (d, \theta_i) \), where the origin is taken to be \( \mu \). Then the polar coordinates of \( s'_i \) are \( (d', \theta_i) \).

3. Return \( S' \).
Figure 2.1: Application of Algorithm 2.6 (a) calculating the probability between point location $d$ and the center of the distribution, (b) find $d'$, the distance between the image of the original sample and the center of the field.

points from less-popular areas are moved closer to the center of the distribution, resulting in the distances between such points getting shrunk, making them more favorable for eviction by QCCM.

2.7 Effect of Storage Management

In this section we develop a simple model to gain insight into how much the storage management of multiple mobile nodes affects their collective probability of success in answering queries. We assume that $n$ mobile nodes roam in a two-dimensional periodic field (i.e., torus) of size $L \times L$. Each node has a storage of size $c$, where $L^2 \gg c$. Nodes are given enough time $T$ to sample the entire field. Nodes answer queries uniformly distributed over the field and of precision $\ell$, where we use $L_1$ (i.e., Manhattan) distance. The mobile nodes move according to some mobility model, and they sample the field along their movement trajectory, applying a storage management algorithm whenever needed. We model any mobility model through a probability distribution $p_{ij}, \forall (i, j) \in [L, L]$, where $p_{ij}$ is the
steady-state probability of any node being in field location \((i, j)\) under that mobility model. A “uniform” mobility model assigns the same probabilities to all field locations, while a “biased” model assigns different probabilities to different field locations (e.g., a random waypoint mobility model results in a higher probability of being in the center of the field). To be amenable to analysis, we assume that any collected sample stays fresh, and so a returned answer is always fresh. This assumption is reasonable if the rate of query/response is much larger than the rate of change in the sampled phenomenon. We relax this assumption in Section 2.8.

The goal of the model is to compare two storage management algorithms: QCCM, and random storage management (RSM) at steady state—we say that the system reached steady state when all nodes have sampled every location in the field. To focus on the efficiency of the storage management algorithm, we assume that nodes flood the field with their queries, so that storage management decisions done at one node affect the probability of success of queries issued at other nodes. We now introduce two lemmas to help us calculate coverage by each storage management algorithm, which, under the uniform query model assumption, is indicative of the query success ratio.

Coverage of a single sample: Assume a node keeps a sample \(e\) at location \((x, y)\), then the coverage of the field attained by keeping \(e\) is a function of \(\ell\), the query precision. The following Lemma defines coverage of a single sample \(R(\ell)\).

**Lemma 1** Let \(\ell\) denote the query precision. Then, in a two-dimensional periodic field, and using the \(L_1\) distance measure, field coverage attained by keeping any sample (assuming no overlap with coverage from other samples) can be calculated by \(R(\ell) = \sum_{i=1}^{\ell} 4i + 1 = 2\ell(\ell + 1) + 1\)

**Proof 1** It suffices to notice that on an \(L \times L\) torus, the number of neighboring locations at distance exactly \(\ell\) from any location equals exactly \(4\ell\), and we add 1, to account for coverage of the field location where the sample lies.

Optimal Inter-Sampling Spacing (ISS) in 2D torus: We need to answer the question:
“How can we place $c$ points on a torus of dimensions $L \times L$, such that the minimum mutual distance between any two points is maximized? And, what would the optimum distance $S_{\text{opt}}$ in this case?” Let’s assume for now that $c$ is a square number, i.e., $c = s^2$ for some integer $s < L$, and $L$ is a multiple of $s$. Then we can very easily argue that placing the $c$ points uniformly on the field maximizes their mutual minimum distance. In such a case, an optimal algorithm would be one that divides the torus into $s \times s$ squares, then places a point in each square. Selecting the corresponding points in each square yields a minimum ISS of $S_{\text{opt}} = L/s$. The following lemma formalizes this fact.

**Lemma 2** In an $L \times L$ torus and given that $c = s^2$, if $L > s$ and $L$ is a multiple of $s$, then $S_{\text{opt}} \geq \frac{L}{s}$.

**Performance of QCCM:** As we discussed above, at steady state, nodes would have sampled the entire field. Recall that QCCM decouples the storage content from the movement trajectory of the nodes. Then we assume that nodes are able to maximize inter-sample spacing, yielding ISS = $S_{\text{opt}} = L/s$. This is always true as long as the mobility model has nonzero probability of visiting all field locations. Since there are $n$ nodes, we know that given any area $A$ of size $= S_{\text{opt}} \times S_{\text{opt}}$, will host exactly one sample from each node, for a total of $n$ samples in $A$. Coverage of $A$, in this case, corresponds to coverage of the whole field, since the coverage pattern in $A$ is repeated over the rest of the field.

To simplify the analysis, we assume that nodes do not optimize their caches with respect to contents in their neighbors’ caches (i.e., nodes do not try to minimize the intersection of coverage achieved by samples in their caches and coverage of samples in their neighbors’ caches). Under this assumption, it follows that $A$ has $n$ randomly placed samples. Figure 2.2 illustrates this setup. Now, consider any field location $(l_{ij})$ in $A$, the probability ($q = \Pr[l_{ij} \text{ covered}]$) of covering this location is proportional to the value of $\ell$, and can be calculated as $q = \frac{R(\ell)}{S_{\text{opt}} \times S_{\text{opt}}}$. This follows from the fact that coverage of any field location is related to coverage of one sample. For example, if $\ell = 0$, $l_{ij}$ has only one chance of being covered (i.e., having a sample at $l_{ij}$). If $\ell = 1$, then $l_{ij}$ has five chances (having a sample
Figure 2.2: Idealized field coverage by four nodes applying QCCM. Notice that any area \( A = S_{opt} \times S_{opt} \) will have exactly one sample from each node.

at location \( l_{ij} \) itself, or having a sample at any of its four neighboring locations), and so on. Now we can view the attempt to cover any field location in \( A \) by the \( n \) samples, as \( n \) independent Bernoulli trials, each with probability of success \( q \). Thus, the probability of covering any field location exactly \( x \) times (i.e., probability \( l_{ij} \) will fall into the coverage area of \( x \) different samples) has a binomial distribution and is given by \( \Pr[B(n, q) = x] \), where \( B(n, q) \) is the Binomial probability. By running a summation of the last quantity for \( x = 1 \cdots n \), we can obtain the probability of success under QCCM as:

\[
\text{Success}_{QCCM} = \sum_{x=1}^{n} \binom{n}{x} q^x (1-q)^{n-x} = 1 - (1-q)^n
\]

(2.2)

**Performance of RSM:** Under RSM, nodes sample the underlying mobility model, hence their storage content will match this distribution. Following the same lines of analysis as we did in QCCM, we have \( n \) nodes, each with storage \( c \), for a total of \( n \times c \) samples in the field. For a given value of \( \ell \), let’s define \( \mathcal{N}_{ij} \) as the set of neighboring locations of \( l_{ij} \) within \( \ell \) distance units. The probability \( \omega_{ij} \) of covering a location \( l_{ij} \) can be calculated as:

\[
\omega_{ij} = p_{ij} + \sum_{l_{xy} \in \mathcal{N}_{ij}} p_{xy}
\]

Hence the probability of \( l_{ij} \) being covered exactly \( x \) times is given by: \( \Pr[B(n \times c, \omega_{ij}) = x] \), and the expected number of locations that are covered exactly \( x \) times is given by:
Figure 2.3: Effect of storage management algorithm on query success ratio for \( n \) mobile nodes. Notice RND2, RSM under mobility model 2 (Figure 2.4 right), is better than RND1, RSM under mobility model 1 (Figure 2.4 left), since the earlier is less skewed than the latter.

\[
\sum_{0 \leq i,j \leq L} \Pr[B(nc, \omega_{ij}) = x].
\]

Then, we can calculate coverage of the field by running a summation for all \( x = 1 \cdots n \times c \), and the success probability is given by:

\[
\text{Success}_{RSM} = \frac{1}{L^2} \sum_{x=1}^{nc} \binom{n}{x} \left[ \sum_{0 \leq i,j \leq L} \omega_{ij}^x (1 - \omega_{ij})^{nc-x} \right]
\]  

Figure 2.3 plots Equations 2.2 and 2.3 for two different mobility models depicted in Figure 2.4. We have numerically confirmed that both mobility models have no \( i,j \) such that \( p(i,j) = 0 \), \emph{i.e.}, nodes can sample the entire field under both models. It is clear that QCCM has a noticeable performance advantage over RSM, as it manages storage content based on the workload, decoupling it from the trajectory of motion of nodes.
2.8 Performance Evaluation

We evaluated APR and DPR using extensive simulations under a variety of settings. In this section we provide the key results from our experiments.

Simulation Model and Setup: we conducted a set of detailed packet-level simulation experiments, in which we used identical mobility and sampling scenarios for the various approaches. Mobility scenarios for our experiments were generated off line using different mobility models, including the corrected version of the Random Waypoint mobility model [LNR04], the Random Direction model [RMSM01] and the Boundless Simulation Area model [Haa97]. Since results were close under different mobility models, we only report results for the corrected Random Waypoint model. In our simulations, we set the minimum and maximum speed of motion to 0.1 m/sec, and 20 m/sec, respectively.

The sampling process used by mobile nodes follows a Poisson process with exponential inter-arrival time of two seconds; a sample at time $t$ constitutes the sensed value of the field at the current location of the node. We report results of simulating 100 mobile nodes moving in a field of $1400 \times 1400$ m, where distance is measured in Euclidean distances. The simulation runs for 5,000 seconds. In the following figures, every point is the average of 20 simulation runs, with 95% confidence intervals shown. Notice that the confidence intervals are extremely small in most cases. Unless otherwise noted, the default parameters are: $r = 160$ m, $\ell = 140$ m, $c = 50$, TTL = 200 secs, PLP = 0, and uniform distribution of queries.

Performance Metrics: The first metric we use is the query success ratio (QSR), which is defined as the ratio between the number of successfully answered queries and the number of all queries. To measure efficiency in terms of consumed energy, we compute the number of successfully answered queries per unit energy, to which we refer using Success Per unit Energy (SPE). We use an energy model based on the model presented in [HCB00] (Equations 1 and 2).

Effect of APR Mechanisms: APR has two main mechanisms: sample diffusion, and
QCCM. In this experiment, we quantify the effect of each of these mechanisms on APR’s performance. To this end, we compare three different versions of APR: (1) APR with both sample diffusion and QCCM, (2) APR with QCCM but no sample diffusion, and 3) APR with sample diffusion, and FIFO storage management. We refer to these versions by APR, No Diffusion, and FIFO, respectively. Figure 2.5 shows the QSR of these APR variants as a function of the communication range $r$. These results show that the combination of sample diffusion and QCCM achieves the highest QSR compared to using one technique only. There is a clear difference in performance between APR and FIFO storage management validating our analytical findings in Section 2.7. On the other hand, disabling the sample diffusion mechanisms hinders the performance of APR.

**Effect of APR Parameters:** in Section 2.6, we alluded to the importance of two parameters in APR, specifically, the rate of sample diffusion $\alpha$, and the size of the diffusion $k$. The following two experiments show the effect of these parameters on the performance of APR.

**Effect of Exchange size $k$:** intuitively, we expect that the larger the exchange size, the higher the query success rate, and the more consumed energy. Figure 2.6(left) shows the effect of $k$ on QSR. It is clear that increasing $k$ improves the success of APR. Improvement of more than 10% can be gained using larger exchange sizes. However, increasing $k$ for queries with tight precision constraints (*i.e.*, small value of $\ell$) does not help the performance much. The reason for this will be explained later when we discuss the effect of the precision $\ell$.
Figure 2.6: Effect of sample exchange size: Query success ratio (left) and Query success ratio per unit of energy (right).

on APR. Figure 2.6(right) shows the effect of $k$ on the SPE. As expected, increasing the exchange size consumes more energy which negatively affects performance. As apparent from the figures, there is no optimum setting for $k$, rather selecting a specific value of $k$ is a compromise between QSR and SPE. In the rest of the experiments we use $k = 4$ to benefit from the sample diffusion mechanism without consuming much energy in the process.

**Effect of Exchange rate $\alpha$:** performing sample diffusion more often should lead to better coverage of the field, leading to higher QSR. Figures 2.7(left) and 2.7(right) show the effect of the silene interval ($=1 / \alpha$) on performance; smaller values of $\alpha$ invigorate more sample diffusion which results in higher QSR. However, exchanging more packets consumes more energy which decreases efficiency (in terms of SPE). Again, in this case there is no clear optimum value for $\alpha$. In later experiments we use $\alpha = 200$, to minimize the energy consumed in sample diffusion, and to avoid packet collisions due to storms of Hello and Exchange packets. The overall conclusion of these two experiments is that, parameterizing APR is platform-specific, i.e., when deployed on energy-sensitive devices, slower sample diffusion and using small exchange size would be an efficient strategy. While when deployed on energy-rich devices, taking advantage of the high QSR presented by higher sample diffusion rate and larger exchange size could be the preferred strategy.

**APR versus DPR:** In this subsection we compare APR to DPR. To that end, we vary the following parameters: communication range $r$, query precision $\ell$, storage size $c$, TTL, and packet loss probability (PLP). We also study the effect of varying the distribution of
queries to a non-uniform distribution, and finally we quantify the effect of mobility (or lack thereof) on both protocols.

**Effect of Communication Range:** The communication range $r$ defines the level of connectivity in the network. We argued that DPR would achieve high QSR only when the network is very-well connected, while APR is able to achieve better QSR in less connected networks. Validating our intuition, Figure 2.8(left) shows query success ratios for APR and DPR using different values for query precision. It is clear that APR outperforms DPR for networks with smaller communication ranges, while the roles are reversed when we increase the communication range. To visualize the impact of network connectivity on QSR for DPR, we also plot the probability of having a connected network as a function of the communication range. This curve is based on the network connectivity model presented in [Bet02]. It is clear that DPR’s performance peaks only when the network is well connected. Increasing the value of $\ell$ (i.e., making precision requirement less strict) helps APR outperform DPR over a wider region of communication ranges. Later, we discuss this effect in more detail.

As for SPE, as shown in Figure 2.8(right), for shorter communication ranges DPR achieves better performance at higher energy consumption level than APR. As the communication range increases, DPR consumes much more energy compared to APR rendering it inefficient in terms of SPE. This is mainly because, unlike APR, DPR depends mainly on multi-hop communication which consumes much energy.

This experiment hints that in loosely-connected networks, APR delivers higher perfor-
mance, but at a higher cost. Increasing the communication range makes APR more efficient in terms of QSR and SPE compared to DPR. When the network is highly connected, DPR delivers better performance in terms of QSR, even for queries with tight precision requirements. For those with loose precision requirements, APR is the best choice for almost all communication ranges.

**Effect of Query Precision:** given a query target, query precision is the maximum distance we allow between the query target and any sample that can be used to answer this query. Assuming uniform distribution of queries, APR aims to give all nodes a uniform view of the entire field. Since nodes’ storage are limited, the supported precision under APR will be unavoidably limited. On the other hand, DPR gives each node a very detailed view of a specific region in the field, suggesting that it should be able to handle queries with tighter precision requirements. Figure 2.9 shows QSR and SPE as a function of the precision. DPR excels in tight precision requirements (and good network connectivity) as measured by QSR, but at a higher energy consumption. As the precision requirement is relaxed, APR catches up and eventually outrivals DPR with much more efficient performance in terms of SPE. Notice the improvement in APR’s performance as we increase the storage size; the larger the storage size, the higher the query success ratio at a given precision requirement.

This experiments suggests that query precision is a dominant factor in APR performance; tight precision constraints hinders the success ratio of APR, making DPR the better choice with respect to QSR. However, this effect of the query precision on APR’s perfor-
Figure 2.9: Effect of query precision: Query success ratio (left) and Query success ratio per unit of energy (right).

Performance could be alleviated by adding more storage (which is a more practical solution since the query precision is application specific and cannot be easily modified). For queries with relaxed precision constraints, APR delivers equally high QSR at about an order of magnitude saving in SPE, even with limited storage size. Recall that in loosely-connected networks, unlike APR, DPR is not able to handle queries with strict nor loose precision requirements. APR is more energy-efficient than DPR under most situations.

**Storage Size:** As we discussed above, increasing the storage capacity of nodes helps APR cover the field with a better precision, while it does not affect DPR much, since adding more storage space does not change the level of network connectivity, the dominant factor in DPR’s performance. Figure 2.10 shows the performance of APR vs. DPR as a function of the storage size. As we noticed, the storage size barely helps DPR’s QSR, while it has a more noticeable performance on that of APR. Beyond a certain storage size, APR’s performance reaches a steady state plateau and remains constant even if we increase the storage size. The reason for is that, the TTL of samples = 200 seconds, and the sampling rate is 0.5 sample/sec (i.e., 1 sample every 2 seconds), suggesting that storage sizes approaching 100 are not particularly useful. On the other hand, as we have already noticed, relaxing the precision constraint improves the performance of APR, while increasing the communication range boosts that of DPR. In terms of energy consumption, APR is more efficient in all cases.

**Sample Freshness (TTL):** APR depends on mobility and one-hop sample exchange to
Figure 2.10: Effect of storage size: Query success ratio (left) and Query success ratio per unit of energy (right).

Figure 2.11: Effect of sample TTL: Query success ratio (left) and Query success ratio per unit of energy (right).

diffuse samples throughout the field, while DPR uses multi-hop communication to achieve the same effect. Since mobility is slower than multi-hop communication, DPR is expected to beat APR for data types with small TTL. However, a larger TTL allows enough time for the sample diffusion process in APR to function properly, resulting in much improved performance. Figure 2.11 shows the performance of APR vs. DPR as a function of the TTL of samples. For samples with short TTL, DPR delivers better performance than APR in terms of QSR. While for samples with longer TTL, APR outperforms DPR. For the same value of TTL, increasing storage size helps APR but not DPR, and increasing the communication range helps both, but its effect is more spoken on DPR. For all values of TTL, APR is more efficient in terms of energy consumption.

Packet Loss Probability (PLP): Figure 2.12 shows the performance of APR vs. DPR as a function of packet loss probability (PLP). PLP effect is more pronounced on DPR. The
Figure 2.12: Effect of packet loss: Query success ratio (left) and Query success ratio per unit of energy (right).

reason is that, as we alluded above, DPR depends on multi-hop communication in sample storage, query forwarding, and query response forwarding, which makes it more vulnerable to packet losses. APR, on the other hand, depends on one-hop communication which makes it more resilient to packet losses. Notice that the difference between APR and DPR in query success ratio at loss probability = 1 is attributed to the storage management algorithm. APR applies QCCM which improves performance over random storage management which DPR applies.

**Performance in Static Networks:** One might expect that in static networks, APR’s performance will deteriorate significantly compared to DPR. In this experiment, we show that, counter-intuitively, lack of mobility does not impact the general behavior of APR’s performance significantly.

In mobile networks, nodes get multiple chances of getting in contact with different neighbors allowing them better sample diffusion and thus an improved view of the entire field. In case of static networks, APR depends, indirectly, on delay-tolerant multi-hop dissemination to achieve this effect. To see why this is the case, consider a node \((z)\) at location \((x_z,y_z)\) in a completely static network. Due to its immobility, all samples cached by \(z\) will be from location \((x_z,y_z)\). This will be true until \(z\) starts the sample diffusion process with its neighbors. At this point, \(z\) will storage samples gathered at locations of its direct neighbors. As the sample diffusion process continues, and QCCM is applied, \(z\) will eventually storage samples gathered at neighbors of its direct neighbors, and so on. This
effect goes on until \( z \) gets a uniform view of the entire field. The combination of QCCM and informed sample diffusion helps to diversify the storage contents of all nodes improving the performance of the entire system. Mobility, only, speeds up this process, especially when the network is not well connected.

The repeated diffusion of samples to nodes farther from the collecting nodes is one form of delay-tolerant multi-hop communication. However, in this case, unlike DPR, nodes on the way get a chance to keep such samples themselves. Figure 2.13 shows the performance of APR and DPR in a static network as a function of the query precision. The relative trend of APR, seen in mobile settings, is still the same (i.e., relaxing the precision constraint improves APR’s performance). This accentuates the effectiveness of APR’s mechanisms in delivering high performance even in networks with no/limited mobility. Regarding energy efficiency, Figure 2.13 (right) shows that APR is always more efficient than DPR.

It is worth pointing out that, the effect of network partitioning is more pronounced when there is lack of mobility. In APR, mobility helps nodes that are temporarily isolated to come in contact with neighbors and exchange valuable samples, which improves the field view at these nodes. When there is no/limited mobility resulting in a partitioned network, disconnected nodes have no such chance, hence their performance deteriorates. This effect is more magnified under DPR, since having persistent network partitions harms the performance of the entire system (due to partitioning the field into RR’s and assigning an RR to each node), as opposed to harming the performance of only the group of disconnected
nodes under APR. Another weakness in DPR is that, since some of the RR will not have
sensors reside in them, queries about these RR will be always missed. Since APR does
not depend on the idea of RR, but rather searches for the sample closest to the query
target, the performance of APR for the same queries is decidedly better. The probability
of this scenario happening increases as we relax the precision constraint and increase the
communication range (see Figure 2.13).

**Non-uniform Query Distribution:** In Section 2.6, we described a solution when the
query targets follows a non-uniform smooth symmetric distribution over the field. As a proof
of concept, we show results when assuming queries follow a bivariate normal distribution
whose mean $\mu$ is the center of the field (i.e., $\mu = (L/2, L/2)$) and a symmetric covariance
matrix. Figure 2.14 (left) shows the effect of applying The Stretching Algorithm on a set of
points sampled according to a bivariate normal distribution with $\mu = (700, 700)$, and a $2 \times 2$
covariance matrix $\Delta$ that has $\sigma^2$ on the diagonal entries and 0.6 at the off-diagonal entries,
and $\sigma^2 = 17000$. It shows the original samples and their mapped images. It is clear that
the distribution of the original samples is skewed (all samples are concentrated around the
mean), while the distribution of the mapped samples approaches a uniform distribution.

Figure 2.14 (center and right) shows the results of APR with out applying the stretching
algorithm, APR with the stretching algorithm and DPR when the query access pattern
follows a bivariate normal distribution with mean $\mu$ that is the center of the field, and
symmetric covariance matrix $\Delta$ with diagonal elements = $\sigma^2$. The x-axis shows the different
values of $\sigma^2$ we tried. For skewed access patterns, the stretching algorithm succeeds in
mapping the samples to a match a uniform distribution yielding superior performance.
When the variance grows such that the query distribution approaches a uniform distribution,
Performance of APR with stretching matches APR with no stretching.

**Summary of Findings:** We conclude this section with a summary of findings from all of
our experiments.

We have shown that: (1) Communication range is the main determinant of DPR’s
performance: a loosely connected network renders DPR dysfunctional. In contrast, APR
Figure 2.14: Effects of non-uniform query distribution. Output of the stretching algorithm (left), QSR (Center), success per consumed power unit (Right).

features higher resilience to network disconnectivity. (2) APR’s performance is not significantly affected by lack of mobility. In fact, when the network is not well connected, lack of mobility negatively impacts the performance of DPR much more than that of APR. (3) APR is more energy efficient than DPR in almost all situations. (4) In well-connected networks, queries with tighter precision constraints are better handled by DPR than APR. Relaxing precision constraints improves APR’s performance. In loosely-connected networks, APR is better than DPR, even for queries with tight precision constraints. (5) In well-connected networks, when the monitored field values have tight freshness constraints (i.e., small TTL values), DPR beats APR in handling queries with stringent precision constraints. APR’s performance improves as we increase the value of TTL. In loosely-connected networks, the performance of APR dominates that of DPR, independent of freshness (TTL) constraints. (6) Unlike DPR, APR is able to take advantage of increased storage sizes in all settings. (7) APR features much higher resilience to packet losses (and node failures) compared to DPR. (8) Applying a linear transformation to sample locations before feeding them to QCCM, enables APR to deliver superior performance when the query distribution is non-uniform over the field.

2.9 Conclusion

In this chapter, we presented a proactive approach, APR, that amorphously places and diffuses sensor data collected by autonomously mobile nodes, allowing nodes (and node
neighborhoods) to compile an integrated view of the monitored field of interest, in anticipation of freshness-constrained and precision-constrained queries thereof. A salient feature of APR is that it enables the management of the nodes’ storage content in such a way so as to match the distribution of query targets, regardless of the distribution of the locations that are collectively visited (and sensed). We demonstrated, by analysis and extensive simulations, how query performance improves under an informed (query-aware) diffusion of sensory samples that maximizes the minimum distance between samples in a node’s cache.

In conclusion, we stress that APR has the spirit of data management algorithms specifically tailored for distributed autonomous systems. APR has a number of characteristics that perfectly matches such systems. We itemize these next.

- APR is a light-weight algorithm, that is suitable for embedded resource-constrained platforms.
- APR does not depend on a routing algorithm, saving operational overhead.
- APR does not involve explicit planning among all participants, which makes adding or deleting members a trivial process.
- APR avoids multihop communication which saves communication overhead.
- APR is able to deliver high query success ratio in almost all conditions (e.g., loosely-connected networks, and networks with little/no mobility).
- APR is workload-aware. This makes it able to adapt its resource management to match the query pattern of users (e.g., favors samples from high-demand areas), resulting in optimized performance.
In this Chapter, we relax the assumption that sensor management is decided by an external entity, driven only by the need to save energy of the device. Rather, we assume that the distributed service has freedom to decide when to sample the current location of the host, in order to optimize its performance. Consequently, besides questions addressed in Chapter 2, we address the question: “When should any sensor sample its current location?” The model we use in this setup closely matches that in Chapter 2, with minor modifications.

We start by presenting the model and the notation that we use. Then, we propose an information-theoretic framework to address the problem of sensor management, storage-management, and query handling. Based on this framework, we propose two solutions, with different types of assumptions about the target phenomenon, and different computational complexities. We, then, compare these two approaches to a naïve storage management technique, and show that our techniques achieve orders of magnitude improvement in the accuracy of the query answers. The contributions of this chapter are as follows.

- Assuming knowledge of the entire spatio-temporal distribution of the target phenomenon, we develop an information-theoretic framework to optimize the cache content, and provide accurate answers to queries (Section 3.2).
• We propose a different approach based on optimizing a correlation-based function relaxing the stringent constraint of full distribution knowledge. We develop a strategy that only requires knowledge of the second order statistics of the phenomenon of interest. Furthermore, this technique lowers the required computational complexity (Section 3.3).

• We provide extensive performance evaluation of our techniques, showing (and quantifying the impact of) the various factors and parameters that affect performance (Section 3.5). We, also, study the robustness of the technique developed in Section 3.3 to model mismatch in case of imperfect knowledge of the correlation structure.

3.1 Problem Definition

The model we use in this setup matches that presented in Chapter 2, with some modifications. The model changes, system parameters, and notation we use are as follows:

• The nodes move in a field $\mathcal{F}$ with area $A = L \times L$.

• While roaming the field, sensor nodes sample a target phenomenon and this process continues for $T$ time units.

• We use capital letters to represent random variables and small letters to represent realizations of these random variables.

• $V_{\ell,t}$ is a random variable that represents the value of the field phenomenon at location $\ell$ and time $t$. $v_{\ell,t}$ denotes a realization of this random variable.

• We use the boldfaced letter $s_i^t = [s_1, s_2, ..., s_c] \in \mathbb{R}^c$ to denote the $c$-dimensional storage content vector of node $i$ at time $t$. To simplify notation and since we would be generally referring to any arbitrary node $i$, we will drop the superscript $i$, unless it is not clear from the context. Note that any cached sample $s_j$ corresponds to a field value $v_{\ell_j,t_j}$, where $\ell_j$ is the location from which this sample was collected and $t_j$ its corresponding time stamp.
• Unlike the approach presented in Chapter 2, a query is only defined by a query target (i.e., no precision constraint). It is assumed that a query posed at any time instant \( \tau \) whose query target is location \( \ell \) targets the value of the field phenomenon \( v_{\ell,\tau} \).

• The field phenomenon is fully characterized by a space-time multivariate probability distribution \( p(\{v_{\ell,t}\}; \ell \in \mathcal{F}, 0 \leq t \leq T) \) with a \( L^2 \times T \times L^2 \times T \) correlation matrix \( R \), such that \( R(v_{\ell_1,t_1}, v_{\ell_2,t_2}) \) represents the correlation between two values of the phenomenon with space-time coordinates \( (\ell_1, t_1) \) and \( (\ell_2, t_2) \), respectively.

• Define the random variable \( L(q) \) as the query target of query \( q \). We assume that \( L(q) \) follows some spatial distribution \( Q \), where \( Q(\ell(q)) \) is the probability of querying field location \( \ell(q) \). \( Q \) is assumed to be stationary. Similarly, we use \( t(q) \) to denote the time at which query \( q \) was posed. Obviously, the best answer to \( q \) would be \( v_{\ell(q),t(q)} \).

**System Goal:** Unlike the system presented in Chapter 2, the goal of the system is to respond to each query with an accurate estimate of the value of the phenomenon at the query target at the time of posing the query (rather than a fresh sample within the specified precision), hence we drop notions of query precision, and sample freshness. Accuracy of a query answer is quantified by minimizing the mean square estimation error (MSE) of the answer. Hence, as in Chapter 2, nodes are required to maintain an efficient storage content to be able to answer queries reliably. In the next sections, we develop different strategies for storage management at the sensor nodes.

### 3.2 Information Theoretic Storage Management

In this section we develop an information theoretic framework to address problems of storage and sensor management, as well as query handling. This framework is based on knowledge of the joint distribution of the target phenomenon.
3.2.1 DEBT Storage Maintenance Strategy

Since nodes have the freedom to keep a sample from the current location of the sensor at anytime, a test has to be performed to gauge the attained utility by keeping a sample from the current location. Towards that end, at each time instant, each node samples its current location, then decides if the newly acquired sample should be kept or not. These decisions are made so as to minimize an entropic utility function that captures the average amount of uncertainty in queries given the probabilistic query target distribution $Q$ — hence the name of the strategy: Distributed Entropy Based Technique (DEBT). Specifically, at each time instant $t$, a node $i$ greedily decides in favor of the set of samples that minimizes the conditional differential entropy averaged over the query distribution $Q$, More formally,

$$s_t = \arg \min_{s_t \in S_t} h(V_{L(q),t}/s_t, L(q))$$

where $s_t \in \mathbb{R}^c$ is the storage content selected by node $i$ at time $t$, and $h(V_{L(q),t}/s_t, L(q))$ is the differential entropy of the values of the phenomenon, conditioned on a given storage content, at the possible query locations $\ell(q)$ which follow a spatial distribution $Q$. $S_t$ is the set of all possible decisions leading to all possible storage contents at node $i$ at time $t$ which is given by:

$$S_t = \{s_t : s_t \in \mathcal{C}_{c,c+1}(s_{t-1} \cup \{v_{\ell_t,t}\})\}$$

where $\mathcal{C}_{c,c+1}(A)$ denotes all the $(c + 1)$ choose $c$ possible combinations of the elements of a set $A$ and $v_{\ell_t,t}$ denotes the value of the phenomenon at the current location of the $i$-th node, $\ell_t$. This expression simply enumerates all the possible storage contents at time $t$; the options being to drop any of the samples from time $t - 1$ and keeping the new sample at the

Note that the differential entropy $h(V_{L(q),t}/s)$ that we use in the minimization of Equation(3.1) is conditioned on a given realization of the storage content. That is to say, no averaging is taken over the conditioning random vector since we are dealing with real-time selection of the samples. This is clearly different from the standard quantity $h(V_{L(q),t}/S)$ with $S$ being a random variable.
current location of node $i$, or just keep the old set of samples, and drop the newly acquired sample.

The intuition behind DEBT is that a node always keeps a storage content that minimizes the uncertainty in the values of the phenomenon (captured by the conditional entropy) given the knowledge of the spatial distribution of the query targets over the field of interest. It might well be true that an old sample taken at a specific location is more valuable, and hence is worth caching than a newer sample taken at a different location given the aggregate effect of the spatial query distribution and the spatio-temporal distribution of the phenomenon\(^2\).

It is worth mentioning that the computation of $h(V_{\ell(q),t}/s_t)$ (Eq.3.3 [CT06]) requires knowledge of the posterior density $p(v_{\ell(q),t}/s_t)$, which can be generally obtained by proper marginalization of the full space-time distribution. For the Gaussian case, this simplifies to a computation of the conditional mean and variance $\mu_{v_{\ell(q),t}/s_t}$ and $\lambda_{v_{\ell(q),t}/s_t}$.

$$h(V_{\ell(q),t}/s) = -\int_{v_{\ell(q),t}} p(v_{\ell(q),t}/s) \ln p(v_{\ell(q),t}/s) dv_{\ell(q),t} \quad (3.3)$$

Notice that, since entropy calculations require knowledge of the values of the samples in the local storage, then, under DEBT, in order to manage their local storage, nodes need to actually acquire a sample from the current location each time unit.

### 3.2.2 Least Square Error (LSE) Query Response Strategy

To answer a posed query $q$, a node computes an estimate of the phenomenon at the query target given its storage content. Given the knowledge of the space-time distribution, it would be natural to resort to a Bayesian Least Square Estimate (BLSE), which is given by the conditional expectation of the posterior density, to minimize the mean square estimation error. Whenever a node receives a query $q$, its task is to compute the expected value of the phenomenon at the query target $\ell(q)$ given its storage content $s$, that is:

\(^2\)This is why we drop the notion of sample freshness, since the utility of a sample, not its age, is what decides whether or not to keep this sample.
\[
\hat{V}_{\ell(q),t(q)} = E[\hat{V}_{\ell(q),t(q)}|s] \quad (3.4)
\]

where \(\hat{V}_{\ell(q),t(q)}\) is the node estimate. Again we point out that this generally requires the computation of the posterior density \(p(\hat{V}_{\ell(q),t(q)}|s)\). Under Gaussian assumptions, the BLSE estimate in Eq.(3.4) is always linear in the storage content, that is the BLSE is equal to the Linear Least Square Estimate (LLSE). For general distributions, the computational complexity could be reduced if we only restrict ourselves to linear functions of the storage content, \(\text{i.e.}\) LLSE, which would only require knowledge of the second-order statistics of the phenomenon. Note that the LLSE, \(\hat{X}_{\text{LLSE}}\), of a random variable \(X\) with mean \(\mu_X\), given a random vector \(Y = y\), with mean vector \(\mu_Y\) is given by \([\text{Tre01}]\):

\[
\hat{X}_{\text{LLSE}} = \mu_X + \Lambda_{XY}\Lambda_Y^{-1}(y - \mu_Y) \quad (3.5)
\]

where \(\Lambda_{XY}\) denotes the cross-covariance between \(X\) and \(Y\), and, \(\Lambda_Y\) is the covariance matrix of the observation vector \(Y\). While the DEBT/LSE techniques outlined in this section are expected to yield accurate performance, they are not practical. Specifically, we note the following limitations on DEBT applicability:

- **Informational Limitations:** DEBT assumes knowledge of the entire distribution of the target phenomenon. Such information may not be always available, or if available (\(\text{e.g.},\) through historical monitoring of the phenomenon of interest), it may not be accurate.

- **Computational Limitations:** In order to provide optimized decisions about whether or not to sample visited field locations, and how to manage the storage, DEBT calculates the conditional differential entropy of the query distribution \(Q\) given any storage setting. This requires performing multiple numerical integration operations, which might not be always suitable due to the limited computational capabilities at the sensor nodes.
• **Practical Limitations:** As we pointed out, DEBT enables nodes to solve the problem of storage management, since it can decide which specific samples to keep in the local storage. However, DEBT solves the problem of sensor management by actually acquiring a sample from the current location at each time unit. Then, it decides whether or not to keep this sample. This behavior is, clearly, not optimal in terms of sensor management.

This motivates taking a different approach that is less-demanding in terms of: 1) knowledge about the spatio-temporal field, 2) computational requirements, and 3) sensor management. In the next section, we propose a more practical (yet quite competitive) strategy that only requires knowledge of the correlation structure \( R \), *i.e.*, second-order statistics.

### 3.3 Correlation-Based Storage Management

In this section, we propose a Correlation-Based Technique (CBT) as a practical alternative to the DEBT approach presented before. CBT avoids the limitations of DEBT by only assuming knowledge of the space-time correlation structure of the field phenomenon \( R \). Namely, instead of calculating the conditional entropy to make caching decisions, CBT decides which samples to store using only the correlation structure of the target phenomenon \( R \). Notice that defining \( R \) implies only knowledge of the second-order statistics of the target phenomenon, as opposed to knowledge of the entire distribution in case of DEBT. Like DEBT, the crux of the CBT technique is to be able to assign a measure of utility capturing knowledge about the field to any given set of samples \( s = \{s_1, s_2, ..., s_c\} \) with respect to the query distribution \( Q \). Then, it retains the set of samples that maximizes the utility. First, we need to assign a measure of utility \( u(q, s) \) to a set of samples \( s \) with respect to a specific query \( q \) with location \( \ell(q) \), and time \( t(q) \). Then by averaging \( u(q, s) \) over the spatial distribution \( Q \), we get a weighted information metric over the entire field, \( M(Q, s) \). More specifically, for a query \( q \), we gauge the utility of \( s \) with respect to \( q \) as follows:
\[ u(q, s) = \frac{Q(\ell(q))}{\Lambda_{q|s}} \]  

(3.6)

Averaging \( u(q, s) \) over \( Q \), we get

\[
M(Q, s) = \int_Q u(q, s) = \int_{\ell \sim Q} \frac{Q(\ell)}{\Lambda_{q|s}}\, d\ell
\]

(3.7)

where \( Q(\ell(q)) \) is the probability of querying field location \( \ell(q) \), and \( \Lambda_{q|s} \) is the conditional covariance of \( q|s \), given by

\[
\Lambda_{q|s} = \Lambda_q - \Lambda_{q,s} \Lambda_s^{-1} \Lambda_{q,s}^T
\]

(3.8)

where \( \Lambda_q \) is the variance of the stationary process, \( \Lambda_{q,s} \) is the cross-covariance between \( q \) and \( s \), and \( \Lambda_s \) is the covariance matrix of the storage content \( s \). Notice that calculation of \( \Lambda_{q|s} \) only requires knowledge of the correlation matrix \( R \). Then, CBT handles both problems of storage and sensor management by maximizing the total utility over the choice of possible storage content \( s \) (i.e., \( \max_s M(Q, s) \)). Moreover, unlike DEBT, to manage their sensors under CBT, nodes need not actually acquire a sample from the current location at each time unit. Rather, a sample is acquired, only, when it is deemed useful to utility of the local storage by CBT.

### 3.4 The Distributed System

So far we have described operation of a single user (i.e., node). However, we argued that, in the distributed systems we are interested in, the “service” is provided to users through cooperative management of resources of these users. Hence, building a distributed system of cooperative users is bound to improve performance of all users. In this setup, we limit our attention to cooperation concerning query response. This is done as follows.

Whenever a node \( i \) gets a query \( q \), \( i \) broadcasts \( q \) to its direct neighbors. Upon receiving the query, each neighbor \( j \) of \( i \) estimates its answer based on its local cache content, then, submits the estimate back to \( i \) along with a measure of confidence in this answer. Node
$i$ performs the same task, and receives query replies from its neighbors. The answer with the highest confidence is used as the query response. In our setting we use the conditional covariance $\Lambda_{q|s}$ (Equation 3.8) as an indication of confidence in the estimated answer. The intuition is that a lower conditional covariance corresponds to less uncertainty about the query, i.e., higher confidence. Notice that, the radius of flooding the query could be increased to values larger than one (i.e., consult nodes beyond direct neighbors), however, we choose not to do this in order to avoid issues of flooding and its associated communication overhead.

Limiting user cooperation to query handling is a different approach than the one that we took in distributed storage management (Chapter 2). There are two reasons for this. First, we have already studied the effect of cooperation on decision making plan (i.e., which samples to locally keep), and we already know that it improves performance. Therefore, to isolate the effect of the information-theoretic framework that we developed, we opted to limit cooperation on query handling.

The second reason is that, when we assume that the field sampling process is controlled by an external entity, driven by the need to save the device battery, the number of samples a node could actually acquire is limited. Hence, introducing node cooperation on the decision making plan (e.g., the sample diffusion process of APR) gives each node a chance to be “exposed” to a larger set of samples. With the end result that each node chooses samples that it locally keeps from a large set of samples, undoubtedly, improving the performance. The case is different, however, when we assume that the distributed service is able to sample the field at any instant. In this case, nodes are already exposed to a large number of samples. Hence, a process similar to the sample diffusion process is still expected to improve the performance, since each node ends up with a better set of samples. However, its effect is not expected to be substantial.
3.5 Performance Evaluation

In this section we evaluate the performance of the different proposed storage and sensor management techniques. We start in Subsection 3.5.1 with a description of the data generation models we used to generate the input data. In Subsection 3.5.2, we provide details of our evaluation methodology. Next, in Subsection 3.5.3, we introduce the performance metrics we use in our evaluation. Finally, we present results of our experiments in Subsections 3.5.4, and 3.5.5.

3.5.1 Data Generation model

In this subsection, we describe the two data generation models we used in this study.

Model 1: A Gaussian Phenomenon: In the first model, the underlying space-time distribution of the phenomenon is a multivariate Gaussian. Thus, the field distribution is fully captured by the mean vector and the joint spatio-temporal correlation (STC) matrix $R, L^2 \times T \times L^2 \times T$. To generate the field, we first generate the data to satisfy the spatial correlation using the standard Cholesky decomposition transformation by pre-multiplying a matrix of independent Gaussian random variables by the square root of the desired spatial covariance [Rub81]. Each individual temporal signal associated with a given location is then filtered using a temporal filter to provide the correct spectral shape. This approach results in an STC covariance structure where the off-diagonal blocks are scalings of the diagonal blocks with a scaling factor that depends on the corresponding time lag. Here we note that other methods based on techniques described in [HY00] could also be used for generation of fields with arbitrary joint space-time correlation.

Model 2: A Random Phenomenon: In the second model, the generated data does not follow a Gaussian distribution. The purpose of this experiment is to study the performance of the CBT technique proposed in Section 3.3, which only requires knowledge of the second-order statistics, when the underlying field follows an arbitrary distribution. We generated data that satisfies a desired STC by first applying a spatial transformation to a vector $V$ of
uniformly distributed random variables, and then by filtering the resulting vector through an autoregressive (AR) digital filter to introduce the desired temporal correlation. The coefficients of the autoregressive filter were obtained using the standard Levinson-Durbin algorithm which takes as input the targeted correlation for the different time lags, and outputs the filter coefficients for the specified order [Kay88]. Since the driving noise (V) we used in the first place is non-Gaussian, the resulting process is also non-Gaussian, and only matches the second-order statistics requirements.

3.5.2 Simulation Model and Methodology

We assume that n nodes, each with a storage of size c, perform a random walk in a 2-D field of dimensions L × L. At every time unit, each node decides whether or not to sample its current location. This decision is made based on the utility that this new sample provides compared to utility of the original storage content. If the new sample does not increase the utility of the storage, it is not kept in the storage. Otherwise, one of the old samples that provides the least utility is evicted in favor of the newly acquired one. After allowing a warmup period of w time units, each node is required to answer a query every time unit. The query specifies a query target, and the answer is an estimate of the value of the phenomenon at the query target given each node’s locally kept field samples. Notice that each node is asked an independent query whose target is drawn from the spatial query distribution Q. This distribution is assumed to be a bivariate normal distribution whose mean is the center of the field, and variance is \( \sigma_Q^2 \times I \), where I is the identity matrix of size 2 × 2. The answer to any query is calculated using Eq. (3.5), where Y in Eq. (3.5) is the vector of locally kept samples by the queried node.

In the experiment with the Gaussian phenomenon, evaluation of the posterior densities by the mobile nodes only required evaluation of a mean vector and a covariance matrix which capture the entire distribution. However, in the non-Gaussian scenario, the computational complexity of DEBT becomes prohibitively expensive, especially for large storage sizes. The reason is that the evaluation of the posteriors requires marginalization of the space-
time distribution over the range of the variables of interest for the entire duration of the evaluation (i.e., length of the simulation in time units). Hence, in the experiment with the Random phenomenon, we only evaluate CBT.

In order to assess the robustness of CBT to model mismatch, we also conducted another experiment in which noise is added to the second-order statistics knowledge used by the nodes for managing their storage (to reflect uncertainty in correlation knowledge). We then evaluate the performance for different signal-to-noise ratios (SNR), where SNR is defined as:

\[
SNR = 10 \log_{10} \frac{\sigma^2}{\sigma^2_{\text{noise}}} \quad (3.9)
\]

where \( \sigma^2 \) is the variance of the phenomenon, and the added noise is Gaussian with mean \( \mu = 0 \), and variance \( \sigma^2_{\text{noise}} \). We experimented with SNR’s = 2db, and 15db.

To quantify the gains achieved by the proposed techniques, we compare them to random storage management, which provides us with a lower bound on performance. With random storage management, at every time unit, each node randomly decides whether or not to sample its current location. If a node decides to sample its current location, and its storage is full, it randomly chooses one of its local samples to be evicted to accommodate the newly acquired sample.

In the following evaluation, we set the default value of the parameters of our simulation and data models as follows. \( L = 8 \), \( c = 10 \), \( n = 5 \), simulation time = 100 time units, warmup time \( w = 50 \) time units, variance of the Gaussian phenomenon \( \sigma^2_G = 50 \), variance of the random phenomenon \( \sigma^2_R = 50 \), and variance of the spatial query distribution \( \sigma^2_Q = 4 \). The default mobility model is a random walk on a 2D discrete field, under which, each node is initially placed at random location in the field. Then at every time unit, each node moves to one of its four neighboring locations with the same probability (i.e., 0.25 for each location).
3.5.3 Performance Metrics

The main performance metric we used in our evaluation is the Mean Squared Error (MSE): Given a specific query, a node returns an estimate of the value of the phenomenon at the query location. We then measure the mean squared error associated with this estimate. Thus, given a query $q$ at time $t$ whose target is $\ell(q)$, the MSE in the estimation of $q$ is:

$$MSE = E[(V_{\ell(q),t} - \hat{V}_{\ell(q),t/s_t})^2]$$

We calculate the MSE for each query received by each node after the warmup period, then we report the average of 20 independent simulation runs.

We start by showing results of a single node as a function of the storage size $c$, and the variance of the query distribution $\sigma_Q^2$. Then we show results of cooperation between a number of nodes. More results can be found in [MBM08a].

3.5.4 Single-Node Results

**Effect of Storage Size:** Figure 3.1 (left) shows the effect of storage size on the MSE of the different considered strategies for a Gaussian and non-Gaussian phenomena. Intuitively, as the storage size increases, the better the MSE performance of CBT and DEBT since a larger storage size implies a better reconstruction of the phenomenon by the queried nodes. DEBT has a lower MSE compared to CBT, however, CBT’s performance is very competitive at a much lower computational cost.

Similar effects could also be observed for the non-Gaussian phenomenon (Figure 3.1 right), regarding the efficiency of CBT. CBT outperforms random storage management by two orders of magnitude. As expected, adding noise to the correlation structure of the phenomenon (i.e., decreasing SNR), degrades the CBT performance. However, even with SNR of as low as 2db, CBT still outperforms random storage management with a significant gain.
Query Spatial Distribution Variance: Figure 3.2 quantifies the effect of a larger variance, $\sigma_Q^2$, for the query distribution on the MSE for both Gaussian and non-Gaussian phenomena. Intuitively, a larger variance implies more uncertainty in the target query locations for a fixed storage size and a fixed number of nodes, which explains the decrease in estimation quality for the various schemes.

In case of a Gaussian phenomenon (Figure 3.2 left) both DEBT and CBT have MSE that is an order of magnitude lower than that of random storage management. While in case of a non-Gaussian phenomenon (Figure 3.2 right), CBT achieves a huge improvement over random storage management, with respect to the MSE. Adding noise to the correlation information decreases the performance of CBT, but is still much better than random storage management.
3.5.5 Multi-Node Results

In the following experiments, we gauge the performance improvement due to cooperation between multiple nodes, as we explained it in Section 3.4 for a non-Gaussian phenomenon. Intuitively, we expect cooperation between nodes to improve the performance of all techniques, where the degree of improvement depends on the density of the nodes. We study this effect by varying the storage size and the number of nodes in the field. We also plot the cooperation gain, which is defined as the ratio between MSE from experiments with one node to MSE of the same node when there are n cooperating nodes in the network. In the following experiments, \( n = 5 \), and communication range = 8.

Effect of Storage Size: Figure 3.3 shows the effect of storage size on the MSE of the different considered strategies for a non-Gaussian phenomenon. The improvement of MSE due to cooperation is evident. It is clear that, after increasing the storage size to a certain point, cooperation causes the gap between random and CBT to shrink. The reason is that, at this point, there is enough storage capacity in the system, such that the performance of a smart algorithm and that of a naïve algorithm seem to be close. However, the improvement of performance comes at a cost of added communication overhead. This is an important factor in system design. It implies that, in dense systems where nodes are not power-limited, a smart caching algorithm is not the only option to consider. However, in sparse systems, or in systems where nodes are power-constrained, applying a smart caching algorithm makes a noticeable difference in performance.

Effect of Query Variance: Figure 3.4 shows the effect of varying the variance, \( \sigma^2_Q \), of the query distribution on the MSE of CBT and random storage management for a non-Gaussian phenomenon. General trends apply, where increasing \( \sigma^2_Q \) increases MSE of CBT, while cooperation helps reduce it. It is clear that cooperation does not help random storage management much, as CBT achieves more than an order of magnitude improvement compared to random storage management.

Effect of Number of Nodes: Figure 3.5 shows the effect of varying the number of nodes, \( n \), on the MSE of CBT and random storage management for a non-Gaussian phenomenon.
Increasing the number of nodes increases the amount of cooperation between nodes, and the storage capacity of the entire system. This improves the estimation by all nodes. Random storage management has noticeable improvement as we increase the number of nodes. This trend matches the expectation that when storage is abundant, the storage management algorithms make a minor difference. However, for all the parameter ranges we experimented with, CBT, even with noisy versions, performs better than random storage management.

3.6 Related Work

Many research efforts addressed the problem of sensor management. In order to save energy in the context of caching, Kotidis [Kot05] tries to optimize energy consumption by trying to put some sensor nodes to sleep mode, without affecting the query ability of the network.
This is done by building a correlation model for the samples of sleeping nodes in neighboring active nodes. However, the built model is only local and can not be used to answer general queries about the entire network. It also involves packet exchange and fitting neighbors’ data to a linear model. In the model we used, given knowledge of the spatio-temporal correlation model, we use it to locally (with no packet exchange) answer queries about the entire network.

Sensor database systems (e.g., [MFHH03, MFHH02, BGS01]) sensors are put into sleep mode to save power, then are scheduled to wake up at specific times to sample the field. In [MFHH03], Madden et al. schedule sensors based on the required reporting rate of any given query. In this model, time is divided into epochs. In order to save battery power, nodes are put in sleep mode for most of each epoch, then wake up to sample their current location, and send their collected samples to the sink, provided that the reporting rate of the query is satisfied. The authors assume that all sensors are synchronized, hence, no samples are lost, due to a node being in sleep mode.

In all these research efforts, sensor nodes are assumed to be static, hence, factors that controls sensor management is either to sample the field to satisfy a given reporting rate, or to help forward collected samples to the sink. Our model is different in that, nodes are mobile, hence, the locale of a node changes while in sleep mode. Hence, the decision to sample the field, is based on the “importance” of the field location where the sensor ends up in, with respect to the spatial query distribution $Q$. 

Figure 3.5: Performance of multiple cooperative nodes as a function of the number of nodes $n$. 

<table>
<thead>
<tr>
<th>Number of Nodes</th>
<th>Mean Square Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td></td>
</tr>
<tr>
<td>CBT</td>
<td></td>
</tr>
<tr>
<td>CBT, SNR = 15db</td>
<td></td>
</tr>
<tr>
<td>CBT, SNR = 25db</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.6: Graph showing the mean square error as a function of the number of nodes.
We utilized information theory to assign a measure of merit to any set of samples. Information theory has been used in similar problems [MDMLN03].

3.7 Conclusion

In this Chapter, we adapted an information-based model to represent the target phenomenon. Then, we presented an information-theoretic framework that solves problems of storage and sensor management, as well as query handling. This framework is powerful to handle any spatial distribution of query interest over the field. We proposed two techniques DEBT and CBT to solve the problem, each with different assumptions about the target phenomenon, and different computational complexities. We evaluated both techniques and showed that the resulting gains in MSE are substantial for both Gaussian and random phenomena. Furthermore, CBT still delivers very good estimation of the field, even when its knowledge about the correlation structure is not perfect.

In conclusion, we point out that, DBET’s assumptions are not practical for deployment on resource-limited embedded sensors. The development of DEBT was motivated by the need to provide some sort of an upperbound, to compare other approaches to. While, we make no theoretical claims about the optimality of DEBT, it has been shown through simulations that its performance is superior to other techniques. On the other hand, CBT is less intense with respect to its information requirements, its computational complexity, and its sensor management. Interestingly, our results show that, CBT is able to deliver very competitive performance compared to that of DEBT. This result is encouraging, since it designates the spatio-temporal structure (STC) as a valuable piece of information, that could significantly improve system performance.

Finally, we stress that the design principles upon which this framework is based are well-suited to distributed resource management of autonomous systems. These principles are:

- No system-wide assumptions are made upon which the functionality of the system is
based.

- Cooperation between users is limited to users that are either co-located, or are physically close to each other. This is done to avoid complexities of multihop routing algorithms, their limitations, and the associated communication overhead.

- Equipping nodes with a knowledge about the STC of the target phenomenon is enough to boost the estimation accuracy of the system by orders of magnitude.
Chapter 4

Distributed Mobility Management in FMAs

In this chapter, we turn our attention to address the problem of service provisioning through distributed mobility management. Due to abundance of conventional resources (e.g., storage, battery power) in these systems, prudent management of these resources is not a major concern of the system design, although it is still a favorable characteristic. The scarce resource in such systems, and the one that we deem worthy of careful coordinated management, is the amount of slack in schedules governing node mobility.

Specifically, we address a case in which, the mobility of each node is coarsely directed by an external schedule. A node’s schedule specifies some waypoints, in the space-time plane, that have to be visited by this node. An important attribute of such a schedule is how tight/relax are the consecutive journeys between waypoints. That is, if a schedule allows much more time than the needed minimum for a node to reach each waypoint, then it would be a relaxed schedule with plenty of slack, otherwise, it would be a tight schedule. Slack is the resource that we identify as worthy of management in such systems to optimize performance of the distributed services provided.

We harness this model in message delivery applications (MDAs) in the context of DTNs in Chapter 5, while in this chapter, we apply this model to field monitoring applications (FMAs). Here, we assume a system of $n$ mobile nodes, such that mobility of each node is governed by a schedule. The goal is to plan mobility of each node in order to: 1)
satisfy schedules of all nodes, and 2) use the collective mobility of nodes to achieve specific monitoring of the field. We coin this problem: the Constrained Mobility Coordination problem for Preferential field Coverage (CMC-PC).

We start by motivating the model we envision, then, present a detailed description of the setup and goal. Then, we show that the CMC-PC problem is NP-complete (by reduction to the Hamiltonian Cycle problem). Next, we survey the related work, pointing out the inefficiency of existing proposals to handle the CMC-PC problem. Then, we propose a distributed heuristic to achieve field monitoring that is as close as possible to the required monitoring distribution. We verify the premise of our mobility planning technique using extensive simulations, as well as taxi logs from a the San Francisco area. Our results underscore the evident performance improvement attained by following our mobility strategy.

Contributions of this chapter are:

- We identify mobility of autonomous nodes as a resource that could be leveraged in order to support distributed service provisioning.
- We develop a framework that captures such mobility and its features (i.e., slack) that should be carefully coordinated in order to optimize performance of distributed services.
- We apply our framework to the problem of distributed field coverage, show that this problem is NP-complete and argue that none of the existing research efforts is adequate to solve the problem.
- We develop TFM, the first mobility coordination strategy that aims to achieve a given distribution of field coverage. Under TFM, in steady state, nodes are in a dynamic (i.e., mobile) state. This salient characteristic of TFM enables it to achieve the required coverage distribution of a spatio-temporal field with a low-density network.
- Using extensive simulations, we compare TFM to random mobility strategy — the latter provides a lower bound on the performance. Our results indicate the big per-
formance gain attained by using TFM over random mobility.

- Furthermore, we perform a trace-driven evaluation of TFM and random mobility. We use cab traces from cabs in the San Francisco area. Results of the trace-driven evaluation underscore the superiority of TFM in practical settings.

## 4.1 Motivation

The problem of controlling mobility of a number of objects (e.g., robots) in order to cover a given field is a well-studied problem in the literature. In this model, node mobility could be used to address any combination of the following problems:

- Circumvent low density of nodes in covering a large field such that there is no static arrangement that could achieve the requested coverage,

- Navigate to hard-to-reach areas (due to natural barriers) in order to achieve uniform coverage of the field,

- React to some change in the environment (e.g., forest fire), or address preferential coverage based on changing demands.

In such a model, it is usually assumed that the mobile nodes are under the control of a single authority that decides the mobility pattern of each mobile node.

In accordance with our model of service provisioning through cooperation and resource management of autonomous users, we envision a different model for field monitoring. We envision a setting in which a number of autonomous mobile users are interested in monitoring a given field according to some distribution. This distribution defines the percentage of time different field locations should be covered by, at least, one sensor. The field monitoring distribution stems from the inherent interest of users to query the state of different field locations. This interest is expressed in querying different field locations according to the target monitoring distribution. We also assume that mobility of each user is coarsely directed by an external schedule. A node’s schedule defines a list of locations and a corresponding
list of times, such that for a node to satisfy its schedule, it has to be present at the specified locations at the indicated times. This view is consistent with our basic assumptions about user autonomy, whereby each user is expected to have some plan dictating waypoints for their general mobility pattern.

In such a case, mobility coordination between nodes is an important factor to achieve the requested monitoring distribution. To see why this is the case, consider a situation where a user moving between two points A and B may have multiple choices of paths of almost equal expected quality (e.g., in terms of travelled distance or time). Taking any of the alternative paths leads to monitoring different field locations. Such a scenario is particularly true for paths between locations in a dense urban setting. As an illustration, Consider Figure 4.9, which shows paths followed by cabs on the streets of the San Francisco Bay area. The grid structure of the paths taken (underscoring the underlying city blocks in SF) demonstrates the existence of multiple routes of indistinguishable lengths, to travel between arbitrary points A and B on the grid. In such a case, it is perceivable that one might think that all nodes would satisfy their own schedules in one of the following manners.

- Nodes would prefer paths leading to monitor the high-demand spots in the field, or
- Nodes would take random routes in each journey between each two consecutive waypoints in the schedule.

In the first scenario, if all users end up monitoring the same (highest-demand) field locations, the rest of the field would be left unmonitored, resulting in missing many of the users queries. On the other hand, if nodes take random paths, as we will show in the evaluation section (Section 4.5), this will lead to poor coverage of the field, since the “importance” of each field location (indicated by the target monitoring distribution) will be ignored in making random mobility decisions.

This accentuates the importance of coordinating the mobility of users, while ensuring that all schedules are satisfied.
<table>
<thead>
<tr>
<th>Time</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>05</td>
</tr>
<tr>
<td>30</td>
<td>25</td>
</tr>
<tr>
<td>45</td>
<td>35</td>
</tr>
<tr>
<td>60</td>
<td>30</td>
</tr>
</tbody>
</table>

Schedule

Figure 4.1: Visualization of a node schedule: Schedule entries are marked with circles. Rectangles mark the legitimate paths a node could take during each journey.

**Leveraging Spatio-Temporal Slack for Mobility Coordination:** Assume (for now) that nodes move in one dimension, and consider a node $q$ whose schedule is given in the table in Figure 4.1. Each entry in this schedule gives the location and a corresponding time for a *waypoint*. For node $q$ to meet this schedule, it has to be present at the given location at the specified time for all waypoints in the schedule. Each two consecutive entries (waypoints) in such a schedule define a *journey*. Without loss of generality, assuming a maximum speed of unity, the schedule given in the table in Figure 4.1 allows $q$ slacks of 9, 5, and 10, in the first, second and third journeys, respectively. Figure 4.1 illustrates this schedule (and the slack it allows) by showing the location coordinate of an entry in the schedule on the x-axis, and the time coordinate of that entry on the y-axis. The rectangles shown in Figure 4.1 enclose the set of feasible (legitimate) paths in space-time that $q$ could take during any journey. The more slack that $q$ has, the wider the rectangles, and vice versa (if there is no slack whatsoever, then the rectangle will be reduced to a straight line, *i.e.*, the shortest path). It is clear that the higher the value of the slack, the more the options available for a node regarding which field locations could be visited while satisfying its schedule, and vice versa. Hence, as we argued before, in this setting, the slack in a node’s schedule is the valuable resource that we need to coordinate in order to achieve the requested field monitoring.
4.2 Problem Definition

In this section, we define the Constrained Mobility Coordination problem for Preferential Coverage (CMC-PC), then show that it is NP-complete.

Definition 1 (Nodes): N autonomously mobile nodes move in the target field. The mobility of each node is externally constrained by a schedule (Definition 2). The prime goal of these nodes is to satisfy their own schedules. While doing so, they also try to cooperatively cover the target field according to the required coverage distribution (Definition 4).

Definition 2 (Schedule L): A schedule of node $n_i$ is a list $L(n_i)$ of tuples of the form $u_{ij} = (\tau_{ij}, l_{ij})$, where $1 \leq j \leq |L(n_i)|$. To satisfy a schedule entry $u_{ij}$, node $n_i$ has to be at location $l_{ij}$ at time $\tau_{ij}$. For $n_i$ to satisfy its schedule, it has to satisfy $u_{ij}$ for all $1 \leq j \leq |L(n_i)|$.

Definition 3 (Field G): The target field is represented as a graph $G = (V, E)$, such that each vertex $v \in V$ represents a field location, and each edge $e \in E$ connects two vertices representing two field locations that could be directly reached from each other.

Definition 4 (Coverage Distribution D): Coverage of a given field is defined by a target coverage distribution $D$, such that $D(v)$ is the relative importance of field location $v \in V$. The coverage distribution $D$ represents the preferential interest in covering different locations in the field, and is application-specific.

Practically, $D$ could be interpreted in a number of ways. For example, we could require that more important field locations be covered more frequently than less important ones. Another interpretation, is to require that more important locations be covered for longer periods compared to less important ones. In this paper, we adapt the latter interpretation. Specifically, we interpret $D(v)$ as the required percentage of time, during which, field location $v \in V$ should be covered, by at least one node. We also note that, at any time, a field location is either covered or not. Hence, covering a given location with only one node is exactly equivalent to covering it with more than one node.
Definition 5 (Communication Range \(r\)): Any two nodes can communicate with each other only if the distance between them is less than or equal to a fixed communication range \(r\).

Definition 6 (Speed of Motion \(\eta_i\)): The maximum speed of motion of a node \(n_i\) is \(\eta_i\). Without loss of generality, we assume that \(\eta_i = \eta_{\text{max}}, 1 \leq i \leq N\).

Definition 7 (The CMC-PC Problem \(P\)): The Constrained Mobility Coordination problem for Preferential Coverage CMC-PC is defined by the tuple \(P(G, D, N, L)\), such that \(G\) is a given field to cover with a target distribution \(D\) using a set of \(N\) mobile nodes, each with its own schedule \(L(n_i)\).

In order to solve a given instance of the CMC-PC problem, we need to coordinate mobility of the \(N\) nodes in order to achieve two goals: 1) satisfy schedules \(L\) of all nodes, and 2) Cover each field location, \(v \in V\), the percentage of time indicated by the target distribution \(D(v)\). Clearly, any feasible solution to the CMC-PC problem must satisfy the maximum speed requirement, \(i.e., \) no node is allowed to move with a speed higher than \(\eta_{\text{max}}\).

Theorem 1 shows that CMC-PC is NP-complete by reduction to the Hamiltonian Cycle Problem. A Hamiltonian Cycle is a cycle in an undirected graph which visits each vertex exactly once and also returns to the starting vertex. Determining (and finding) whether a Hamiltonian cycle exists in a given graph is NP-complete.

**Theorem 1** The CMC-PC problem is NP-complete.

**Proof 2** We show that a simple instance of the CMC-PC problem is equivalent to the Hamiltonian cycle problem. Consider the following parameterization of the CMC-PC problem. The target field is represented by a connected graph \(G = (V, E)\), such that \(|V| = T\). The monitoring distribution function is a uniform function. This means that each field location has the same importance as every other location, so the goal is to spend the exact amount of time at each location. In graph terminology, this corresponds to visiting each
vertex \( v \in V \) exactly the same number of times. Also, assume the number of nodes in the system \( N = 1 \). The schedule of this node has only two entries \( s_0 = \{(1, v_0), (T + 1, v_0)\} \), i.e., this node starts at some field location \( v_0 \in V \) at time 1 and ends at the same location at time \( T + 1 \). Coupling the uniform monitoring distribution with the given node’s schedule, we end up with the requirement that this node is required to visit each field location \( v \in V \) exactly once (recall that \( |V| = T \)). Hence, we need to plan mobility of the given node that starts and ends at location \( v_0 \) and visits each field location \( v \in V \) exactly once. This defines an instance of the CMC-PC problem as \( P'(G, \text{uniform}, 1, L(n_1) = \{(1, v_0), (T + 1, v_0)\}) \).

Solving the instance \( P' \) amounts to finding a Hamiltonian Cycle in the graph \( G \) starting and ending at vertex \( v_0 \). Since finding Hamiltonian cycles are NP-complete, then so is the CMC-PC problem.

### 4.3 Related Work

The problem we study here is related to main research areas which received a lot of attention from the community. Specifically, these areas include sensor deployment and redeployment, field coverage, and motion planning.

Studying coverage of static sensor networks is a fairly well-studied problem. Multiple research efforts [MKPS01, HT03, DC03, SSS03] concentrated on calculating the coverage level attained by a static network. Meguerdichian et al. [MKPS01] find the maximal breach and support paths resulting from a static configuration of sensors in the field, while Huang et al. [HT03] study the decision problem if each field location is covered by at least \( k \) different sensors in a static sensor networks. Dhillon et al. [DC03] formulate the coverage problem as an optimization problem where they optimize placement of sensors in the field to maximize attained average coverage of the field. They also model preferential coverage of certain field spots. Following a theoretical approach, in [SSS03], the authors study the field coverage and connectivity of the network as a function of the communication range and probability of node failure, when the number of nodes in the network goes to infinity. As we
mentioned, all these research efforts study the coverage properties of a static configuration of nodes in the field.

Another group of research efforts concentrate on the effect of mobility on network coverage [GGT06, ZC03, HMS02, LDJ07, OECC04, YC07]. Most efforts in this group start from a sub-optimal deployment of nodes in the field (e.g., random), calculate an “optimal” deployment, and then move each node to its newly calculated location. These efforts differ, basically, in the way they calculate the new locations of sensors. In [HMS02], Howard et al. introduce the idea of controlling nodes mobility using potential fields. In this framework, nodes are viewed as objects of some mass that exert either attractive or repulsive forces on each other, based on distance between them. Field boarders also exert repulsive forces on nodes. Mobility of any node is the result of the sum of all these forces. Based on the same idea, Zou et al. [ZC03] add to the model forces resulting from preferential coverage, and propose VFA, a one-time centralized mobility strategy to redeploy sensor after initial deployment. In [LDJ07], the authors use a hierarchical structure to apply the potential field idea to coordinate mobility of larger groups of nodes (members of different groups might not have direct communication to coordinate their mobility). In [GGT06], Wang et al. propose three distributed algorithms to achieve uniform coverage of the field. Their algorithms calculate the current Voronoi diagram of the network and move nodes to cover voids in the network. In [YC07], the authors design a centralized algorithm to redeploy sensors in order to equalize power consumption in different areas of the network and alleviate problems resulting from battery depletion of nodes closer to the sink in a uniform deployment.

A common aspect of these efforts is that the steady state of the network is a static configuration of nodes that is considered to be “optimal” in some way (e.g., cover the entire field uniformly, or equalize power consumption at different nodes). So, the network starts from a static configuration, then nodes move once to reach another optimal static configuration. Since, the new configuration is considered to be optimal, there is no need to move nodes again, until a new trigger is introduced (e.g., a node’s battery dies) which
might require redeployment of sensors.

Another group of research efforts concentrated on the attained dynamic coverage of a mobile network. In [LBD+05], authors study the efficacy of a mobile network in field surveillance. They gauge the ability of the network to detect a static and a mobile intruder. They conclude that, random node mobility is the best strategy to detect any mobile intruder. Basnik et al. [BAI06] study the effectiveness of mobile sensors in detecting events taking place on a closed-curve. Events take place at specific locations on this curve. Each event alternates between on and off states, where the length of the interval in each state is exponentially distributed. Finally, Yu et al. [YR08] relax the assumption that events take place only on a closed-curve. They propose a distributed mobility strategy to detect and monitor events taking place in a general two dimensional field. A common factor in these efforts is that the steady state of the network is a dynamic one, unlike previous research efforts. The problem we address resembles problems addressed by these efforts in this regard. However, we address the more general problem of constrained mobility planning of nodes in order to achieve some given monitoring distribution.

Motion planning has been studied in the robotics field [LaV06, Lat91]. Coupling robotics and sensor networks concepts has also been studied [DRS+05, BP03, MGM03]. These efforts study problems of sensors carried by robots, and the required modifications in robots mobility planning in order to support tasks of sensor networks. The problem addressed in this chapter is also different from these efforts in that, we assume that sensors are embedded into platforms that are mobile by nature, and whose mobility has a limited degree of freedom (i.e., slack) that could be planned to optimize performance of the embedded sensor network.

### 4.4 Targeted Field Monitoring (TFM) Mobility Strategy

In Section 4.2, we showed that CMC-PC is NP-complete. In this section we propose the Targeted Field Monitoring (PFC) mobility strategy, a distributed heuristic to solve the problem. To execute this algorithm, each mobile node \( n_i \) needs to know its own schedule
$L(n_i)$, and the target coverage distribution function $D$. This algorithm does not assume existence of a centralized decision-making facility nor knowledge about schedules of other nodes.

TFM uses another algorithm to assign a utility value to each field location, based on the coverage distribution $D$. Then, at each time unit, TFM plans node mobility by selecting the field location to be visited at the next time unit such that it maximizes the utility of visited field locations, satisfying node’s schedule at all times.

Specifically, let us denote the current location of node $n_i$ as $v_c$. Let us also denote the immediately accessible field locations from the current location, the set of neighboring locations, by $N(v_c)$. $N(v_c)$ is the set of field locations $v_f \in V$ such that there exists a direct route (i.e., edge between $v_c$ and $v_f$). Formally,

$$N(v_c) = \{v_f \in V|v_f \neq v_c \text{ AND } e = (v_c, v_f) \in E\} \quad (4.1)$$

Figure 4.2 (left) shows field location $v_c$, and its neighboring locations $N(v_c)$. In order to plan its mobility, node $n_i$ needs to decide at each time unit, which of the neighboring field locations, $N(v_c)$, it will move to next. To that end, $n_i$ executes the Two-phase Utility Assignment (2UA) Algorithm to assign a utility value to each location $v_f \in N(v_c)$. Node $n_i$, then, decides on the next step greedily to maximize the utility of visited locations. We will now describe the two phases of the 2UA algorithm.

**Phase One of the 2UA Algorithm:** During this phase, $n_i$ assigns an initial utility value $U_i(v)$ to each field location $v \in V$. The utility of each field location is a function of the popularity of this location (defined by $D(v)$), the specific node carrying out the calculations $n_i$, and the time of performing this calculation.

More specifically, $n_i$ keeps a local “view” of the field representing the last time each field location was last visited by *any* of the mobile sensors. Let us denote the local view of node $n_i$ as $C_i$, where $C_i(v)$ is the last time field location $v$ was last visited by *any* node, according to node $n_i$ (Figure 4.2 center). Node $n_i$ updates its local view of the field at two occasions: 1) Whenever it visits a new field location, it updates the last time this location
was visited to the current time, and 2) Whenever it encounters another node \( n_j \), the two nodes exchange their views of the field. The result of this exchange is that, each node keeps the most recent version of the two views.

Using its current view of the field \( C_i \), node \( n_i \) calculates utility of field location \( v \) as

\[
U_i(v) = D(v) \times (t_c - C_i(v))
\]

(4.2)

where \( t_c \) is the actual time of performing the utility calculation (i.e., the current time).

Notice that Equation 4.2 is a linear function of the popularity of the location, \( D(v) \), and the length of the interval since location \( v \) was last visited \((t_c - C_i(v))\). Equation 4.2 is just an example for utility calculations, which could take any different form (e.g., exponential in the location utility). Notice also that this equation is related to our interpretation of \( D(v) \) as the required percentage of time during which field location \( v \) should be covered.

**Phase Two of the 2UA Algorithm:** In this phase, node \( n_i \) calculates a coarse utility value, \( \hat{U}_i(v_f) \), for each of the directly neighboring locations, \( v_f \in N(v_c) \). The coarse utility of \( v_f \) is calculated as the sum of utilities of field locations comprising the highest-utility path of length \( h \) that starts from \( v_f \). More specifically, for each \( v_f \in N(v_c) \), we find all paths of length \( h \) that start from \( v_f \), called Potential Future Paths (PFPs). The utility of each PFP is the sum of initial utilities (calculated in Phase 1 of 2UA) of field locations
comprising this path. The coarse utility of $v_f$ is the highest utility of all such PFPs starting at $v_f$. Formally,

$$\hat{U}_i(v_f) = U^p(P_{\text{best}}(v_f, h)) = \sum_{v_x \in P_{\text{best}}(v_f, h)} U_i(v_x)$$  \hspace{1cm} (4.3)$$

where function $U^p(P)$ calculates the sum of utilities of locations in PFP $P$, and $P_{\text{best}}(v_f, h)$ is the PFP with the highest utility and is defined as,

$$P_{\text{best}}(v_f, h) = \{ P'(v_f, h) : U^p(P'(v_f, h)) \geq U^p(P(v_f, h)) \},$$ \hspace{1cm} (4.4)$$

where $P(v_f, h)$ is the set of all PFPs of length $h$ that start from location $v_f$, and $P(v_f, h) \in P(v_f, h)$ is defined as a series of $h$ connected field locations, the first of which is $v_f$. Figure 4.2 (right) shows the current location $v_c = (3,3)$, its directly neighboring locations $N(v_c) = \{(2,3), (4,3), (3,2), (3,4)\}$, and the range of PFPs from the neighboring locations of length $h = 1$. For example, for $v_f = (2,3)$, $P(v_f, 1) = \{\{(2,3), (1,3)\}, \{(2,3), (2,2)\}, \{(2,3), (2,4)\}\}$.

**Scope of PFPs:** In the strategy we described, PFPs constitute some form of lookahead in order to optimize performance, and $h$ is a tunable parameter that defines the exact amount of lookahead to perform. Hence, we need to answer the question: “What is the optimum range of PFPs to consider? Consequently, what is the best value for $h$?” It is natural to think that the higher the value of $h$, the longer the range of considered PFPs, the longer it takes to plan mobility, and the more optimal the mobility decisions are. There is, however, a dynamic restriction on the value of $h$ to be used at any neighboring location $v_f$. Theorem 2 states this restriction, then we give an example to illustrate it.

**Theorem 2** In a field coverage system with a single node, while determining the PFPs $P(v_f, h)$ of a location $v_f$, only field locations that could be actually reached by the node (due to scheduling constraints) should be included in PFPs. Hence, the optimum value of the locale radius $h$ is the amount of slack $k$ available to node at the current time. Using any other values of $h$ could lead to sub-optimal decisions.
Proof 3  The crux of the theorem is that: in order to optimize the mobility planning process, all reachable field locations, and only these locations, should be included in PFPs. Hence we need to show that, in determining locations to be included in PFPs, the following two alternatives may lead to sub-optimal decisions: 1) not including all reachable field locations, and 2) including field locations that are not reachable.

1. First, we show that not including reachable field locations could lead to sub-optimal decisions, i.e., using a value of $h < k$. To show that, we define two sets of PFPs for each neighboring field location $v_f \in N(v_c)$, the first, $\mathcal{P}(v_f, h)$, is of length $h < k$. This set leaves out PFPs that consume more than $h$ units of slack. The second set, $\mathcal{P}(v_f, k)$, includes all PFPs that consume up to $k$ units of slack. Let us also denote all field locations that belong to PFPs in $\mathcal{P}(v_f, h)$ by $\mathcal{A}(v_f, h)$, formally

$$\mathcal{A}(v_f, h) = \{ v : v \in P \text{ AND } P \in \mathcal{P}(v_f, h) \}$$  (4.5)

Similarly, we denote all field locations that belong to PFPs in $\mathcal{P}(v_f, k)$ by $\mathcal{A}(v_f, k)$.

$$\mathcal{A}(v_f, k) = \{ v : v \in P \text{ AND } P \in \mathcal{P}(v_f, k) \}$$  (4.6)

As a result, the following relation is true by definition:

$$\mathcal{A}(v_f, h) \subset \mathcal{A}(v_f, k)$$  (4.7)

We refer to the set of locations in $\mathcal{A}(v_f, k)$ but not in $\mathcal{A}(v_f, h)$ as set of exclusive locations $\mathcal{E}$, formally

$$\mathcal{E} = \mathcal{P}(v_f, k) - \mathcal{P}(v_f, h)$$  (4.8)

Notice that, by definition, all exclusive locations $v \in \mathcal{E}$ are within reach, subject of scheduling constraints, of the mobile node (since we include only locations that consume up to the maximum available slack $k$, and not more).
Now, let us assume that depending on PFPs $P \in \mathcal{P}(v_f, h)$ results in more optimal coverage of the field than depending on $P \in \mathcal{P}(v_f, k)$, where utility of a given coverage is the sum of utilities of field locations actually visited by the node. This assumption means that $U^p(P_{\text{best}}(v_f, h)) \geq U^p(P_{\text{best}}(v_f, k))$. The last relationship cannot be true for the following reasons: 1) By definition $P_{\text{best}}(v_f, h)$ has the highest utility of paths of length $h$, which means that $P_{\text{best}}(v_f, h)$ has the $h$ locations with the highest utilities among all paths of this length. 2) On the other hand, the selection $P_{\text{best}}(v_f, k)$ is optimized with respect to all path of length $k$, which by definition, selects the most optimum first $k$ steps. 3) By definition of a node’s schedule, since this node has a maximum slack of $k$ to reach the destination waypoint, it ends up following a path of length $k$. It is clear that $P_{\text{best}}(v_f, h)$ does not optimize its decisions to select the highest-utility path of length $k$ (the actual length of the path that a node ends up taking), while $P_{\text{best}}(v_f, h)$ does exactly this. and 4) Since we consider a system with a single node, then location utilities can not be altered (after making mobility decisions) by having other nodes visit them, hence past decisions could not be invalidated in the future. Hence we have a contradiction of the initial assumption, and as a result, we prove that depending on PFPs $P \in \mathcal{P}(v_f, h)$ can not result in more optimal coverage of the field than depending on $P \in \mathcal{P}(v_f, k)$.

2. Second, we show that including field locations that could not be reached due to scheduling constraints (i.e., using a value of $h > k$) leads to sub-optimal decisions. Similar to Case 1, we define two sets of PFPs for each neighboring field location $v_f$, the first, $\mathcal{P}(v_f, h)$, is of length $h > k$, and the second $\mathcal{P}(v_f, k)$. Sets $\mathcal{A}(v_f, h)$, and $\mathcal{A}(v_f, k)$ are also defined as in Equations 4.7 and 4.8, respectively. Notice that, unlike the previous case, since $h > k$, then $\mathcal{A}(v_f, k) \subset \mathcal{A}(v_f, h)$. We refer to the set of locations in $\mathcal{A}(v_f, h)$ but not in $\mathcal{A}(v_f, k)$ as exclusive locations $\mathcal{E}$, it is defined as

$$\mathcal{E} = \mathcal{A}(v_f, h) - \mathcal{A}(v_f, k) \quad (4.9)$$
Notice that, by definition, locations \( v \in \mathcal{E} \) are not within reach of the mobile node (since these locations are beyond locations that consume up to the maximum available slack \( k \)).

Following a similar agreement to the one we used in Case 1, we assume that depending on PFPs \( P \in \mathcal{P}(v_f,h) \) would lead to more optimal coverage of the field compared to PFPs in \( \mathcal{P}(v_f,h) \). The last statement could not be true because of the following reasons: 1) By definition \( P_{\text{best}}(v_f,k) \) has the highest utility of paths of length \( k \), which means that \( P_{\text{best}}(v_f,k) \) has the \( k \) locations with the highest utilities among all paths of this length. \( P_{\text{best}}(v_f,h) \) is optimized given the infeasible length \( h \), which does not necessarily select the most optimum first \( k \) steps (like \( P_{\text{best}}(v_f,k) \)). 2) By the definition of the node’s schedule, it ends up actually taking only \( k \) steps (and no more) to reach the destination waypoint. 3) As we argued in Case 1, since the system has a single node, then location utilities cannot be altered by having other nodes visit them, hence past decisions could not be invalidated in the future. Hence we prove that depending on PFPs in \( \mathcal{P}(v_f,h) \) can not lead to more optimal coverage of the field compared to PFPs in \( \mathcal{P}(v_f,h) \).

We state two remarks about Theorem 2

- Theorem 2 addresses the case of a field coverage system comprised of a single node. The reason is that, in case of multiple nodes, mobility planning decisions made by one node in the past, could be “invalidated” in the future due to mobility of other nodes in the system. For example, a field location \( v_x \) of high initial utility according to node \( n_i \) in the past, could be visited by other node \( n_j \) causing its current utility to drop when actually visited by \( n_i \). We argue, however, that Theorem 2 still holds in systems of sparse deployment, we give evidence to this insight in our trace-driven evaluation Section 4.5.

- Theorem 2 does not address optimality of PFPs of any length other than \( k \), the reason is that, in a general multi-peaked coverage distribution \( D \), it could be the case that in
some situations increasing $h$ leads to better performance, and in some other situations, the same increase leads to worse performance, depending on the actual location of the node performing these calculations. In a “simple” single-peaked coverage distribution $D$, it is generally desirable to increase $h$ as much as possible provided that is less than $k$. We show an evidence of this observation in our trace-driven evaluation Section 4.5.

The following example illustrates the idea of Theorem 2, it concentrates on Case 1 of Theorem 2 showing that not including all reachable field locations in deciding the PFPs of each $v_f$ results in sub-optimal coverage of the field.

Table 4.1 gives schedule of node $n_1$, while Figure 4.3 left shows $v_c$ the current location of node $n_1$, along with field locations directly accessible from $v_c$, $N(v_c) = \{(2,3),(3,4)\}$. It also shows the initial utility of visiting each location, the result of phase one of the 2UA algorithm. Notice that locations $(1,4)$ and $(2,5)$ are not members of the set $N(v_c)$, the reason is that, schedule of $n_1$ does not allow enough slack to visit any of these locations. Figure 4.3 left also marks the set of field locations that could be reached given the schedule of $n_1$.

Node $n_1$ needs to assign a coarse utility value $\hat{U}_{i}(v_f)$ to each of the neighboring locations. Figure 4.3 (center) shows the PFP with the highest utility for each $v_f$ such that $h = 3$, which matches the time needed to get to the destination. $P_{\text{best}}((3,4),3) = \{(3,4),(4,4),(4,3),(4,2)\}$, while $P_{\text{best}}((2,3),3) = \{(2,3),(3,3),(4,3),(4,2)\}$. In this case, both PFPs visit the high-utility location $(4,3)$. $U^p(P_{\text{best}}((3,4),3)) = 4.3$, while $U^p(P_{\text{best}}((2,3),3)) = 4.1$. Based on these calculations, $n_i$ moves to $(3,4)$ as its next step.

Figure 4.3 (right) shows the same paths when $h = 1$. Notice that in this case, not all reachable field locations are included in the range of PFPs. $P_{\text{best}}((3,4),1) = \{(3,4),(4,4)\}$,
Figure 4.3: Location $v_c$, its neighboring locations $N(v_c)$, the destination waypoint, and the utility of each location (left), $P_{\text{best}}(v_f, 3)$ (center) and $P_{\text{best}}(v_f, 1)$ (right).

while $P_{\text{best}}((2, 3), 1) = \{(2, 3), (2, 2)\}$, and $U^p(P_{\text{best}}((3, 4), 1)) = 0.9$, while $U^p(P_{\text{best}}((2, 3), 1)) = 1.0$. Based on these calculations, $n_i$ moves to $(2, 3)$ as its next step, which is clearly a sub-optimal decision.

### 4.5 Performance Evaluation

In order to evaluate the efficacy of TFM in achieving a specific coverage distribution of a given field, we developed a mobility simulator. Our simulator models the mobility of nodes by keeping track of the location of each node at each time unit. It also models exchange of local views between nodes upon an encounter. Since our goal is to evaluate the synthesized mobility of our detour-based techniques, we make simplifying assumptions about the communication model as we assume that nodes within a certain communication range could successfully exchange data. We assume that the size of exchanged messages is small with respect to the bandwidth in a single contact between two nodes. We also, willingly, overlook storage limitations. We do this motivated by current advances in storage technology that make memory devices of tens of gigabytes available off-the-shelf.

**Performance Metrics** The performance metrics we use are the Kullback-Leibler (KL) distance, and the query success ratio (QSR). The KL distance is a measure of distance between distributions [CT06]. Having a true distribution $P$, and an approximated one $Q$,
the KL-distance between $P$ and $Q$, $DL(P\|Q)$, is calculated as follows:

$$DL(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$  \hspace{1cm} (4.10)

Mobility of nodes over the field induces a distribution $Q$ of the length of periods during which each location is covered. We measure the KL distance between the required coverage distribution, $D$, and the induced distribution $Q$, $DL(D\|Q)$. Lower values of the KL distance indicate that the induced distribution is close to the required distribution $D$, which is the prime requirement in field coverage.

Then, we assume that nodes have unlimited storage in which they keep collected samples from the field. A node keeps a sample from each field location it visits. A sample is assumed to have a time-to-live (TTL), during which this sample is considered to be fresh, i.e., an accurate representation of the target phenomenon at field location where it was collected. Only fresh samples are kept in the local storage, while expired ones are evicted. Nodes are independently queried about the state of the field. A query is defined by a tuple $q = (v_q, p)$. $v_q$ is the query target, the location in which the inquirer is interested, while $p$ is a measure of tolerable imprecision in the answer. The specific locations of query targets follow some spatial distribution over the field. The answer to a query $q = (v_q, p)$ is a sample collected at location $v$, such that the distance between $v$ and $v_q$ is less than $p$ (i.e., $|v - v_q| \leq p$). We also assume that node cooperation is limited to query handling (APR-style Section 2.6). Specifically, to answer any query: a queried node searches its local storage to find a sample that could be used as an answer to the query. If found, then the query is counted as a success. Otherwise, the queried node forwards the query to its direct neighbors only. If one of these neighbors has an answer to the query, this neighbor sends the answer back to the queried node, and the query is counted as a success. If neither the queried node nor any of its neighbors has an answer to the query, it is counted as a missed query. In order to assess the efficacy of each mobility strategy in achieving the required probability distribution, we

\footnote{We can think of this as if the user/owner hosting the mobile node is interested in the state of some location in the field, so he uses the local device, to which the sensor is attached, to submit queries and receive answers back from the distributed system.}
matched the distribution of query targets to that of the required coverage distribution $D$ (i.e., field locations that are required to be monitored for longer periods of time have higher probability of being query targets). We define the query success ratio (QSR) as the ratio between the number of successfully answered queries to the total number of queries.

The point of the two performance metrics is to gauge the degree to which each mobility strategy can match the required coverage distribution $D$. If a mobility strategy could closely match this distribution, this should be manifested in achieving a small KL distance, and high query success ratio.

**Competing Strategies:** We compare TFM to random mobility strategy (RND), in which nodes move randomly between every two consecutive waypoints provided that the schedule is satisfied. In the trace-driven evaluation, we compare TFM to Wait-at-Destination (WAD), a variation of RND. Under WAD, nodes moves to the destination waypoint using the shortest path, where they wait spending all the available slack, if any. Clearly, both RND and WAD represent a lower bounds on performance, since they do not actively attempt to coordinate nodes’ mobility to improve coverage of the field.

In the synthetic evaluations, we evaluated the performance of TFM and RND with respect to query handling in two different setups, distributed and centralized. In the distributed setup, only nodes within communication range $r$ could communicate to cooperate in query handling. This version is marked as “DST” in the following graphs. In the centralized setup (marked as “CTR” in the following graphs), query handling is done in a centralized facility. In this case, we simulate the case where all collected samples are forwarded to a central processing facility, and all queries are directed to this facility.

**4.5.1 Evaluation Using Synthetic Workloads**

**Schedule Generation:** Every node starts at time $= t_{current}$ (initially, $t_{current} = 1$) at a random location in the field $loc_1$. The entry $(t_{current}, loc_1)$ is added to the schedule. Then we randomly select another location $loc_2$ in the field such that the minimum time to move
from \( loc_1 \) to \( loc_2 \) is \( t \). For the \( loc_2 \) we assign time \( t_s \)

\[
t_s = t_{\text{current}} + t + (\kappa \times \rho)
\]  

(4.11)

where \( \kappa \) is the maximum slack we allow in any journey, and \( \rho \) is a uniform random variable such that \( \rho \in [0, 1] \). The entry \((t_s, loc_2)\) is appended to the schedule. We repeat this process until the end of the simulation time is reached.

**Baseline Parameters:** We simulated a field of 10x10 where nodes can communicate only when they are at the same field location. Simulations run for 100 seconds. In the following graphs, each point is the average of 20 simulation runs, with the 95% confidence interval shown as well. The required field coverage distribution is assumed to be a symmetric bivariate normal distribution centered in the center of the field, with variance \( 4 \times I \), where \( I \) is the identity matrix of size \( 2 \times 2 \). In the following experiments, unless otherwise stated, the default value of the maximum PFPs \( h = 1 \), number of nodes \( N = 30 \), and query precision \( p = 1 \).

**Effect of Number of Nodes:** Figure 4.4 shows the KL distance of RND and TFM as a function of the number of nodes in the system, using two different values of the maximum slack \( \kappa = 0 \) (Figure 4.4 left), and \( \kappa = 20 \) (Figure 4.4 right). When there is no slack in the schedule \( (\kappa = 0) \), TFM achieves between 37% and 42% lower KL distance than RND; the difference between TFM and RND is noticeable, albeit not huge. Increasing the available slack allows TFM more freedom to match the required coverage distribution \( D \), resulting in a considerably improved KL distance. A maximum slack of \( \kappa = 20 \) allows TFM to achieve KL distance that is up to 10-times lower than that of RND.

Figure 4.5 shows the effect of increasing the number of nodes on the performance of the system in handling queries. TFM achieves at between 2-fold to 3-fold improvement in QSR over RND. Increasing the TTL of collected samples or the maximum slack \( \kappa \) improves performance of both protocols, especially centralized versions, however, TFM is always superior to RND. It is also interesting to notice that the distributed version of TFM achieves close performance to that of the centralized one. Increasing either TTL or maximum slack
\( \kappa \) diminishes the gap between the two versions. This confirms the premise of TFM in a distributed practical setup. While, there is a noticeable gap between QSR of the distributed and centralized versions of RND, as expected of a naïve approach.

We conclude from Figure 4.5 that the distributed version of TFM has very compatible performance to that of the centralized version. The reason is that: to achieve the requested coverage distribution, different nodes could visit the same field location(s) over a period of time, which requires coordination between nodes, over some period of time. At finer time intervals, only a small group of nodes could visit field locations in a specific area of the field (\( i.e. \), only nodes that are currently physically close to this area). Hence, in order to coordinate mobility of this group of nodes, only these nodes need to communicate. In a distributed setting (where only nodes at the same intersection could communicate), exactly such group of nodes could communicate and successfully coordinate their mobility, due to their proximity. Communication (and coordination) between this group of nodes and different groups that are physically distant (as is the case in a centralized setting), has a limited effect on performance, since there is no potential overlap, \textit{on the short run}, between field locations that could be visited by the different groups, due to their physical distance.

**Effect of Maximum Slack \( \kappa \):** Figure 4.6 shows the KL distance of RND and TFM as a function of the maximum slack \( \kappa \) of the schedule, in a system of 5 nodes (Figure 4.6 left), and 15 nodes (Figure 4.6 right). As we pointed out earlier, increasing \( \kappa \) allows TFM to
match the required distribution resulting in a smaller value of the KL distance. This is true for systems of both high and low densities. Increasing the number of nodes has a more pronounced effect on the KL distance of RND compared to TFM.

Figure 4.7 shows the effect of $\kappa$ on the QSR. Similar to the KL distance, increasing $\kappa$ improves QSR of TFM. This is not always the case for RND, since, nodes under RND do not coordinate usage of their slack to optimize coverage of the field. TFM achieves more than 2-fold improvement in QSR over RND. Also, increasing TTL improves QSR for both mobility strategies. For values of $\kappa > 0$, the distributed version of TFM achieves close QSR to that of the centralized version, again, underscoring the usefulness of TFM in distributed settings.

**Effect of Partially Following Detours:** The goal of this experiment is to measure the effectiveness of TFM when only a given percentage of the nodes follow detour hints provided by TFM. This scenario is motivated by the observation that some nodes may not be willing
Figure 4.7: Query success ratio of TFM and RND as a function of the Maximum slack for different TTL values.

Table 4.2: KL distance of resulting distribution when only a percentage of nodes follow the detours of TFM.

<table>
<thead>
<tr>
<th>percentage</th>
<th>0</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\kappa = 0$</td>
<td>0.8781</td>
<td>0.8537</td>
<td>0.8221</td>
<td>0.7934</td>
<td>0.7774</td>
<td>0.7472</td>
</tr>
<tr>
<td>$\kappa = 20$</td>
<td>0.7385</td>
<td>0.5319</td>
<td>0.4018</td>
<td>0.2984</td>
<td>0.2124</td>
<td>0.1407</td>
</tr>
<tr>
<td>$\kappa = 40$</td>
<td>0.7706</td>
<td>0.5133</td>
<td>0.3588</td>
<td>0.2726</td>
<td>0.1731</td>
<td>0.1175</td>
</tr>
<tr>
<td>$\kappa = 60$</td>
<td>0.7804</td>
<td>0.5209</td>
<td>0.3560</td>
<td>0.2647</td>
<td>0.1834</td>
<td>0.1368</td>
</tr>
<tr>
<td>$\kappa = 80$</td>
<td>0.7326</td>
<td>0.4679</td>
<td>0.3437</td>
<td>0.2523</td>
<td>0.1944</td>
<td>0.1434</td>
</tr>
<tr>
<td>$\kappa = 100$</td>
<td>0.7415</td>
<td>0.4780</td>
<td>0.3424</td>
<td>0.2424</td>
<td>0.1858</td>
<td>0.1586</td>
</tr>
</tbody>
</table>

to participate in the field coverage application, and opt to spend their slack in a different way. Table 4.2 shows the resulting KL distance. Obviously, as the percentage of nodes following TFM hints increases the resulting KL distance decreases. When 60% of the nodes following TFM hints, the resulting KL distance decreases by up to 67% ($\kappa = 100$).

Figure 4.8 shows the QSR as a function of the density of nodes following TFM hints in a distributed setting. Where density is defined as the number of nodes $N$ over monitored area $A$ (i.e., density = $N/A$). We experimented with three values of $\kappa$: 0 (left), 20 (center), and 100 (right). In this experiment the total number of nodes is fixed at 30. Similar to KL distance, increasing the percentage of nodes following TFM hints causes linear increase in the QSR of the system. Figure 4.8 reveals two interesting facts: 1) the “height” (i.e., success level) of each curve is a function of the TTL of samples. Higher values of TTL lead to higher QSR. we have seen similar effects in pervious experiments, and 2) the slope of the
Figure 4.8: Effect of partially following hints on query success ratio for three different values of slack= 0, 20, and 100.

linear increase is a function of the slack. Intuitively, the impact resulting from an additional percentage of nodes following TFM hints is a function of how much slack these nodes have. The more slack the higher the impact (i.e., the higher the QSR). This effect is manifested in the increased slope of the QSR curves. We conclude that, using more nodes to cover the field enable the distributed service to attain better coverage of the field. However, it is clear that, with enough schedule slack, as low as 0.3 density of nodes following TFM hints could achieve QSR approaching 100%.

4.5.2 Trace-Driven Evaluation

Following our motivating application, and to use even more realistic evaluation of the protocol we proposed, we used taxi traces [cab] for cabs in the San Francisco area as input to our models. The goal is to show that, with little coordination between cabs, they could function as an effective distributed field coverage system.

**Methodology:** For each cab, the traces show location updates of the cab. Each update is composed of latitude and longitude of the cab location, the time of the location update, along with the cab status: metered (hailed) or not (not hailed). We gathered more than a full day’s worth of data for more than 450 cabs. In the traces we collected, some cabs have as many as 400 location updates, while others have as few as 5 updates. We used all location updates for all cabs to construct a “map” of the San Francisco area. We represented the map as an undirected graph $G = (V, E)$. $V$ is the set of all legitimate locations any cab can be in at any time, where a location is defined by its latitude and longitude coordinates.
In the data we collected, the total number of locations is 40399, and the number of unique locations, $|V| = 39,103$ locations. To determine the relation between different locations (i.e., the edges, $E$), we used a threshold-based neighborhood algorithm with a threshold value $r_{th}$. This means that, for any two locations $a, b$, such that the distance between them is $Dist(a, b)$, if $Dist(a, b) \leq r_{th}$, then we add an edge between $a$ and $b$ whose cost = $Dist(a, b)$. We used $r_{th} = 200$ meters ($\approx 0.12$ miles = 656 feet). This value of $r_{th}$ partitioned the unique field locations into different partitions, with the largest partition consisting of 36,368 unique locations. We used this partition as a representative of the map. A depiction of this map is given in Figure 4.9.

Finally, out of the 450 cabs, we selected the 50 cabs with the highest number of location updates. We mapped the location updates of the cabs to the map we generated, and used the map to “fill” in the gap of the missing location updates for the first 150 minutes. This is done by mapping each two consecutive updates to the map, finding the shortest route between them. Next, we interpolate a number of locations along this route that is equal to the number of minutes between the location updates. This process allows us to infer the location of cabs at one-minute granularities. The cab status for those interpolated locations is set to be its reported status in the last location update.

Based on each cab’s mobility profile (obtained as described above), we defined the schedule of the cab as follows: every time the cab is metered, its location is added to the
schedule of the cab. This means that, if the cab is hailed (according to location updates), then it has to be in the indicated location at the indicated time. In other words, we can not change the location of a hailed cab. This leaves room for offering hints to the cab only when it is not hailed.

We compared TFM and Wait At Destination (WAD), a variant of the RND protocol. Under WAD, when not hailed, a cab moves to the next location where it picks up its next customer, as early as possible, and spends its slack time there waiting for the customer. Throughout the trace evaluation, we assumed that cabs do not exceed speed of 30 mph, which is quite conservative.

**Target Distribution** $D$: We assume that the goal of these cabs is to cover the city tracking a specific phenomenon breaking out at random locations. An Amber alert is issued specifying the break out location (the center of the phenomenon) which is given the highest level of attention. Attention awarded to neighboring locations is a function of their distance from the center of the phenomenon. To model this application, we define target coverage distribution $D$ as follows: We start with a maximum utility value $M$ and a utility decrement value per hop $\delta$. Then we randomly select a field location, $v_1$, to be the center of the distribution. We assign virtual utilities $u_v$ as follows

$$u_v(v_i) = M - \delta \times |v_1 - v_i|$$  \hspace{1cm} (4.12)

where $u_v(v_i)$ is the virtual utility assigned to field location $v_i$, and $|v_1 - v_i|$ is the number of hops between $v_1$ (the center of the distribution) and $v_i$. This function assigns to $v_1$ the maximum value of utility, $M$. The utility value of field location $v_i$ drops as a linear function of the number of hops between $v_i$ and the center of the distribution, $v_1$. To get the required distribution $D$, which specifies the percentage of time $D(v_i)$ that field location $v_i$ should be covered, we use the following equation.

$$D(v_i) = u_v(v_i) / \sum_{v_x \in V} u_v(v_x)$$  \hspace{1cm} (4.13)

Figure 4.10 illustrates an example distribution over a compact version of the map.
Figure 4.10: Example of the distribution we used with the San Francisco cab traces. The lighter the area the higher the coverage percentages \( D(v) \). \( v_1 \) in this case is marked by the red (dark) dot in the light area.

Table 4.3: KL distance resulting from applying TFM on cab traces with different values of maximum length of PFPs \( h_{max} \).

<table>
<thead>
<tr>
<th>( h )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>KL distance</td>
<td>1.0986</td>
<td>0.8975</td>
<td>0.7441</td>
<td>0.6528</td>
</tr>
</tbody>
</table>

**Results:** In comments on Theorem 2, we argued that in case of systems with single nodes, the longer the PFPs used to estimate the coarse utility of directly neighboring locations, the better the performance, under two conditions: 1) we limit our consideration to field locations that are reachable under the scheduling constraints of the node, and 2) the desired coverage distribution \( D \) is simple single-peaked (as the one we used in the trace-driven evaluation). In this experiment we aim to evaluate this Theorem in systems with multiple nodes, but low node density. Towards this end, we calculated the KL distance of TFM using different values of the maximum length of PFPs \( h_{max} \). For each value, we run 20 simulations using the inferred schedule and the generated map. Each simulation has a different distribution center \( v_1 \). Table 4.3 shows the average KL distance of the different values of \( h \).

It is clear that the performance improves by increasing \( h \), provided that we only consider reachable field locations, confirming our expectations (formalized in theorem 2). Since \( h = 3 \) yields the lowest value of KL distance, we use this value in the query-based performance evaluation. The KL value of WAD = 2.1, three times that of TFM with \( h = 3 \).
Figure 4.11: QSR of TFM and WAD as a function of TTL of samples (TTL measured in minutes). The graph show results when using two values of the communication range: 600 and 1200 meters.

Figure 4.11 shows the query performance of the two mobility strategies. TFM achieves from 30% to 120% improvement over QSR of WAD. Increasing the communication range improves performance of both protocols. However, we found out in another experiment in which we measured performance as a function of the communication range (results not shown here) that this improvement reaches a fast steady state.

4.6 Conclusion

In this Chapter we discussed service provisioning through distributed mobility management. We argued that, in such systems, the amount of freedom in constraints on node mobility is the scarce resource to be coordinated among different nodes. We showed that this problem is NP-complete. Then, we proposed Targeted Field Monitoring (TFM), a distributed computationally-scalable mobility coordination scheme to control node mobility to achieve the required monitoring distribution of the field, while satisfying schedules of all nodes. We evaluated TFM through synthetic as well as trace-driven simulations using cab traces. Our results indicated that, under TFM, nodes achieve a very close field monitoring distribution to the required distribution. This is also manifested in achieving an excellent performance in terms of query success ratio.

In conclusion, we stress that, to our knowledge, TFM is the first mobility strategy that
aims to achieve some distribution of field coverage, such that in steady state, nodes are in a dynamic (i.e., mobile) state. This salient characteristic of TFM enables it to achieve a specific monitoring distribution of a spatio-temporal field with a low-density network. As we argued in Section 4.3, under other mobility planning algorithms, nodes are static, in an “optimal” configuration to cover the field. All these strategies fail to address the problem we addressed in this Chapter for two reasons:

- In case the density of nodes is low, a static configuration of the nodes will persistently miss some field locations. Hence, we need a mobility planning algorithm, under which, nodes are mobile in the steady state to solve this problem.

- Node schedules constantly force nodes to be mobile. Current mobility planning algorithms can not handle this case, since they need to put nodes in a static configuration.
Chapter 5

Distributed Mobility Management in MDAs

We continue our consideration of service provisioning through distributed mobility coordination. In this chapter, we concentrate on message delivery applications (MDAs) in the context of delay tolerant networks (DTNs). Similar to Chapter 4, we assume that node mobility is controlled by a schedule featuring some slack, and that the slack of the schedule is the resource that we coordinate among different users of the system. However, unlike Chapter 4 where the idea of mobilizing nodes to achieve field coverage was proposed in the literature albeit in different contexts, the idea that we propose in this chapter is original.

We summarize the best contributions of this chapter as follows.

- Our framework is general enough that it allows us to model and evaluate some of the existing message delivery schemes in DTNs, including data mules and message ferries.

- We formally define the problem and show that it is NP-hard.

- We propose two detour-based approaches to solve the problem. The first (DMD) is a centralized heuristic that leverages knowledge of the message workload to suggest specific detours to optimize message delivery. The second (DNE) is a distributed heuristic that is oblivious to the message workload, and which selects detours so as to maximize node encounters.

- We perform extensive performance evaluation of our proposed detour-based approaches
using based on synthetic workloads. We, also, use the same traces that we used to evaluate FMAs in Chapter 4. Our evaluation shows that our centralized, workload-aware DMD approach yields the best performance, in terms of message delay and delivery success ratio, and that our distributed, workload-obliviou DNE approach yields favorable performance when compared to approaches that require the use of data mules and message ferries.

\section{Motivation}

In a Delay-Tolerant Network (DTN), it is generally assumed that, on the one hand, there is no end-to-end path between a message’s source and its destination, but that on the other hand, messaging between mobile nodes does not require immediate delivery. Email delivery is a canonical example of a DTN application over traditional IP networks. There are many motivations for assuming an infrastructure-less networking environment. In some cases, such an assumption is necessary as is the case with networking applications envisioned for rural, under-developed, or impoverished milieus. As we argued in Chapter 1, the cost of laying down supporting infrastructure in these areas is a strong incentive for depending on a delay-tolerant-type model, where message delivery is done in a distributed fashion. In other cases, such an assumption may be motivated by pricing considerations, given the amount of real-time traffic that needs to be carried as is the case for amorphous sensing applications, for example.

Even in settings – e.g., large metropolitan areas – where networking (cellular or wireless 802.11) infrastructures may exist, issues of trust, privacy, and anonymity may make the use of infrastructure-less networks quite desirable. Example DTN applications along these lines include: anonymous tipping for crime prevention and law-enforcement purposes, and communication between covert agents in hostile countries for homeland security purposes. Given the proliferation of personal communication devices and computing platforms, it is conceivable that individuals will allow their mobile personal or vehicular communication
devices to be used as part of an infrastructure-less overlay network to facilitate message delivery in such applications.¹

While one may assume that the infrastructure-less overlays we envision could be quite dense – e.g., if all vehicles in a metropolitan area join it – thus enabling the formation of an ad-hoc mesh network, it is more likely that for the set of motivating applications we presented, such overlays will be sparse – e.g., if vehicles belonging to a specific organization (a taxi company or cars owned by members of a university, etc.) join it. By sparse, we mean that for most of the time, nodes in such an overlay are not within communication range of one another and hence the existence of an end-to-end path between the source and destination of a message in such a network is highly unlikely, rendering useless conventional ad hoc routing techniques. Instead, in such sparse overlays, node mobility is exploited to circumvent the lack of an end-to-end path. A store-carry-and-forward model is adopted to deliver messages to their destinations minimizing the total delay of each message.

Towards Realistic Mobility Processes for MDAs in DTNs: Commonly, research work message deliver in the context of DTN assumes that node encounters are predestined, in the sense that they are the result of unknown, exogenous processes that control the mobility of these nodes. For example, a taxi hired for a trip between points A and B would use (say) the shortest path between points A and B, making that taxi’s encounters with other vehicles that belong to the DTN overlay “predestined”. While the above assumption (that node encounters are predestined) may make sense in some settings, it is too restrictive in general. We provide a supportive argument below.

As we argued in Section 4.1, when moving between two points A and B, a node (taxi) may have multiple choices of paths of almost equal expected quality (e.g., in terms of travelled distance or time), each of which resulting in potentially different encounters. The availability of multiple paths between two waypoints presents an opportunity for mobility coordination across nodes to improve DTN performance. Indeed, results we show later in the chapter using the real taxi trips shown in Figure 4.9 indicate that mobility coordination

¹The incentive to contribute one’s storage and communication resources to establish such an overlay is not too different from the incentives for setting up Thor overlays (onion-routing) for anonymous file sharing.
has the potential of a multi-fold improvement of the performance of DTN routing both in terms of message delivery and throughput (see Figure 5.10).

The above observation could be made about mobility of nodes in general (individuals, vehicles, etc.): namely, that in most settings, the exogenous processes that drive the mobility of nodes do not predetermine paths, but rather they establish constraints on the spatio-temporal coordinates of the start and end points of the node’s journey. Thus, we argue that the specific path that the node may take in space-time, and hence the set of nodes it may encounter could be influenced/controlled in such a way as to improve performance of MDAs in DTNs.

**Leveraging Spatio-Temporal Slack for Mobility Coordination:** In Figure 4.1, we presented an example of a node’s schedule and visual depiction of this schedule. Since some of the legitimate paths shown the figure could lead to useful encounters with neighbors while other paths could miss such encounters, it becomes evident that judicious mobility coordination by leveraging slack in node schedules could potentially improve the performance of MDAs (e.g., improve message delivery rate, or decrease latency of message delivery). Coordinating this slack in order to improve performance of a DTN is the main proposal of this chapter.

### 5.2 Related Work

Our work is relevant to a number of research communities, including: delay-tolerant networks, vehicular networks, and robot mobility planning.

Research in DTNs [dtu] assumes lack of end-to-end connectivity between communicating nodes, and leverages node mobility to transport messages. Research efforts in DTNs concentrate on finding an optimized algorithm to forward messages between nodes upon an encounter. The result is a routing protocol that determines what messages to forward to which neighbor when an encounter takes place. The simplest solution is epidemic routing [epi, VB00], whereby all messages are replicated upon an encounter. Gossip routing
and probabilistic routing [HHL06, LDS03] are more efficient by being judicious in terms of utilizing available bandwidth and storage. Various other solutions (e.g., [SJ03, LW07]) have been proposed with different assumptions about the requisite knowledge of the node encounter pattern and messages workload.

The detour-based approaches we advocate in this chapter differ from these efforts in that a node’s motion (path) is viewed as a controllable variable as opposed to a fixed, uncontrollable input. While constrained by node schedules (among possibly other constraints), the mobility of a node could be manipulated to improve the performance of the entire system.

There have been some recent proposals for controlling/planning the mobility of a group of nodes in an ad-hoc network. The first group of such proposals [LFC05, KSJ+04, LR00] focused on actively mobilizing some nodes to bridge unconnected islands of nodes, hence improving instantaneous end-to-end network connectivity. The second group of proposals [SRJB03, ZAZ04, TAZ06, WSC05] suggested the use of special nodes as ferries/mules. These special nodes which are unconstrained in terms of their communication, computation or power resources act as “postmen”; collecting messages from sources and delivering them to destinations, improving temporal connectivity.

In our work we show that judicious mobility coordination of nodes spares the need for external helper nodes (e.g., ferries) while meeting all functional requirements of the nodes (e.g., spending a given percentage of the time monitoring the environment in a sensor network, or satisfying a given node schedule of locations and deadlines in an ad-hoc network). Notice that the use of “helper nodes” (ferries or mules) implies the use of an infrastructure of sorts. As we alluded earlier, a major motivation for the use of DTN overlays is to avoid the use (and hence the need to trust) any infrastructure, making these approaches less attractive for such applications.

Information dissemination in vehicular networks [M.06, WFGH04, CB07] is another example of DTN applications. The main difference here is that mobility takes place on mostly one-and-half dimension (i.e. mobility on a network of roads) at higher speeds. As an example of “thick” platforms, vehicular networks are less concerned with energy, storage,
and communication constraints as it is conceivable that vehicles can easily host powerful computing platforms (compared to e.g., hand-held devices). Also, the DNE technique we devise in this chapter differs from these efforts in that, it attempts to “guide” node mobility (as opposed to react to it) in order to increase the number of node encounters leading to an improved message delay.

5.3 Constrained Mobility Coordination for Message Delivery (CMC-MD)

In this section, we define the Constrained Mobility Coordination Problem for Message Delivery (CMC-MD), show that this problem is NP-hard, and formulate it as a constrained optimization problem. We, then, show how this formulation can be specialized to model two existing message delivery techniques in DTNs, specifically, message ferries, and data mules.

Definitions and Notation: We consider a DTN overlay consisting of $n$ mobile nodes. We assume that any two nodes within distance less than or equal to a fixed communication range $r$ can communicate. We also assume that the maximum speed of motion for a node $i$ is $\eta_i$, and without loss of generality assume that $\eta_i = \eta_{\text{max}}$. We define a message (or communication) workload $G$ in a DTN to be a vector of $m$ of messages in the system. Any message $g \in G$ is a tuple $g = (t, o, d)$, where $t$ is the time at which message $g$ originates (i.e., arrives), $o$ and $d$ are the identifiers of the source and destination for message $g$, respectively. Each node $i$ in the DTN has a schedule $s_i$ that consists of a list of $L(s_i)$ tuples of the form $u_{ij} = (\tau_{ij}, l_{ij})$, where $1 \leq j \leq L(s_i)$. To satisfy a schedule entry $u_{ij}$, node $i$ has to be at location $l_{ij}$ at time $\tau_{ij}$. For $i$ to satisfy its schedule, it has to satisfy $u_{ij}$ for all $1 \leq j \leq L(s_i)$.

The CMC-MD problem: Given a set of $n$ nodes, each with its own schedule, and given a message workload $G$, the CMC-MD problem is to find a set of node encounters that minimize message delivery delays while satisfying all node schedules. Solving the CMC-MD problem amounts to synthesizing the mobility profile for each node. The mobility profile
for node \( i \) gives the location of node \( i \) at time \( t \) for \( 1 \leq t \leq T \), where \( T \) is the evaluation epoch. Any feasible solution to the CMC-MD problem must satisfy the maximum speed requirement, \( i.e., \) no node is allowed to move with a speed higher than \( \eta_{\text{max}} \). Message exchange/delivery is only performed through node encounters induced by node mobility profiles. Node encounters must satisfy the communication range requirement, \( i.e., \) nodes can only communicate if the distance between them is less or equal to \( r \). We show that CMC-MD is NP-hard by reduction to the Minimum Latency Tour (MLT) problem [BCC+94], which we define next.

The Minimum Latency Tour (MLT) Problem: Given a set of field locations in a metric space \( P = \{p_1, p_2, \ldots, p_n\} \) where a symmetric distance function \( d_{i,j} \) is defined between each pair of locations \( p_i \) and \( p_j \), the MLT problem amounts to finding a tour on the set \( P \) minimizing \( \sum_{i=1}^{n} \ell(i) \), where \( \ell(i) \) is the latency to visit location \( p_i \) for a mobile element starting at some given location \( p_{\text{init}} \). The MLT problem is known to be NP-hard for general metric spaces [BCC+94].

The CMC-MD problem is NP-Hard: To show that the MLT problem is a special case of the CMC-MD problem, consider a DTN with \( n + 1 \) nodes. The initial locations of the first \( n \) nodes is set to \( P = \{p_1, p_2, \ldots, p_n\} \), and the initial location of the \((n+1)^{th}\) node is set to \( p_{n+1} \). The schedule of the first \( n \) nodes is set to their respective locations for the entire time \( i.e., \) schedule \( s_i \) of node \( i \) is given by \( s_i = \{(t, p_i)\}, 1 \leq t \leq T \). The communication workload of the first \( n \) nodes is empty. This in effect “pins down” the first \( n \) nodes to their initial locations throughout the epoch \( T \). The schedule \( s_{n+1} \) of node \( n + 1 \) consists of two entries \( s_{n+1} = \{(1, p_{n+1}), (T, p_{\text{fin}})\} \). For some random field location \( p_{\text{fin}} \) and some time \( T \) such that \( T \gg 1 \). This schedule gives node \( n + 1 \) the freedom to roam around the field long enough to have visited all \( n \) fixed locations \( \{p_1, p_2, \ldots, p_n\} \), and finally goes to some random location \( p_{\text{fin}} \). The communication workload of node \( n + 1 \) is set so as to deliver \( n \) messages \( \text{(one to each of the static nodes)} \), such that the origination time of all messages = 1. This reduction to the MLT problem proves that CMC-MD problem is NP-hard.

CMC-MD as an Optimization Problem: It should be noted that the CMC-MD prob-
Problem could also be cast as an optimization problem (Appendix A). However, the resulting formulation is an integer non-linear problem, whose solution requires searching the entire space, which is only feasible for the smallest of problem sizes. While such formulation does not yield a practical solution for realistically-sized DTNs, it provides some insights into the main optimization variables and the constraints that shape the solution of the problem.

**CMC-MD Modeling of Message Ferries:** Message Ferries (referred to as NIMF in [ZAZ04]) are external helper nodes that are not limited in power, computation nor communication capabilities. A ferry has a *well-defined* route in the field. When a node has enough slack, it approaches the route of the ferry, and upon encountering it, the node unloads outgoing messages unto the ferry, and gets messages destined to itself from the ferry. Our CMC-MD framework is able to model a message ferry with a node whose schedule is to go in a some defined route in the field, which is known to all other nodes. No messages are originated nor destined to this node (i.e., the ferry). To approach the ferry whenever it is possible, all other nodes spend all the slack in their schedules to get as close as possible to the ferry route.

**CMC-MD Modeling of Data Mules:** Data mules [SRJB03] (also referred to as FIMF in [ZAZ04]) are similar to message ferries, except that whenever a node needs to send messages to any other node, it “calls” the data mule. The data mule, having all the node requests, schedules its own mobility to serve as many requests as possible. Whenever any of the nodes change its location, it sends a *location-update* message to the mule so that it can update its mobility accordingly. After encountering a node, the mule selects the closest node with a standing request as its next target, an so on. In CMC-MD, a data mule could be modeled by a node that can move with higher speed than normal nodes. Nodes send message delivery requests to the data mule. The mule updates its mobility strategy to accommodate these requests. Upon encountering the data mule, a node unloads all outgoing messages unto the mule, and gets messages for itself from the mule.
5.4 Detour for optimized Message Delivery (DMD)

So far, our formulation of the CMC-MD problem, though complete, is impractical to solve. Therefore, in this section we propose to solve a serialized version of the problem. By serialized we mean that we optimize the delay for each message in the message workload $G$ in order of message origination time. More specifically, we consider one message at a time and identify for this message the node encounters that help minimize the delay delivering it, subject to constraints of the current schedule of all nodes. Each such encounter is then committed by adding the spatio-temporal coordinates of the encounter in the schedule of the nodes involved in that encounter, forcing these nodes to take the necessary “detours” to synthesize these encounters. This process is then repeated for each subsequent message in order of origination time. Notice that decisions made to optimize delay for message $g$ are considered as input when optimizing delay for message $g + 1$ – hence the “serial” nature of this optimization as opposed to the optimization approach we presented in Section 5.3, which optimizes the detours that each node takes for the entire message workload.

The output of the DMD approach is an augmented schedule for all nodes. An augmented schedule is a copy of the original schedule plus more tuples of the form $(\tau_{ij}, l_{ij})$, making an augmented schedule more restrictive (i.e., featuring less slack) compared to the original one. Notice that by definition, an augmented schedule is always feasible, since detours are only added to a schedule if they are feasible for the node to satisfy (subject to maximum speed constraints, etc.)

The Potential Encounter Graph (PEG): In order to obtain the feasible detours as described above, we represent potential encounters between nodes as a directed graph that has two groups of vertices $V_1$, and $V_2$, and two groups of edges $E_1$, and $E_2$. $V_1$ represents actual nodes in the system, while, $V_2$ represent potential encounters between any two nodes. There are no edges between vertices $v \in V_1$. Two vertices $v_a, v_b \in V_2$ are connected if there

---

2The final mobility of each node could then be determined by any basic technique (e.g., move randomly between entries in the augmented schedule, move at maximum speed and wait at destination, etc.) as long as the augmented schedule is satisfied. Notice that the resulting mobility will satisfy the original schedule since the augmented schedule is more restricted than the original one.
exists a node $n_c$ that can have both encounters represented by $v_a$ and $v_b$ simultaneously (i.e., taking part in both encounters is physically possible, based on the physical distance between the two encounter locations, their respective times, and the maximum speed). We call these edges *vertical* edges, $E_2$. If two vertices in $V_2$ (i.e., encounters) are connected with a vertical edge, then the cost of this edge is the difference between the earliest times each of the two encounters could take place. There are also edges between vertices in $V_1$ and vertices in $V_2$ such that each node (represented by a vertex in $V_1$) is connected to all vertices representing encounters that this node takes part in. We call these edges *horizontal* edges, $E_1$. Horizontal edges have an associated cost of 0, initially.

To handle a message $g$ arriving at node $n_{source}$ and targeting node $n_{dest}$ at time $t_x$ we do the following: (1) Temporarily eliminate all horizontal edges between all vertices in $V_1$ representing nodes other than $n_{source}$ and $n_{dest}$; (2) Assign direction from $V_1$ to $V_2$ to all horizontal edges coming out of node $n_{source}$; (3) Assign direction from $V_2$ to $V_1$ to all horizontal edges going into node $n_{dest}$; (4) Eliminate all edges incident to either $n_{source}$ or $n_{dest}$ connecting these two nodes to encounters taking place earlier than $t_x$; and (5) Each horizontal edge coming out of $n_{source} \in V_1$ to a vertex (encounter) $e \in V_2$ is assigned cost that is the difference between the time of message arrival, $t_x$, and the time of having encounter $e$. Finding the shortest path between $n_{source}$ and $n_{dest}$ in the resulting graph amounts to finding the list of encounters, which if *committed*, would deliver the message.

Figure 5.1: A three-node schedule (left) with potential encounters 1 through 6 (center) and PEG graph (right).
from the source to the destination incurring the least possible latency.

To illustrate the above process, consider three nodes $n_1$, $n_2$, and $n_3$ in a one-dimensional field of size 60. The nodes' schedule is given in Table 5.1. Figure 5.1 (left) gives a visual representation of this schedule. By inspecting Figure 5.1 (left), it is possible to locate the potential encounter points between the different journeys of the three nodes, we number these encounters 1 through 6 (Figure 5.1-center). Encounters 1 and 3 take place between $n_1$ and $n_2$, encounter 2 takes place between $n_1$ and $n_3$, while encounters 4, 5, and 6 takes place between $n_2$ and $n_3$. From this graph we can construct the potential encounter graph (PEG), shown in Figure 5.1-right. Notice that for the sake of clarity, Figure 5.1-right shows only horizontal vertices, $E_1$, but does not show vertical edges, $E_2$. Table 5.2 gives the edges in $E_2$. In Table 5.2, the label of the row gives the source vertex of the edge while the label of the column is the destination vertex of this edge. A blank entry in Table 5.2 means that there is no node that can carry a message between two encounters. For example, encounter 1 takes place between $n_1$ and $n_2$. Encounter 4 takes place between $n_2$ and $n_3$. However, $n_2$ cannot simultaneously satisfy both encounters (given its original schedule), hence there is no vertical edge between their corresponding vertices.

A detailed description of the PEG construction process, along with an illustrative example is given in Appendix B.

**Detour Synthesis using PEG:** Once constructed, the PEG graph is used to find the set of encounters that minimize the delay for each message, in order. As we alluded before, DMD considers one message at a time.

For a message $g$ originating from node $n_1$ to node $n_2$ at time $t_x$, we proceed as follows: (1) We temporarily eliminate all horizontal edges between all vertices representing nodes other than $n_1$ and $n_2$. This is done since we need to find the set of encounters to deliver the message (*i.e.*, vertices in $V_2$), hence there is no need to going back to $V_1$. (2) We assign direction from $V_1$ to $V_2$ to all horizontal edges coming out of node $n_1$. Since the only time we cross from $V_1$ to $V_2$ is when the message originates at $n_1$. (3) We assign direction from $V_2$ to $V_1$ to all horizontal edges going into node $n_2$. Since the only time we cross back to $V_1$
Table 5.1: Example of a node schedule

<table>
<thead>
<tr>
<th></th>
<th>n₁</th>
<th>n₂</th>
<th>n₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>location</td>
<td>time</td>
<td>location</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td>20</td>
<td>13</td>
<td>38</td>
<td>38</td>
</tr>
<tr>
<td>50</td>
<td>33</td>
<td>70</td>
<td>23</td>
</tr>
</tbody>
</table>

from V₂ is to deliver the message the message to n₂. (4) We eliminate all edges incident to either n₁ or n₂ connecting these two nodes to encounters taking place earlier than tₓ. This is done to prevent past encounters from being used to deliver future messages. And, (5) to each of the remaining horizontal edges going out of the message source, we assign a cost that equals the difference between the time at which the message originates and the time the respective encounter takes place. This represents the amount of time a message waits in the source node until the first encounter. Similar wait times in intermediate destinations are represented by weights of the vertical edges E₂.

Finding the shortest path between the n₁ and n₂ in the resulting graph amounts to finding the list of encounters, which when committed would result in the delivery of the message from the source to the destination, while incurring the least possible latency.

Figure 5.2 shows this procedure for a message g that originated at node n₂ to node n₁ at time 18. Figure 5.2(A) shows applying the five steps on the PEG, while Figure 5.2(B) shows finding the shortest path on the resulting graph. The resulting path yields a minimum delay of 18 secs. In this path, the message waits at n₂ for 12 secs, then is transported to n₃ (encounter 5), where it waits for another 6 secs, and is finally delivered to n₁ (encounter 2). Notice that for message g to be delivered in this delay, encounters 2 and 5 must be confirmed. Encounter 2 is confirmed by adding it to the schedules of n₁ and n₃, while
Computational Complexity: We now analyze the time complexity for the DMD algorithm. For each given message, DMD performs a number of steps in order to decide on the encounters that will minimize the message delivery delay. These steps are enumerated below.

Step (1): Construct the basic intersection graph (an example of this graph is shown in Figure 5.1 Left). Given schedules of \( n \), each of which with a maximum of \( L \) entries over an epoch \( T \), the time complexity for this step is \( O(Tn) \) – basically, given the schedule of each node, we determine the spatial range that this node could visit at each time unit.

Step (2): In order to construct the PEG graph, we need to determine the encounter points (vertices in \( V_2 \)). Encounter points are the earliest points in the space-time graph at which any two journeys of two nodes intersect each other. In this case, we have \( n \) nodes, each with \( L \) journeys. On average, we can assume that journeys are of similar lengths across all nodes. Hence any journey rectangle will intersect at most a fixed number of journeys \( K \), where \( K \leq L \). This leads to a total number of encounter points of \( (L n) \times K (n-1) \sim L K n^2 = O(Ln^2) \). In the worst case, when a given journey rectangle intersects any other journey rectangle, we have a maximum number of \( (L n) \times L (n - 1) = O(L^2 n^2) \) encounter points. Notice that, we could start constructing the PEG graph in this step, and assign the

---

3We could calculate the earliest and the latest times at which two nodes encounter each other. This might lead to a better approximation of the optimal solution, but also complicates the algorithm considerably. We have opted not to consider this option in the current version of the algorithm.
horizontal edges (i.e., link each node to encounter points in which it takes a part).

Step (3): Find vertical edges in $V_2$, along with their weights. Let us denote by $\epsilon$ the number of encounter points for a single node. Then on average $\epsilon = L K n$ (and in the worst case, $\epsilon = L^2 n$). The number of vertical edges between encounters of one node can be at most: $(\epsilon - 1) + (\epsilon - 2) + \cdots + 1 = \sum_{i=1}^{\epsilon-1} i = (\epsilon - 1)/2$. Then, on average, the total number of vertical edges is $n \times L K n(L K n - 1)/2 = O(L^2 n^3)$, and the worst case is: $n \times L^2 n(L^2 n - 1)/2 = O(L^4 n^3)$. Notice that, applying the five-step transformation on the PEG graph could be done in steps 2 and 3 above.

Step (4): Given a specific message, find the shortest path between the message source and destination on the resulting PEG graph. To do this we run Dijkstra’s algorithm which takes time $O(|E| \log|V|)$, where $E$ is the set of edges, and $V$ is the set of vertices in the graph. The number of vertices in the modified PEG graph is at most the number of potential encounters ($|V_2|$) plus 2 (the source and the destination), and the number of edges equal the number of edges in $V_2$. Hence, this step takes $O(L^2 n^3 \log(L n^2))$ on average, and $O(L^4 n^3 \log(L^2 n^2))$ in the worst case.

Step (5): Augment schedules in order to deliver the message. In the worst case, a message is delivered using all $n$ nodes, which means adding, at most, two meeting points to each node (only one for the source and destination). Hence, at most we have to add $2n$ encounter points to schedules. Assuming this addition takes constant time, This step costs $O(n)$ time.

It should be clear that handling a single message is dominated by the cost to perform the shortest path algorithm. Hence, the average time to run the DMD algorithm for $M$ messages is $O(M L^2 n^3 \log(L n^2))$, while in the worst case, it runs in $O(M L^4 n^3 \log(L^2 n^2))$.

**DMD versus Optimal CMC-MD:** DMD is an approximation for the NP-hard optimal solution of the CMC-MD problem. To get a feel for how close DMD’s performance would be compared to an optimal CMC-MD solution (OPT), we used simulations to compare DMD’s performance to that of OPT. OPT is implemented using an exhaustive search that allows us to find the schedule of all nodes that minimizes the sum of the delay of all messages, such
Figure 5.3: Comparing DMD to OPT, the optimal mobility planning. relative delay (left), relative throughput (right)

that an undelivered message incurs a large delay value that is less than infinity. Details of the simulation setup are given in Subsection 5.6.1. Due to the exponential computational complexity of OPT, we were able to simulate only toy-sized problems – namely we simulated two nodes in a field of 5x5 over a period of 10 seconds. In the following graphs, each point is the average of 20 simulation runs. We compare the performance of DMD, and OPT as a function of the maximum slack in the nodes schedule (referred to as $\kappa$ in Equation 4.11). Figure 5.3 (left) reports the average delay of DMD compared to that of OPT. In this figure, messages that were not delivered add delay of $(T + 1)$ to the sum of message delays, where $T = 10$ is the evaluation epoch Figure 5.3 (left) shows that the delay of DMD is between 1 to 2.2 times that of OPT. This figure suggests that the delay of DMD is within a constant factor (i.e., about 2.2) from that of OPT. Figure 5.3 (right) shows the relative throughput of DMD compared to that of OPT. At no slack (i.e., $\kappa = 0$), OPT and DMD have the same throughput, since nodes basically take the shortest-path. Increasing the slack (i.e., $\kappa = 1$), the difference between OPT and DMD becomes clear, since, there is very little slack to manage, and missing the “very few chances” to deliver messages is reflected in the relative throughput. As the value of the slack is increased, the throughput of DMD improves. This trend is clear and does not show signs of reaching a low level steady state value until the maximum slack, $\kappa$, is equal to 5. We could not explore values of $\kappa$ beyond 5 due to the intensive computational requirements of OPT.
5.5 Detour for maximizing Node Encounters (DNE)

The DMD approach is centralized in nature, thus imposing limitations on its applicability in practical settings. In this section we propose a heuristic that introduces detours so as to maximize the number of Node Encounters. Using this DNE heuristic, instead of trying to explicitly minimize the delay of every message in the system (as in DMD), we rely on increasing the number of encounters between nodes in the system. The motivation is that by using the slack in the schedules to create new encounters between nodes, we are likely to increase the probability of having useful encounters, which could lead to maximizing the success ratio of message deliveries and minimizing message latencies.

In DNE, we assume that there is an ordered set $\Omega$ of suggested encounter locations along with a frequency parameter $\mu$ and start time $t_0$. The set $\Omega$ as well as $\mu$ and $t_0$ are known to all nodes in the system. Based on its schedule, a node $i$ identifies the locations in $\Omega$ that could be visited at time $t_0$ without violating its own schedule. Let us denote these feasible locations by $\omega \subseteq \Omega$. In case $\omega \neq \phi$, i.e., node $i$ could make one or more of the proposed locations at time $t_0$, $i$ selects its target location based on the order of the original set $\Omega$. If $\ell_0 \in \omega$ is the highest-ranked location in $\Omega$, then node $i$ adds the tuple $(t_0, \ell_0)$ to its schedule. This has the effect of “committing” that node $i$ will be at location $\ell_0$ at time $t_0$. Node $i$ then repeats the same process for all times $t_k = t_0 + k \times \mu$, for $k = 1, 2, \ldots$. The outcome of this procedure is an augmented schedule, with the locations of the added entries being all from the same set $\Omega$. Hence there is higher chance of having the same meeting point added to the schedule of more than one node, which in effect creates new encounters.

As with DMD, the actual motion of the node is determined using any basic strategy as long as the augmented schedule is satisfied.

**Computational Complexity:** It is clear that any node, needs to check, at most, which of the $|\Omega|$ locations fits its schedule. This calculation is repeated $T/\mu$ times, where $T$ is evaluation epoch. Hence, the total time needed for DNE is $O(T/\mu |\Omega|)$. 
5.6 Performance Evaluation

In order to evaluate the efficacy of our detour-based approaches, we used the same mobility simulator that we used in evaluating FMAs (Section 4.5).

We compare the mobility resulting from the use of our approaches to four basic alternatives. The first two are wait-at-source (WAS), and wait-at-destination (WAD) approaches. In WAD, given a schedule, nodes take the shortest path to the destination of the current journey and wait there, i.e., spend all the slack time waiting at the destination. In WAS, all the slack time is spent at the source of the journey, and then nodes take the shortest path to the destination of the journey. The third approach is random mobility (RND), in which nodes move randomly between every two consecutive waypoints provided that the schedule is satisfied. Clearly, WAS, WAD, and RND represent a lower bound on performance, since they do not actively attempt to manage node mobility for improved message delivery performance.

It should be clear that we are not trying to design a routing algorithm, nor a message forwarding technique. Rather our work focuses on the synthesis/coordination of node mobility subject to schedules and message workloads. Hence, after obtaining the node location across time (i.e., the result of applying the various mobility synthesis/coordination approaches), we can easily infer the contact model induced by the synthesized node mobility. The resulting node encounters, along with the message workload can be fed to any message routing algorithm to decide which messages to forward to which neighbor upon an encounter. The details of the specific routing algorithm are orthogonal to our work. In this evaluation, we choose to use an optimum algorithm that calculates the optimum forwarding path for every message, given the current node contacts. This means that, results we report here are the best case performance for all mobility synthesis approaches. Notice that the exact performance of the optimization program could be attained in a distributed fashion using epidemic routing [epi, VB00]. A more efficient algorithm is to use gossiping [HHL06, LDS03] to avoid much of the problems associated with epidemic routing while
reaping some of its benefits. In short, we stress that the message forwarding technique is orthogonal to our work, and any technique could be used here.

The optimized algorithm we used to find the optimum path for every message to reach its destination is based on the formulation given by Jain et. al [SJ03].

5.6.1 Evaluation Using Synthetic Workloads

Schedule Generation: Schedules are generated as we described in Subsection 4.5.1.

Message Generation: Message sources and destinations are randomly generated such that the source and destination of any message are not the same. The message arrival process follows a Poisson process with mean 2.

Performance Metrics The performance metrics we use are the delivery ratio and average delay. Delivery ratio is the ratio of successfully delivered messages to the total number of messages generated. Average delay is measured for delivered messages only.

Comparison with Naive Techniques: We simulated a field of 30x30 city blocks where nodes can communicate only when they are at the same intersection. The simulation runs for 100 seconds. In the following graphs, each point is the average of 20 simulation runs, with the 95% confidence interval shown as well.

In the first set of experiments, we compare DMD and DNE to the basic WAD, WAS, and RND approaches. In these experiments, the maximum slack allowed $\kappa$ was set to 15.

Figure 5.4 (left) shows the delivery ratio of the five approaches. As expected the delivery ratio of all approaches improve as we increase the number of nodes, which in turn increases the number of encounters, thus enabling more messages to get delivered. This effect is more evident for WAS, WAD, and RND. DMD is able to achieve from 80% to two times higher delivery ratios than the basic algorithms. This underscores the importance of our PEG-based approach, and the value of the encounters it chooses. DNE yields from 30% to 80% higher delivery ratios compared to the basic algorithms, confirming our intuition that a simple distributed mobility coordination algorithm that focuses only on increasing the number of encounters (while being oblivious to the message workload) is bound to improve
the delivery ratio.

Figure 5.4(right) shows the average delay of delivered messages. The difference between DMD and the other mobility synthesis approaches is very clear; it has between 13% and 170% less delay compared to the basic RND, WAS, and WAD techniques. On the other hand, DNE achieves from 13% to 40% lower delay than WAS, WAD, and RND. An interesting point is that increasing the number of nodes increases the average message delay for all approaches, except for DMD, i.e., for all distributed workload-incognizant approaches. The reason is that, DMD creates encounters between nodes that are certain to help minimize message delay. While, using the other approaches, increasing the number of nodes creates more encounters that help deliver more messages but not necessarily on the most optimum path, yielding higher average message delay.

To summarize, DNE improves the delivery ratio and the average message delay compared to the basic approaches. DNE’s efficiency is more evident in networks with low node density (typical in DTN networks). DMD achieves the best message delivery ratio and average delay.

**Effect of Partially Following Detours:** The goal of this experiment is to measure the effectiveness of DNE in two cases – namely (1) when all nodes follow DNE hints with some probability (Figure 5.5), and (2) when only a given percentage of the nodes follow detour hints provided by DNE (Figure 5.6). Both of these scenarios are motivated by the fact that autonomous users may opt not to follow routes suggested by DNE. Figure 5.5 shows the performance as a function of the probabilities of following hints for different number of nodes.
Clearly, the performance improves as nodes follow hints more consistently. As we hinted before, experiments with more nodes have higher success ratio and longer average message delays. It is interesting to see that the gain is almost linear (as a function of the percentage of nodes following hints). The rational is that as nodes try to make the meeting points more often, the higher the chance of actually having a useful encounter that could be used to deliver messages. Figure 5.6 shows the effect of having different percentages of nodes completely follow the detours proposed by DNE, while the rest of the nodes completely disregard these proposals. Similar to Figure 5.5, performance improves linearly as the percentage of complying nodes increases. This is expected from a distributed algorithms, where nodes have no way of knowing other nodes decisions. The performance of the entire system improves as more nodes comply with the distributed protocol. Interestingly, results of this experiment are similar to the corresponding experiment in Chapter 4 (Table 4.2, and Figure 4.8). The conclusion from both experiments is that, if the provided service depends on cooperation among multiple users, then the more cooperating users, the better the performance of the system.

**Comparison With Ferries and Data Mules:** As we discussed in Section 5.2, we propose better planning of slack time instead of relying on external helper elements: *i.e.*, message ferries and data mules to deliver messages. Of course, if there is very little slack, then there is no way to improve the system performance but to rely on the help of external nodes. However, as the results we present later in this section suggest, the availability of even a
modest level of slack might prove very useful, resulting in performance that is even better than relying on external helper nodes, such as ferries or mules.

We can categorize DMD, DNE, ferries and mules along three dimensions. The first dimension is the distributivity of the solution. While DNE and ferries are distributed approaches, DMD and data mules are both centralized approaches, in the sense that knowledge about the message workloads and node locations must be aggregated and processed centrally.\(^4\) The second dimension is whether the message workload is used in coordinating node mobility. DMD, data mules, and message ferries are workload-cognizant while DNE is workload-oblivious. In DMD, data mules and message ferries, node mobility is derived, at some point in time, by the knowledge that there is a message that needs to be communi-

\(^4\)Zaho et al. [ZAZ04] propose a distributed approach to implement data mules by allowing a node to use long-range communication to inform the mule about the requests of service and location updates. We do not argue the practicality of this proposal and just note that the solution is centralized in nature.
cated to some other node in the network. This is not the case in DNE, under which, nodes take mobility decisions motivated by the desire to increase their chances of encountering other nodes, irrespective of the message workload. The third dimension is the dependence on external helper nodes (i.e., some form of “infrastructure”). It is clear that both DMD and DNE do not depend on external helper nodes, while message ferries and data mules do.

In Section 5.3, we outlined how to use our CMC-MD framework to represent the functionalities of message ferries and data mules. In particular, we noted that the flexibility of our scheme enables modelling both ferries and mules by specific parameterization of the node schedules and the message workload: A message ferry is modelled as a node whose schedule is to go in a predetermined path in the field, whereas a data mule is modelled as a node with an empty schedule, enabling it to move anywhere in the field. Both ferries and mules are considered “helper” nodes, and as such they are not themselves origins or destinations of DTN messages.

In addition to this general modelling approach for ferries and mules within our CMC-MD framework, there are a number of specific implementation details that we have adopted in our experimental evaluation. We summarize these next.

Unlike message ferries, the mobility of the data mules has to be “planned” to minimize the time that nodes have to wait to be served by the mule. Shah et al. [SRJB03] proposed that mules use a random walk on the field, whereas Zhao et al. [ZAZ04] showed that planning the mobility of mules is an NP-hard problem, and proposed different heuristics to
solve it. One of the heuristics they proposed is the nearest neighbor heuristic, in which the mule moves to meet the closest node with a request. This is the heuristic we used in our experiments. Notice that under this model, the mobility of normal nodes is not affected by the communication workload, i.e., nodes do not approach the data mule to speed up the message exchange as in the case of ferries. Rather, it is the data mule that approaches them to collect messages. To enable this to happen, in our experiments, we assigned to data mules a speed that is double that of normal nodes, and we let normal nodes follow a Wait-At-Source (WAS) strategy in order to remain static for most of the time, allowing the mule a higher chance to reach them.

In our evaluation, we did not limit message exchanges to be only with a ferry (or a mule), rather encounters between normal nodes could also be used to deliver messages.

Since DMD and DNE work by leveraging the slack that exists in the schedules, in the set of experiments where we compare DMD and DNE to ferries and data mules, we vary the amount of slack available to nodes ($\kappa$ in Equation 4.11).

Figures 5.7 and 5.8 show the performance of the DMD, DNE, ferry, and data mules with 5 and 10 nodes, respectively. As we increase the slack in the schedules, the performance of DMD, DNE, and ferry improves. The reason is that both DMD, and DNE explicitly benefit from relaxed schedules since they depend on the available slack. As for ferries, more slack allows nodes more chances to encounter the ferry, since nodes try to move towards the current ferry location, if their slack permits. In the case of data mules, as we mentioned above, nodes do not approach the mule to speed up the exchange, hence, increasing their slack does not improve the performance.

It should be noticed that the delivery ratio of data mules is always better than that of ferries (which is consistent with what has been reported by Zhao et al. in [ZAZ04]). The reason is that, unlike ferries, data mules have more freedom in terms of their route in the field, as they can change their route based on the current workload. Moreover, data mules move with double the speed of normal nodes making them more effective than ferries in message delivery.
Figures 5.7 and 5.8 show that data mules are more effective than DNE, but only for the tightest of schedules. Increasing the slack improves the performance of DNE until it surpasses that of mules. DMD has the best performance for all parametrizations we experimented with.

**Comparison Using A Vehicular Networks Simulator:** We used Groovenet [GRO] to compare DNE and WAD. The goal is to use a more realistic setting in comparing our techniques to baseline models. Groovenet is a vehicular networks simulator. It allows creating different vehicles with different characteristics including the communication model, the link layer model, the physical layer model, and the mobility model. We used Groovenet to create a mobility schedule for nodes. Schedules were created by selecting random locations within a given city, finding the time to travel between every two consecutive locations and deciding the schedule time using Equation 4.11, with $\kappa = 600$ seconds. Then, we extended Groovenet with two mobility startegies: WAD, and DNE. We ran simulations of 100 vehicles in an a square area of 10 miles $\times$ 10 miles in a US city in the Northeast. The message arrival process is a Poisson Process with mean 5 seconds. We ran simulations for 40,000 seconds (about 11 hours). Figure 5.9 (left) shows the throughput of DNE and WAD. The throughput of WAD improves noticeably by increasing the number of nodes, while DNE is able to maintain a high throughput even for cases with few number of nodes. Figure 5.9 (right) shows the resulting delay. The delay of WAD is from 160% to 260% of that of DNE. Increasing the number of nodes helps the delay of WAD to reach a steady state, after which, delay is not improved much by adding more nodes.

**5.6.2 Trace-Driven Evaluation**

We also used the San Francisco cab traces that we used to evaluate performance of the distributed field coverage (Subsection 4.5.2). The goal is to show that, with little coordination between cabs, they could function as an effective DTN system.

To that end, we used the same methodology to generate a “map” of the SF area, and to generate schedules of nodes.
We compared two mobility synthesis approaches: WAD and DNE. Under WAD, when empty, a cab moves to the next location where it picks up its next customer, as early as possible, and spends its slack time there waiting for the customer. However, under DNE, we divided the field into 11 big sections and selected some location in each section as the potential meeting location with other cabs. When a cab is empty, it calculates the distance between its current location and all suggested meeting points, and if there is enough time, it moves to the closest location to spend its slack time. When the slack time is over, it moves to the next location where it picks up its next customer. In doing this we assumed a maximum speed of 30 mph, which is quite conservative.

**Message Workload Generation:** We generated message workloads similar to those used in our synthetic simulations. Specifically, we used a Poisson arrival model with a mean of 0.75 message/minute. Messages sources and destination were randomly selected from the
nodes.

**Results:** Figure 5.10 give the results of this experiment. DNE yields superior performance compared to WAD in terms of average message delay and delivery ratio. Figure 5.10 shows that, similar to the simulation-based evaluation, increasing the number of nodes increases both the delivery ratio and the average message latency.

It should be noted that assuming higher maximum speeds could improve the performance of DNE even more. Another important factor is the number of cabs we conducted the study on; increasing this number is bound to improve the delivery ratio.

### 5.7 Conclusion

In conclusion, we stress the novelty of the model that we presented in this chapter. All prior work in DTN concentrated on dealing with a set of given encounters. Assuming some message workload, these efforts tried to devise an optimal routing protocol to minimize delivery of messages, consuming the least amount of resources (\(e.g.,\) generate the least number of message duplicates). In this chapter, we argued that, the actual mobility of user could be fine-tuned in order to generate a set of useful encounters to optimize performance of MDAs in DTNs. To the best of our knowledge, this is the first time to propose such a model in DTNs. We showed that this problem is NP-hard and proposed two heuristics to solve it. The first heuristic (DMD) adopts a centralized workload-cognizant approach, whereas the latter heuristic (DNE) adopts a distributed workload-oblivious approach. We confirmed the premise of our hypothesis through extensive simulations and using real-life cab traces from the San Francisco Bay area. Moreover, we demonstrated that careful planning of users’ mobility have more profound effects on system performance compared to other approaches that used external elements to help the process (\(e.g.,\) data mules and message ferries).

We believe that the model that we developed in this chapter could be extended to support many other applications, for example the service of matching drivers with passengers in carpooling [MAS]. Another potential example is fleet management for transportation
systems. In such a system, a single authority have multiple trucks transporting goods between numerous warehouses. Instead of dedicating a single truck to deliver a given object from the source to the destination warehouses, a similar model to DMD could be applied in order to minimize delay of such deliveries, maximizing the profit. The only difference between such applications and our model is that message transfers between nodes is not done instantaneously, rather there is some time and cost for riders to change cars, or to transfer goods between trucks.

Appendix A: CMC-MD as an Optimization Problem

In this Appendix, we cast the CMC-MD problem as an optimization problem. The resulting formulation is an integer non-linear problem requiring a search of the entire space, which is only feasible for the smallest of problem sizes. Nevertheless, while such formulation does not yield a practical solution for realistically-sized DTNs, it provides the reader with insights into the main optimization variables and the constraints that shape the solution of the problem.

As before, we consider a DTN over an epoch $T$, with $n$ mobile nodes, each with a maximum speed $v$, a communication range $r$. Let $G$ represent the message workload consisting of $m$ messages. Furthermore, for any given message $g$, we use $\Theta(g)$, $O(g)$, and $D(j)$ to denote the origination time, source and destination of $g$. We also use $\text{Dist}(a, b)$ to denote the distance between two field locations $a$ and $b$.

Let the mobility matrix $X$ be a $T \times n$ real matrix, such that $X(t, i)$ denotes the derived location of node $i$ at time $t$. Let the message carrier matrix $Y$ be a $T + 1 \times n \times m$ binary matrix, such that $Y(t, i, g) = 1$ if and only if node $i$ buffers message $g$ at time $t$. We add an extra time unit after the end of the evaluation, such that the message reaches its destination at time $T + 1$, if it has not reached it already. Let the neighborhood matrix $E$ be a $T \times n \times n$ binary matrix such that $E(t, i, k) = 1$ if and only if nodes $i$ and $k$ are neighbors at time $t$. Let the message host matrix $H$ be a $T \times m$ integer matrix such that $H(t, g)$ is the ID of the
node that hosts message $g$ at time $t$. Finally, let the delivery time matrix $\Delta$ be an integer vector of length $m$, such that $\Delta(g)$ is the time that message $g$ reaches its destination. In the following equations we use: $i, k$ as node indices ranging from $1 \ldots n$, $g$ as a message index ranging from $1 \ldots m$, $j$ as an index in the schedule of a given node $i$, $j$ ranges between $1 \ldots L(s_i)$, and $t$ as a time index ranging between $1 \ldots T$ (unless specified otherwise).

The CMC-MD problem could be formulated as an optimization problem – namely to minimize the objective function.

$$\sum_{j=1}^{m} \Delta(j) - \Theta(j)$$  \hspace{1cm} (5.1)

subject to the following constraints:

$$X(\tau_{ij}, i) = l_{ij}$$  \hspace{1cm} (5.2)

$$\text{Dist}(X(t,i), X(t-1,i)) \leq v_i, 2 \leq t \leq T$$  \hspace{1cm} (5.3)

$$H(t, g) = \sum_{i=1}^{n} i \cdot Y(t, i, g)$$  \hspace{1cm} (5.4)

$$E(t, i, k) = \begin{cases} 1, & \text{if } \text{Dist}(X(t,i), X(t,k)) \leq r \\ 0, & \text{otherwise} \end{cases}$$  \hspace{1cm} (5.5)

$$Y(t, i, g) = \begin{cases} 0, & 1 \leq t < \Theta(g) \\ 1, & i = O(g), t = \Theta(g) \end{cases}$$  \hspace{1cm} (5.6)

$$\sum_{i=1}^{n} Y(t, i, g) = 1, \Theta(g) \leq t \leq \Delta(g)$$  \hspace{1cm} (5.7)

$$Y(t, i, g) \leq E(i, H(t-1, g), t), \Theta(j) < t \leq \Delta(j)$$  \hspace{1cm} (5.8)

$$\Delta(g) = \sum_{t=1}^{T+1} t \cdot Y(D(g), g, t)$$  \hspace{1cm} (5.9)

$$\sum_{t=1}^{T+1} Y(D(g), g, t) = 1$$  \hspace{1cm} (5.10)

The role of the above constraints can be explained as follows: Equation 5.2 constrains the mobility matrix in order to satisfy the schedule of each node. Equation 5.3 constrains the mobility of all nodes such that the travelled distance during any time unit does not
exceed the maximum speed of mobility. Equation 5.4 defines the host of each message at all time units (this is set to zero before the message arrives at its origin and after it is delivered). Equation 5.5 constrains encounters to be between nodes within communication range of each other. Equation 5.6 ensures that no node would host a message before this message originates, and Equation 5.7 ensures that when a message \( g \) originates, it is only hosted at the node \( O(g) \) that originated it. Equation 5.8 ensures that messages are not duplicated. Equation 5.9 ensures that messages are communicated between nodes only when nodes come into contact with one another. Equation 5.10 defines the time of delivery of all messages. Equation 5.11 means that the destination will get the message eventually, even if after the normal evaluation epoch (\( T \)). This equation simply forces delivery of the messages, if possible. Equations 5.10 and 5.11 mean that in order to minimize the objective function, a solution where messages are delivered earlier (i.e., before time \( T + 1 \)) should be found.

As we mentioned above, solving the above optimization problem entails solving an integer non-linear problem, which is not tractable for practical systems.\(^5\)

**Appendix B: PEG Construction**

We denote the Potential Encounter Graph, \( PEG = (V,E) \), where \( V = V_1 \cup V_2 \), and \( E = E_1 \cup E_2 \).

For every vertex \( v \in V_1 \), \( \gamma(v) \) gives the id of the node represented by this vertex. Any vertex \( v \in V_2 \) is a tuple of the form \( (\nu_1, \nu_2, \lambda, \omega) \) such that \( \nu_1 \) and \( \nu_2 \) are the IDs of the two nodes having the encounter, \( \lambda \) is the location of the encounter, and \( \omega \) is the earliest time at which the encounter can take place.

For every encounter \( v \in V_2 \), \( \gamma_1(v) \) and \( \gamma_2(v) \) give the IDs of the two nodes in the encounter, respectively, \( \theta(v) \) is the earliest time this encounter can take place, and \( \ell(v) \) is the location of the encounter. For any edge \( e \in E \), the functions \( \gamma_1(e), \gamma_2(e), \) and \( C(e) \) give the source, destination, and cost of the edge, respectively.

---

\(^5\)Non-linearity stems from the definition of a contact between two nodes (Equation 5.5).
**constructPEG()**

**Input:** Schedule $S = \bigcup_{i=1:n} s_i$

**Output:** $\text{PEG} = (V, E)$

1. Set $V_1 = \{1, 2, \ldots, n\}$
2. $P = \text{calculate-potential-encounter-points}(S)$
3. Initialize $V_2 = \phi$, $E_1 = \phi$, $c = 0$
4. For every $p \in P$
   1. Add $\{p\}$ to $V_2$ as follows: Set $c = c + 1$; Let $v_c = p$, $V_2 = V_2 \cup \{v_c\}$
   2. Add horizontal edges to $E_1$ linking $v_c$ to the 2 nodes in that encounter.
5. For encounters $v_1$ and $v_2$ with common node $i$, let $v_1$ be the earlier encounter.
   1.1 If (can-do-both-encounters($i, v_1, v_2$) == true), then
   1.2 Let $C(e) = \theta(v_2) - \theta(v_1)$.
6. set $V = V_1 \cup V_2$, and $E = E_1 \cup E_2$.

In the `constructPEG` Algorithm, steps 1-3 construct $V_1$, $V_2$ and add the horizontal edges to $E_1$, while steps 4 and 5 add vertical edges to $E_2$. Function `calculate-potential-encounter-points(S)` returns a list of the points where nodes might encounter each other, along with the time of such potential encounters.

**calculate-potential-encounter-points(S)**

**Input:** Schedule $S = \bigcup_{i=1:n} s_i$

**Output:** List of encounter points $P = \{p_1, p_2, \ldots\}$, $p_i = (\nu_1, \nu_2, \lambda, \omega)$

1. For every two nodes $i$, $j$ such that $i \neq j$.
   1.1 For every journey $a$ ($1 \leq a \leq L(s_i)$) in $i$’s schedule, mark each field location ($\text{loc}$) with $\text{er}_i(\text{loc})$ and $\text{lat}_i(\text{loc})$: the earliest and latest times $i$ could be at $\text{loc}$ during this journey.
   1.2 Do the same for every journey $b$ ($1 \leq b \leq L(s_j)$) in $j$’s schedule to get $\text{er}_j(\text{loc})$, and $\text{lat}_j(\text{loc})$.
2. Let $\zeta$ be all field locations at which nodes $i$ and $j$ could meet at journeys $a$ and $b$. Associate with each point $z \in \zeta$ the earliest time $\theta(z)$ at which an encounter can take place at this location.
   1.4 If $\zeta$ is not empty, let $p = \{v = (i, j, \lambda, \omega) : v \in \zeta; \omega \leq \theta(z), \forall z \in \zeta\}$. i.e., $\lambda$ is the meeting location at which the earliest encounter could take place (i.e., at time $\omega$) during journeys $a$ and $b$.
3. Let $P = P \cup \{p\}$. 
Function \emph{calculate-potential-encounter-points} depends on marking field locations with the earliest and latest times a node could be at this location during any given journey.

To illustrate this point, consider the first journey of node \( n_1 \) shown in Table 5.1. The starting point of this journey is location 8 at time 1, and its destination is location 13 at time 20. Figure 5.1 (left) reveals that node \( n_1 \) can only visit locations 1 through 20 in the field (the span of the first rectangle of \( n_1 \) on the \( x \)-axis), which means that locations 21 through 60 are unreachable during this journey, hence they will be marked as so. For locations 1 to 20, the earliest time \( n_1 \) could be at these locations is given by the lower two sides of the rectangle representing this journey. While the latest times to visit these locations corresponds to the upper two sides of the same rectangle. Figure 5.11 shows the marking of corresponding field locations during this journey. By intersecting different such tables of different journeys \( i.e., \) rectangles in Figure 5.1 left) of different nodes, we are able to determine the locations and times at which different nodes could encounter each other.

Finally, function \emph{can-do-both-encounters}(\( i, v_1, v_2 \)) returns a boolean answer indicating whether or not a given node \( i \), whose schedule is \( s_i \), can have two meetings at the two given meeting points.

can-do-both-encounters

\textbf{Input:} Node \( i \) with schedule \( s_i \), first and second encounters \( v_1 \) and \( v_2 \) such that \( \theta(v_1) \leq \theta(v_2) \)

\textbf{Output:} True if node can be at both meeting points, false otherwise.

1. Let \( wp_1 \) be the journey in \( s_i \) such that \( \theta(v_1) \geq \) starting time of \( wp_1 \) and \( \theta(v_1) \leq \) end time of \( wp_1 \). Similarly let \( wp_2 \) be the journey where \( \theta(v_2) \) falls.
2. If \( wp_1 \neq wp_2 \) return true, otherwise go to step 3.
3. Let dist = \( \text{Dist}(\ell(v_1), \ell(v_2)) \). and \( \delta = \theta(v_2) − \theta(v_1) \).
4. If \( \delta \times v_{\text{max}} \geq \text{dist} \) return true, else return false.

The reason Algorithm \emph{can-do-both-encounters} returns \texttt{true} in line 2 is that, any two meeting points in different journeys, could be trivially met by a node, given that each meeting point fits the original node schedule. In other words, two different meeting points in two different rectangles in Figure 5.1 could be met by the node irrespective of the actual
Figure 5.11: Earliest and latest times that node $n_1$ could visit various locations during its first journey (start ($\tau_{11} = 1$, $l_{11} = 8$), end ($\tau_{12} = 20$, $l_{12} = 13$)). UR stands for “unreachable”

distance and time interval between them – given that they actually are on or inside two different rectangles. Otherwise, if the two meeting points fall within one rectangle, we need to check that the distance between them is less than the maximum distance the node can cover during the time interval between the two meeting points.

Notice that we call function can-do-both-encounters with two meeting points, each of which is known to fit the original schedule (see Algorithm constructPEG, line 5.1).
Chapter 6

Conclusion

In this thesis, we proposed a new approach to service provisioning in mobile networks. Our approach leverages ubiquity of personal computing devices and embedded sensors to construct distributed systems. Users of these systems are autonomous mobile nodes, who are willing to contribute a limited fraction of their resources to support the distributed system. Services are provided through cooperation and coordinated management of these fraction of resources. Hence in our model, system users are the service providers. We argued that this model enables providing large-scale services, without the overhead of laying down infrastructure. Nevertheless, this model poses a number of new research challenges that have to be addressed in order to make it viable. The first challenge is the often resource-limited nature of the computing platforms hosting such services. This mandates using smart resource management techniques to maximize utility of these resource. This challenge is only made more arduous by the fact that these resources are distributed and are under the control of different parties. The second challenge is the fact that nodes participating in the service provisioning/access are autonomous. The need to preserve autonomy of these nodes, while managing some of their resources constitutes a hard challenge, even in the case where the host platforms are resource-rich.

We studied two main types of resources that users share in order to support distributed services: conventional resources, and mobility. By conventional resources (Chapters 2 and
3), we mean storage, battery power, and ability to control a peripheral sensor. The other type of resources we studied is mobility (Chapters 4 and 5). We argued that, in some cases, the coarse mobility behavior of the host could be controlled by an external schedule. This schedule is expected to allow some level of slack. We argued that this slack could be effectively coordinated to optimize performance of distributed systems.

We evaluated the premise of the distributed services provisioning model in two different application domains: Field Monitoring Applications (FMAs) (Chapters 2, 3 and 4), and Message Delivery Applications (MDAs) (Chapter 5).

In Chapter 2, we addressed supporting FMAs. We assumed that the distributed service is allowed to control a limited fraction of the host storage. Neither the sensor scheduling nor the host mobility is controlled by the service. The goal of this system is to answer as many user queries as possible consuming the least amount of resources. To achieve this goal, we designed APR [MBM08b], a novel data management technique that avoids multi-hop communication, rather depends on direct communication between nodes. To achieve its goals, APR depends on two main mechanisms: query-cognizant cache management, and sample diffusion. We showed through analysis and simulation that APR was superior to techniques in the literature in terms of query success ratio and consumed power.

In Chapter 3, we adapted similar setup with minor changes. We assumed that the distributed service had control over when to use the sensor to sample the current host location. We also assumed that there was some form of correlation between values of the target phenomenon at different field locations at different times. The goal of the system, in this case, is to estimate answers to queries minimizing the estimation error. We designed an information-theoretic framework [MABM08] to handle sensor management, storage management, and query handling. Based on this framework, we proposed two techniques with different assumptions about the accessible information. We showed that our techniques provided much more accurate query answers compared to random caching.

In Chapter 4, we addressed the problem of mobility management to support FMA. We assumed that there were no constraints on the storage nor sensor scheduling of the
host. However, mobility of the host is constrained by an external schedule that specifies, for each node, a list of points in the space-time plan that must be visited by this node. We, also, assumed that the distributed service had control over mobility of each node, provided that schedules of each node was satisfied. The goal of the system, in this case, is to coordinate mobility of nodes in order to simultaneously achieve some given monitoring distribution of the field, and satisfy schedules of all nodes. We, first, showed that this problem is NP-complete, then, we proposed TFM, a distributed mobility strategy to solve the problem. We showed that TFM achieved very close distribution to the required one. We also showed that, under TFM, nodes achieved, at least, 2-fold improvement in the query success ratio compared to that under the random mobility strategy. Furthermore, we confirmed the premise of TFM using cab traces from the San Francisco area. results of the latter evaluation similar to those of the earlier, underscoring potential of TFM.

In Chapter 5, we studied mobility management to support MDA in DTNs. We followed a similar setup to that of Chapter 4. In this setup, each node has a number of messages to be delivered to other nodes in the system. The goal is to plan mobility of the nodes in order to minimize the average delay of messages delivery. We showed that this problem was NP-hard, then, proposed two heuristic solutions [MBM08c]. The first is centralized workload-aware that plans mobility to explicitly minimize delay of delivering each message. The second heuristic is distributed workload-oblivious that depends on increasing node encounters, hoping that it would create useful encounters to minimize delay of delivering the messages. We compared the two heuristics to the random strategy model, message ferries, and data mules. We showed that these heuristics were superior to other mobility strategies. Similar to Chapter 4, we used the same cab traces to evaluate our heuristics. Results of the trace-driven evaluation showed that the distributed version is superior to random mobility.

We believe that the service provisioning model that we presented here is very suitable to modern trends of pervasive computing, and equipping mobile computing platforms with increased resources.
Throughout this thesis we stressed two main messages.

- The first message is that, in managing the limited resources under their control, the distributed systems that we develop make a crucial use of different pieces of information. This knowledge helps to immensely improve performance of the provided distributed services.

  In the context of FMAs, the system uses characteristics of the target phenomenon, as well as the workload. The workload in this case is represented by the specific pattern of users’ interest in the field to manage limited storage. The characteristics of the target phenomenon define the amount of information provided by samples collected at a certain location and time about the value of the phenomenon at the current time, at any field location. While the workload defines which areas of the field are highly targeted by user queries. Combining the two pieces of information, the system would give higher priority to samples which provide more information about high-demand areas compared to other samples which give less information about such areas. Likewise, in the context of MDAs, knowledge of the workload greatly improves performance of the system. The workload in this case is information about the communication pattern of users. Specifically, workload is the source, destination and origination time of each message. Acquiring, and leveraging such knowledge enables distributed systems to greatly improve performance of MDAs.

- The second message is about users autonomous mobility. First, we recognize this mobility as a valuable resource worthy of management and coordination. Furthermore, we show that careful coordination of such mobility has a huge untapped potential to improve distributed service provisioning in both FMA, and MDA. We show that such coordination is so powerful, that its effects surpass that of using external helper nodes without relying on mobility coordination.
6.1 Future Work

Ideas, models, and algorithms presented in this thesis can be extended in a number of ways. In this section, we identify four such venues.

6.1.1 Load Balancing

To make our models applicable in practical settings, we need to address issues of load balancing among system members. Specifically, defining the requirements of system membership, and the allowable workload from each member are important issues to address. For example, we need to answer questions like “Is there a lower limit on the amount of resources each member should contribute to the distributed system?”, “Is there an upper bound on the number of queries per unit time that each member is allowed to pose to the system?”, and “If the answer to either of the previous questions is yes, then what is the limit in this case?”. Leaving such issues un-handled opens the door of system abuse, and unfair treatment to some system members. This could motivate selfish or dishonest behavior, which is in sharp contrast with the distributed service provisioning principles.

6.1.2 Correctness/Security issues

Another important direction is handling issues of correct performance and security of the system members. Consider for example a FMA in which we monitor temperature of a given field. Assume also there is some member whose sensor is malfunctioning. All samples collected and reported by this member constitute false representation of the temperature. We need to arm the system with mechanisms to detect and isolate such cases, otherwise, the provided service would not be trusted by users.

Another example from MDA, if there is a member that uses the system to deliver its own messages. However, this member does not actively follow detour hints offered by the system to help deliver other member messages. Leaving such a door open gives incentives to abusing the system and breaking the entire service.
6.1.3 Pricing Models

Most research work in distributed systems assumes that system participants have the intention to actively participate in the operation of the system. In this thesis, we followed similar assumptions. Practically, this assumption might not be always true. Specifically, it could be the case that a group of “members” are not willing to participate in the operation system (i.e., contribute their resources) unless offered some incentives. Such incentives could be in the form of similar or different services received from the system, or could be in an entirely different form. The need to quantify the compromise between contributions of system members against incentives offered by the system is an interesting research problem. This includes attaching a price to each of the member contributions and a similar price to the “payback” from the system, in order to make an optimized decision.

6.1.4 New Application Domains

We believe that the general service provisioning model that we developed in this thesis is applicable in other application domains. Vehicular networks is an example of such domains. It is conceivable that, using our model, we can deploy distributed traffic monitoring system. Interfacing such a system with the Internet would provide a useful service to allow users to query the congestion state of any street, and get real-time responses. Using cameras deployed in many vehicles today (e.g., rear-view camera) for accident detection is another application. A real-time image processing program is needed to process the received images and decide whether there is a situation (fire, accident, etc) that warrants intervention or not. Similar application is to detect road conditions in terms of presence of ice, snow, rain in the Winter. This information could be also reported to users on the Internet. Information like this could help plan trips to avoid icy roads in the Winter.

A different example comes from the world of the thriving social networks (e.g., Facebook, MySpace, Orkut, etc). Consider for example the introduction of such applications to handheld devices. Drifting from the centralized client/server application, exchange of users information, profiles, pictures, comments, and other files could be done in a peer-to-peer
fashion. This model is motivated by non-flat pricing models, in which users are billed according to how much bandwidth they consume. In order to employ our model of service provisioning in this domain, we need to design a utility assignment mechanism to assign some form of merit to locally storing other people’s information. For example, a user who is a member of some group might be more interested in keeping information of other members of the group than keeping information of strangers. As we pointed out multiple times, this information could be used to assess in the management of the limited storage available to this application. A user interested in finding out some information about some other user could pose this query to the system to locate the needed information. This model could be extended to help a member in some group to physically locate other members of the group.

The above are just two potential applications of the distributed model of service provisioning in mobile networks. We do believe that this model has potential in many other domains. Over time, we expect that these domains will only increase, due to the need to growing need to deploy services in new locations, and the rising cost of laying down the needed infrastructure.
Bibliography


[BLB02] Allan Beaufour, Martin Leopold, and Philippe Bonnet. Smart-tag based data dissemination. In *WSNA '02: Proceedings of the 1st Association for Comput-


Washington, DC, USA, 2001. Institute of Electrical and Electronics Engineers (IEEE) Computer Society.


[FCAB00] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z. Broder. Summary cache: a scalable wide-area web cache sharing protocol. Institute of Electrical and Electronics Engineers (IEEE)/Association for Computing Machinery (ACM) Transactions on Networking, 8(3):281–293, 2000.


[MABM08] Hany Morcos, George Atia, Azer Bestavros, and Ibrahim Matta. An information theoretic framework for field monitoring using autonomously mobile sensors (best paper in the applications track). In *Proceedings of the 4th Institute of Electrical and Electronics Engineers (IEEE) International Conference on*


[MFHH03] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. The design of an acquisitional query processor for sensor networks. In SIGMOD ’03: Proceedings of the 2003 Association for Computing Machinery (ACM) SIGMOD international conference on Management of data, pages


[SH03b] Tara Small and Zygmunt J. Haas. The shared wireless infostation model: a new ad hoc networking paradigm (or where there is a whale, there is a way). In MobiHoc ’03: Proceedings of the 4th Association for Computing Machinery (ACM) international symposium on Mobile ad hoc networking & computing, pages 233–244, New York, NY, USA, 2003. Association for Computing Machinery (ACM) Press.


Curriculum Vitae

Hany Morcos

CONTACT INFORMATION

Computer Science Department
Boston University
111 Cummington Street,
Boston, MA 02215, USA

Phone: +1 (617) 888-9838
Fax: +1 (617) 353-6457
E-mail: hmorcos@cs.bu.edu
Web Page: http://cs-people.bu.edu/hmorcos/

EDUCATION

Ph.D., Computer Science
College of Arts and Sciences, Boston University, Boston, Massachusetts, USA
January 2004 - August 2008
Current GPA: 3.84/4 (January 2009)
Thesis title: Service provisioning in mobile networks through coordinated resource management.
Committee: Azer Bestavros, Ibrahim Matta, Mark Crovella, and John Byers

MEE in Electrical and Computer Engineering
George R. Brown School of Engineering, Rice University, Houston, Texas, USA
August 2001 - January 2003
GPA: 3.37/4

B.Sc. in Computer Engineering
College of Engineering, Cairo University, Cairo, Egypt
July 1998
Grade: Distinction with honor

RESEARCH INTERESTS

Design, analysis and evaluation of caching and mobility protocols for delay tolerant, vehicular, mobile ad-hoc, and wireless sensor networks.
TEACHING AND ACADEMIC POSITIONS

Research Assistant
Computer Science Department, Boston University
June 2004 - August 2008

Teaching Assistant
Computer Science Department, Boston University
January 2004 - January 2006

Research Assistant
Electrical and Computer Engineering, Rice University
August 2001 - August 2002

Teaching Assistant
Computer Science Department, Cairo University
September 1998 - June 2001

RESEARCH ACHIEVEMENTS

Mobility Planning in Sensor Networks (with Azer Bestavros and Ibrahim Matta, Boston University)
We study the problem of mobility coordination of a number of mobile nodes in order to achieve a specific monitoring of a given field. Mobility of nodes under consideration is coarsely controlled by a schedule that features some slack. The problem is to coordinate such slack to monitor the field. We show that this problem is NP-complete, then propose TFM, the first mobility coordination algorithm that under which nodes are mobile in the steady state. This allows achieving the required distribution even with a small number of mobile nodes. We compare TFM to random mobility and use both synthetic and trace-driven evaluations. Our results show that the performance under TFM is orders of magnitude better than that under random mobility.

Mobility Planning in Delay Tolerant Networks (with Azer Bestavros and Ibrahim Matta, Boston University)
Mobility could be adapted as a control parameter in Delay Tolerant Networks (DTN). The goal is to maximize the message delivery and minimize average message delay. We designed two approaches to solve the problem: a centralized workload-cognizant approach, and a distributed workload-incognizant approach. We show that our proposals offer a tremendous improvement with respect to message delivery and average delay.

Caching in Ad hoc/Sensor Networks (with Azer Bestavros and Ibrahim Matta, Boston University)
We designed distributed algorithms to enable a system of autonomously mobile ad hoc/sensor nodes to perform distributed field monitoring. This is done to serve user queries about the field. The system achieves very high success ratio in query answering using a very little amount of communication energy. It performs fairly well in partitioned network, networks with high probability of packet loss, and networks with limited/no mobility. We developed a full C++ wireless network simulator that simulates perfect communication in mobile wireless sensor networks.

Routing in Sensor Networks (with Ibrahim Matta and Azer Bestavros, Boston University)
We designed and evaluated a novel routing protocol for wireless sensor networks. Our routing protocol \( M^2RC \), exploits the relation between the amount of consumed power to forward packets between two nodes and the physical distance between them. Since this relation does not obey the triangular inequality, \( M^2RC \) forwards packets to the closest neighbors in areas of high reliability, and uses high-powered transmissions in areas of lower reliability to overcome node failures and random packet losses. Our protocol shows superior performance over other protocols like GRAB, Minimum-Power routing and Maximum-Power routing.

Routing in Ad-hoc Networks (with David B. Johnson, Rice University)
We introduced multiple modifications and improvements to the Dynamic Source Routing (DSR) protocol. Our contribution is three fold. First, we revised the implementation of the Address Resolution Protocol (ARP). This modification cut the maximum packet latency to 10% of its original value. Second, we proposed and simulated a novel technique to handle route breaks that also decreased the average latency for constant-bit-rate traffic by 20%. Finally, we proposed another new technique to handle route breaks that exploits overhearing of ongoing communication to maintain connectivity even when a node on the route moves out of its neighbors’ reach. This technique improved the packet delivery ratio of DSR.

EXPERIENCE

Fidelity Investments Marlborough, Massachusetts (summer intern) May 2006 - August 2006
- Analyzed requirements to build an Emergency Change Request system.
- designed, and implemented the system using C# in Visual Studio .Net and SQL server.
- Implemented feedback by system users and senior Management.

DataShack, Inc. Mississauga, Ontario, Canada January 2003 - November 2003
- Analyzed interaction between SAP and Optivision software through E-link.
- Analyzed side-effects of upgrading SAP to version 4.6 on E-link.
- Identified potential problems and functions that may work incorrectly.
- Worked on both SAP and E-link to solve problems due to the upgrading process.
- Tested the new system using real-life data.
- Verified that the upgraded system works as expected.

Intelligent Systems Cairo, Egypt (part-time job) September 1999 - May 2001
- Designed solutions based on a primary analysis of the customers’ needs and requirements.
• Designed database needed to implement solutions.
• Implemented solutions in PowerBuilder.
• Tested, verified and validated implemented systems.
• Installed products at customer sites.
• Analyzed customers’ feedback and implemented needed updates.
• Trained customer employees to use products.

**IBM Corporation** Cairo, Egypt (summer intern) May 1997 - August 1997

• Worked on the development of database applications using OS2 scripts.
• Implemented another database system using VisualAgeC++ and DB2.

**Publications**


• Hany Morcos. Introducing Route Elongation to DSR. Master Project report. ECE Department Rice University, December 2002.

Conference Presentations


• “Amorphous placement and informed diffusion for efficient field monitoring by autonomously mobile sensors”. In IEEE SECON, San Francisco, CA, June 2008.


Posters


Skills

Platforms: Unix, Linux and Windows.
Databases: SQL server, Oracle, Sybase, and Access.
Scientific Tools: Matlab, and Mathematica.
Simulators: Network Simulator (NS2), and Groovenet (vehicular network simulator).

Awards

Graduate Student Fellowship, Boston University, January 2004-2008.
Graduate Student Fellowship, Rice University, 2001-2002.
Graduate Scholarship, Cairo University, 1998-2001.
Distinction Medal, Cairo University, 1998.

PROFESSIONAL ACTIVITIES

Volunteered in the Organizing Committee of the Fourth Workshop on Applications and Services in Wireless Networks (ASWN 2004), Boston, Massachusetts.