HAL
open science

Energy Conservation for Collaborative Applications in Wireless Sensor Networks

Oualid Demigha

To cite this version:

Oualid Demigha. Energy Conservation for Collaborative Applications in Wireless Sensor Networks. Networking and Internet Architecture [cs.NI]. Université de Bordeaux; Ecole Nationale Supérieure d'Informatique (ESI) - Alger, 2015. English. NNT : 2015BORD0058 . tel-01282912

HAL Id: tel-01282912

https://theses.hal.science/tel-01282912v1

Submitted on 4 Mar 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

DISSERTATION PRESENTED
FOR THE DEGREE OF

DOCTOR FROM

NATIONAL HIGH SCHOOL
OF COMPUTER SCIENCE, ESI - ALGIERS

AND UNIVERSITY OF BORDEAUX

Department: COMPUTER SCIENCE

By Oualid DEMIGHA

Energy Conservation for Collaborative Applications in Wireless Sensor Networks

Supervised by Professor Walid-Khaled HIDOUCI
and Professor Toufik AHMED

Defended publicly on November 29, 2015 in front of thesis committee members:

- Mr. Djamel Eddine ZEGOUR, Professor at ESI-Algeria (Examiner)
- Mr. Bouabdellah KECHAR, Professor at Oran University-Algeria (Examiner)
- Mrs. Samira MOUSSAOUI, Professor at USTHB-Algeria (Reviewer)
- Mr. Hacène FOUCHAL, Professor at Univ. of Reims-France (Reviewer)
- Mr. Walid-Khaled HIDOUCI, Professor at ESI-Algeria (Advisor)
- Mr. Toufik AHMED, Professor at ENSEIRB-France (Advisor)

THÈSE EN COTUTELLE PRÉSENTÉE
POUR OBTENIR LE GRADE DE

DOCTEUR DE

L'UNIVERSITÉ DE BORDEAUX

ET L'ECOLE NATIONALE SUPÉRIEURE
D'INFORMATIQUE, ESI - ALGER

Spécialité: INFORMATIQUE

Par Oualid DEMIGHA

Conservation d'énergie pour les applications collaboratives dans les réseaux de capteurs sans fil

Sous la direction du Professeur Toufik AHMED
et Professeur Walid-Khaled HIDOUCI

Soutenue publiquement le 29 Novembre 2015 devant les membres de jury:

- M. Djamel-Eddine ZEGOUR, Professeur à l'ESI-Algérie (Examinateur)
- M. Bouabdellah KECHAR, Professeur à l'Univ. d'Oran-Algérie (Examinateur)
- Mme. Samira MOUSSAOUI , Professeur à l'USTHB-Algérie (Rapporteur)
- M. Hacène FOUCHAL, Professeur à l'Univ. de Reims-France (Rapporteur)
- M. Walid-Khaled HIDOUCI, Professeur à l'ESI-Algérie (Directeur de thèse)
- M. Toufik AHMED, Professeur à l'ENSEIRB-France (Co-Directeur de thèse)

To the memory of my mother and my father
“God bless them”

To my darling daughter, Zeineb

© 2015 Oualid Demigha

This page is intentionally left blank

Abstract

Wireless Sensor Networks is an emerging technology enabled by the recent advances in Micro-Electro-Mechanical Systems, that led to design tiny wireless sensor nodes characterized by small capacities of sensing, data processing and communication. To accomplish complex tasks such as target tracking, data collection and zone surveillance, these nodes need to collaborate between each others to overcome the lack of battery capacity. Since the development of the batteries hardware is very slow, the optimization effort should be inevitably focused on the software layers of the protocol stack of the nodes and their operating systems.

In this thesis, we investigated the energy problem in the context of collaborative applications and proposed an approach based on node selection using predictions and data correlations, to meet the application requirements in terms of energy-efficiency and quality of data. First, we surveyed almost all the recent approaches proposed in the literature that treat the problem of energy-efficiency of prediction-based target tracking schemes, in order to extract the relevant recommendations. Next, we proposed a dynamic clustering protocol based on an enhanced version of the Distributed Kalman Filter used as a prediction algorithm, to design an energy-efficient target tracking scheme. Our proposed scheme use these predictions to anticipate the actions of the nodes and their roles to minimize their number in the tasks. Based on our findings issued from the simulation data, we generalized our approach to any data collection scheme that uses a geographic-based clustering algorithm. We formulated the problem of energy minimization under data precision constraints using a binary integer linear program to find its exact solution in the general context. We validated the model and proved some of its fundamental properties. Finally and given the complexity of the problem, we proposed and evaluated a heuristic solution consisting of a correlation-based adaptive clustering algorithm for data collection. We showed that, by relaxing some constraints of the problem, our heuristic solution achieves an acceptable level of energy-efficiency while preserving the quality of data.

Résumé

Les réseaux de capteurs sans fil est une technologie nouvelle dont les applications s'étendent sur plusieurs domaines: militaire, scientifique, médical, industriel, etc. Cette technologie a vu le jour grâce aux récents progrès en matière des Systèmes Micro-Electro-Méchaniques (MEMS) qui ont permis de construire des nœuds capteurs sans fil de taille miniature.

Quand ces nœuds sont déployés en grand nombre (à l'ordre de 10.000 à 100.000 nœuds) dans une zone à surveiller ou à contrôler, l'interaction entre eux devient une véritable opportunité pour combler le manque de ressources par l'agrégation des moyens de calcul et de capacités de stockage des nœuds singuliers. Ceci permet la réalisation des applications aussi complexes que le pistage d'objets mobiles, la collecte de données multimodales ou bien la surveillance de zones sensibles.

Cependant, la miniaturisation malgré qu'elle facilite le déploiement et réduit les couts, induit des défis en matière de capacités de traitement, de stockage et de transmission de données. Les nœuds capteurs sans fil sont généralement dotés de puces de calcul d'une faible fréquence d'horloges et ne dispose que de quelques mégas voire des kilos d'octets de mémoire (vive et de stockage). De plus, les rayons de communication et de capture sont très limités ($100m$ au maximum). Ceci impose impérativement que les nœuds communiquent en multi-sauts, et complique davantage les opérations de routage et de partage de médium sans fil.

D'un autre point de vue, la possibilité de tirer profit de la densité du réseau et de l'autonomie des nœuds permet de mettre en place une collaboration entre eux afin de réaliser les applications citées ci-dessus. Ainsi, la fusion des données redondantes et multimodales issues de différents nœuds augmente la qualité d'estimation et aide à planifier les actions des nœuds selon leurs rôles et leurs ressources d'énergie.

La batterie d'un nœud capteur, comme tout autre composant, est soumise à la contrainte de la taille ce qui réduit sa capacité. De plus, la contrainte matérielle sur le développement des batteries des nœuds capteurs semble être un problème persistant. D'où la nécessité de l'optimisation logicielle dans les différentes couches

de la pile protocolaire et du système d'exploitation des nœuds. Ainsi, le problème d'énergie dans les réseaux de capteurs sans fil se transforme en un problème d'optimisation des algorithmes et des protocoles dans les différentes couches de la pile protocolaire qui font fonctionner le nœud.

L'une des méthodes les plus utilisées dans ce contexte est la sélection des nœuds basée sur la prédiction et la corrélation des données. Puisque les nœuds capteurs sans fil ne sont pas de simples interfaces de communication sans fil mais des sources de données (*grandeurs physiques*), ces données peuvent être utilisées pour optimiser leur énergie tout en préservant la qualité de données. Ceci constitue la problématique traitée dans cette thèse.

Pour approcher cette problématique, nous élaborons quatre (04) contributions pour la conservation des ressources énergétiques du réseau en utilisant la prédiction comme un moyen pour anticiper les actions des nœuds et leurs rôles afin de minimiser le nombre de ceux qui sont impliqués dans la tâche en question.

Dans la première contribution, nous dressons un état de l'art des différentes méthodes et approches récemment proposées dans la littérature. Nous prenons l'application de pistage d'objets mobiles comme un cas d'étude. Nous proposons une classification qui s'inspire du fait que le sous-système de capture et le sous-système de communications d'un nœud capteur peuvent interagir via l'algorithme de prédiction pour optimiser la consommation d'énergie. Le sous-système de capture fournit des données de lectures bruitées et redondantes qui sont fusionnées par l'algorithme de prédiction afin d'extraire des informations *utiles*. Il estime l'état courant du système et génère une ou plusieurs prédictions pour les prochaines étapes. Ces prédictions sont utilisées par le sous-système de communication afin de réveiller les nœuds les plus probables pour prendre en charge la tâche de pistage dans les prochaines étapes. Le sous-système de communication organise le réseau sous forme d'arbres ou de clusters, détermine la zone d'activation et envoie les messages de réveil. De cette façon, nous épargnons l'énergie des nœuds qui sont loin de l'évènement d'intérêt et nous obtenons une qualité meilleure des données de mesure car l'objet mobile est supposé être proche des nœuds.

Par ailleurs, nous avons distingué deux sous-classes de chaque classe de méthodes à savoir: les méthodes de routage et d'agrégation et les méthode d'auto-organisation du réseau pour la classe des approches basées sur le sous-systèmes de communication, et les méthodes de traitement de signal à nœud unique et les méthodes de traitement de signal collaboratif pour la classe des approches basées sur le système de capture. Nous avons détaillé chaque sous-classe de méthodes avec plusieurs exemples récents afin de montrer l'idée de base de chaque méthode, son algorithme de prédiction, son mécanisme de transfert de données, son mécanisme d'activation des nœuds, sa métrique de la qualité de détection et enfin sa structure logique du réseau. Nous avons mené une comparaison théorique entre toutes les méthodes étudiées afin d'extraire les recommandations et les perspectives de recherche.

Dans la deuxième contribution, nous proposons un schéma de pistage d'objets mobiles basé sur le clustering dynamique qui utilise une variante modifiée du filtre de Kalman distribué. Le but est de *localiser* le traitement des données dans la région où est détectée la cible. Ainsi, les nœuds établissent de manière distribuée un cluster dynamique qui suit la trajectoire de la cible, et change de structure chaque fois que l'état de la cible change. La distribution du filtre de Kalman aide à *alléger* le calcul des estimations et des prédictions sans avoir besoin de transmettre toutes les données de lecture à un centre de fusion globale. Les nœuds détectent la cible en utilisant un modèle de détection probabiliste et exécute un consensus des filtres de Kalman distribués en se basant sur un modèle de communication par messages. Nous avons établit une règle d'élection du cluster-head qui utilisent des informations sur les nœuds se trouvant dans la liste des candidats telles que: (1) la distance entre le nœud et la cible, (2) le nombre des nœuds actifs dans le cluster, et (3) l'énergie résiduelle des nœuds. La comparaison avec d'autres schémas de pistage d'objets mobiles basés sur le filtre de Kalman montre bien l'efficacité énergétique de la méthode proposée.

Dans la troisième contribution, nous généralisons les résultats obtenus par les simulations de notre algorithme de clustering dynamique à tout autre algorithme de collecte de données basé sur le clustering géographique et qui soit efficace en énergie. Nous formalisons le problème que nous avons appelé EMDP pour *Energy Minimization under Data Precision constraints Problem* à l'aide d'un programme linéaire à variables binaires (BILP). Ce programme nous donne la solution exacte qui permet de déterminer à chaque étape de collection de données quels sont les nœuds sélectionnés pour faire la capture, les nœuds qui jouent le rôle de relais et les nœuds d'agrégation (les clusterheads ou les nœuds racines).

Deux ensembles de variables sont définies à chaque étape: les variables a_i^t et les variables b_{ij}^t. Une variable a_i^t détermine si le nœud s_i est sélectionné pour participer à la capture à l'étape t et une variable b_{ij}^t indique si le nœud s_j est sélectionné comme parent du nœud s_i dans l'arbre de collection de données à l'étape t. L'arbre ainsi construit est dynamique est il est reconfiguré à chaque étape. La fonction objectif est: *minimiser* la somme des énergies des nœuds dépensées dans la capture, la transmission et le traitement des données. Ce modèle est soumis à un ensemble de contraintes qui régissent la structure du réseau, l'élection des nœuds d'agrégation et de relais, la mise à jour des énergies des nœuds et l'acheminement des données. La contrainte sur l'acheminement des données a fait l'objet d'une preuve mathématique formelle afin de prouver l'exactitude du modèle. Par ailleurs, le modèle a été implémenté et testé en utilisant le langage GMPL (*GNU MathProg modeling Language*) pour de petites instances. Étant donnée la complexité du problème EMDP, nous avons jugé nécessaire de proposer des solutions heuristiques qui s'approchent de la solution exacte pour les grandes instances du problème. Ce qui a fait l'objet de la quatrième contribution.

Dans cette dernière contribution, nous avons proposé une solution heuris-

tique qui consiste en un algorithme de clustering dynamique basé sur la corrélation des données, appelé CORAD (*Correlation-Based Adaptive Dynamic Clustering Algorithm*). CORAD adapte la topologie du réseau au dynamisme des données capturées afin d'optimiser la consommation d'énergie en exploitant la corrélation qui pourrait exister entre les données.

C'est un algorithme gourmand inspiré de l'heuristique du *plus proche d'abord*. A chaque étape, CORAD sélectionne le nœud avec le maximum rapport *proximité/budget d'énergie* comme étant le nœud d'agrégation et ensuite construit l'arbre de collection de données. Pour ce faire, CORAD utilise un modèle d'énergie basé sur la propagation dépendante de la distance et les données capturées par les nœuds membres du même cluster.

A chaque étape, CORAD estime la qualité des données et la compare à un seuil défini: si la qualité estimée est en baisse, alors CORAD augmente le nombre de nœuds participant à la collecte des données à la prochaine étape. Ceci augmentera évidemment la consommation d'énergie mais sauvegarde la qualité des données. Sinon, si la qualité des données estimée par CORAD est en hausse, alors le nombre de nœuds qui seront impliqués dans la prochaine étape de collecte de données sera baissé. En faisant ainsi, nous préservons les ressources d'énergie des nœuds. Notre solution est testée par simulation et les résultats obtenus montrent bien son efficacité en termes d'énergie et de qualité de données.

Notre thèse ouvre de nombreuses perspectives quant à la caractérisation du problème EMDP et sa structure ainsi qu'aux possibilités d'application d'autres heuristiques ou méta-heuristiques afin d'améliorer les solutions approchées.

Acknowledgments

This thesis would not have been possible without the help of God Allah. Thanks to Allah for giving me the chance and the courage to accomplish it.

First, I would like to express my gratitude to my advisors Professor Toufik Ahmed and Professor Walid-Khaled Hidouci for supporting me incessantly to study problems that I considered interesting myself and come up with my own ideas, yet they always helped me in thought-provoking discussions when I needed inspiration. I would like to thank them for giving me valuable feedback. They have been fantastic mentors. I count myself extremely lucky that I have been jointly supervised by two advisors getting me to enrich my experience.

I have to extend my thanks also to my most prolific collaborators members of the Artificial Intelligence Lab at Ecole Militaire Polytechnique: Associate Professor Mohamed Aissani, PhD Dr. Ali Yachir, PhD Dr. Mustapha-Rada Senouci and PhD student Mohamed Boudali. I shared with them my working environment during the four past years.

I am also very grateful that I could share my time during my research visits at LaBRI laboratory with PhD students: Omar Tout, Christian Glacet, Gabriel Renault, Samir Medjiah and Abbas Bradai who are now doctors.

If it was not for the continued support of my family: my sisters and my brothers, I could not be in the position that I am today. I will forever be grateful for their encouragement and unconditional love. I also want to thank my friends Bilel Laloui, Issad Djeltane and Amir Abbas for being a great brothers and an even better friends.

Last but certainly not least, I would like to thank my wife, for always believing in me, for always standing by me, and for always being there for me.

List of Acronyms

ADC	Analogical-Digital Converter
ADCT	Adaptive Dynamic Cluster-based Tracking
AIT	Alert-In Triangulation
AMPL	A Mathematical Programming Language
AR	Auto-Regressive
ASAP	Adaptive Sampling APproach
BAN	Body Area Networks
BFST	Breath First Search Trees
BILP	Binary Integer Linear Programming
BN	Boundary Node
BPOM	Binary Proximity Observation Model
CAG	Clustered AGgregation
CCA	Clear Channel Assessment
CGS	Controlled Greedy Sleep
CN	Computation Node
COBOM	Continuous BOundary Monitoring
CODA	Continuous Object Detection and Tracking Algorithm
CollECT	Collaborative Event detECtion and tracking
CORAD	Correlation-Based Adaptive Dynamic Clustering
COST	Commercial Off-The-Shelf
COZ	Compare-One-Zero
CRC	Cyclic Redundancy Check
CSP	Collaborative Signal Processing
CRLB	Cramer-Rao Lower Bound
CV	Constant-Velocity
CVN	Changed-Value Node
DKF	Distributed Kalman Filter
DKF_DC	Distributed Kalman Filter with Dynamic Clustering
DKF_GOSSIP	Distributed Kalman Filter with Gossip Communication

DKF_LSR	Distributed Kalman Filter with Limited Sensing Range
DPM	Dynamic Power Management
DPT	Distributed Predictive Tracking
DSP	Digital Signal Processor
DSPS	Dual Shortest Path Selection
DSTA	Distributed Spanning Tree Algorithm
EAST	Efficient Data Collection Aware of Spatial-Temporal Correlation
ECOT	Energy-efficiency Continuous Object Tracking protocol
EEDC	Energy-Efficient Data Collection Exploiting Spatial-Temporal Correlation
EEPROM	Electrically Erasable Programmable Read-Only Memory
EKF	Extended Kalman Filter
Elink	in-network Clustering
EMDP	Energy Minimization under Data Precision Constraints
FOTP	Face-based Object Tracking Protocol
FSK	Frequency Shift Keying
GB	Giga-Bytes
GLPK	Gnu Linear Programming Kit
GMPL	Gnu Math Prog Language
GRASP	Greedy Randomized Adaptive Search Procedure
GSSM	Greedy-Selection Sensor Management
HCTT	Hybrid Cluster-based Target Tracking
HN	Helper Node
HTTP	Herd-Based Target Tracking Protocol
IDSQ	Information-Driven Sensor Querying
ILP	Integer Linear Programming
ISM	Industrial-Scientific-Medial Radio Frequency Band
ISI	Inter-Sensor Interferences
ITS	Intelligent Transport Systems
KB	Kilo-Bytes
kbps	kilo bits per seconds
KCF	Kalman Consensus Filter
KF	Kalman Filter
LQI	Link Quality Indicator
MAC	Medium Access Control
MAV	Micro Air Vehicles
MB	Mega-Bytes
MEMS	Mechanic Electronic Micro Systems
MIPS	Million Instructions Per Second
MN	Main Node
NLMP	Network Lifetime Maximization Problem
NN	Nearest Node
NSPS	Naive Shortest Path Selection

P2P	Peer To Peer
PF	Particle Filter
P-GEP	Parallel Gene Expression Programming
PID	Proportional-Integral-Derivative
POWERTOSSIM	Energy Oriented TinyOS Simulator
PTT	Polygon-based Target Tracking
PV	Position-Velocity
QoS	Quality of Service
RADAR	RAdio Detection And Ranging
RAM	Random Access Memory
RFID	Radio Frequency IDentification
RN	Representative Node
RISC	Reduced Set Instruction Computer
ROM	Read-Only Memory
RPLMP	Routing Path Length Minimization Problem
RSSI	Received Signal Strength Indicator
SIR	Sampling Importance Re-sampling
SP	Signal Processing
SSP	Sensor Selection Problem
TCP	Transport Control Protocol
TEKWEM	Timely Energy-Efficient k-Watching Event Monitoring
TinyOS	Event-Driven Operating System for senor nodes
TOCOB	Tracking Of Continuous Objects
TOSSIM	TinyOS Simulator
UDP	User Datagram Protocol
UKF	Unscented Kalman Filter
VD	Voronoi Diagram
UAV	Unmanned Aerial Vehicles
VF	Variational Filter
WHSN	Wireless Heterogeneous Sensor Networks
WN	Worker Node
WSN	Wireless Sensor Networks
WSPS	Weighted Shortest Path Selection
YEAST	dYnamic and scalablE tree Aware of Spatial correlaTion

This page is intentionally left blank

Contents

1 Introduction **21**
1.1 Wireless Sensor Networks . . . 22
1.1.1 Wireless Sensor Nodes . . . 22
1.1.2 Applications . . . 27
1.1.3 Challenges . . . 28
1.2 The Energy Problem . . . 32
1.2.1 Protocol Stack . . . 32
1.2.2 Collaborative Applications . . . 35
1.2.3 Our Contributions . . . 35
1.3 Research Publications . . . 37
1.4 Manuscript Organization . . . 37

2 Energy Efficient Collaborative Target Tracking: A Survey **38**
2.1 Schemes Classification . . . 40
2.1.1 Target Tracking Schemes Characterization . . . 40
2.1.2 General Classification . . . 42
2.2 Collaborative Signal Processing Methods . . . 44
2.2.1 Information-Driven Techniques . . . 44
2.2.2 Data Filtering Techniques . . . 45
2.3 Network Self-Organization Approaches . . . 47
2.3.1 Sleep Scheduling . . . 47
2.3.2 Node Selection . . . 51
2.3.3 Dynamic Clustering . . . 54
2.4 Classification of Specific Approaches . . . 56
2.4.1 Continuous Object Tracking . . . 56
2.4.2 Tracking with Binary Sensors . . . 58
2.5 Schemes Comparison and Discussion . . . 59
2.6 Conclusion . . . 61

3 Energy Efficient Target Tracking Scheme with Limited Sensing Range 63
3.1 System Model and Assumptions 65
3.1.1 WSN Model 65
3.1.2 Centralized Kalman Filter 66
3.1.3 Consensus on Kalman Micro-Filters 68
3.2 Proposal of Dynamic Clustering Based on Distributed Kalman Filter 70
3.2.1 Cluster Formation and Leader Election 71
3.2.2 Cluster Reconfiguration 73
3.2.3 Target State Estimation 75
3.3 Simulations and Results 78
3.3.1 Simulation Setup 78
3.3.2 Energy Consumption 78
3.3.3 Quality of Estimation 79
3.4 Conclusion 83

4 Energy Modeling under Data Precision Constraints 84
4.1 Background and Related Work 85
4.2 Problem Definition & Formulation 86
4.2.1 System Model & Parameters 87
4.2.2 Binary Integer Linear Programming Formulation 87
4.2.3 Proof of Constraint 4.9 91
4.3 Model Validation 95
4.4 Conclusion 96

5 Correlation-Based Adaptive Dynamic Clustering Algorithm for Data Collection 97
5.1 Basic Idea & Objectives 99
5.1.1 Network Model 100
5.1.2 Energy Model 101
5.1.3 Data Source Model 101
5.1.4 Detection Model 102
5.2 Static Clustering Scheme 102
5.3 CORAD: Correlation-Based Adaptive Dynamic Clustering Scheme 103
5.3.1 Data Correlation Capture Model 105
5.3.2 Dynamic Clustering Rules 108
5.4 Simulation Results & Analysis 113
5.5 Conclusion 121

6 Conclusions & Future Work 122
6.1 Contributions 123
6.2 Future Work 123
6.2.1 Characterization of EMDP 123

6.2.2 Meta-Heuristics 124

A The Kalman Filter **137**
A.1 Preliminaries 137
A.1.1 Probability 137
A.1.2 Random Variables 138
A.1.3 Mean, Variance & Covariance 139
A.1.4 Gaussian Distribution 140
A.2 Discrete Kalman Filter 140
A.2.1 Computational Origins of KF 141
A.2.2 High-Level Operation of KF 143
A.3 Extended Kalman Filter 145
A.3.1 Computational Origins of EKF 145

B Gnu Linear Programming Kit **148**
B.1 Gnu MathProg Language 148
B.1.1 Coding Rules 149
B.1.2 Compilation 151
B.1.3 Expressions 151
B.1.4 Modeling Objects 157
B.2 GLPK Implementation for EMDP 161

List of Figures

1.1 Typical architecture of a sensor node. 22
1.2 A set of sensor nodes with different sizes. 26
1.3 Examples of WSN applications: (a) Soil humidity measurement in agriculture, (b) Precise irrigation, (c) Volcanic irruption surveillance, and (d) Generic WSN testbed. 27
1.4 A simulation-generated topology for 150-nodes clustered WSN deployed in 100×100 square area. 30
1.5 Stack protocols of sensor nodes. 32

2.1 Example of a prediction-based scheme. 39
2.2 Target tracking scheme components. 41
2.3 General classification. 43
2.4 Illustration of periodic scheduling. 46
2.5 Effect of empty slots. 46
2.6 Example of FOTP operations. 50
2.7 Example of DSTA protocol operations. 54
2.8 Specific approaches. 57

3.1 Probabilistic detection model. 66
3.2 Kalman filter steps. 68
3.3 Micro-filter architecture of a DKF node. 69
3.4 Example of the dynamic clustering protocol. 71
3.5 Cluster formation operations (Steps 1 and 2). 73
3.6 Cluster formation operations (Steps 3 and 4). 74
3.7 Cluster formation operations (Step 5). 75
3.8 Cluster reconfiguration. 76
3.9 State-transition diagram of the proposed clustering protocol. 77
3.10 Energy consumption of the network vs. sampling period. 79
3.11 Energy consumption of the network vs. network density. 80
3.12 Quality of estimation vs. sampling period. 80

3.13 Estimation quality for a sampling period of $3s$. 81
3.14 Performance vs. target speed (1). 81
3.15 Performance vs. target speed (2). 82

4.1 Cases of direct neighborhood of nodes. 91
4.2 Illustration of the data forwarding constraint. 92
4.3 Illustration of the base case. 94
4.4 Illustration of the recursion case. 95

5.1 Static clustering with $m = 100$, $A = 100m \times 100m$, $R_c = 30$ and $n = 1$. Red points represent the cluster-heads and black (blue) points represent the members. The line segments represent direct communication links. 105
5.2 Static clustering with $m = 150$, $A = 80m \times 80m$, $R_c = 15$ and $n = 2$. Red points represent the cluster-heads and black (blue) points represent the members. The line segments represent direct communication links. 106
5.3 Static clustering with $m = 150$, $A = 100m \times 100m$, $R_c = 10$ and $n = 4$. Red points represent the cluster-heads and black (blue) points represent the members. The line segments represent direct communication links. 107
5.4 Example of event regions. 108
5.5 Data correlation model. 109
5.6 Determination of the clusters based on data correlations and neighboring information. 110
5.7 Example of parent selection. 111
5.8 Initialization phase of CORAD. 112
5.9 Cluster reconfiguration. 114
5.10 CORAD performance vs. $\mathcal{S}_{min}$. 116
5.11 CORAD performance vs. n. 118
5.12 CORAD performance vs. **Thd**. 119
5.13 CORAD performance vs. ϵ. 120

List of Tables

1.1 Examples of energy harvesting sources. 23
1.2 Features of some of transceivers. 24
1.3 Characteristics of some of microprocessors. 25
1.4 Properties of some sensors. 25
1.5 Characteristics of some sensor nodes. 26
1.6 Differences between wireless computer networks and WSN. 29
1.7 Power requirements for different operations in mica motes. 31
1.8 Power requirements of different radio interfaces. 33

2.1 Comparison between schemes objectives. 60
2.2 Comparison between mechanisms of sensing-related schemes. . . . 61
2.3 Comparison between mechanisms of sleep scheduling based schemes. 61
2.4 Comparison between mechanisms of node selection and dynamic clustering based schemes. 61
2.5 Comparison between mechanisms of node selection and dynamic clustering based schemes (continued). 61

3.1 Roles of messages. 72

5.1 Simulation parameters. 115

B.1 Reserved keywords of GMLP. 150
B.2 Delimiters in GMLP. 151
B.3 Main options for compilation using glpsol. 151
B.4 List of the available functions in GLPK. 153
B.5 List of the iteration operators provided by GLPK. 153
B.6 List of the arithmetic operators. 154
B.7 Hierarchy of the arithmetic operators. 154
B.8 Set operators. 156
B.9 Logic operators. 156

B.10 Parameters of the GLPK implementation. 161

This page is intentionally left blank

Chapter 1

Introduction

WIRELESS SENSOR NETWORKS (WSN) is one of the ten emerging technologies that will change the world [1]. They are made possible by the recent technological advances in MEMS (Micro-Electro-Mechanical Systems) and the increasing rate of circuit integration and size reduction.

The idea behind WSN is to address the lack of single sensor nodes capacities in terms of power, data processing and communication by aggregating their capabilities to perform complex tasks. These tiny nodes that are in most cases randomly deployed in hostile or inaccessible areas, represent a real mine of multimodal redundant and correlated data, making WSN a new paradigm of extraction and collection of data, due to their proximity to the event of interest.

In addition to the intrinsic characteristics of WSN, integration with the *Internet-of-Things* makes them a keystone of a wide range of applications from military to health-care through industry and agriculture. The market of WSN is increased from $500 millions in 2005 to $4.6 billions in 2011 and more than 4.1 millions of sensors are produced in 2010 [2]. This shows the growing interest of this new technology and the applications that can provide.

However, the energy problem remains a real bottleneck for WSN applications proliferation because of the hardware limitation. Thus, the collaboration between the nodes seems to be a promising strategy to overcome this constraint. Before addressing this problem and exploring the possible tracks and approaches for its solution, we introduce first the concept of WSN. Thereby in this chapter, we give details about WSN characteristics, applications and challenges in order to draw the context of this thesis. In Section 1.1, we give some basic definitions and concepts to understand the problem that we deal with. Then in Section 1.2, we give a rough description of this problem and our contributions to resolve it. These contributions are published and briefly presented in Section 1.3. Finally, we briefly describe the manuscript organization in Section 1.4 to make the reader more comfortable with the next chapters.

1.1 Wireless Sensor Networks

WSN can be abstracted as a set of a large number of tiny low-cost sensor nodes with limited energy, that collect data about a physical phenomenon evolving in the surrounding environment. These nodes can sense, process and send data to a base station for post-processing.

To perceive the importance of WSN, it is necessary to understand the characteristics of their basic blocks which are: the *wireless sensor nodes*.

1.1.1 Wireless Sensor Nodes

A wireless sensor node is typically composed of a set of necessary or optional components. As shown in Figure 1.1, the necessary components are: the energy sources, the power management unit, the radio unit, the microprocessor, the ADC unit, the sensors and the data storage units. Each component has a specific role and unique characteristics.

Energy Source/Harvesting Unit. It ensures the autonomy of the sensor node. The energy units are generally non-rechargeable batteries coupled with *renewal energy harvesting* units using vibrations, solar radiation or heat or any *natural* source of energy to produce electricity of the node when the batteries discharge. These batteries are not replaceable especially when sensor nodes are deployed in hostile or inaccessible environment. Thus, the use of energy harvesting devices or energy conservation techniques is necessary for long-term operation of the network.

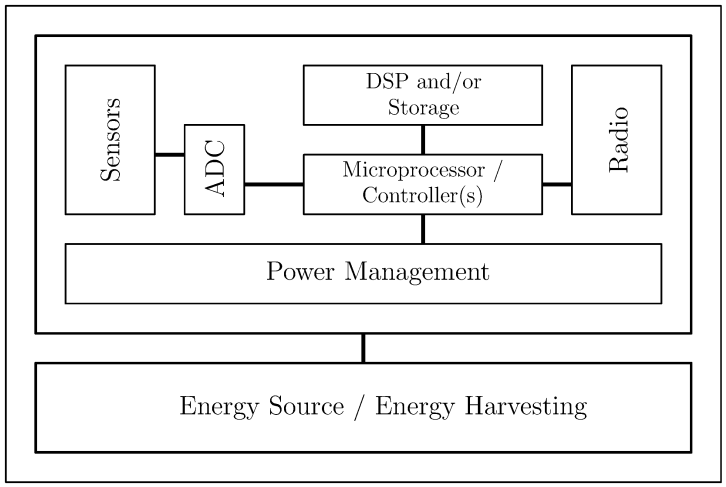


Figure 1.1: Typical architecture of a sensor node.

Table 1.1 shows some sources of energy harvesting with their respective features [3]. The major sources of energy harvesting are: radiant, thermal and mechanical. In the scavenging of radiant energy, a node converts the energy from the signals sent in the RF spectrum. The solar cells are the most mature energy harvesting technology. They can provide up to 15 mW/cm^2 from direct sunlight in outdoor conditions. However, the performance decreases significantly in indoor conditions. In the scavenging of thermal energy, a node converts the energy from the temperature difference between both of its surfaces. A difference of $10°C$ can be sufficient to produce electrical energy. Vibration is also used as a source for harvesting mechanical energy using piezoelectric conversion.

Table 1.1: Examples of energy harvesting sources.

Source	**Power Density**	**Duration**
Solar cell (direct sunlight)	15 mW/cm^2	Continuous
Solar cell (well illuminated room)	10 $\mu W/cm^2$	Continuous
Piezoelectric	200 $\mu W/cm^3$	Operation
Temperature Difference	40 $\mu W/cm^3/5°$ C	Continuous
Air flow	380 $\mu W/cm^3/5m/s$	Continuous

Power Management Unit. It is a microchip to manage the power of the node. Its role is to estimate the power supply at the ends of the energy unit, the current consumption and the level of residual energy.

Radio Unit. It is composed of radio antennas and wireless interfaces. Its role is the sensing of the carrier wave, the modulation/demodulation of the radio signals, and its transmission over the wireless medium. It also implements certain physical and MAC-layer protocols (details are in subsection 1.2.1). For example, an IEEE 802.15.4 compliant PHY includes: data frame synchronization for perceiving the start of an incoming frame, Clear Channel Assessment (CCA) for detecting ongoing traffic on a frequency channel, Received Signal Strength Indicator (RSSI) and Link Quality Indicator (LQI) for measuring signal strength and estimating link quality to neighboring nodes, respectively, Cyclic Redundancy Check (CRC) calculation for checking bit errors on received frames, data encryption/decryption for improving network security and automatic acknowledge transmissions after receiving frames. When these features are implemented in hardware, they can improve energy-efficiency. However, the increased complexity of the hardware increases its cost. Thus, in practice, we use COST (Commercial Off-The-Shelf) to reduce costs but this limits the features.

In general, the radio units of sensor nodes use a specific radio frequency band which is the license-free ISM (Industrial-Scientific-Medical) spectrum (see Table

1.2). For example, the chipcon CC1000 operates on the 433-915 MHz band and offers up to 50 programmable channels using FSK modulation. The CC2420 interface which succeeds to CC1000, is included in MicaZ motes and built to comply with the IEEE 802.15.4 standard for low data rate. It operates on 2400 MHz. Another example is the TR1000 family that can dynamically change its transmission power up to 1.4 mW, and transmits up to 115.2 kbps.

Table 1.2 provides the key figures about the characteristics of some transceivers in terms of data rate, RF band, buffer size, sleep energy, power and energy consumption of transmission and reception [3]. The major manufacturers documented are: MC (Microchip), NS (Nordic Semiconductor), RFM (sRF Monolitics), SE (Semtech) and TI (Texas Instruments).

As we can see on Table 1.2, the data rate and frequency band have a low effect on current consumption and the radios operating on 2.4 GHZ frequency band are the most energy-efficient, which is mostly caused by their high data rate.

Table 1.2: Features of some of transceivers.

Radio	Data Rate (kbps)	Band (MHZ)	Buffer (B)	Sleep (μA)	RX Power (mA)	TX Power (mA)	RX En. (nJ/b)	TX En. (nJ/b)
MC MRF34J40	250	2400	128	2	18	22	264	216
NS nRF2401A	1000	2400	32	0.9	19.0	13.0	39	57
NS nRF24L01	2000	2400	32	0.9	12.3	11.3	17	18
NS nRF905	50	433-915	32	2.5	14.0	12.5	750	840
RFM TR1001	115.2	868	-	0.7	3.8	12	313	99
RFM TR3100	576	433	-	0.7	7.0	10	52	36
SE XE1201A	64	433	-	0.2	6.0	11.0	516	281
SE XE1203F	152.3	433-915	-	0.2	14.0	33	650	276
TI CC2420	250	2400	128	1	18.8	17.4	209	226
TI CC2500	500	2400	64	0.4	17.0	21.2	127	102
TI CC1000	76.8	433-915	-	0.2	9.3	10.4	406	363
TI CC1100	500	433-915	64	0.4	16.5	15.5	93	99

Microprocessor/Controller Unit. This is the core component of the sensor node that executes programs. The microprocessor unit is usually based on a 8-16 bits RISC architecture and few number of registers, and its performance is limited. For example, Atmel128L microprocessors can process a maximum rate of 8 MIPS (Millions of Instructions Per Second) when running at 8 MHz [4].

Table 1.3 shows and compares the characteristics of some microprocessors of different manufacturers. It indicates that Semtech XE8802 and Texas Instruments MSP430F1611 are the most energy-efficient chips [3].

Table 1.3: Characteristics of some of microprocessors.

Microprocessors	Flash (KB)	SRAM (BK)	EEPROM (KB)	Sleep (μA)	1 MIPS (mA)
Atmel AT89C51RE2 (8051)	128	8	0	75	7.4
Atmel ATmega103L (AVR)	128	4	4096	1	1.38
Atmel AT91FR40162S (ARM)	2048	256	0	400	0.96
Cypress CYC29666	32	2	0	5	10
Freescale M68HC08	61	2	0	22	3.75
Microship PIC1LF8722	128	3.9	1024	2.32	1.0
Microship PIC24FJ128	128	8	0	21	1.6
Semtech XE8802 (CoolRISC)	22	1	0	1.9	0.3
TI MSP430F1611	48	10	0	1.3	0.33

ADC Unit. It performs analog-digital conversion of raw data to generate real values usable by the microprocessor/controller. In some cases, it is integrated with the microprocessor unit.

Sensors Units. They are a variety of different sensors for measuring different physical quantities such as: electromagnetic, vibration, temperature, humidity, luminosity, movement, etc. They generate and store the raw sampled data before conversion by the ADC unit. The requirements of sensor units are energy saving (low power) and early detection (short sampling interval). In addition, adequate accuracy is required in entire temperature range. Table 1.4 shows the features of some sensors [3]. It indicates that most of the sensors fulfill the above requirements well.

Table 1.4: Properties of some sensors.

Physical Qty.	Sensor	Accu.	Power (μA)	Time (ms)	Consumption (μJ)
Acceleration	VTI SCA3000	1%	120	10	3.6
Air pressure	VTI SCP1000	150 Pa	25	110	8.3
Humidity	Sensorionn SHT15	2%	300	210	190
illumination	Avago APDS-9002	50%	2000.0	1.0	6.0
Infrared	Fuji MS-320	-	35	cont.	-
Magnetic field	Hitachi HM55B	5%	9000.0	30	810
Position	Fastrax iTRAX03	1.0 m	32000	4000.0	380
Temperature	Dallas DS620U	0.5° C	800	200	480

DSP and/or Storage Units. They are flash memories of RAM type or EEPROM type for storing volatile or permanent data. The storage units are usually small

and unstructured. They can be represented by a long-array of 8-16 bits words.

In addition to these components, sensor nodes may contain application-specific equipments or optional components such as position-finding units, mobilizing unit, etc.

All these components are limited in size and capacity because they should fit in small boxes (see Figure 1.2). Many sensor nodes are designed as commercial products or scientific prototypes, such as: Mica [5], Mica2 [6] and MicaZ [7], Telos [8], TelosB [9] and imote2 [10] of Crossbow, Smart Dust [11] of Berkeley, Arduino [12], etc. Table 1.5 clearly shows the above-described characteristics [13]. As we can see on this table, Telos mote is the smallest in terms of size but it is equipped with MSP430 CPU that has a low speed. Imote2 on the other hand, is the most powerful mote with up to 416 MHZ of CPU frequency, but it requires much more energy ($3 \times AAA$ batteries) and has a double size compared to Telos.

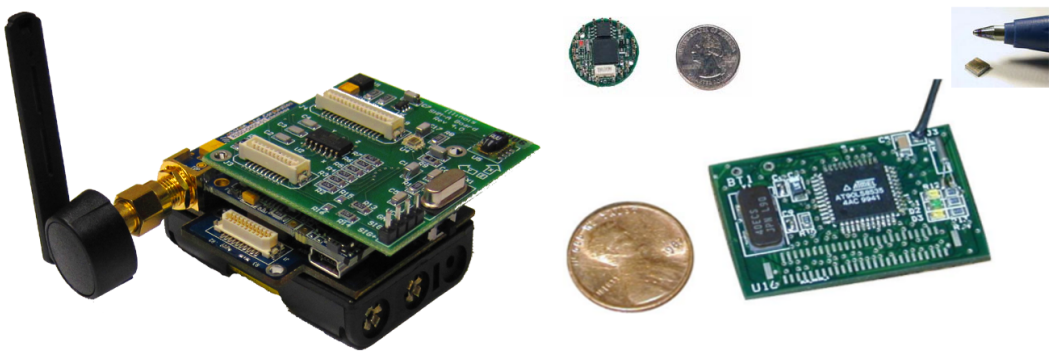


Figure 1.2: A set of sensor nodes with different sizes.

Table 1.5: Characteristics of some sensor nodes.

Mote	**MicaZ**	**Telos**	**Imote2**	**Wavenis**
Microprocessor	Atmel AT-Mega128L	TI MSP430	Intel PXA271	TI MSP 430
Processor Speed	16 MHZ	8 MHZ	13 - 416 MHZ	8 MHZ
Memory (KB)	4	2-10	256	2
Program Space	128K	60K - 48K	32K	128B
Flash	512K	256K	32M	None
Battery	2×AA	2/3 AA	3×AAA	2×AA
Voltage(V)	2.7	1.8 - 3.6	3.2 - 4.5	2.4 - 5
Radio	CC2420	CC2420	CC2420	ASIC Wavenis
Frequency(MHZ)	2400-2483	2400-2483	2400-2483	433/868/915
Data rate(KB/s)	250	250	500	4.8 - 153
Dimension(mm^3)	$58 \times 32 \times 7$	$13 \times 26 \times 5$	$36 \times 48 \times 9$	$26 \times 20 \times 4.5$

1.1.2 Applications

Today, WSN are in the heart of the Internet-of-Things, as they have a huge potential for applications. From military to health-care through automatic industrial chains control and agriculture, WSN have a place in every aspect of life. Figure 1.3 shows some applications of WSN.

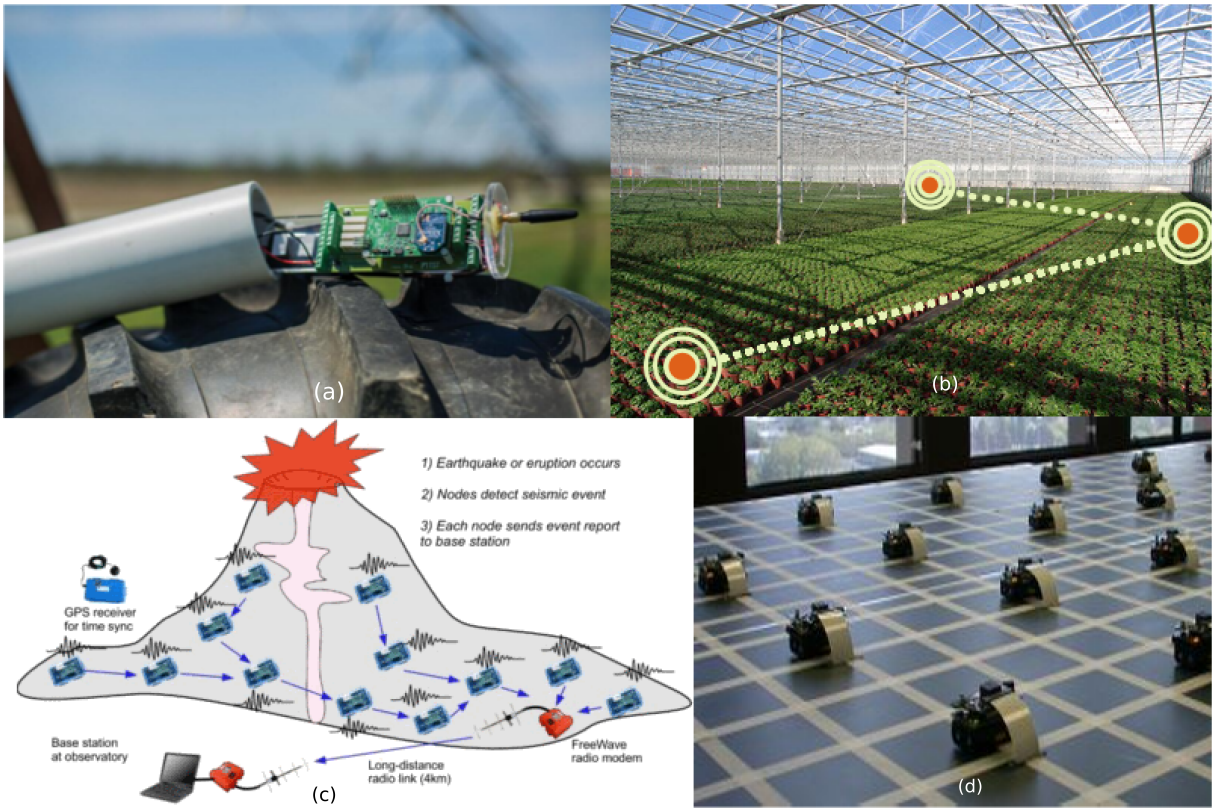


Figure 1.3: Examples of WSN applications: (a) Soil humidity measurement in agriculture, (b) Precise irrigation, (c) Volcanic irruption surveillance, and (d) Generic WSN testbed.

- In the military field, WSN are the basis of new concepts of modern battlefield, as they enable low-cost and efficient zone surveillance, barrier surveillance and target tracking. Hence, the random deployment with reduced operating costs allow fast and efficient control of the battlefield. The sensors can be deployed via UAV, artillery or may be broadcast from rolling vehicles like seeds [11]. Wireless sensors also allow early and rapid detection of chemical and biological agents in combat. They also can be used to monitor precursors of chemical usage at and around manufacturing facilities. Interrogation of the WSN is possible via UAV or MAV systems [11].

- In industry, wireless sensors are replacing increasingly wired sensors in embedded systems and automation control systems in production thanks to cost and maintenance effectiveness [14–18].
- In agriculture, WSN are used in smart irrigation and precision agriculture. For example, sensor nodes take measurements on soil moisture and air humidity and send this information to a control center so that it provides instructions to irrigation actuators [19, 20].
- In the urban monitoring, structures such as buildings, bridges and transports can be monitored by WSN. They can help predict adverse events such as: bridge collapse or road congestion, thus providing accurate information to make intelligent decisions. WSN are also used in ITS (Intelligent Transportation Systems) to control traffic on the intersections of roads and to reduce congestion [21–23].
- In the domain of rescue operations, natural disasters such as volcanic irruption [24], deluge and floods, can be predicted by WSN which help the operations to search for victims. The fire-fighting is also a potential field of applications of WSN. Sensor nodes deployed in a forest can immediately alert the authorities before a fire begins to spread uncontrollably [25].
- In health-care applications, The WSN are the basis of the concept of BAN (Body Area Network) where tiny sensor nodes are implanted on and in the patient's body to monitor vital signs [4]. Patients are free to move around while they are under constant supervision and the doctors can be alerted and dispatched automatically to respond to an emergency [26].

WSN also find their use in home automation, habitat monitoring [27, 28], etc.

1.1.3 Challenges

Although WSN offer huge potential of applications, they are faced with many technological barriers and raise many challenges that require considerable research effort. Some challenges are due to cost constraints and hardware limitation of wireless sensor nodes. Others are common challenges for most networking technologies.

Even with many similarities that WSN share with other distributed systems, they are subject to specific challenges and constraints that affect their design, which leads to algorithms and protocols that differ from their counterparts in other distributed systems (see Table 1.6 [3]).

In this section, we present and classify these challenges into four main categories, namely: energy, communication protocols, application requirements, and security and reliability.

Table 1.6: Differences between wireless computer networks and WSN.

Requirement	Computer Networks	WSN
Resource constraints	Low	Very high (1-2 MIPS, 32-128 KB)
Adaptivity	Static	Dynamic environment
Scalability	Moderates (10-100 nodes)	High (Up to 10000 nodes)
Latency	High (250 ms - 1 s)	Variable (1 s - 1 H)
Throughput	Very high (MB/s)	Low-moderate (bits/s - Kbits/s)

Energy. Energy is a fundamental constraint for WSN because sensor nodes are equipped with tiny batteries that have small capacities. In addition, the technological development of batteries is very slow and the nodes are increasingly involved in energy-consuming tasks such as sampling, transmission and reception of packets and data aggregation. Also, WSN are supposed to work for long-term autonomous operations.

To achieve such a long-time operation, the nodes must operate in very low duty-cycles and must save the transmission and reception of packets as much as possible because they are the most energy-consuming tasks, and this is a challenging task. For example, when the microprocessor runs 1000 instructions, it consumes an amount of energy equivalent to that consumed by the transceiver to send only one bit of data over the radio.

Thus, it is also important to set the maximum number of nodes that are not involved in WSN operations or have no *useful information* in the deep sleep state as long as possible to conserve their energy resources. On the other hand, the quality of data should not be degraded because of missing some important data when putting the node in sleep mode. Also, it is important to establish reliable estimations of the information utility and to trade between energy consumption and quality of data. Besides, the energy is also a key parameter in the routing and wireless medium sharing operations.

Communication Protocols. For the success of the task of data reporting, the nodes in WSN need to be interconnected to reach the base station (or the sink node). However, short-range wireless communications and random deployment of the nodes require ad-hoc multi-hop routing with data forwarding via relays. In addition, the high-density of WSN and its scalability add more complexity to the MAC-layer problems such as collision and contention for the wireless medium.

The challenges related to communication protocols concern also the routing task and the multi-objective function that should satisfy[1]. Since WSN do not use PP (Point-to-Point) communications and the nodes do not have complete information about the entire network and are not individually addressable, new com-

[1]End-to-end delay, $k-$path routing, throughput, overhead are examples of these objective functions.

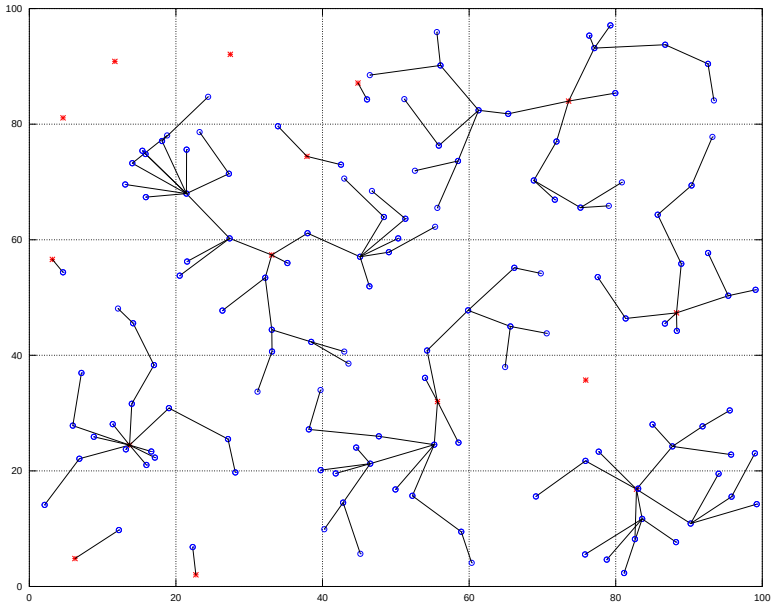


Figure 1.4: A simulation-generated topology for 150-nodes clustered WSN deployed in 100×100 square area.

munication paradigms based on data-centric view have been designed, such as directed-diffusion [29].

Topology control is also challenging in WSN because the nodes have to setup ad-hoc communication structures (such as trees or clusters) and maintain them each time the network dynamics change. As illustrated in Figure 1.4, a distributed algorithm is needed to identify the clusters and construct each cluster and maintain it.

It is worth noting here that communication is the most dominant energy consumption factor [4] among all the other components as illustrated in Table 1.7 and will be explained in subsection 1.2.

Applications' Requirements. WSN is an application-oriented technology where hardware and software solutions are specific to unique application. Due to the limited storage capacity of the nodes and for optimization of the transmission, aggregation is a widely applicable technique in WSN. However, the local aggregation of data can be challenging because the nodes have to define the limits of the *aggregation zone* in order to reduce the need for communications and thus to make it more energy-efficient. In addition, once the WSN is deployed and for reasons of autonomy and dependability, the lifetime should be as long as pos-

Table 1.7: Power requirements for different operations in mica motes.

Operation	Power (nAh)
30-bytes packet transmission	**20000**
30-bytes packet reception	**8000**
1 ms radio listening	**1250**
Sensor analog sample	**1080**
Sensor digital sample	**0.347**
Reading sample from ADC	**0.011**
Flash read data	**1.111**
4-bytes write/erase data	**83.333**

sible, and the scalability and nodes' heterogeneity may add more constraints to WSN-based applications.

Besides, even with redundant and correlated readings, data fusion remains challenging because readings are inaccurate due to the hardware differences. The requirements of applications vary depending on many QoS parameters such as: the coverage, the degree of exposure, the number of sensors to be selected for data collection, the accuracy and the sensitivity of data, etc.

Security and reliability. Security is essential for any mission-critical system as autonomous and unattended as WSN. The information in military or health-care WSN-based applications *must* be confidential. However, the nature itself of WSN exposes the system to various risks in security such as malicious intrusion, denial of service, etc. The wireless communications make it easy for an adversary to eavesdrop on sensor transmissions and put the system out of service using jamming. Therefore, security threats in WSN are more challenging than in other systems given the constraints of resources of sensor nodes, which require new solutions for key establishment and distribution, node authentication, etc.

Since WSN are supposed to operate without human intervention and nodes are exposed to both system dynamics and environmental dynamics, reliability becomes a significant challenge. Nodes have to be self-managing devices to monitor their surroundings, adapt to changes in the environment and cooperate with their neighbors to self-organize the network. For example, a node can choose its transmission power to maintain a certain degree of connectivity to prevent the occurrence of voids in the networks. Moreover, the ability to self-heal allows sensor nodes to discover, identify, and react to network disruptions and achieve *fault-tolerance*. All these *non-functional* features should be implemented without incurring excessive energy and communication overheads [30].

1.2 The Energy Problem

In this section, we describe in depth the energy problem to understand the problematic addressed in this thesis. We present the layers of the protocol stack of sensor nodes and the factors that may effect energy consumption, and the behavior of each layer to deal with this issue.

1.2.1 Protocol Stack

As shown in Figure 1.5, there are typically five layers in the protocol stack [4].

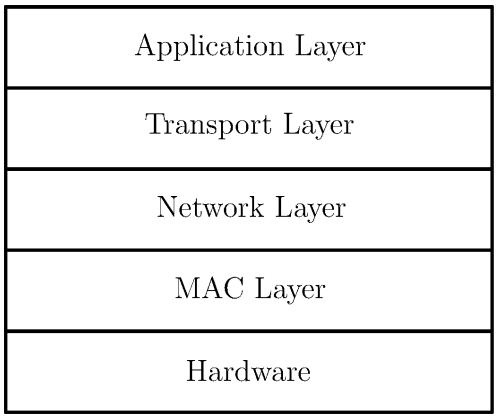


Figure 1.5: Stack protocols of sensor nodes.

Physical Layer (or the Hardware). It is responsible for Point-to-Point wireless digital communications. It is composed of the transmitter, the channel and the receiver. It executes the operations of generating the analog baseband message signal, its conversion into discrete signal (both in time and amplitude) in order to be processed by the processor[2]. Then, the discrete signal is converted into binary stream using *source encoding* techniques. This operation should satisfy bandwidth and power requirements. After that, the physical layer executes the channel encoding to make the signal robust to noise and interferences. After the channel encoding, the modulation takes place. It transforms the baseband signal to a passband signal. It is useful when the nodes have short antennas. At the receiver side, the reverse process is executed.

The choice of modulation scheme and transmission rate further affects the resources and the energy requirements of the sensor nodes. A *power* efficient modulator enables a communication system to reliably transmit information at the

[2]This operation requires sampling at Nyquist rate, so that the information will not be lost.

Table 1.8: Power requirements of different radio interfaces.

Operation	RFM TR1000	RFM TR3000	MC12302	CC1000	CC2420
Date Rate (kbps)	115.2	115.2	250	76.8	250
Transmit Current (mA)	12	7.5	35	16.5	17.4
Receive Current (mA)	3.8	3.8	42	9.6	18.8
Idle Current (mA)	3.8	3.8	0.8	9.6	18.8
Standby Current (μA)	0.7	0.7	102	96	426

lowest practical power cost. A *spectrally* efficient modulator enables a communication system to send as many bits of data as possible within a limited bandwidth. Power and spectrum efficiency cannot be achieved at the same time.

In WSN, since nodes produce small-volume of data, the power efficiency is a major concern while the bandwidth is not. For example, it is not allowed to use *over-emitting*, i.e. using larger transmit powers than necessary because it is a major contributor to excessive energy consumption.

MAC Layer. The MAC protocol is responsible for regulating the access to the common medium. Most sensing applications rely on radio transmission in the unlicensed ISM band which may affects significantly communications by the noise and interference. The choice of the MAC protocol has a direct impact on the reliability and efficiency of network transmissions. Other issues have to be faced by the MAC protocol include signal fading, simultaneous medium access by multiple nodes, asymmetric links, etc.

In the MAC protocol, energy is not only consumed for transmission and reception, but also for sensing the medium for activity, data packets retransmissions (due to collisions), packet overheads, control packet transmissions, and transmit at power levels that are higher than necessary to reach a receiver. Table 1.8 shows and compares the energy consumption of different radio interfaces [30]. As we can see on this table, there is a huge difference between TR family and CC family transceivers in terms of data rate and power requirements ($0.7\mu A$ of standby current in TR1000 and $426\mu A$ in CC2420). This is due to the difference of applications in which TR and CC families can be used.

It is common for a MAC protocol in WSN to trade between energy efficiency for increased latency or reduced throughput or fairness.

A common technique used to preserve energy is DPM (Dynamic Power Management), where a resource can be moved between different operational modes such as active, idle and sleep. Without power management, most transceivers switch between transmit, receive and idle modes, although idle and receive modes are typically similar in their power consumption. However, dramatic energy savings can be obtained by putting the device in *low-power sleep* mode. Moreover, using periodic traffic models, the network application can benefit from the MAC

schemes that do not require nodes to be active at all times. Instead, they allow nodes to obtain periodic access to the medium for a very small fraction of time called *duty cycle* and they return to the low-power sleep mode.

In addition to idle listening, the overheads are also caused by inefficient MAC protocol designs such as: large packet headers, reliability requirements (such as collisions requiring retransmissions, error control mechanisms, etc.), control messages (to address the *hidden-terminal problem*), etc.

Network Layer. The network layer is responsible for establishing paths from a source to a sink across one or more relays. When nodes are scattered into an environment randomly, the resulting topology is non-uniform and unpredictable. Thus, it is essential for the nodes to cooperate to determine their positions, identify their neighbors, and discover the paths to the sink.

The most crucial aspect in routing is energy-efficiency with many metrics that can be applied with various interpretations such as: minimum energy consumed per packet, maximum time to network partition, minimum variance in node power level, maximum average energy capacity, and maximum minimum energy capacity. Each formulation of energy-awareness lead to very different routing protocols that differ in their results and overheads.

Transport Layer. The transport layer is responsible for managing end-to-end connections using reliable stream-based communication (TCP or TCP-like protocols), or unreliable packet-based communication (UDP or UDP-like protocols). The use of one of the two protocols or both of them depends on the application, and has an impact on the energy consumption.

A reliable transport protocol maintains state information in each node on the path from the source to the destination which incurs more energy consumption. Energy is also excessively consumed when retransmissions of data packets are needed due to packet loss. That is why for a reliable transport protocol, to be energy-efficient, should deal with mobility of nodes as inherent characteristic not as an exception.

Unreliable transport protocol is suitable for multimedia WSN where the packet sequence is insignificant and the applications tolerate data loss. It does not maintain any information about the data transfer session.

Application Layer. The application layer gives the user a view of the WSN as a database that he can interrogate via queries. It hides all the details and complexities of the network and/or the node. Based on that, there are two types of application protocols: *node-centric* protocols and *application-centric* protocols.

In node-centric protocols, the overall network-wide sensing application is described as a collection of pairwise interaction of individual nodes. At the op-

posite, the application-centric protocols focus on programming groups of sensor nodes and treat the entire network as a single entity.

The application protocols should deal with the frequent dynamic changes such that the WSN continues to serve its intended purpose even when some parts of the network fails. It should hide many faults due to topology changes, link errors, void appearance, etc. In addition, it should provide support for self-management and self-configuration to make WSN scalable.

One WSN-specific concept to deal with all of these problems is *data-centric* applications. Since in WSN not only individual nodes are of interest but also the data that they generate and disseminate, collaboration is needed for some applications to analyze and process data *in-network*. At the opposite of other applications that are concerned only by collecting data at a central point, data-centric applications require to early identify if data should be propagated further or acted upon. This could help optimizing energy at the application layer.

1.2.2 Collaborative Applications

In this thesis, we address the problem of energy-efficiency at the application layer by exploiting the intrinsic characteristics of WSN such as: locality, distribution, data correlation and density. We believe that collaboration between nodes helps increasing the quality of data while reducing transmissions. For some applications such as target tracking, selective activation of some nodes and putting the other nodes in low-power operating mode enable network lifetime extension.

Therefore, the prediction of events and the exploitation of data and their correlation in collaborative applications can be of interest in optimizing the energy in WSN. That is what makes the subject of our thesis.

1.2.3 Our Contributions

Our contributions to tackle the above-described problems are four:

Contribution 1: is a survey of recent energy-efficient target tracking schemes proposed in the literature. We propose a *novel classification* of these schemes based on the interaction between the communication subsystem and the sensing subsystem in a sensor node. We are interested in collaborative tracking systems instead of single-node tracking systems.

We show that energy-efficiency in a collaborative target tracking scheme can be achieved via two classes of methods: sensing-related methods and communication-related methods. We illustrate both of them with several examples. We show also that these two classes can be linked to each other through a prediction algorithm to optimize the communication and the sensing operations.

In addition to this general classification, we discuss also a particular classification of certain protocols that put specific assumptions on the target nature and/or use *non-standard* hardware for sensing. In the end, we make a theoretical comparison between all these schemes in terms of objectives and mechanisms. Finally, we give some recommendations to assist the design of energy efficient target tracking schemes in WSN.

Contribution 2: is a target tracking scheme called DKF_DC (Distributed Kalman Filter with Dynamic Clustering). It exploits the predictions generated by the DKF algorithm to select the most appropriate nodes for the task. After selection, it constructs dynamic clusters that changes each time the target state changes.

Our proposed scheme exploits the fact that the presence of the target is a localized event. Therefore, to reduce energy consumption, we consider WSN with nodes of limited sensing range.

The simulation data show a clear improvement of network energy consumption, but the quality of estimation is slightly degraded compared to centralized approaches and other tracking schemes with limited sensing range that do not limit the set of involved nodes.

Our tracking scheme reduces the number of tracking nodes which reduces the network energy consumption.

Contribution 3: is a generalization of contribution 2 to any collaborative application that uses geographic-based clustering for data collection. We propose a Binary Integer Linear Programming (BILP) model for the problem of Energy Minimization under the constraint of Data Precision in the context of correlated data collection in Wireless Sensor Networks, called EMDP.

The exact solution of our BILP model determines, in each round of data collection, the role of each node in terms of sensing, data relaying and processing. It gives the baseline for optimal network operations and helps characterizing the complexity of EMDP problem.

Contribution 4: is a Correlation-Based Adaptive Clustering Algorithm called CORAD. Based on data correlations, CORAD reconfigures the network topology each time the dynamics of the events change.

Depending on the nature of the physical phenomenon, CORAD defines the structure of the network as a set of multi-hop reconfigurable clusters. Its goal is to avoid energy wasting when nodes report useless data that contribute very little in the improvement of the quality of data required by the end-user.

CORAD achieves also a certain level of data accuracy by selecting appropriate nodes for sampling: i.e. the nodes with the maximum energy resources and that are the closest to the physical phenomenon. It reconfigures the created clusters

based on the current data accuracy and the set of participating nodes in sampling and data reporting operations. CORAD depends on several parameters whose sensitivity is analyzed via simulation.

The simulation results support our different proposed models for detection, data correlation and clustering. The performance evaluation demonstrates that CORAD performs well in presence of highly dynamic events and extends considerably the network lifetime with balanced distribution of energy consumption.

1.3 Research Publications

Our thesis yielded four completed publications and communications and one ongoing publication listed below:

1. O. Demigha and W. Hidouci. *Energy Reduction Techniques for Collaborative Target Tracking in Wireless Sensor Networks*. Proc. 7ème Conférence sur le Génie Electrique (CGE'07). EMP, April, 12-13, 2011. Algiers. ALGERIA.

2. O. Demigha, H. Ould Slimane, A. Bouziani and W. Hidouci. *Energy efficient target tracking in wireless sensor networks with limited sensing range*. Proc. The 6th International Conference on Systems and Networks Communications (ICSNC 11). 23-28 Oct. 2011. Barcelona SPAIN.

3. O. Demigha, W. Hidouci and T. Ahmed. *On energy-efficiency in collaborative target tracking in wireless sensor networks : a review.* IEEE Communications Surveys and Tutorials journal. Vol 15, No 3. Pages 1210-1222, 2013.

4. O. Demigha, W. Hidouci and T. Ahmed. *A Novel BILP Model for Energy Optimization under Data Precision Constraints in Wireless Sensor Networks.* IEEE Communications Letters. Vol. 18, Issue 12, Pages 2185 - 2188. 2014.

5. O. Demigha, T. Ahmed and W. Hidouci. *Energy-Efficient Data Collection under Data Constraints in Wireless Sensor Networks.* Submitted to International Journal of Sensor Networks, Inderscience Publishers (Awaiting Reviewer Assignment).

1.4 Manuscript Organization

The thesis manuscript is composed of six (06) chapters: Chapter 1 and Chapter 6 are respectively the introduction and the conclusion, and the four remaining chapters are devoted to the body of our four contributions. In addition, two appendices are added at the end of the manuscript to help the reader understanding related concepts: Appendix A for the Kalman Filter and Appendix B for the Gnu Linear Programming Kit.

Chapter 2

Energy Efficient Collaborative Target Tracking: A Survey

THANKS TO THEIR ADVANTAGES in terms of cost-efficiency, WSN can be considered as an alternative technology to expensive conventional tracking technologies such as RADARS. As, they are deployed without any centralized or pre-installed infrastructure, its operating mode is easy compared to RADARS [31].

When sensor nodes are deployed with high density in the surveillance area, they allow to setup collaborative data processing in contrast to single-node tracking systems. These nodes exchange information between each other and cooperate to report the sensed data to a special node called the *sink* or the base station for post-processing.

However, in target tracking applications, energy is a challenging issue because of the limited capacities of sensor nodes. As we have explained in Chapter 1, the sensor nodes are equipped with low-cost small-capacity batteries which are, in most cases, non-rechargeable and irreplaceable, and their sensing, processing and communication capacities are minimal. In addition, the target tracking application is aimed to meet the same performance as the conventional technologies in terms of time and accuracy.

Furthermore, the measurements sent by the nodes to the sink are in most cases redundant, noisy and non-synchronized, and the inter-node communication is an energy-consuming task. Hence, neither reliable communication protocols nor complex data processing algorithms can be implemented on a sensor node because of its limited processing and communication capacities.

From this point of view, energy conservation in target tracking applications should be *data-centric* and can be achieved via collaboration. Different methods are proposed in the literature [32–34] and the prediction coupled with selective activation of nodes is one of such methods.

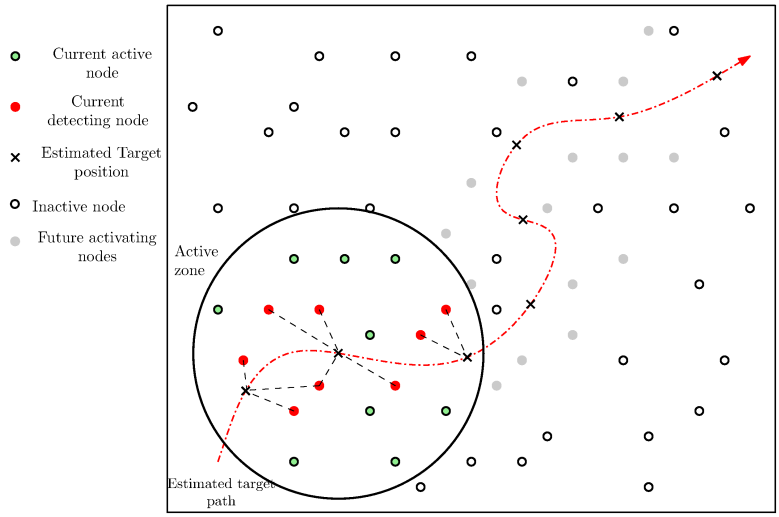


Figure 2.1: Example of a prediction-based scheme.

Figure 2.1 shows an example of target tracking scenarios in event-driven WSN, where nodes are woken-up *explicitly on-demand* following the target trajectory. The previous active nodes predict the activation zone to which the target will probably go and a subset of nodes within this zone will be explicitly activated. These active nodes collaborate between each other to generate an accurate estimation of the target state using *in-network light-weight* data fusion algorithms such as the Kalman Filter. The gain of such algorithms is twofold: (i) it generates state-estimates of the target, and (ii) it produces state-predictions for the next sensing round.

Therefore, these predictions are the basis for node selection which leads to the problem called the *Sensor Selection Problem (SSP)* [35]. Basically, this problem consists in finding the subset of nodes with minimum cost that provide the information with maximum utility among all the nodes of the network. In the case of target tracking problem, the cost and the information utility can be respectively defined by the energy consumption and the accuracy of data [36].

Another technique for energy conservation is to define a schedule for activation and/or deactivation of nodes. Depending on the target trajectory predicted by the prediction algorithm, nodes are *woken-up* to perform some sensing and communication tasks for a very short period of time, and then they return back to a *deep-sleep* state. Other nodes that are not involved in the tracking task are put in the sleep state to preserve their energy resources. However, the schedule should not miss the target as it passes through the sensing zones of the nodes and it should concern only the nodes with the maximum energy resources.

This procedure requires collaboration between the nodes and coordination between the communication-related and the sensing-related operations because

of several reasons. First, the sensing measurements are redundant and noisy, and the multi-node target detection in contrast to single-node detection generates correlated measurements that can be fused. Second, the communication links are lossy which can be overcome by using collaborative protocols to deal with the loss of messages. And finally, the dense nature of the WSN requires self-organization to reduce energy consumption.

In this chapter, we draw a taxonomy of some recent proposed approaches that aim to achieve all the above-described requirements by organizing the network in clusters and/or trees. Our goal is to extract general recommendations to design an energy-efficient target tracking scheme.

Nonetheless, we do not consider MAC-layer mechanisms such as *duty-cycling* [37] because they are out of scope of this work.

2.1 Schemes Classification

Before presenting our proposed taxonomy, we first give some definitions that help characterizing a typical target tracking scheme [38].

2.1.1 Target Tracking Schemes Characterization

Typically, a target tracking scheme consists of three subsystems, namely: the sensing subsystem, the estimation/prediction subsystem, and the communication subsystem. Figure 2.2 shows the relationship between the sensing subsystem and the communication subsystem. The estimation/prediction scheme extracts useful information from the heterogeneous and redundant data transmitted by active nodes. It uses this information to extrapolate the position of the target in the future and then organizes the network to follow-up the target trajectory. The communication subsystem creates and maintains the clusters and/or the trees structures to make the communications more efficient. Besides, the prediction algorithm predicts the state of the target for the next sensing rounds on the basis of which, the instants and the durations of the sensing operations can be optimized.

In our classification, we assume that all the nodes should be initially in a sleep state, except the ones that are on the borders of the surveillance area. These nodes perform the first operations of detection and identification of the target, and then activate other nodes via *external activation* messages transmitted over a low-power channel called the *paging channel*.

The tracking process is generally divided into successive rounds whose durations are constant or variable depending on the estimation/prediction algorithm. In each round, the activation message is disseminated in a zone called the *activation zone* whose range depends on the estimated velocity of the target and the measurements' error in the current tracking round.

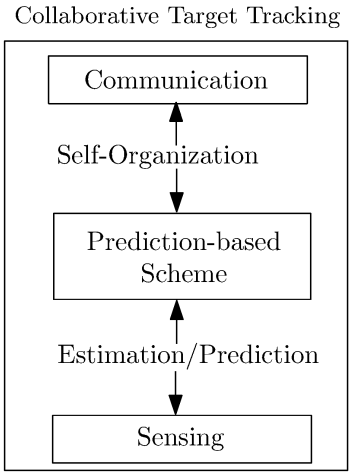


Figure 2.2: Target tracking scheme components.

After initialization of the network, the estimation/prediction algorithm generates a reliable estimation of the target state for the current tracking round, and one or more predictions for the next rounds. If the target has a dynamic behavior during the current tracking round, then a cluster and/or a tree reorganization is triggered to follow-up the target trajectory. It is the task of the current leader and/or the current root to generate state estimations of the target and report data to the sink.

For designing an energy-efficient target tracking scheme, the following elements should be considered:

Quality of Detection. According to the network coverage ratio (which is related to the initial deployment, the sensing range, the network density, etc.), the target can be *detected* by one or many nodes, thereby generating correlated measurements. A tracking scheme should be able to decide which nodes should be selected for the next round? How long should be the diameter of the activation zone? How many nodes should be selected? etc., in order to obtain the required quality of detection which helps computing the current estimation error.

Estimation/Prediction Algorithm. The prediction algorithm should be *distributed* and *light-weight* depending on the state model equation of the target (linear or non-linear), the noise model of the sensor readings (Gaussian, non-Gaussian), the nature of data (mono-modal, multi-modal), etc. As sensor nodes have limited resources, the Kalman Filter algorithm (KF) [39] is widely used for estimation/prediction because its distributed variants such as the Distributed Kalman Filter (DKF) [40] and the Kalman Consensus Filter (KCF) [41] are easy to implement for linear systems.

However, for non-linear systems, more sophisticated data filtering algorithms are needed, such as: the Particle Filter (PF) [42], the Variational Filter (VF) [43], the Extended Kalman Filter (EKF) [44], the Unscented Kalman Filter (UKF) [45], etc.

Data reporting mechanism. After the estimation and prediction of the target state, the target tracking scheme have to choose the nodes that report data to the sink. Ideally, when connectivity is ensured, the nodes that are close to the target with the maximum energy resources should be selected. However, the reconfiguration of the network may lead to a situation where the reporter node is far away from the sink and/or the target. In this case, the selection of *backup* reporter nodes or the establishment of a hybrid (static/dynamic) network structure can be applied.

Activation mechanism. The activation range depends on the target velocity. To avoid the loss of the target, a multi-step activation mechanism with variable activation range can be applied. The activation plan can be static (pre-established at the beginning of the tracking process) or dynamic, according to the current estimated state and the measurements' error.

Logical network structure. To optimize the communications, a flat network structure is not the better solution. By constructing temporary clusters and/or trees, data fusion can be *localized* within few number of nodes. However, the target tracking scheme should deal with some issues due to dynamic changes such as: leader election, cluster/tree reconfiguration, clusters boundary determination, etc.

2.1.2 General Classification

Figure 2.3 shows our proposed classification which relies on two complementary aspects namely: the *sensing-related* aspect and the *communication-related* aspect.

In this general classification, we divide the sensing-related approaches into two subclasses: *Single-node Signal Processing* approaches (SP) and *Collaborative Signal Processing* approaches (CSP). The first class is out of scope of our work and the second class is split into two other subclasses, namely: *information-driven* subclass and *data filtering* subclass. The difference between the two subclasses is that the first exploits the data content to optimize the future readings, however the second generates accurate information from noisy readings.

We also split the communication-related approaches into two subclasses, namely: the *Routing/Aggregation* approaches and the *Network Self-organization* approaches.

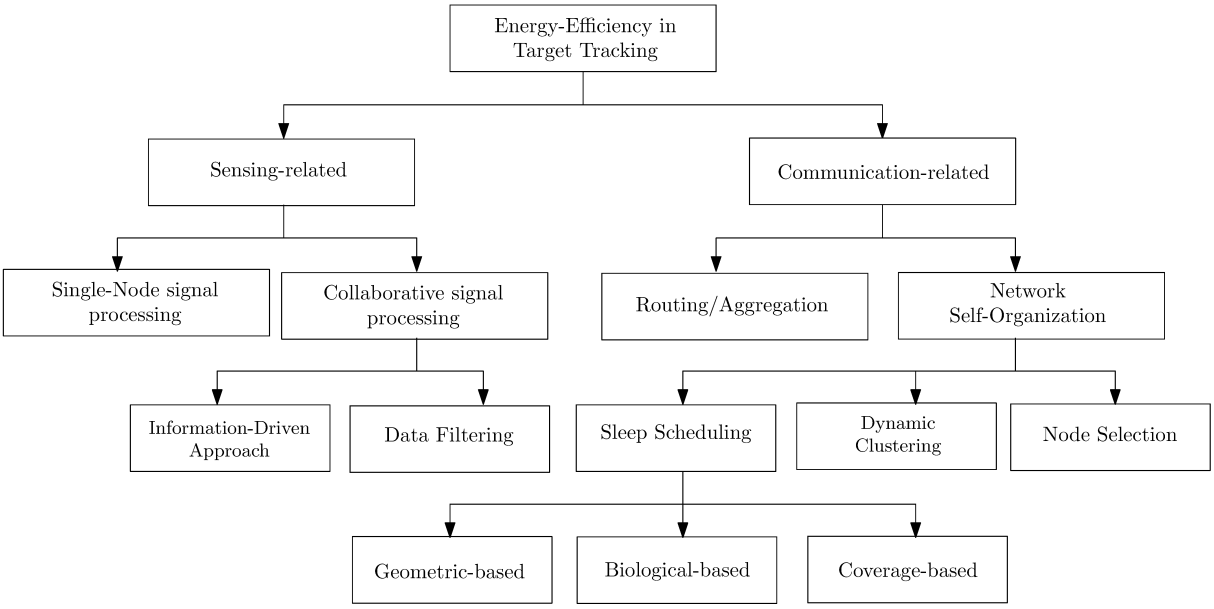


Figure 2.3: General classification.

As routing and aggregation techniques are common to all WSN-based applications, and they are well surveyed in the literature [46], we omit the first subclass and we detail only the second one which is directly related to the applications of target tracking. We divide it into three subclasses, namely: the *node selection* subclass, the *sleep scheduling* subclass, and the *dynamic clustering* subclass.

In node selection techniques, the resources of the nodes and their probability of target detection are estimated. Based on that, the contribution of each node in the target *belief state* [47] is computed, and the algorithm decides which node to activate and which one to put in the sleep state.

Since the sleep scheduling subclass is more important, we divided it again into three subclasses which are: *geometric-based* subclass, *biological-based* subclass and *coverage-based* subclass. These subclasses differ between each others with respect to the strategy of each subclass to activate and deactivate the nodes. An in-depth description of each approach is given in Section 2.2 and Section 2.3.

It is worth noting that it exists a previously proposed approach that merges Distributed Predictive Tracking (DPT) with regional Collaborative Signal Processing (CSP) and uses them alternately to track a group of objects. This approach is proposed in [48] and it uses the same concepts that rule our classification.

2.2 Collaborative Signal Processing Methods

In collaborative signal processing techniques (called also distributed in-network data processing techniques), instead of sending data to the sink node to be processed by the end-user application, sensor nodes collaborate between each others to retrieve the required information. They decide about which data to deliver and which one to aggregate, or to compress or to suppress (drop). The goal here is to optimize the network communications and to reduce the number of nodes involved in the tracking process as well as the volume of messages exchanged between them. That is what we refer to by *data-centric* approach. In the following subsections, we present two different techniques which are: the information-driven techniques and data filtering techniques.

2.2.1 Information-Driven Techniques

To the best of our knowledge, Information-Driven Sensor Querying (IDSQ) technique has been first proposed in [47]. The basic idea behind it is to explore the content of the data captured by the sensor nodes to optimize the future readings. Specifically, it aims to determine which sensor should take the measurements, and to whom it should send them. IDSQ requires collaboration among sensor nodes because the targets may have sparse spatial-temporal distributions. Hence, the target tracking process in IDSQ can be seen as a *sequential Bayesian estimation problem* in which we can use different measures of the information utility such as: the *Mahalanobis distance* and the *entropy-based* utility measure.

When we use the entropy-based utility measure [49], the activation mechanism selects an informative sensor such that the fusion of the selected sensor observation with the prior target location distribution would yield, on average, the greatest or nearly the greatest reduction in the *entropy* of the target location distribution.

Therefore, the problem raised in [49] is how to efficiently evaluate the expected information gain attributable to each candidate node to selection without actually retrieving sensor data. The authors define the entropy-based heuristic to measure the quality of detection using the sensor's view about the target location (which is the geometric projection of the target location onto that of the sensor's observation perspective). This metric is a function of both target location and sensor location, and it is simpler than mutual-information method [50]. The main difference between the the two methods proposed in [49] and in [51] is that, the former involves only one sensor, while the later selects many sensors.

The Distributed Kalman Filter (DKF) with information-driven estimation algorithm is proposed in [52] by Olfati-Saber. He showed that the common objective of improving individual information value of the sensors would force to perform an unplanned moving rendezvous near the mobile target. Collision avoid-

ance between agents leads to a flocking behavior. He proposed a metric that measures the information value similar to the *Fisher Information* [53]. He showed also that adding the agent-target interaction to the flocking algorithm [54, 55] is a way of taking the information value of sensor measurements into account in motion planning of agents toward the target.

Another problem related to computational efficiency in information-driven methods is when we have to activate nodes with maximum information-utility. This problem can be non-trivial because the function of the next-step error covariance matrix can be *non-convex*. The authors in [56] proposed to minimize the trace of the next step error covariance matrix to find the maximum of the function. They proposed a relaxation approach that searches for the computationally feasible sub-optimal solution. Thus, the trace of the next step error covariance matrix becomes a convex function and its minimum can be easily computed using convex optimization.

2.2.2 Data Filtering Techniques

With respect to the constraints of sensor nodes in terms of computation and communication, light-weight versions of classical filtering algorithms have been proposed. For example, in [41] the Kalman Consensus Filter (KCF) algorithm is proposed for nodes with limited sensing range. In this kind of network, not all the nodes of the network can observe the target but only a subset of them. The authors proposed a consensus-based filtering algorithm implemented over a logical P2P network of micro-filters. Each micro-filter is a *local estimator*. A high-level fusion center aggregates the local state-estimates and the error covariance matrix of each micro-filter. The goal of KCF algorithm is to reach for a consensus on estimates obtained by local KFs rather than distributing the construction of the fused measurements and the covariance information of the central KF, which is complex. The fusion center in this case (KCF) does not receive a large amount of data because of the hybrid architecture.

The extended variant of the Kalman Filter (EKF) can also be used as an estimation algorithm in cluster-based tracking schemes such as in [57]. The tracking process is run as successive selections of nodes. To select a node, the leader needs to know its *target detection probability* (which can be deduced from the target state equations) and then it computes the *Joint Detection Probability of all detecting sensors*. Unfortunately, this process is very complex and requires a Monte-Carlo simulation method. The authors propose a greedy approach in which, the sensors with high detection probability are selected at first.

Another aspect of EKF when it is used as an estimation/prediction algorithm is the problem of the *Inter-Sensor Interferences* problem (ISI). This problem appears with active sensors that track non-collaborative targets [58]. To resolve this problem, a time-division distributed technique is proposed in which each sensor

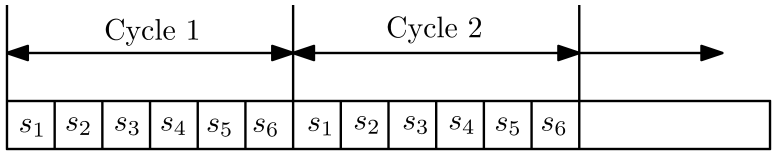


Figure 2.4: Illustration of periodic scheduling.

senses the target alternatively within a predefined number of slots.

The first variant of this technique, called the *periodic scheduling*, is based on the division of the time into periodic cycles each of which is assigned to a sensor (see Figure 2.4). If the scheduled sensor detects the target in its time slot, it computes the difference between its time of measurements and the previous time step and fuses its measurements with the existing target estimation using EKF. Finding the minimum number of time-slots can be modeled as a *graph coloring problem* which is known as NP-Complete.

The second variant is called the *adaptive scheduling*. Its goal is to eliminate empty slots by scheduling the next tasking node for the next time step according to the predicted tracking accuracy derived from the trace of the covariance matrix of state estimation using Unscented Kalman Filter (UKF). Figure 2.5 illustrates the problem of empty slots (nodes s_2, s_3 and s_4 have empty detections because the target passes far from their sensing zones).

EKF can also be used by the current cluster-head to estimate the distance to the target. This technique has been proposed in [59] where detecting nodes send their measurements to a selected cluster-head which in turn sends a command message to the second nearest sensor to get initial coordinates of the target. After that, it executes the Least Square Estimation method to obtain a *good* estimation

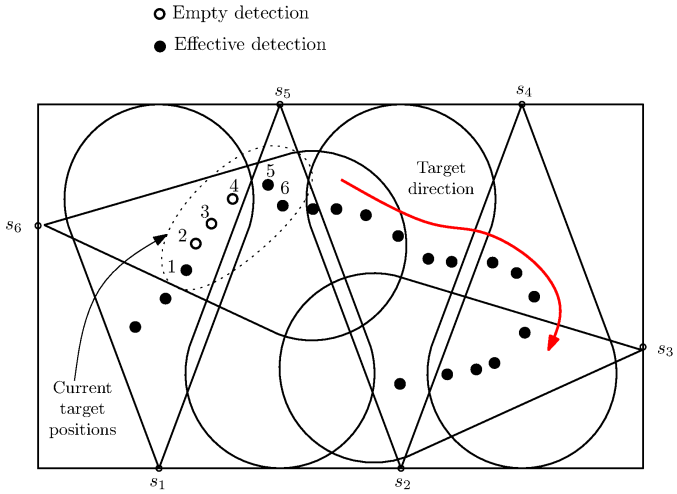


Figure 2.5: Effect of empty slots.

of the target position. The EKF algorithm is then triggered to estimate the current and the next target state. The target model in this scheme is Position-Velocity (PV) with 4 sensor distance measurements. The sampling period (δT) is computed according to the target velocity.

It is worth to note that when we use EKF as an estimation/prediction algorithm, it is important to find the most energy-efficient logical network topology (ex. a tree) that satisfies state estimation constraints. This problem is addressed in [60] and it is shown that choosing the tree with the minimum energy consumption is very difficult. Thereby, the authors propose a tree reconfiguration algorithm composed of three procedures: (1) a recursive tree initialization procedure that uses the minimum power transmission to establish connections of each node with its immediate neighbors, (2) a switching tree topology procedure that is triggered when the desired quality is not achieved. It transforms some two-hop neighbors to one-hop neighbors, and (3) a minimum energy subtree procedure that finds all the possible subtrees satisfying the required quality. It returns the tree with the minimum overall energy cost.

In [60], the authors show that the minimization of the overall energy consumption may not lead to the maximization of the network lifetime. Thus, a tree scheduling algorithm that chooses M trees from N^{N-2} possible trees is necessary[1]. A linear programming solution has been proposed to search for that M trees, efficiently.

In all the above-described schemes, the main objective of collaborative signal processing methods is to measure the data quality delivered by the nodes to choose the most appropriate ones for activation. They focus on reducing the noise of measurements and predicting the target behavior in the future without paying attention to the network structure to reduce energy consumption in communications. Thereby, we address this complementary aspect in the next section.

2.3 Network Self-Organization Approaches

The objective of the network self-organization approaches is to extend the network lifetime by eliminating unnecessary wakening of the nodes, by planning rigorously their sleep state and adapting the network topology to the changes of the target dynamic. In the following subsections, we describe three subclasses that fit in these approaches.

2.3.1 Sleep Scheduling

The first subclass of the network self-organization methods is sleep scheduling. It focuses on planning the sleep state of the nodes along with the trajectory of the

[1]Optimal scheduling of such trees is also known to be an NP-Complete problem.

target. As the presence of the target in the surveillance area is a *localized event* (because of the limited observability of the target, the limited sensing range of nodes, and the attenuation of the energy emitted by the target), only a subset of nodes located close to the target should be activated. The others should stay inactive until they receive an explicit activation message from the previous tracking nodes. It indicates that the target is *probably* in their sensing ranges and thus they can sense it. The predictability of the target trajectory helps to determine, with a sufficient degree of confidence, which nodes to wake-up? At which instant? And for which duration? From this standpoint, the sleep scheduling (or sleep planning) approaches should consider certain constraints such as: the coverage and the connectivity of the network, the network lifetime, the detection reliability, the accuracy of tracking, etc.

In what follows, we present three subclasses of sleep scheduling methods, namely: biological-based methods, geometric-based methods and coverage-based methods.

Biological-Based Approaches

Biological-based approaches use biological-inspired concepts such as insect communities, gene programming, ant colony, etc. [61, 62].

In [61], the authors abstract the target as a virtual chemical emitter. They propose to construct contours of influence around the target whose strength decreases with the distance from the target. Then, they select the sampling period based on meta-data of the target using its *net traveled distance* (its past behavior). When the target enters the surveillance region, it is detected by the border nodes. After that, a group of nodes is pro-actively selected and assigned as Main Node (MN) or Helper Node (HN) [2].

In [62], the concept of Parallel Gene Expression Programming (P-GEP) is used to schedule the sleeping state of the nodes. The target trajectory is modeled as a piecewise function divided into different shorter portions. This function is unknown before the target appears, but it can be extracted. The future positions can be predicted from the past locations. P-GEP includes a distance-based lightweight localization algorithm to estimate the current target position. During the trajectory mining process, some past location information are unnecessary, so they can be discarded using a *sliding window* mechanism whose size determines how many previous information is needed according to the prediction accuracy. This later is measured using the distance between the prediction position and the actual position. Given the prediction accuracy, an upper-bound and a lower-bound trajectories are computed. A fitness function is proposed to evaluate individuals in P-GEP: the higher is the function value, the better is the individual (the

[2]The MN selects the next tracking group based on the predicted target state and the node's *centrality* which should be the largest.

the prediction error is low). The node scheduling algorithm is based on a single-step or multi-step prediction model that uses the trajectory found to determine which nodes to wakeup at time t_{i+j} from the historical information up to time t_i.

Coverage-Based Approaches

The goal of coverage-based approaches is to preserve the requirements of event coverage while running the sleeping plane of the nodes. Different problems arise.

First, the problem of determining the optimal length of the activation period in the sleep schedule of the *Controlled Greedy Sleep* algorithm (CGS) [63]. It is addressed in [64] and modeled as a bi-parti graph whose properties determine the static and the dynamic *k-coverage* requirements. The basic idea behind this technique is that each node can estimate the number of neighbors that will benefit from its duty period. Based on this estimation, it decides to become active or not. The authors recommend to consider a short period of activation for dynamic networks.

Second, the problem of *off-duty* eligibility rules, which is addressed in [65]. The objective is to identify the redundant nodes to put them in off-duty mode without using any location information. The basic idea is that before scheduling the sleeping state of the nodes, the user application specifies the desired ratio of coverage loss. Then, the corresponding threshold is determined using a given mathematical expression or using the collected data.

Last, the multi-node event watching problem i.e. TEKWEM[3] which is formulated in [66]. The authors propose an algorithm that finds the sets of detection nodes that satisfy the warning delivery delay and the network lifetime constraints. The algorithm uses a color-based method to construct all the *Breath-First-Search Trees* (BFST) each of which is rooted at a gateway. Each tree corresponds to a detection set. The gateway adds greedily the sensors whose sensing components can help to k-monitor the atomic events, into the detection set. It can also add useless nodes for connectivity purposes. After that, the gateway builds the working schedule then broadcast it to all the sensor nodes. Multiple selection heuristics can be proposed in this context.

Geometric-Based Approaches

The geometric-based approaches use computational geometry to construct application-oriented network topology. They profit from the location information of the nodes to optimize the localization and tracking of the target. In this subsection, we describe three different schemes, each one of them uses a different geometric concept.

[3]Timely Energy-Efficient k-Watching Event Monitoring.

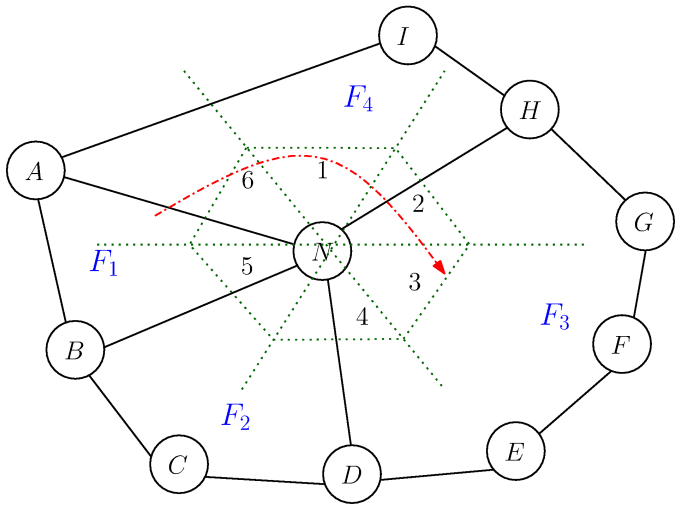


Figure 2.6: Example of FOTP operations.

Face-based Object Tracking (FOTP) scheme [67] uses a face-based architecture with a *hexagon* algorithm for prediction. It achieves energy efficiency by reducing the number of active faces and the number of waking nodes. A face is defined as "the subdivision of the maximal connected subset of the plane that does not contain a point on an edge or a node" [67]. The main idea behind FOTP is as follows:

First, some active nodes called "soldiers" detect the target then wake-up all the nodes in the face by which the object entered. In Figure 2.6, the target enters by face F_1 and nodes A, B and N are woken-up to detect it. The nodes in the current face can estimate the distance to the target. After that, they select the nearest node (NN) to the target as a leader. In the example of Figure 2.6, node N is the leader. The NN determines by which edge the target enters and obtains its last position. Then, it computes its speed and direction. The current NN determines the next edge to which the target is going through. In Figure 2.6 it is the edge $N - H$. After that, it selects the next NN from the set of nodes in the adjacent faces that intersects with the predicted edge. In Figure 2.6, the next NN is node F. If the next NN cannot detect the target in *its time slot*, it sends a *loss* message to the last NN which will activate all its adjacent faces. If the target is not detected again, this last NN sends a message back to the base station.

Another concept used in geometric-based schemes is *planar graphs*. A typical scheme that demonstrates this concept is Polygon-based Target Tracking scheme (PTT) [68], where a measure of the information-utility based on the Cramer-Rao Lower Bound (CRLB)[4] of the variance is used. The objective here is to minimize the number of nodes that participate in the tracking process. The algorithm is

[4]CRLB is the inverse of Fisher Information Matrix.

based on the *brink construction* procedure: it determines the critical region by connecting an edge called the *brink* to the active polygon. This critical region helps to confirm if the target is leaving the current polygon and entering another one, or not. The PTT scheme uses also a node selection procedure based on both the information-usefulness and the energy cost: before the target crosses the brink, a control message which contains the target state estimation is sent to the nodes in the forwarding polygon. The receiving nodes combine their measures with the received estimation to compute their weights. Each node can locally decides whether it should join the tracking operation or not, and thereby it determines its couple nodes.

Finally, *Kinematics* are used to reduce the active tracking area in multiple-target tracking [69]. The sensor nodes are classified into three categories: Boundary Nodes (BN), Worker Nodes (WN) and Computational Nodes (CN). The tracking area is mapped to a *Voronoi Diagram* (VD) and three different cases are identified depending on the overlapped area of interest. Larger overlapped area of interest between two polygons results in small number of sensors in that polygons, and vice versa.

2.3.2 Node Selection

The second subclass of network self-organization approaches is node selection approaches. As the network lifetime maximization problem is often formulated as an optimization problem under constraints, node selection can be seen as the result of the resolution of such a problem.

A a possible formulation of this problem is the *knapsack model* [70]. The goal is to maximize the residual energy of the network while meeting the application QoS requirements. In this model, the duration of the submitted task T is subdivided in multiple rounds of size t. The execution of the algorithm is alternated between different subsets of active nodes whose role will not change during the round. The algorithm seeks to select the best subset of nodes according to different objective function such as: minimum energy consumption, maximum residual energy, etc. According to this model, the nodes are the objects of potential relevance estimation R_i and residual energy U_i. They have to be placed in a knapsack of capacity M which represents the energy budget. The energy cost of each node is its energy spent in sensing and communication activities. The other constraints of coverage, connectivity and QoS are included in the dynamic program that resolves the problem.

Other possible formulations are the Network Lifetime Maximization Problem (NLMP) and Routing Path Length Minimization Problem (RPLMP) which are jointly presented in [71]. NLMP aims to maximize the number of sensor nodes which are kept in sleep state, while RPLMP aims to minimize the routing path length. The activated nodes remain in active state until the end of the round.

These two problems are *proved to be NP-Complete* and three heuristics are proposed to solve them, namely: the Naive Shortest Path Selection heuristic (NSPS), the Dual Shortest Path Selection heuristic (DSPS) and the Weighted Shortest Path Selection heuristic (WSPS). NSPS is better in delay minimization while WSPS and DSPS are better in network lifetime maximization. The choice of one heuristic among the others depends on the application requirements.

The problem of node selection raises the question of finding the upper bound of the network lifetime for any collaborative protocol [72]. A *Role Assignment* based approach is proposed in [72] where three basic roles for nodes are defined, namely: sensing, gateway and aggregation. The question is first modeled as a linear optimization model under the constraints of network topology and sensing operations. After that, it is transformed into a flow maximization problem in order to reduce the complexity of computation. A point is worth to note here is that energy consumption in the MAC layer should be incorporated into the proposed model in order to give a reliable estimation of the network lifetime upper-bound.

Node selection usually refers to *sensor management* with respect to energy-efficiency. This approach uses many concepts such as: state-centric strategy [73], rechargeable sensors [74] and profile information of the target [75, 76], for the purpose of managing the sensing activities of the nodes.

For example, in [73], the selection of active nodes is based on a state-centric strategy. The prediction of the target state is computed whatever is the number of hops between the current data fusion center and the next one. In this scheme, the selection of the fusion center is based on the energy cost, and the data transmissions are all routed along the paths of minimal energy cost.

The sensor activation range is also an important factor in tracking mobile target with high acceleration [75]. A Proportional-Integral-Derivative (PID) control system is used to update the activation range in each sampling period. The proposed algorithm [77] measures the effective tracking quality and compares it with the required tracking quality. The average error determines the number of nodes to be activated and the activation range as well. In this algorithm, the recovery process is based on the number of consecutive misses, the distance between the predictive positions, etc.

Another example of the use of sensor management is the *greedy selection*. The GSSM (Greedy-Selection Sensor Management) scheme is proposed in this context to assign a proper subset of sensors to track multiple targets [76]. GSSM uses an information filter [78] for multi-sensor data fusion. Sensor management in this scheme is based on the fact that *not all* measurements contribute in improving the tracking accuracy[5]. The information propagation uses two mechanisms to decide which subset of nodes receive the target information: (1) the *predicted*

[5]This is called the Sensor Management Problem.

subset mechanism which is more optimal, but less-effective, and (2) the *nearest subset mechanism* which is sub-optimal but communication and delay effective. In GSSM, the sensor management problem with the maximization of the information contribution of the sensors is formulated as a binary optimization problem. GSSM finds a near-optimal solution of this problem compared to the *branch-and-bound* algorithm. However, this later is more difficult to implement in a localized fashion.

The sensor selection problem with rechargeable batteries that use energy harvesting techniques is proposed in [74]. The basic idea is that the energy consumption depends on the current state of the nodes, namely: active state, inactive state, wakeup state, energy harvesting state and observation state. As in ANSWER architecture [79], a set of scheduling techniques that take into account uncertainties of the energy incomes are used. For example, in the *static active time* approach, the complete duration of the day contains a regular alternation between active and inactive time. This approach increases the observation quality but suffers from the lack of adaptation to unexpected events. To overcome this drawback, a *multi-parametric heuristic based* approach is proposed. Multiple parameters such as the current stored energy, the probability of encountering an event, etc. are used to predict the length of the next active period. A third approach based on the *utility active time* is proposed to overcome the problems of the heuristic approach.

From the perspective of the target nature, node selection is also affected by the target speed. For example, a scheme called Distributed Spanning Tree Algorithm (DSTA) is proposed in [80] to track fast targets. Its goal is to decrease the probability of target loss. For this purpose, it generates successive predictions to awakening nodes at $t+\delta t$, $t+2\delta t$, $t+3\delta t$, etc. where δt is the duration of the tracking step. The protocol is designed in two layers: the lower-layer and the upper-layer. The lower-layer runs a spanning tree-based clustering protocol that builds a cluster-based network structure (see Figure 2.7(a) for illustration: DSTA forms three clusters: the current cluster $\{s_1, s_2, s_3, s_4\}$ with node s_1 as the cluster-head, the wakeup cluster (i) $\{s_5, s_6, s_7, s_8, s_{14}, s_{15}\}$ with node s_{14} as the cluster-head and the wakeup cluster $(i+1)$ $\{s_9, s_{10}, s_{11}, s_{12}, s_{13}\}$ with s_{10} as the cluster-head). The upper-layer runs the awakening algorithm where the clusters form a spanning tree rooted at the sink node. Due to the average speed of the target and its direction, the clusters in the same direction are woken-up. Their number depends on the target speed.

The node selection depends also on the profile information of the target. For example, the Global Prediction-Based Algorithm (GPBA) [81] considers two main parameters in this context: (1) the sampling duration and (2) the reporting frequency (to the sink). It uses global profile rather than local profile because it is accessible by all the nodes of the network. The mobile object (target) in GPBA can collect the information about its behavior from the network. Thus, nodes can use this information to activate a specific set of nodes. This approach is specific

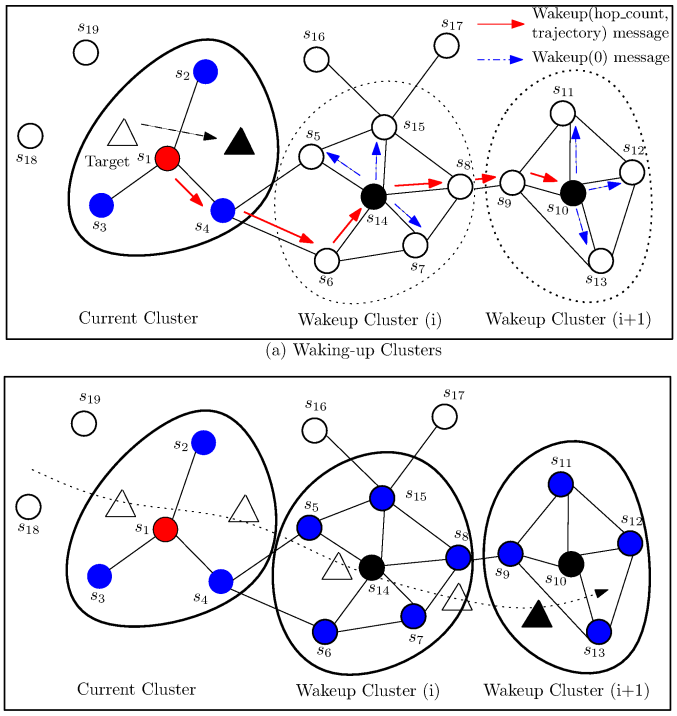


Figure 2.7: Example of DSTA protocol operations.

to objects tagged with a unique global ID. A learning phase is necessary to record the frequency of the object movements between the nodes. Each node saves the frequency of each node by which the object is traversing. After that, based on the object profile, a *tracking leader* is elected as the current cluster-head that detects the target. The object profile is updated each time the object is lost.

2.3.3 Dynamic Clustering

A WSN-based target tracking system is generally built on a cluster structure because of its aggregation and data fusion characteristics. In the literature, different types of clustering schemes exist such as: pure dynamic clustering schemes and hybrid (static/dynamic) schemes. The cluster formation, maintenance, reconfiguration and the cluster-head election are the main issues related to these schemes. In this section, we describe and discuss some cluster-based protocols such as: ADCT, HTTP and HCTT, etc. Each protocol has a different method to tackle the above-mentioned issues.

As a starting example, we have the Adaptive Dynamic Cluster-based Tracking (ADCT) protocol which is proposed in [82], and whose cluster formation procedure is based on a two-phase mechanism: a broadcast phase and a notification

phase. The node with the smallest distance/ID is chosen as a cluster-head. The sensor selection procedure is based on an optimal selection function which is a mixture of both data usefulness and energy cost. The usefulness of the nodes can be deduced from the *bid messages* sent by the members to the cluster-head. The cluster reconfiguration procedure is triggered when the predicted position of the target is on the boundary of the current cluster. In this case, the cluster-head sends a command message to the neighbor node nearest to the predicted position. The receiving nodes send an election message to their neighbors and select the first replying one as the new cluster-head. The recovery mechanism is based on acknowledgment messages and waiting timers.

Cluster-based schemes can also be coupled with the Particle Filter (PF) [83]. It uses a re-sampling method (SIR) to reduce the computation complexity of PF by eliminating samples with small weights and preserving samples with big weights: that is called *bootstrap filter*. This approach suffers from degeneracy problem: the system may collapse to a single point. The solution proposed in [83] is a *local linearization* using EKF or UKF. The utility function used in node selection is defined as the uncertainty of the target reduced by the additional measurements. It can be represented by the entropy of the belief state which can be used to select the best node among sensor candidates to maximize information gain. The cost includes: the bit rate between the cluster-head and the neighbor, the distance from the sensor node to the cluster-head and to the target, and the energy needed to receive one bit from the neighbors. An optimization problem is formulated and two scenario-dependent solutions are proposed: a *meta heuristic called GRASP* for the static scenario and a *branch-and-bound* method for the dynamic scenario.

Another protocol called Herd-Based Target Tracking Protocol (HTTP) is proposed in [84]. It uses a state transition model with three states, namely: sensing, sleeping and tracking. Each node computes its weight and decides to participate in the tracking process or not. The nodes in the tracking state form a cluster surrounding the target. The *backup herd node* is a node that has the same role as the *herd node* but it does not send the data to the base station. The geographic region of the network is divided into virtual grids, each of which is monitored by a cluster-head. A node in the sensing state computes its weight periodically then checks if it exceeds a specified threshold. Then, it changes to the tracking state. Note here that the weight of a node depends on its distance to the target, and the number of clusters that can participate in the tracking task is determined by the grid size. When the target moves out from the current grid to a new one, the nodes within it can change either to the tracking state or to the sensing state depending on the measurements' data. Meanwhile, the nodes of the previous grid return back to the sleeping state because they cannot sense the target anymore. The herd reconfiguration is triggered based on the distance between the current herd and the target.

An example of hybrid cluster-based protocol is HCTT (Hybrid Cluster-based Target Tracking) [85]. It deals with the problem of the *boundary of clusters* (which increases the tracking uncertainties) by integrating a dynamic on-demand clustering protocol and a static cluster-based target tracking scheme. To solve this problem, HCTT checks first whether there exist neighbor nodes that belong to another cluster or not. If yes, then these are boundary nodes and the cluster region is divided into three types: safety region, boundary region and alert region. Consequently, the dynamic cluster includes active boundary nodes that detect the target. The hand-off between static and dynamic clusters is based on the sensing data received from the nodes within these different regions.

The scheme proposed in [86] uses a backoff procedure to deffer the broadcast of messages, in order to reduce the energy consumption and to achieve data accuracy by varying the transmission range. It consists of two processes: the selection process and the release process. In the selection process, the nodes that detect the target simultaneously trigger the backoff procedure. The node with the lowest backoff period, broadcasts a *DETECT message* to its neighbors (nodes that are in its transmission range). The receiving nodes stop their backoff procedures and the other nodes (that are out of the transmission range and which has the lowest backoff time) will not receive the DETECT message. Thus, the node with the lowest backoff time will track the target. Collisions occur when two or more nodes transmit their DETECT message simultaneously because of their identical backoff time. In the release process, when the target moves out of the sensing range of the detecting nodes, a *RELEASE message* is transmitted to the neighbors. Upon receiving this message, nodes trigger a backoff procedure. Similarly, nodes having the lowest backoff time will be selected to track the target.

2.4 Classification of Specific Approaches

In this section, we describe certain schemes that put special assumptions on the sensor capacities and/or track special targets. These schemes do not fit in our classification proposed in Section 2.1 because they use different methods to achieve energy efficiency. However, they are related to the general classification by the ability of using predictions to optimize the sensing and communication operations. In the following subsections, we present the most important schemes that fit in this category.

2.4.1 Continuous Object Tracking

Continuous objects contrary to single or individual objects, have a geometric shape and may expand in a large area such as: gas leaks, animal troupes, etc. Generally, this kind of objects cannot be represented by a single point or an atomic

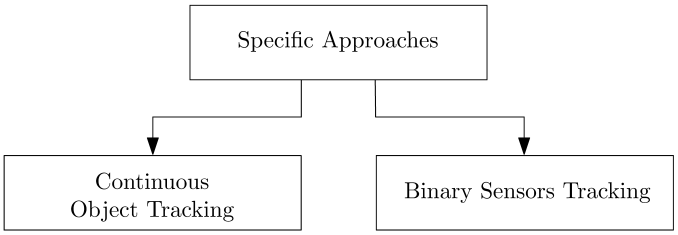


Figure 2.8: Specific approaches.

event. They often need multiple attributes to be described with. Thus, sensor nodes should have multiple sensing modalities to track such objects, i.e. they form a *Wireless Heterogeneous Sensor Network* (WHSN), where one sensor node can detect multiple attributes of a target. WHSN are suitable for detecting and tracking composite events (ex. fire or pollution). A challenging task in tracking continuous objects is the *event boundary determination*. Contrary to individual object tracking, in which the main problem is how to predict the next location of the target, and how to inform the next tasking nodes to take charge of the tracking task, composite events occupy a large area. The goal here is not to construct and maintain a network topology but to estimate attribute regions and to determine the event region.

According to [87], there exist four methods to track continuous objects:

1. All the sensors in the phenomenon area report data to the sink.
2. Only nodes nearby the boundary area of the phenomenon are selected.
3. Nodes outside and inside the boundary area are selected (more nodes than in the previous method).
4. Few representative nodes that report data.

For example, in ECOT [87], an adjustable sensing range technique is used where nodes are classified in five types: (1) undetect-to-detect, (2), detect-to-undetect, (3) border node, (4) representative node and (5) border point. Boundary detection is achieved by adjusting the sensing range of nodes: nodes of type 1 diminish their sensing range and nodes of type 2 increase their sensing range.

TOCOB is another scheme [88] where each node wakes-up periodically and makes a local observation. A node becomes a CVN (*Changed Value Node*) when it observes a value in the current sampling period different from its last recorded value. This node broadcasts a COZ (*Compare One Zero*) message containing its ID and its status. A node becomes BN (*Boundary Node*) if it receives a COZ message with different value; It counts the COZ messages received during a period of time in order to decide to become a RN (*Representative Node*) or not. Contrary to the COBOM algorithm [89], not all the nodes nearby the boundary will become

BN but only those that receive different readings. RN selection is based on the number of COZ messages received by the BN nodes to determine the backoff time: the higher is the number of COZ messages, the shorter is the backoff time. Hence, the nodes near the boundary will have high probability to become RN nodes.

CODA [90] uses a hybrid static/dynamic clustering approach by constructing a static cluster backbone and determining the boundary sensors which then form a dynamic cluster to monitor the continuous object profile. CODA uses also the *Graham Scan algorithm* [91] to resolve the *Convex Hull* problem i.e.: determine the sensors located at the boundary of the cluster by the cluster-head. Boundary detection is based on the number of the static clusters that detect the object. The cluster-head is notified via *sense* messages. Then, it executes the Graham Scan algorithm and constructs a dynamic cluster. When the object boundary moves out of the sensing range of the current boundary sensors, new clusters are formed.

In CollECT [92], the accuracy of *event determination* is achieved by subdividing the estimated attribute regions into multiple non-overlapping faces. In each face, the nodes determine whether the event has occurred or not. Some test rules such as AIT (*Alert-In Triangulation*), active role transition and passive role transition are used to implement these procedures of event determination and border node selection. A simplified strategy of logical neighbor selection called *short diagonal wins* is used. More general design strategies are needed to prove the effectiveness of such technique.

2.4.2 Tracking with Binary Sensors

Binary sensors generate one bit of information indicating that the target is approaching or moving away, or it is present in the sensing range of the sensor or absent. These primitive sensors indeed minimize the volume of data transmitted from the nodes to the sink. However, the quality of tracking may degrade when noisy measurements and/or lossy links are present.

In [93], the authors propose a minimalist scheme of binary sensors that broadcast only one bit of information to the base station to indicate that the target is approaching or moving away. A derived tracking algorithm based on PF is proposed. The algorithm generates a set of particles whose weights are computed based on the probabilities of moving from one point to another and the distances between the sensors. The acceptance criterion of a particle is its associated probability which should be above of a certain threshold.

Although this model is energy-efficient because it generate small amount of data, it can not distinguish two targets close to each other or that move parallel to each other. Hence, an enhancement with proximity information is proposed in [94] to overcome this problem.

Binary proximity sensors can be used for tracking a group of targets as a con-

tinuous region [94]. Sensor nodes compute their probability of detection based on the duration of the target presence in their sensing field. They use this probability to determine if the target is really detected or not. In order to localize the group of the targets, authors propose two algorithms to determine the monitoring region: the first one which is accurate but complex, computes the convex hull using the graham scan algorithm. However, the second one which compute the circle that contains the convex hull, is less complex but less accurate. The algorithm of selection of the reporter node chooses the node closest to the plus (+) sensors. Based on the distance transmitted by the data, and after iteration, the proposed algorithm gives the near-optimal coordinates of the reporter node. Finally, a redeployment control algorithm is proposed when the target group moves away from the current one. This algorithm aims to reduce the energy consumption by reducing the distances of the data transmission.

Another binary proximity sensor tracking scheme is proposed in [43]. For non-Gaussian target state distribution, the tracking process becomes intractable. Instead of using a PF, the authors propose a *Variational Filtering* approach (VF) to reduce inter-cluster communications. In addition to this, a Binary Proximity Observation Model (BPOM) with a predefined threshold is used. This gives a solution to the problem with minimalist model. The advantage of VF over PF is the compression of the statistics required to update the filtering distribution between successive instants. In order to reduce the energy consumption, the authors adopt a non-myopic cluster activation based on the prediction generated by the VF.

2.5 Schemes Comparison and Discussion

All the above-described schemes share the double-objective of minimizing the energy consumption and improving the data accuracy:

- Energy-efficiency is related to the sensing and the communication operations.
- Data-accuracy can be expressed as the precision of the estimations or the amount of data extracted by the WSN given a certain network energy budget.

As it is shown in Table 2.1, on one hand, the sensing-related methods select nodes using prediction algorithms with respect to coverage, connectivity and network lifetime constraints. They use data fusion and data compression algorithms to minimize the volume of transmitted data. On the other hand, the communication-related methods optimize the routing and data reporting. They select reporter nodes based on the distance estimated between the current active

Table 2.1: Comparison between schemes objectives.

Objectives	Sensing-Related	Communication-Related
Energy Consumption	Information-Utility measures Measurements' Error Data fusion & Compression	Selection of data reporting node Role assignment Multi-prediction Application-oriented topologies
Data Accuracy	Target Observability Sampling period length Sensor management	Network coverage & connectivity Geographic position information Topology reconfiguration

nodes and the target. They use role-assignment and multi-step prediction approaches to balance energy consumption between the nodes. Another objective of the network self-organization techniques is to construct application-oriented topology that helps optimizing data-fusion process and extend the network lifetime.

Data accuracy and energy-efficiency should be traded as they are opposite objective. According to our study, data accuracy depends on the network coverage and the target observability. Constructing an *efficient* topology with respect to the coverage constraints and the message communication cost is a challenging problem. Furthermore, the use of a minimalist model of target observation such as binary sensors, adds additional constraints but helps reducing the energy consumption. The specification of the information-utility measurements and the target profile definition are also key problems to enhance the data accuracy.

Table 2.2, Table 2.3, Table 2.4 and Table 2.5 show the comparisons that we conduct between the sensing-related, the sleep scheduling, the sensor selection and the dynamic clustering mechanisms, respectively. It is easy to observe that few schemes consider *all* the energy-efficiency aspects of a typical target tracking scheme (subsection 2.1.1). All the schemes address only two or three aspects (except two schemes [57], [75] that consider four aspects). Furthermore, the activation mechanism is the most important aspect addressed by the different schemes followed by network logical structure and the estimation/prediction algorithm. Hence, more attention should be put on the other aspects that are *less-handled* such as the data reporting mechanism.

As we can see on the tables, sensing-related schemes focus on the estimation/prediction algorithm in contrast to sleep scheduling based schemes and sensor selection and dynamic clustering based schemes which concentrate, respectively, on the quality of detection and the logical structure of the network.

Table 2.2: Comparison between mechanisms of sensing-related schemes.

Mechanisms	[47]	[49]	[52]	[56]	[41]	[57]	[58]	[59]	[60]
Quality of Detection	✓	✓	✓	✓	-	✓	-	-	-
Estimation/Prediction Algorithm	✓	-	✓	-	✓	✓	✓	✓	✓
Data Reporting Mechanism	-	-	-	-	-	-	-	-	-
Activation Mechanism	✓	✓	-	✓	-	✓	✓	✓	✓
Logical Network Structure	-	-	✓	-	✓	✓	-	-	✓

Table 2.3: Comparison between mechanisms of sleep scheduling based schemes.

Mechanisms	[61]	[62]	[64]	[65]	[66]	[67]	[68]	[69]
Quality of Detection	-	✓	✓	✓	✓	-	✓	-
Estimation/Prediction Algorithm	-	✓	-	-	-	✓	-	-
Data Reporting Mechanism	-	-	-	-	-	-	-	-
Activation Mechanism	✓	✓	✓	✓	✓	✓	✓	✓
Logical Network Structure	✓	-	-	-	✓	✓	✓	✓

Table 2.4: Comparison between mechanisms of node selection and dynamic clustering based schemes.

Mechanisms	[70]	[71]	[72]	[73]	[75]	[76]	[74]	[80]	[81]
Quality of Detection	-	-	-	-	✓	✓	✓	-	✓
Estimation/Prediction Algorithm	-	-	-	-	✓	-	-	✓	-
Data Reporting Mechanism	-	-	✓	✓	✓	✓	-	-	✓
Activation Mechanism	✓	✓	-	✓	✓	-	-	✓	-
Logical Network Structure	✓	-	-	✓	-	-	-	✓	✓

Table 2.5: Comparison between mechanisms of node selection and dynamic clustering based schemes (continued).

Mechanisms	[82]	[83]	[84]	[85]	[86]
Quality of Detection	-	-	✓	-	-
Estimation/Prediction Algorithm	-	✓	-	-	-
Data Reporting Mechanism	-	-	-	-	-
Activation Mechanism	✓	✓	✓	-	✓
Logical Network Structure	✓	✓	✓	✓	✓

2.6 Conclusion

The energy problem in WSN is still an active research area where a huge body of research is produced. In this chapter we surveyed, some of the most recent

target tracking schemes whose goal is to preserve the energy of the network while maintaining an acceptable level of data accuracy. All the schemes that we have described and discussed above try to resolve the energy problem by enabling the interaction between the sensing layer and the communication layer and allowing each layer to take profit from the other.

However, we believe that the energy can be more likely optimized in the communication layer and the research effort should be focused on it. For this purpose, Chapter 4 and Chapter 5 deal with this aspect. In Chapter 4, we propose to model the energy problem by a BILP model to find the exact solution in the general context. And in Chapter 5, we relax the model by using a heuristic solution called CORAD.

Nonetheless, before going in depth in the communication related aspects, we first explore in Chapter 3 the possibilities of optimization in the sensing layer. We propose a dynamic clustering algorithm based on an enhanced version of the Distributed Kalman Filter for target tracking using sensors with limited sensing range.

Chapter 3

Energy Efficient Target Tracking Scheme with Limited Sensing Range

TARGET TRACKING APPLICATIONS in WSN can use either long sensing range or limited sensing range sensors, or both of them depending on the energy supply and consumption of the nodes. When the nodes use sensors with long sensing range, they consume energy excessively, especially if they are equipped with limited capacity batteries. At the opposite, when they use sensors with limited sensing range, they conserve the energy of their batteries due to the small quantities that they spend to *sense* the targets located in their restricted proximity.

The benefit of such sensors is also to *localize* the data fusion process. When the target is detected by a restricted set of nodes, the network can be *self-organized* into disjoint dynamic clusters that move along with the target trajectory. This will limit the number of nodes involved in the data fusion process. However, two problems are raised by this approach:

1. The *distribution* of the data fusion algorithm: since data are received from multiple sources and the data filtering algorithms are centralized which is not suitable for the sensor nodes, a *lightweight distributed* version of these algorithms could be used to limit the processing task over the set of detecting nodes.

2. The prevention of the target loss: as the detection zone is limited, then at each target state update, a network reconfiguration operation is required to prevent the loss of the target. The overhead of the reconfiguration should also be reduced to make the operation more efficient.

In this chapter, we propose a dynamic clustering approach based on a modified variant of the distributed Kalman Filter which we use as a data fusion al-

gorithm to optimize the network communications and to reduce the energy consumption in target tracking applications.

Our data fusion algorithm is based on the work proposed in [41], in which the nodes implement local micro-filters and reach a consensus using *message passing* communication model. This scheme makes the assumption that passive nodes (nodes that do not detect the target) are considered with *no contribution*. Despite this, they are included in the fusion process.

There are also other versions of the Kalman Filters that refers to the Distributed Kalman Filter such as: the Kalman Consensus Filter (KCF) proposed in [95]. It uses a set of *k Kalman micro-filters* to fuse heterogeneous data received from sensors with non-linear sensing models. There are two variants of this approach: one fuses the measurements and the other fuses the estimations. In the variant of measurements' fusion, low-pass and band-pass filters are modified into *high-gain* high-pass filters. In the other variant, the estimations are fused instead of the measurements in order to accelerate the consensus convergence. This filter uses power-consuming complex matrix computations that generate latency and may fail detecting fast targets. Furthermore, the algorithm makes the assumption that all the nodes can observe the target simultaneously; which may not hold all the time.

Many approaches are proposed to distribute the Kalman Filter such as: biparti graph [96], gossip communications [97], etc. In biparti graph approach [96], the global model is decomposed into n_l ($n_l \ll n$ and n is the network size) reduced sub-models, each one is executed by a micro-filter in a single node. Each node computes its local estimation and fuses it with the received estimations. Biparti graphs are used when dependencies exist between these sub-models. This method is suitable for the fusion of estimations because it includes data correlation between local estimations.

Distributed Kalman filter with Gossip communications is proposed in [97]. Each node can sense only a part of the observed phenomenon, i.e., each node can measure or estimate a subset of the target state attributes and communicates them to its neighbors and then deduces the missed attributes. There are two drawbacks to this method: (1) message communication complexity, i.e. nodes exchange many messages (estimations and error covariance matrix), and (2) topology dependency model, i.e. a strong network connectivity is required for communicating estimations between neighboring nodes.

Instead of sending long messages, authors of [98] propose to send only *one bit* of information. A *quantification function* is defined to represent the estimation of the node and then the filter is executed in two distinct procedures: an observation-transmission procedure and a reception-estimation procedure. The main disadvantage of such approaches is that the quantification function may induce information loss when lossy links are present in the network.

All the above-mentioned approaches do not consider the problem of limiting

the number of nodes participating in the tracking task and suppose that the target can be observed by the *whole* network. However, these two assumptions do not hold in all the cases: i.e. in a 2D ground deployed WSN, only nodes that are close to the phenomenon can sense it; the other nodes cannot. In addition, low-power nodes have limited sensing ranges and can communicate only with their direct neighbors. This aspect can be exploited to reduce the energy consumption in target tracking applications, but also arises other issues.

3.1 System Model and Assumptions

In this section, we give some basic definitions of the WSN model and the centralized Kalman Filter. We describe the mathematical model of the Distributed Kalman Filter used in our proposed method.

3.1.1 WSN Model

WSN is modeled as a undirected graph $G(V, E)$ where $V = \{s_1, s_2, \cdots, s_n\}$ is a set representing the nodes and $E = \{(s_i, s_j) \mid \|s_i - s_j\| \leq R_c\}$ is a set representing the communication links. R_c is the communication range of each node and $\|s_i - s_j\|$ is the Euclidean distance between nodes s_i and s_j. The sensing range R_s is assumed uniform among all the nodes and it is supposed to verify the condition of coverage and connectivity, i.e.:

$$R_c \geq 2R_s$$

The target state is supposed also a 4-tuple vector:

$$X = (x, y, \dot{x}, \dot{y}) \in R^4$$

where (x, y) and $(\dot{x}, \dot{y})$ are respectively, the target position coordinates and the velocity components along X and Y axes. Each node measures the distance to the target ρ, and the angle between the X axis and the target position vector θ.

For target detection, we use a probabilistic model formulated by the equation 3.1:

$$p_s(q) = \begin{cases} 0 & \text{if } r + r_e \leq \|s - q\| \\ e^{-\alpha(\|s-q\| - (r - r_e))^\beta} & \text{if } r - r_e \leq \|s - q\| \leq r + r_e \\ 1 & \text{if } r - r_e \geq \|s - q\| \end{cases} \tag{3.1}$$

where $\|s - q\|$ is the Euclidean distance between the sensor s and the target q, r is the sensing range of s and r_e is the sensing error ($r_e \ll r$). α and β are constants (see Figure 3.1).

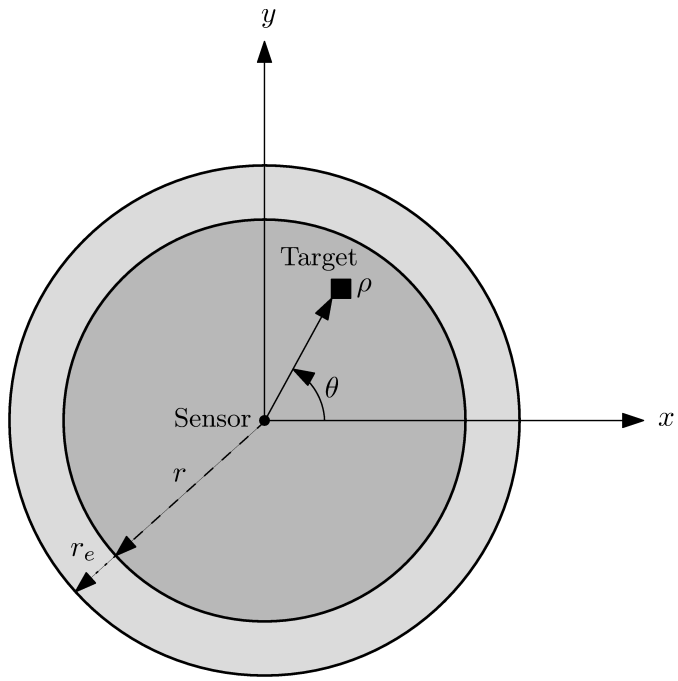


Figure 3.1: Probabilistic detection model.

The nodes are assumed initially in the sleep state which guarantees minimum energy consumption. In fact, in this state, all the hardware units of the node are off, except the processing unit and a low-power *paging channel* to receive *wake-up* messages. Upon receiving a wake-up message, the nodes start up all their hardware units. We assume also that each node maintains a list of its neighboring nodes with their corresponding distances to it. All the nodes are aware of their geographic positions using either GPS-based or localization-based techniques to measure such distances.

The first target detection is supposed done successfully and the first activation is performed via an *external* activation message.

3.1.2 Centralized Kalman Filter

We assume that the target state and the measurement models are respectively defined by the following recursive linear equations:

$$x_{k+1} = A_k x_k + B_k u_k + w_k$$

$$z_k = H_k x_k + v_k$$

where: A_k is the matrix that relates the previous target state to the current one, B_k is the matrix that relates the commands to the current target state, w_k is the

system noise, H_k is the matrix that relates the measurements to the current target state and v_k is the measurements' noise at time step k. x_k is the target state vector at time step k.

We suppose that w and v are white noises with respective Q and R covariances:

$$p(w) \sim N(0, Q)$$
$$p(v) \sim N(0, R)$$

Also, we suppose that matrices A_k and H_k are detectable and all the matrices A_k, B_k, H_k, Q_k and R_k are time-independent, i.e:

$$\exists A, B, H, Q, R : \forall k, A_k = A, B_k = B, H_k = H, Q_k = Q, R_k = R$$

For *Constant-Velocity* (CV) target model, matrix A_k and H_k have these values:

$$A_k = A = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, H_k = H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

We denote $\hat{x}_k^-$ and $\hat{x}_k$ as the *a priori* and the *a posteriori* target state estimations at the time step k, respectively.

Similarly, the a priori and a posteriori estimation error covariance matrices P_k^-, P_k at time step k are defined by:

$$P_k^- = E[(x_k - \hat{x}_k^-)(x_k - \hat{x}_k^-)^T]$$
$$P_k = E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T]$$

The Kalman gain factor at time step k is defined by:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1}$$

We summarize the Kalman model by the following diagram shown in Figure 3.2. The two recursive steps of the Kalman Filter are [1]:

1. The prediction step:

   - Next step state prediction:

$$x_{k+1}^- = A_k \hat{x}_k + B_k u_k \tag{3.2}$$

   - Next step error covariance prediction:

$$P_{k+1}^- = A_k P_k A_k^T + Q_k \tag{3.3}$$

[1]Exhaustive description of the Kalman Filter is given in Appendix A.

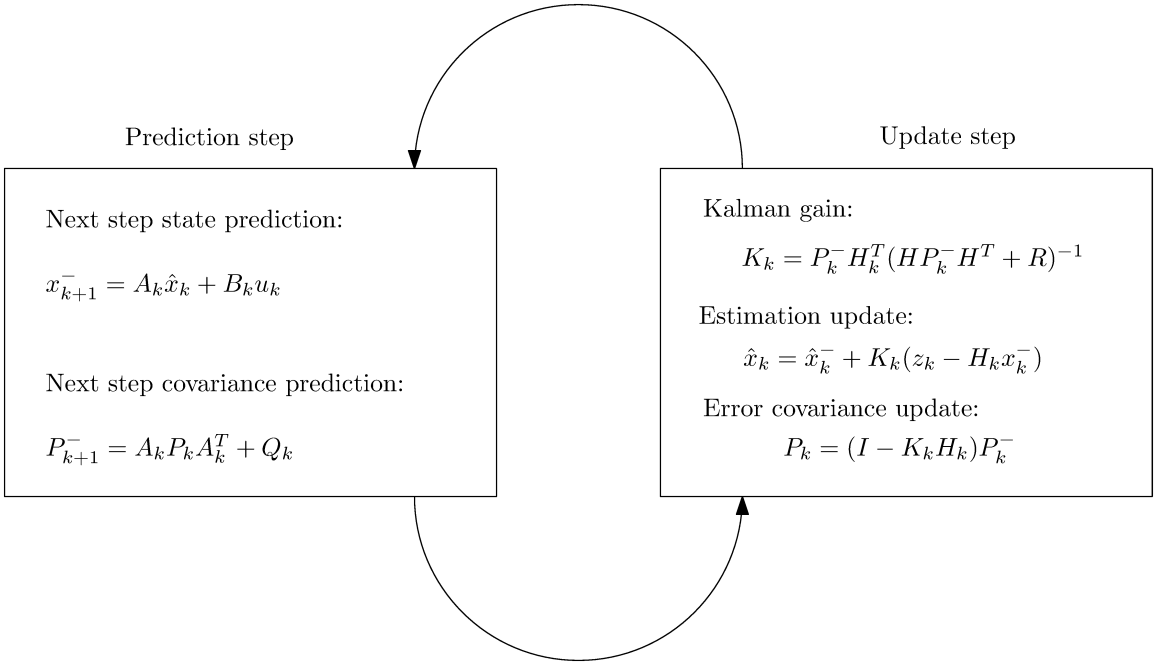


Figure 3.2: Kalman filter steps.

2. The update step:
   - Kalman gain:
   $$K_k = P_k^- H_k^T (H P_k^- H^T + R)^{-1} \tag{3.4}$$
   - Estimation update:
   $$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H_k x_k^-) \tag{3.5}$$
   - Error covariance update:
   $$P_k = (I - K_k H_k) P_k^- \tag{3.6}$$

3.1.3 Consensus on Kalman Micro-Filters

The Distributed Kalman Filter (DKF) gives the target state estimation using a set of k micro-filters. This model requires *all-to-all* communications between all the nodes. In [99], the authors propose an algorithm of DKF using high-pass and low-pass filters for data fusion. It fuses heterogeneous data generated by a non-linear sensing model:

$$y_i = h_i(x_k) + v_i^k$$

All the nodes have the same architecture illustrated in Figure 3.3.

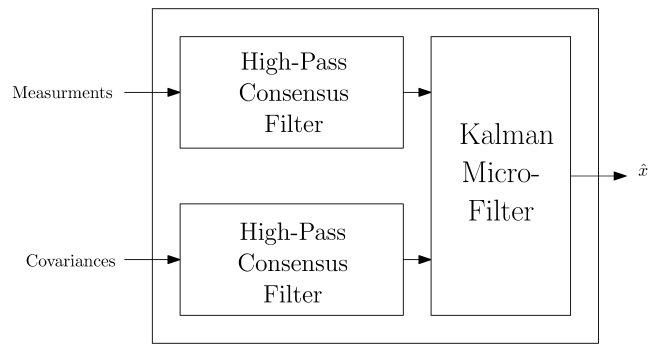


Figure 3.3: Micro-filter architecture of a DKF node.

The observed system is modeled as follows:

$$x_k = A_k x_{k-1} + B_k w_k$$

$$y_i^k = H_i^k x_k + v_i^k$$

In addition to the equations of the Kalman Filter model described in Section 3.1.2, two new variables are included into the model, namely: (1) the reverse of the error covariance matrix, and (2) the fusion of the measurements. These two variables are given respectively by:

$$S_k = \frac{\sum_{i=1}^{n} H_i^{kT} R_i^{-1} H_i^k}{n}$$

$$y_k = \frac{\sum_{i=1}^{n} H_i^{kT} R_i^{-1} y_i^k}{n}$$

Each node executes the following calculations:

$$M_i^k = (P_{i,k}^{-1} + S_k)^{-1}$$

$$\hat{x}_k = \hat{x}_k^- + M_i^k (y_k - S_k \hat{x}_k^-)$$

$$P_i^{k+1} = A_k M_i^k A_k^T + B_k Q_i^k B_k^T$$

$$\hat{x}_{k+1} = A_k \hat{x}_k$$

with: $Q_i^k = nQ_k$ and $P_i^0 = nP_0$. The estimations are identical in all the nodes, i.e.,

$$\hat{x}_i^k = \hat{x}_k, \forall i$$

The filter dynamic is given by equation 3.7 (for more details see [100]):

$$\begin{cases} \dot{q}_i = -\beta \hat{L} q_i - \beta \hat{L} u_i \\ p_i = q_i + u_i \end{cases} \tag{3.7}$$

where $\hat{L} = L \bigotimes I_m$ is the $m-$dimension graph Laplacian, u_i is the node input, q_i is the Kalman Filter state and β ($\beta > 0$) is the gain (it should be big enough for random deployed topologies). The filter output is computed according to equation 3.8:

$$\begin{cases} \dot{q}_i = \beta \sum_{j \in N_i} (q_j - q_i) + \beta \sum_{j \in N_i} (u_j - u_i) \quad \beta > 0 \\ y_i = q_i + u_i \end{cases} \tag{3.8}$$

With N_i is the set of neighbors of node i (the reader could refer to [101] to learn more about the filter discretization in connected graphs).

3.2 Proposal of Dynamic Clustering Based on Distributed Kalman Filter

Figure 2.2 in Chapter 2 shows the relationship between the sensing component and the communication component in a sensor node that uses a prediction-based scheme. A *light-weight* estimation-prediction algorithm can be used to estimate the target state and predict its next position. This helps selecting and awakening the most appropriate nodes and reducing the network communications. As non-selected nodes remain in the *sleep* state, the energy is conserved more than in *periodic sampling*-based approach [2].

Intuitively, the periodic sampling provides accurate data but it wastes the energy resources of the nodes. At the opposite, the prediction-based schemes are more appropriate for dense networks where it is not necessary to wake-up all the nodes. However, they raise two main issues with respect to the quality of data: (1) the distribution of the estimation algorithm over the subset of detecting nodes, and (2) the dynamic changes of the group of tracking nodes according to the dynamic of the target.

To resolve these issues we propose a Distributed Kalman Filtering approach with Dynamic Clustering, which we call DKF_DC [102].

Our approach is inspired from the work proposed in [41]. But instead of activating all the network, our approach uses a dynamic clustering protocol to limit the communications between participating nodes. Our dynamic clustering algorithm consists of two phases: the leader election phase and the cluster reconfiguration phase.

The leader election is executed among *active* nodes that are close to the target. The other nodes stay inactive to conserve their energy resources. Figure 3.4 shows an example of 36 sensor nodes deployed in the surveillance area. Only nodes

[2]Contrary to the periodic sampling based approach, our proposed scheme refers to the event-driven approach.

s_{11}, s_{12}, s_{16} and s_{22} are active. All the 32 remaining nodes stay inactive in the first tracking step.

In the second tracking step, node s_{22} is still active and nodes s_{20} and s_{21} are activated by the previous cluster nodes.

Therefore, the nodes are woken-up only when they receive activation messages to adhere to the current cluster. Unlike centralized fusion methods, the cluster-head in our method is not considered as a fusion center but as a cluster manager, as well, which is responsible for its reorganization. Hence, the communications are performed between all the active nodes and not only between the active nodes and the cluster-head.

The continuity of target tracking is guaranteed by allowing a subset of the last cluster members to adhere to the current cluster (such as node s_{22} in Figure 3.4). That is what ensures the propagation of the estimation information along with the target trajectory.

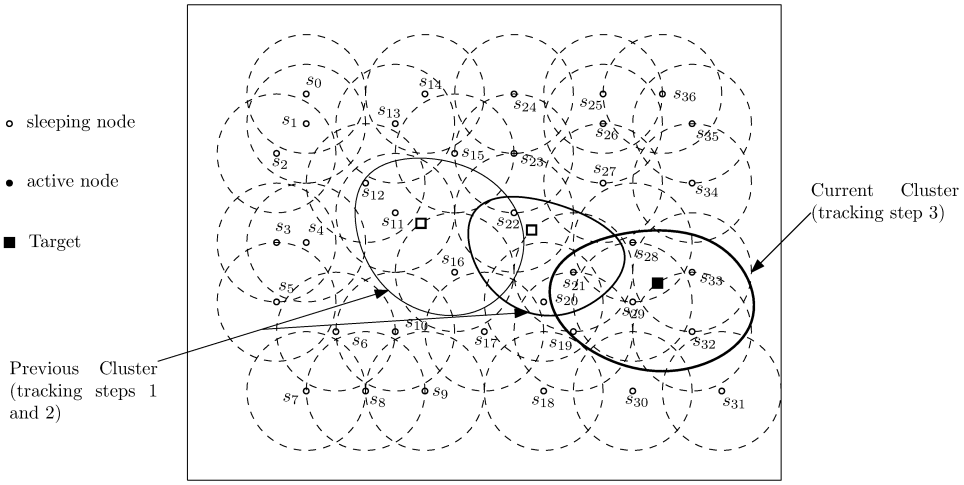


Figure 3.4: Example of the dynamic clustering protocol.

The role of each message exchanged between nodes is described in Table 3.1.

3.2.1 Cluster Formation and Leader Election

When a node receives an external intrusion message "MSG_INTRUSION" that contains the target state estimation recorded by some border nodes, it wakes-up and triggers the leader election phase. In this phase, the node sends first a wake-up message "MSG_WAKEUP" to all its neighbors, then it broadcasts a cluster creation message "MSG_CREATECLUSTER" containing the first detected position.

The nodes receiving "MSG_CREATECLUSTER" message, compute a *local decision value* using measure functions such as: (1) the distance between the node

Table 3.1: Roles of messages.

Message	Role
MSG_INTRUSION	It is an external message sent by border nodes via a low-power channel, indicating that a target is present in the surveillance field.
MSG_WAKEUP	It is broadcast by the first node receiving MSG_INTRUSION message. The nodes receiving this message change to active state.
MSG_CREATECLUSTER	It triggers the cluster formation operation. It contains the estimated position of the target.
MSG_CHREADY	It is sent by the nodes in the waiting state upon expiration of their waiting timer. It indicates that the sender node is on the top of its list of candidates, and it can be elected as a leader.
MSG_JOIN	It is sent by the leader node to force the other nodes to adhere as members in its cluster.
MSG_NOTCH	It is sent by the leader to indicate that it is no longer leader.
MSG_QUITCLUSTER	It is the opposite of MSG_JOIN message. It forces the member nodes of a cluster to leave this cluster.

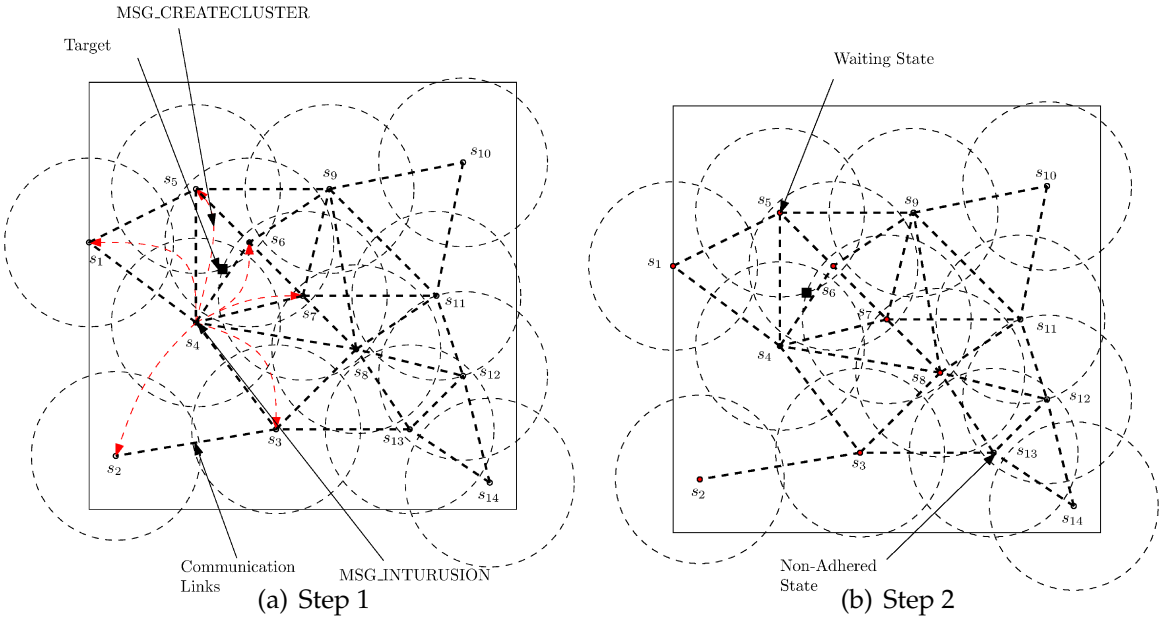
and the target, (2) the last estimation quality measured by the covariance matrix P_k issued from the Kalman Filter or (3) the residual energy of the node. A node leaves the *non-adhered* state to go back either to the *sleep* state when the waiting timer expires, or to the *waiting* state when it receives a "MSG_CREATECLUSTER" message.

Figure 3.5 illustrates this process. Node s_4 receives the "MSG_INTRUSION" message and it sends "MSG_CREATECLUSTER" message to all its neighbors: s_1, s_2, s_3, s_5, s_6 and s_7 (see Figure 3.5(a)). These nodes enter the waiting state and the other nodes return back to the non-adhered state and then to the sleep state (see Figure 3.5(b)).

The node in the waiting state, computes its decision value and broadcasts it to its neighbors. If it receives a decision value sent by another neighbor, then it updates its *list of candidates* by including the couple (sender ID, value). Then, another timer is alarmed to wait for receiving such decision values. After expiration, the waiting node decides to become leader if it is on the top of the list of candidates. Otherwise, it changes to a *temporary member* state.

Figure 3.6 illustrates the steps described above. Nodes s_1, s_4, s_5, s_6 and s_7 change to temporary-member state because of their exchanged values. However, nodes s_2 and s_3 are discarded from the election process because they have null values (that represent non-significant detected values).

During the waiting time, if a node receives a "MSG_CHREADY" message containing the cluster-head ID, then it adheres as a member to this cluster. Temporary-



(a) Step 1 (b) Step 2

Figure 3.5: Cluster formation operations (Steps 1 and 2).

member nodes discard themselves from the top list of candidates and wait for receiving a "MSG_CHREADY" message to adhere to the cluster. If they do not receive such a message and they are on the top list of candidates then, they declare themselves leaders (cluster-heads).

Similarly, a member node leaves this state when it receives a "MSG_NOTCH" message sent by a leader. Consequently, it goes back to the non-adhered state.

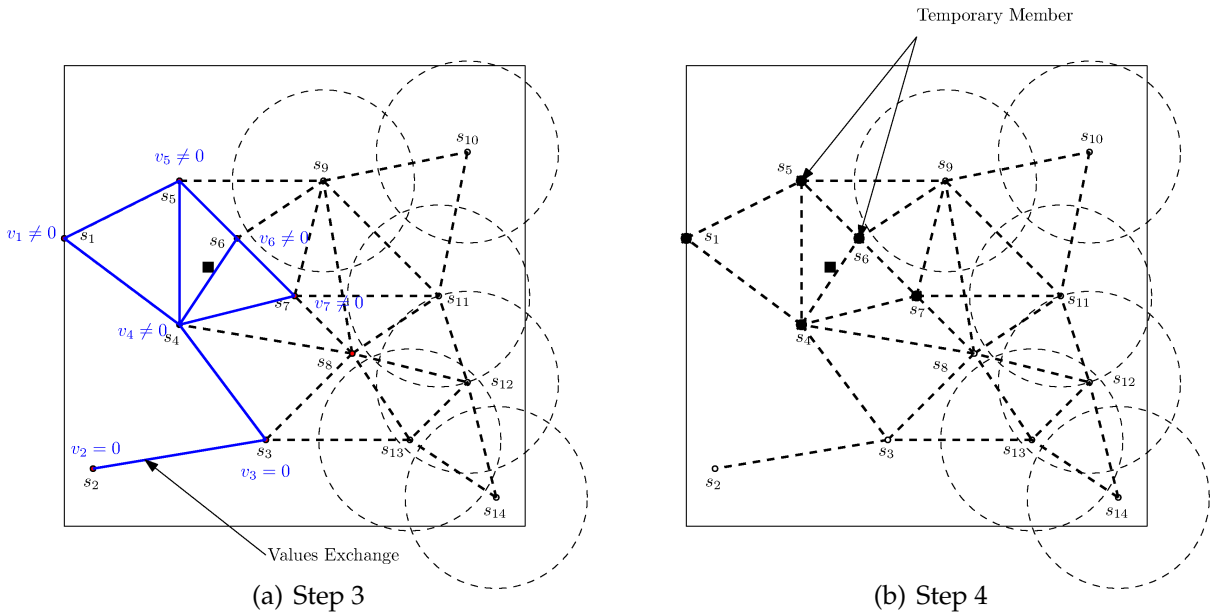
Figure 3.7 illustrates this last step. Node s_1 sends a "MSG_CHREADY" message to the temporary-member nodes s_4, s_5, s_6 and s_7, which adhere as members in the created cluster.

3.2.2 Cluster Reconfiguration

The leader node checks the target state estimation to decide about the cluster reconfiguration. When it detects that the target is lost, then it performs the following two tasks:

1. Sending back a "MSG_JOIN" message to force the senders of "MSG_CHREADY" or "MSG_CREATECLUSTER" messages to adhere to its cluster.

2. Updating the list of the cluster members upon receiving a "MSG_JOIN" or a "MSG_QUITCLUSTER" messages.



(a) Step 3 (b) Step 4

Figure 3.6: Cluster formation operations (Steps 3 and 4).

A leader leaves this state when one of the following events occurs:

- Receiving a "MSG_JOIN" message to become a member of the cluster of the message sender.
- Losing the target. In this case, the cluster should be reconfigured and the nodes return back to the non-adhered state.

The cluster reconfiguration operation consists of updating the list of candidates and informing the member nodes that the leadership has changed using a "MSG_NOTCH" message. Upon receiving this message, the nodes trigger a new election process that may include the previous members.

Figure 3.8 illustrates this operation. When the target moves away in the right direction, node s_1 (the cluster-head) can no longer detect it. Then it sends a "MSG_NOTCH" message to all its cluster members. This message triggers a new election process. Note that, nodes s_6 and s_7 are also members of the newly created cluster because they still detect the target in the new election process.

On the contrary to the scheme proposed in [103], our dynamic clustering protocol prevents the tracking of one target by multiple clusters by letting only direct neighbors to adhere to the cluster and forcing the other nodes to return back to

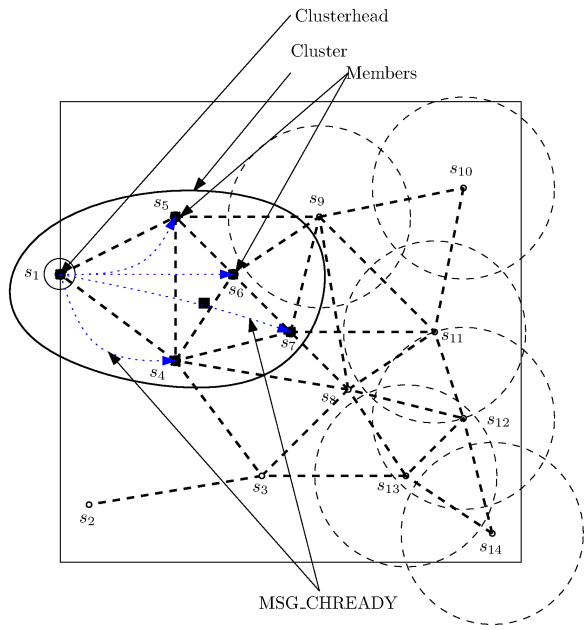


Figure 3.7: Cluster formation operations (Step 5).

the sleep state. After construction of the tracking cluster, the data fusion phase takes place and generates a target state estimation. All the above-described steps are summarized in the state-transition diagram of DKF_DC in Figure 3.9.

3.2.3 Target State Estimation

The member nodes of the current cluster perform the sampling to detect the target. They compute their information matrix u_i and U_i (including the leader node) as follows:

$$u_i = H_i^T R_i^{-1} z_i \tag{3.9}$$

$$U_i = H_i^T R_i^{-1} H_i \tag{3.10}$$

Equations 3.9 and 3.10 represent respectively the measurements' information and the measurements' error. The Kalman Filter fuses the estimation errors to generate an updated state estimation. After that, each node broadcasts a message $m_i = \{u_i, U_i, \bar{x}_i\}$ containing the measurements u_i, the measurements' error U_i, and the last state estimation $\bar{x}_i$, to all the cluster members. Each node waits for receiving such messages from the other members to fuse the information matrix

.

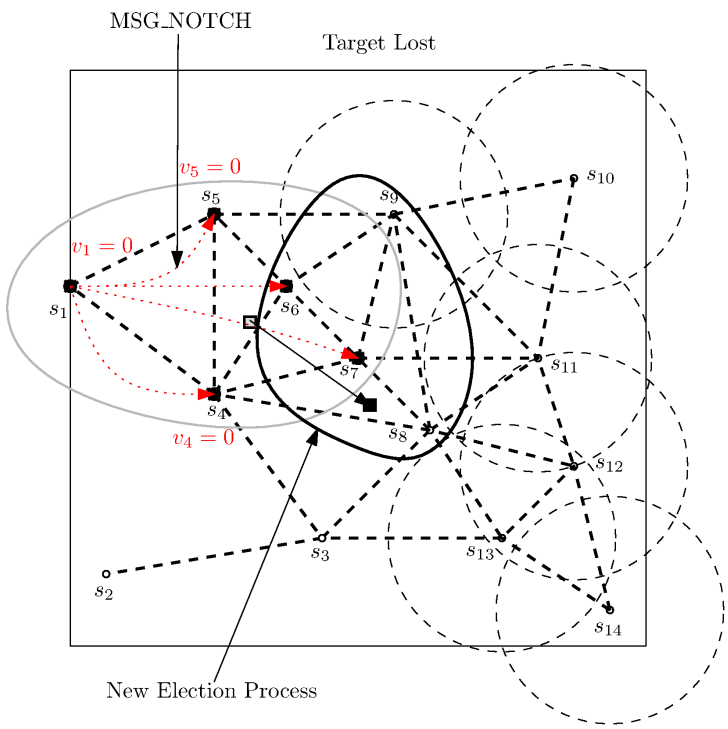


Figure 3.8: Cluster reconfiguration.

and generate the vectors y_i and S_i given by:

$$y_i = \sum_{j \in J_i} u_j$$

and

$$S_i = \sum_{j \in J_i} U_j$$

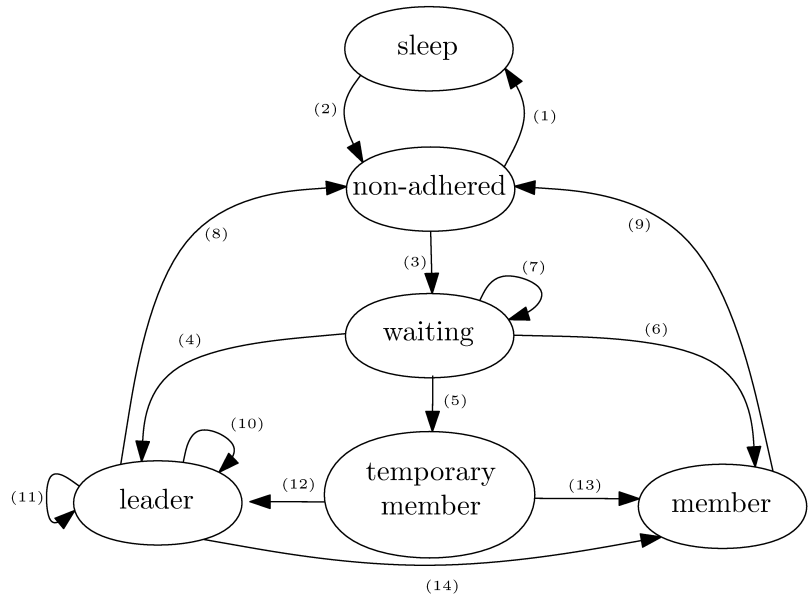
At the end of the data fusion phase, each node estimates the target state using the KCF equations:

$$M_i = (P_i^{-1} + S_i)^{-1} \tag{3.11}$$

$$\hat{x}_i = \bar{x}_i + M_i(y_i - S_i\bar{x}_i) + \gamma M_i \sum_{j \in N_i} (\bar{x}_j - \bar{x}_i) \tag{3.12}$$

After that, each cluster-member node updates its state using equations 3.13 and 3.14:

$$P_i = AM_iA^T + BQB^T \tag{3.13}$$



(1): Timer expired

(2): recev(MSG_INRUSION) or recev(MSG_WAKEUP)

(3): send(MSG_WAKEUP) or send(MSG_CREATECLUSTER)

(4): Timer expired(ID=Top List Candidate)

(5): Timer expired and ($CurrentNode \neq TopListCandidate$)

(6): recev(MSG_CHREADY), send(MSG_JOIN)

(7): Update Candidate List

(8): send(MSG_NOTCH) send(MSG_CREATECLUSTER)

(9): recev(MSG_NOTCH)

(10): recev(MSG_CHREADY) or recev(MSG_CREATECLUSTER) and send(MSG_JOIN)

(11): recev(MSG_JOIN) or recev(MSG_QUITCLUSTER) and Update Candidate List

(12): Timer expired and ($CurrentNode = TopListCandidate$)

(13): recev(MSG_CHREADY) send(MSG_JOIN)

Figure 3.9: State-transition diagram of the proposed clustering protocol.

$$\bar{x}_i = A\hat{x}_i \tag{3.14}$$

Each node can use a combined formula of the following information to update its list of candidates:

- The distance to the target.
- The number of active nodes in the cluster.
- The residual energy of the nodes.

Based on this information, the node on the top list of candidates is elected as a leader.

3.3 Simulations and Results

We use TOSSIM simulator [104] to implement and evaluate DKF_DC algorithm. TOSSIM is a discrete-time simulator of TinyOS operating system for wireless sensor nodes.

We compare DKF_DC with three different target tracking schemes described at the beginning of this chapter which are: (1) the Centralized Kalman Filter (CKF) [39], (2) the Distributed Kalman Filter with limited sensing range (DKF_LSR) [41] and (3) the Distributed Kalman Filter with gossip communications (DKF_GOSSIP) [97]. The CKF is considered here as the base reference for our comparisons.

3.3.1 Simulation Setup

We vary the sampling period, the network size (or density) and the target speed and we measure the energy consumption and the estimation quality. The communication range is set to 50 m, the sensing range is set to 15 m and the mobility model of the target is Gauss-Markov.

The nodes' energy consumption is evaluated using POWERTOSSIM and the estimation quality is measured by the mean square error between the real target position and the estimated target position:

$$\epsilon = \sqrt{(x - \hat{x})^2 + (y - \hat{y})^2}$$

In the following subsections, we present the simulation results we have obtained in terms of energy consumption and estimation quality.

3.3.2 Energy Consumption

In the first set of simulations, we create a network of 100 randomly deployed nodes in a $200 \times 200m^2$ 2D area. We vary the sampling period in $\{1s, 2s, 3s\}$. The obtained results are depicted in Figure 3.10.

As we can see in Figure 3.10, the network average energy consumption of the different simulated schemes is inversely proportional to the sampling period time because of the number of data messages exchanged between the nodes. CKF and DKF_LSR schemes consume much more energy than DKF_GOSSIP and DKF_DC because CKF is based on a centralized approach and DKF_LSR does not limit the number of participating nodes. DKF_DC reduces the network energy consumption thanks to the dynamic clustering protocol that limits the number of nodes involved in the tracking task.

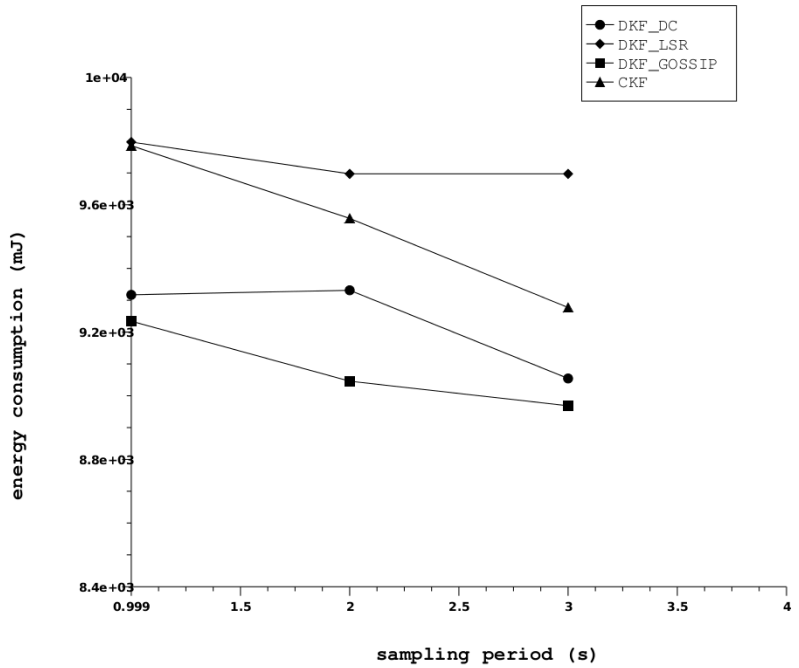


Figure 3.10: Energy consumption of the network vs. sampling period.

We also evaluate the network energy consumption of the simulated schemes with respect to the network size. We vary the network size in $\{225, 150, 100\}$ and we set the sampling period to $1s$. The obtained results are shown in Figure 3.11.

We can see that the dense nature of WSN impacts the network energy consumption. In CKF and DKF_LSR schemes, the number of tasking nodes is bigger than in DKF_DC which induces excessive energy consumption because all the nodes execute the sensing operation in each tracking step. Note that the sensing and the communication energy consumptions are bigger than the processing energy consumption which explains the performance of our DKF_DC scheme.

3.3.3 Quality of Estimation

The quality of estimation of the simulated schemes is evaluated for different sampling periods (Figure 3.12(a), Figure 3.12(b) and Figure 3.13 show the quality of estimation for $1s$, $2s$ and $3s$ sampling period, respectively).

As we can see on the three figures, the estimation quality of our proposed scheme DKF_DC is less than those of CKF and DKF_LSR, and the CKF scheme outperforms all the other schemes with respect to the different sampling periods.

In DKF_DC scheme, the reduced set of participating nodes in the estimation

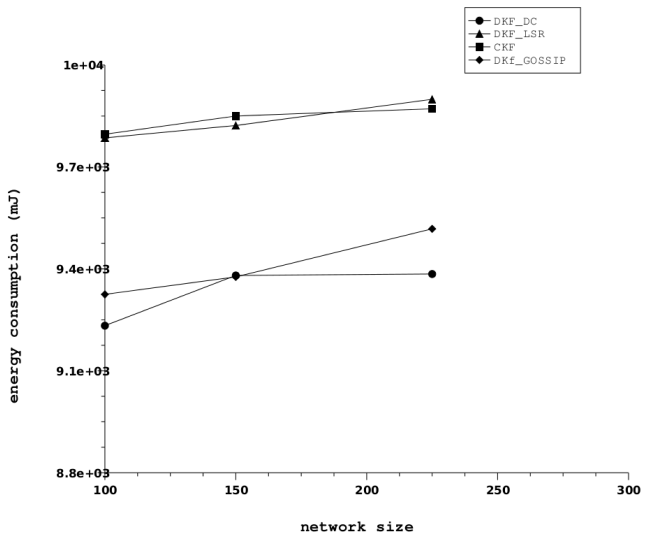
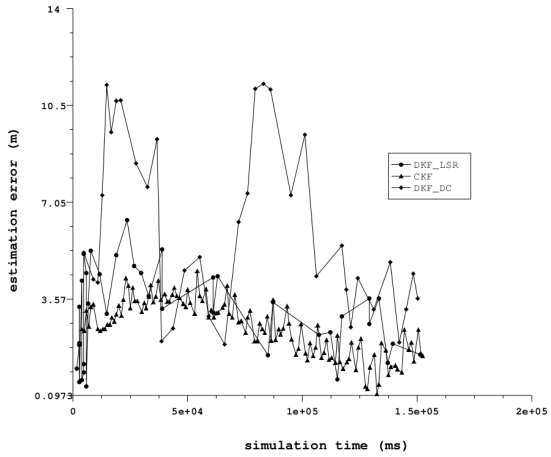
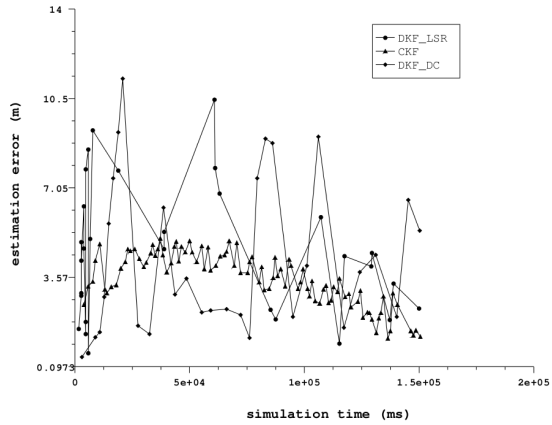


Figure 3.11: Energy consumption of the network vs. network density.



(a) Estimation quality for a sampling period of $1s$



(b) Estimation quality for a sampling period of $2s$

Figure 3.12: Quality of estimation vs. sampling period.

process may decrease the total utility of the nodes when inappropriate nodes are chosen. Therefore, including all the nodes in the estimation process and considering a uniform distributed noise model, improves the quality of estimation and speeds-up the model convergence, because lot of amount of data are fused despite the fact that they contribute or not in the estimation process.

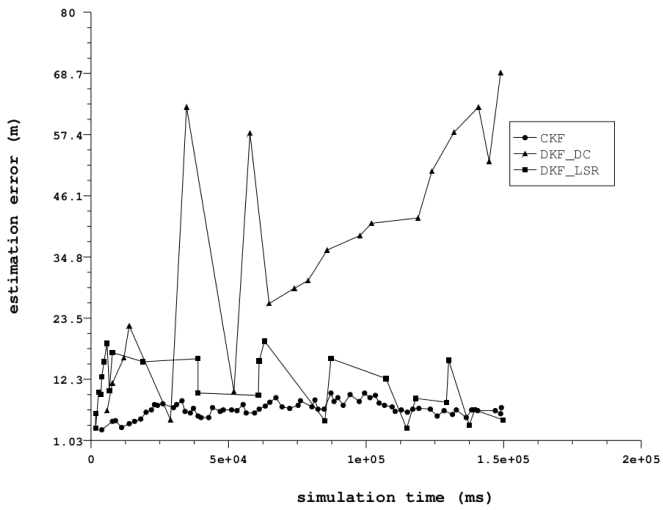
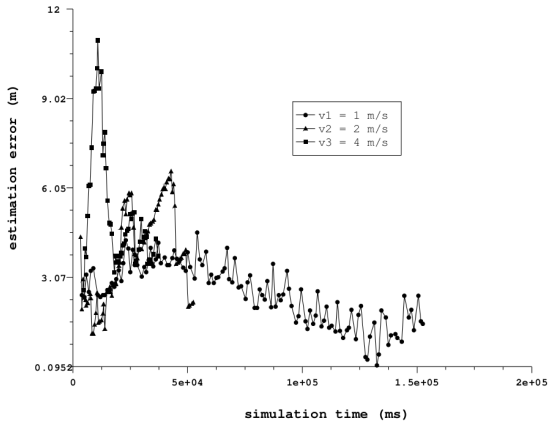
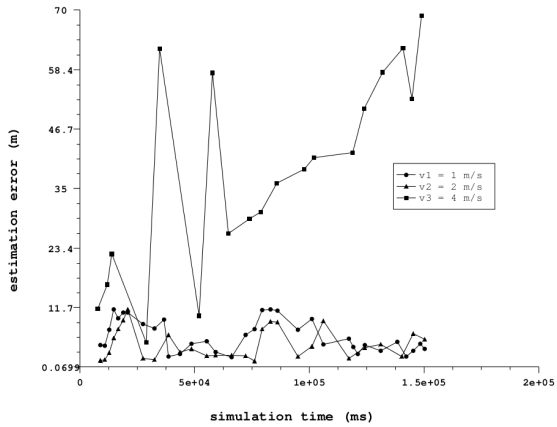


Figure 3.13: Estimation quality for a sampling period of $3s$.

Peaks can be also observed on the graphs of DKF_DC due to the cluster reconfiguration process. Figure 3.13 shows that with a large sampling period, our method presents poor estimation quality because of the latency generated by the clustering protocol. This can be considered as a *drawback* of DKF_DC scheme.

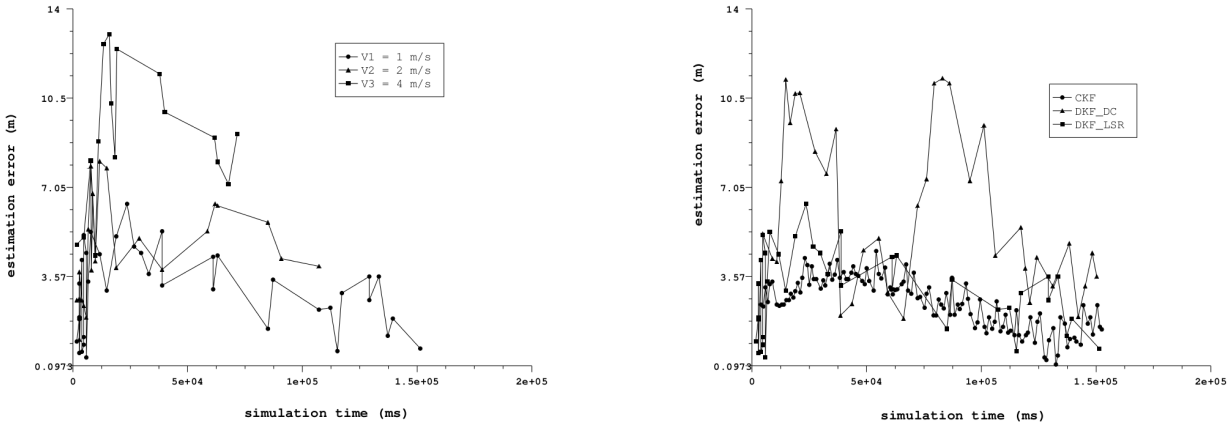


(a) CKF estimation quality vs. target speed



(b) DKF_DC estimation quality vs. target speed

Figure 3.14: Performance vs. target speed (1).



(a) DKF_LSR estimation quality vs. target speed

(b) Estimation quality of the different schemes

Figure 3.15: Performance vs. target speed (2).

Finally, the quality of estimation of CKF, DKF_LSR and DKF_DC schemes with respect to the target speed is presented in Figure 3.14(a), Figure 3.14(b) and Figure 3.15(a), respectively. We vary the target speed in $\{1m/s, 2m/s, 4m/s\}$ in each simulation.

As we can see in Figure 3.14(a), the estimation error of CKF scheme decreases for the different target speeds. Indeed, we can observe that the Kalman Filter presents some peaks in the beginning of the estimation process but they disappear after that and the estimation error converges to zero due to the recursive nature of the KF algorithm.

In Figure 3.14(b), we can see that for a target moving with a relatively low speed, the estimation error of DKF_DC scheme converges to zero. However, when the target speeds up, it overcomes the cluster reconfiguration mechanism. Thus, we observe a degeneration of the estimation quality for targets with velocity $v_3 = 4m/s$. A recovery process should be setup here to deal with this problem.

The estimation error of DKF_LSR scheme is presented in Figure 3.15(a). As we can see, this scheme converges also for the different target speeds but in a slow rate compared with CKF, because the state vector and the covariance matrix are exchanged between a large number of nodes which slow down the convergence speed.

Figure 3.15(b) shows the convergence speed of the different simulated schemes. The estimation error of all the schemes converges to zero but with different rates. CKF converges quickly than the other schemes.

3.4 Conclusion

In this chapter, we proposed a Distributed Kalman Filtering algorithm coupled with Dynamic Clustering protocol (DKF_DC) that helps reducing the network energy consumption in WSN with limited sensing range.

DKF_DC prevents nodes from awakening up periodically and limits the selection process within the area close to the target. It structures the detecting nodes as a cluster that moves along with the target trajectory.

We implemented our DKF_DC scheme for simulation and we compared it with three other schemes that use the Distributed Kalman Filter algorithm as a prediction algorithm.

The results show a clear improvement of the energy efficiency of the network in DKF_DC than in the other schemes. However, the estimation quality has slightly degraded when tracking high-speed targets.

Exploiting the data correlation existing between the sensor readings and considering a realistic detection model of the sensor nodes can help improving data quality while achieving energy efficiency. We consider that this is a promising track for energy optimization in target tracking applications as well as other collaborative applications for data collection in WSN. This aspect is explored in the next chapter.

Chapter 4

Energy Modeling under Data Precision Constraints

As SENSOR NODES HAVE LIMITED sensing and communication ranges, they *must* cooperate with each others to forward sensed data to the base station after possible aggregation.
However, for optimizing the energy consumption in WSN, the research efforts have focused on the communication layer by using general-purpose methods previously applied to Mobile Ad hoc Networks (MANET). In this chapter, we argue and prove that *data* can also help improving the energy consumption in the data collection process in WSN.

Our proposition is based on two fundamental aspects:

- First, the density of deployed nodes in the surveillance area makes them close to each others which increases the probability to generate correlated data about the *localized* phenomenon. Thus, the sensor nodes can take profit from this spatial and temporal correlated data to reduce the energy consumption in wireless communications, and to schedule the future sampling operations.

- Second, the activation of the whole network to perform data collection tasks is not practical. A *natural* way to do so is to put all the nodes in a low-power operating mode except the ones who detect the events or relay the messages or process the data. However, to prevent the degradation of the quality of data, the nodes with the most *meaningful* information should be selected for sampling operations and the ones with the maximum energy resources should be selected for data relaying and processing operations.

From this point of view, an optimization problem arises here: *how to select the best nodes in terms of quality of data with the minimum cost in terms of energy consumption?*

To resolve this problem, we first start by defining the network topology as a set of cluster-trees, each of which collects the data about the nearby monitored phenomenon. We use this network structure because of its intrinsic characteristics in terms of network scaling and data aggregation. The problem is then transformed into a problem of finding the most energy-efficient data reporting cluster-trees (forest) that maintain the data precision within a certain interval and consume the minimum energy during the sampling, transmitting, receiving and processing operations.

We call this problem EMDP: *Energy Minimization under the Data Precision constraint*, and we present its formal definition in Section 4.2.

4.1 Background and Related Work

In this section, we present some related works that treat EMDP−like problems. Correlated data gathering subject to latency and energy-balancing constraints is addressed in [105] and [106], respectively. The authors of [105] predict the practical limits on achievable energy improvements and usable delays when data correlation is included in the optimization problem. They found that the energy improvement is not significant compared to cost and complexity of the problem. On the other hand, the authors of [106] showed that when all the nodes act as samplers, energy is naturally balanced. However, when only a subset of the nodes acts as data sources, aggregation helps balancing energy among them. We note here that in [105] and [106], the optimization effort is done in centralized fashion, and in our case, we relax the delay constraint but we keep the centralized optimization.

An interesting result reported by [107] is that the standard solutions to the maximum flow problem as a model of EMDP-like problems may be sub-optimal because they do not consider data correlation. Thus, including correlation when solving an EMDP−like problem is necessary and it can be done using two methods as suggested by the authors: the Slepian-Wolf coding and the joint-entropy coding with explicit communications. EMDP problem fits in the second method.

Another EMDP-like problem is the RSP problem defined in [108] where aggregation is used to maximize the network lifetime. The authors define the RSP problem as a joint routing/aggregation linear optimization problem to which they propose a heuristic algorithm based on load balancing to reduce the complexity.

A worth noting point here is that in a cluster-tree topology, sensor nodes may have wide set of clusters to which they can adhere. Thus in [109], the authors propose an association scheme based on linear optimization to improve the network throughput and to balance energy. However, their scheme is only applied when strong overlapping exists between the clusters.

We also note here that the energy minimization technique is often based on the optimization of the distance of transmission. When the distance between the nodes and their cluster-heads is small, their energy consumption will be small too. That is what we find in [110] under the constraint of clusters balancing.

Our work is closely related to the model proposed in [111] for the PCEO problem. While the objective functions are similar, the constraint on the data precision differs. The authors of [111] define the data precision as the mean square error between the estimated readings and the real values, i.e. data distortion. In our model, we define the data precision as a data sensitivity interval. We do not consider the data rate between the nodes because we assume it is constant, and we do not try to restore the non-collected data from the collected data.

In this chapter, we propose a *formal* definition of the EMDP problem using a binary integer linear program [112]. We discuss and analyze the resulting model and we prove certain of its properties for validation.

4.2 Problem Definition & Formulation

Modeling the EMDP problem as a BILP program is motivated by finding the exact solution to the problem of energy minimization under data precision constraints.

The correlation between data of the nodes is fundamental here. As a simple definition, we propose that two nodes have correlated readings if the absolute difference between these readings is less than a threshold ϵ (a user-defined constant). We also assume that:

- The structure of the network (topology) is defined as a set of data reporting cluster-trees, each of which is rooted at a special node called the cluster-head.
- The nodes send their readings periodically for k rounds to their corresponding relay nodes in the collection tree in order to be forwarded to the cluster-heads (roots).
- All the trees should have a depth at most of n hops.
- A selected node at round t, $0 \leq t \leq k$ should have a *non-zero* energy budget ($E_i^t > 0$) and the readings of *all* the nodes in the same reporting tree should be correlated.

In addition, as the inputs of the EMDP problem, we have the data sensitivity parameter τ, a reference value v_0 and the set of nodes readings $\hat{X}$.

Definition 1 (EMDP)**.** *Given a WSN topology $G(V, L)$, a number of data collection rounds $k \geq 1$, a sensitivity parameter $0 < \tau < 1$ and a reference value v_0. Find the*

successive minimum energy consumption data reporting tree(s) over the k rounds that maintain the data precision in the sensitive interval $[(1-\tau)v_0, (1+\tau)v_0]$ in each round.

Optionally, we can define the LMEB problem (*Lifetime Maximization under the constraint of the nodes' Energy Budget*) as the dual problem of EMDP.

Definition 2 (LMEB)**.** *Given two real numbers $E > 0$ and $\tau > 0$. All the sensor nodes have initial energy budget E. Find the maximum number of data collection rounds that maintain the data precision in the interval $[(1-\tau)v_0, (1+\tau)v_0]$ and no node runs over its energy budget.*

4.2.1 System Model & Parameters

Given a large number of sensor nodes statically and densely deployed in the surveillance area that report data to a base station after aggregation. The nodes can be grouped on *local* clusters to monitor a local phenomenon. They communicate in multi-hop fashion. The nodes may have different roles: sampling nodes, relay nodes and cluster-heads.

We represent the WSN by an undirected graph $G(V, L)$ where $V = \{s_1, s_2, \ldots, s_m\}$ is the set of vertex representing the sensor nodes and L is the set of edges representing the communication links. The network size is $m = |V|$. An edge $e_{i,j} = (s_i, s_j) \in L$ exists if a direct transmission link exists between node s_i and node s_j. We have $e_{i,j} \in L \Rightarrow e_{j,i} \in L$. We denote by E_i^t the energy budget remaining in node s_i at round t. This energy is subject to decrease after each operation of sampling, transmitting, receiving or processing operations as we will show later in subsection 4.2.2.

The nodes report data periodically for k rounds. In each round, the data precision is ruled by a parameter τ and a reference value v_0. They are organized in cluster-trees of maximum depth n.

The amount of energy consumed in sampling, receiving, transmitting and processing operations are respectively denoted by e_s, e_r, $e_{i,j}^t$ and e_p. Each node s_i may generate a set of readings $\hat{X}_i = \{\hat{v}_i^{t_1}, \hat{v}_i^{t_2}, \ldots, \hat{v}_i^{t_{k_i}}\}$, where $\hat{v}_i^t$ is node's s_i reading in round t and v_i^t is the real value of the physical phenomenon at the position of node s_i. $\hat{X} = \bigcup_{t=1,2,\cdots,k} \hat{X}_i$ is the set of all the readings of all the nodes.

We assume also that each node s_i can construct the set of its direct (one-hop) neighbors $N_i = \{s_j \in V, d(s_i, s_j) \leq R_c\}$ and the set of its $n-$hop neighbors $N_i^n = N_i^{n-1} \bigcup_{s_j \in N_i^{n-1}} N_j$, recursively.

4.2.2 Binary Integer Linear Programming Formulation

We define two sets of decision variables:

- $a_i^t = 1$ if node s_i participates in the sampling operations at round t. 0, otherwise.
- $b_{i,j}^t = 1$ if node s_i is parent of node s_j in a collection tree[1] at round t. 0, otherwise.

Based on these decision variables, we define the following sets of nodes that help characterizing the EMDP problem:

- $A^t = \{s_i \in V, a_i^t = 1\}$ is the set of all selected nodes at round t. $A = \bigcup_{t=1}^{k} A^t$ is the set of all selected nodes in all the rounds.
- $C^t = \{s_i \in V, a_i^t = 1 \,\&\, \sum_{s_j \in N_i} b_{j,i} = 0\}$ is the set of all elected cluster-heads at round t. $C = \bigcup_{t=1}^{k} C^t$ is the set of all elected cluster-heads in all the rounds.
- T_i^t is the data collection tree rooted at node s_i at round t. It is defined using the variables a_i^t and $b_{i,j}^t$ as follows:

$$\begin{aligned} T_i^t =& \{s_i \in C^t\} \cup \{s_j \in A^t : \exists s_{j_1}, s_{j_2}, \ldots, s_{j_k}, s_i \in A^t : \\ & b_{j,j_1}^t = 1, b_{j_1,j_2}^t = 1, \ldots, b_{j_k,i}^t = 1 \text{ and } s_i \in C^t\} \end{aligned}$$

- $T^t = \bigcup_{s_i \in C^t} T_i^t$ is the set of all data collection trees (forest) at round t.

The total energy consumption in sampling, data reporting and processing in *one round* t is given by:

$$E_{round}^t = e_s \sum_{s_i \in V} a_i^t + (e_r + e_p) \sum_{s_i \in V} \sum_{s_j \in N_i} b_{i,j}^t + \sum_{s_i \in V} \sum_{s_j \in N_i} b_{j,i}^t \cdot e_{i,j}^t \tag{4.1}$$

As we can see in equation 4.1, the amount of the energy consumption depends on the roles of the nodes:

- Sampling: for all selected nodes.
- Receiving and processing (aggregating): for all selected nodes except the leaves (the nodes with no childes).
- Transmitting: for all the nodes except the roots (the cluster-heads).

Our program should determine who are the leaf nodes, the relay nodes and the root (cluster-head) nodes based on the following objective function:
Minimizing the total energy consumption over all the k rounds[2]:

$$\min \sum_{t=1}^{k} E_{round}^t \tag{4.2}$$

[1] To be determined by the program.

[2] The amount of energy consumption depends on the variables a_i^t and $b_{i,j}^t$ and $e_{i,j}^t$. i.e. it depends mainly on the sensor selection for sampling, data relaying and data aggregation. Theses aspects are defined in the constraints section.

s.t.

$$\sum_{s_i \in V} a_i^t - \sum_{s_i \in V} \sum_{s_j \in N_i} b_{j,i}^t \geq 1, \forall t : 1 \leq t \leq k \tag{4.3}$$

$$\sum_{s_j \in N_i} b_{i,j}^t \leq a_i^t \cdot (|N_i| - 1), \forall s_i \in V, \forall t : 1 \leq t \leq k \tag{4.4}$$

$$\sum_{s_j \in N_i} b_{j,i}^t \leq a_i^t, \forall s_i \in V, \forall t : 1 \leq t \leq k \tag{4.5}$$

$$\sum_{s_i \in V} a_i^t \cdot \hat{v}_i^t \geq |V| \cdot (1 - \tau) v_0, \forall t : 1 \leq t \leq k \tag{4.6}$$

$$\sum_{s_i \in V} a_i^t \cdot \hat{v}_i^t \leq |V| \cdot (1 + \tau) v_0, \forall t : 1 \leq t \leq k \tag{4.7}$$

$$\begin{aligned} &(a_i^t - \sum_{s_j \in N_i} b_{j,i}^t) \cdot E_i^t \geq (a_i^t - \sum_{s_j \in N_i} b_{j,i}^t - 1 + a_j^t) \cdot E_j^t, \\ &\forall s_i \in V, \forall s_j \in N_i^n, \forall t : 1 \leq t \leq k \end{aligned} \tag{4.8}$$

$$\begin{aligned} &a_j^t - \sum_{s_p \in N_j : d(s_p, s_i) < d(s_j, s_i)} b_{p,j}^t \leq 1 - (a_i^t - \sum_{s_q \in N_i} b_{q,i}^t), \\ &\forall s_i \in V, \forall s_j \in N_i^n, \forall t : 1 \leq t \leq k \end{aligned} \tag{4.9}$$

$$\begin{aligned} E_i^{t+1} = E_i^t - [a_i^t \cdot e_s + \sum_{s_j \in N_i} b_{i,j}^t \cdot (e_r + e_p) + \\ \sum_{s_j \in N_i} b_{j,i}^t \cdot e_{i,j}^t], \forall s_i \in V, \forall t : 1 \leq t \leq k - 1 \end{aligned} \tag{4.10}$$

$$a_i^t \in \{0, 1\}, b_{i,j}^t \in \{0, 1\}, \forall s_i \in V, \forall s_j \in N_i, \forall t : 1 \leq t \leq k \tag{4.11}$$

Constraint 4.3 ensures that at least, one cluster should be constructed in each round. It is easy to observe that a cluster-head s_i at round t is characterized by (see Figure 4.1(d)):

$$a_i^t - \sum_{s_j \in N_i} b_{j,i}^t = 1$$

When it is summed over all the nodes of the network, the right side of equation 4.3 becomes greater than one.

Constraints 4.4 and 4.5 ensure that a non-selected node should not have a child or a parent, i.e. when a node s_i is selected, it must have at most one parent and $|N_i| - 1$ child. Reciprocally, before being a parent or a child, a node should first be selected ($a_i^t = 1$).

Constraints 4.6 and 4.7 specify the criterion of node selection: the nodes are selected based on the cumulative data among all the nodes which should fit in the interval defined by the sensitive parameter τ and the reference value v_0.

Constraint 4.8 states that an elected cluster-head should have the maximum energy budget among all its $n-$hop neighbors. It ensures that for a node s_i with equation $a_i^t - \sum_{s_j \in N_i} b_{j,i}^t = 1$, all its $n-$hop *selected* neighbors (with $a_j^t = 1$) should have their energy budget at round t less than E_i^t. Otherwise, if $a_j^t = 0$, i.e. s_j is not selected, inequality 4.8 ensures that $E_i^t \geq -E_j^t$ which is evident. On the other hand, when node s_i is not elected as a cluster-head, i.e. $a_i^t - \sum_{s_j \in N_i} b_{j,i}^t = 0$, the right side of inequality 4.8 becomes 0 and for any node $s_j \in N_i$, either selected or non-selected. Therefore, inequality 4.8 is verified.

Constraint 4.9 specifies the data forwarding rule. When a node s_i is elected as a cluster-head among all its $n-$hop neighbors, i.e.

$$a_i^t - \sum_{s_q \in N_i} b_{q,i}^t = 1$$

the left side of inequality 4.9 is set 0 thereby forcing the right side of this inequality to be 0 because of the constraint of binary variables (constraint 4.11). This ensures that the other selected nodes within the $n-$hop neighborhood of node s_i except the cluster-head (with $a_j^t = 1$) should have their parents, i.e.

$$\sum_{s_p \in N_j} b_{p,j}^t = 1$$

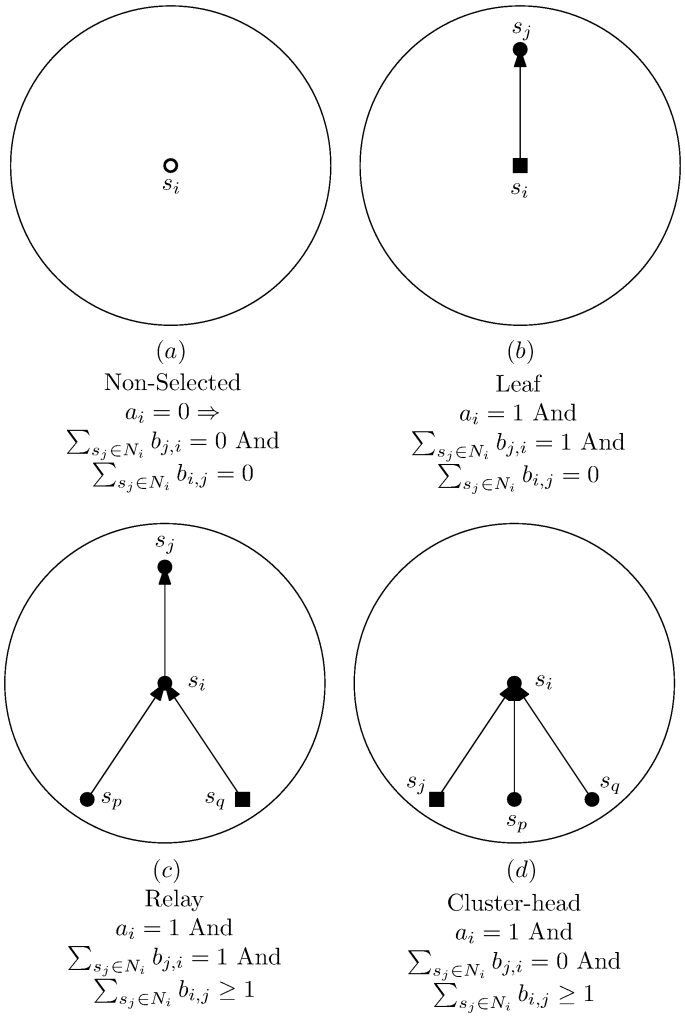
as shown in Figure 4.1. In addition to its role in connecting all the selected nodes, this constraint ensures also that the data are forwarded toward the cluster-heads. This property is not evident and it needs to be proved mathematically. The proof is given in Section 4.2.3

Constraint 4.10 specifies the energy updating rule of the nodes between successive rounds. Each selected node s_i for sampling at round t (with $a_i^t = 1$) consumes e_s (Joules) of energy. Similarly, when it receives and process data from its child nodes s_j i.e. with $b_{i,j}^t = 1$, it spends $e_r + e_p$ of its energy budget. And finally, when it serves as a relay node, it sends data to its parent s_j and consumes $e_{i,j}^t$. Note that all these amounts of energy are independent from the data rate between the nodes.

Constraint 4.11 ensures that decision variables a_i^t and $b_{i,j}^t$ are binary.

The above-discussed constraints can be grouped into three categories:

- Constraints on the network structure (cluster-trees): 4.3, 4.4, 4.5 and 4.9.
- Constraints on the energy: 4.8 and 4.10.
- Constraints on the data precision: 4.6 and 4.7.



(a)
Non-Selected
$a_i = 0 \Rightarrow$
$\sum_{s_j \in N_i} b_{j,i} = 0$ And
$\sum_{s_j \in N_i} b_{i,j} = 0$

(b)
Leaf
$a_i = 1$ And
$\sum_{s_j \in N_i} b_{j,i} = 1$ And
$\sum_{s_j \in N_i} b_{i,j} = 0$

(c)
Relay
$a_i = 1$ And
$\sum_{s_j \in N_i} b_{j,i} = 1$ And
$\sum_{s_j \in N_i} b_{i,j} \geq 1$

(d)
Cluster-head
$a_i = 1$ And
$\sum_{s_j \in N_i} b_{j,i} = 0$ And
$\sum_{s_j \in N_i} b_{i,j} \geq 1$

Figure 4.1: Cases of direct neighborhood of nodes.

4.2.3 Proof of Constraint 4.9

Proposition 1. *For a cluster-head s_i, a selected node s_j member of the same cluster as s_i, that satisfies the constraint 4.9 is a relay node or a leaf node.*

Proof. A leaf or a relay node s_j at data collection round t should satisfy the following condition (as shown in Figure 4.1(b) and Figure 4.1(c)):

$$a_j^t = 1 \text{ and } \sum_{s_p \in N_j} b_{p,j}^t = 1 \tag{4.12}$$

For simplicity and without loss of generality, we restrict the proof to only one round in which the constraint 4.9 becomes:

$$\forall s_j \in N_i^n, a_j - \sum_{s_p \in \boldsymbol{F}_{j,i}} b_{p,j} \leq 1 - (a_i - \sum_{s_q \in N_i} b_{q,i}) \tag{4.13}$$

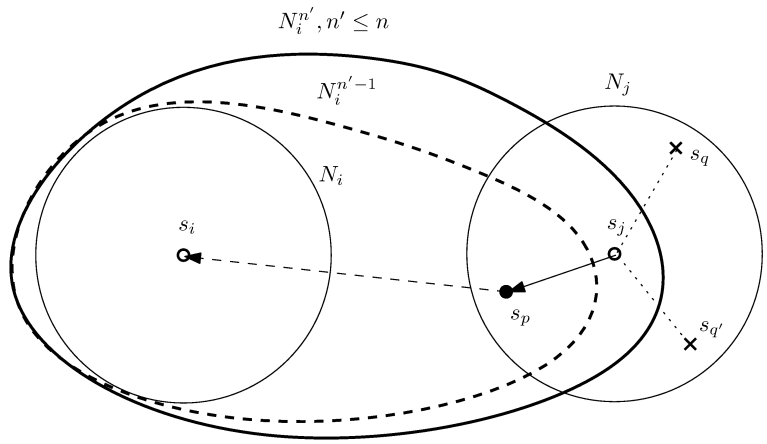


Figure 4.2: Illustration of the data forwarding constraint.

With:

$$\boldsymbol{F}_{j,i} = \{s_p \in N_j : d(s_p, s_i) < d(s_j, s_i)\}$$

is the *direct forwarding* set as illustrated in Figure 4.2. Since node s_i is selected as a cluster-head, condition 4.14 is achieved:

$$a_i - \sum_{s_q \in N_i} b_{q,i} = 1 \tag{4.14}$$

Then, constraint 4.9 gives:

$$\forall s_j \in N_i^n, a_j - \sum_{s_p \in \boldsymbol{F}_{j,i}} b_{p,j} \leq 0 \Rightarrow \forall s_j \in N_i^n, a_j - \sum_{s_p \in \boldsymbol{F}_{j,i}} b_{p,j} = 0 \tag{4.15}$$

Based on this last condition (4.15), two cases arise:

Case 1. $a_j = 0$ and $\sum_{s_p \in \boldsymbol{F}_{j,i}} b_{p,j} = 0$. This case does not fit into Proposition 1 because it does not consider non-selected nodes.

Case 2. $a_j = 1$ and $\sum_{s_p \in \boldsymbol{F}_{j,i}} b_{p,j} = 1$. Since $\boldsymbol{F}_{j,i} \subset N_j$, condition 4.12 is effectively verified. □

Proposition 2. *Constraint 4.9 ensures data forwarding from leaf and relay nodes to their corresponding cluster-heads.*

Proof. For simplicity and without loss of generality, we consider only one round of data collection t. We have to prove that for a cluster-head s_i and a member node $s_j \in N_i^n$, we have:

$$\begin{aligned}&\exists s_{j_1} \in \boldsymbol{F}_{j,i}, \exists s_{j_2} \in \boldsymbol{F}_{j_1,i}, \ldots, \exists s_{j_n} \in N_i : \\ &b_{j_1,j} = 1, b_{j_2,j_1} = 1, \ldots, b_{j_n,i} = 1\end{aligned} \tag{4.16}$$

With:

$$\boldsymbol{F}_{j,i} = \{s_p \in N_j : d(s_p, s_i) < d(s_j, s_i)\}$$
$$\boldsymbol{F}_{j_1,i} = \{s_p \in N_{j_1} : d(s_p, s_i) < d(s_{j_1}, s_i)\}$$
$$\ldots$$
$$\boldsymbol{F}_{j_n,i} = \{s_p \in N_{j_n} : d(s_p, s_i) < d(s_{j_n}, s_i)\}$$

This proof is done recursively on n.

Base case. ($n = 1$) It means nodes s_j and s_i are direct neighbors. Then, the constraint 4.9 becomes:

$$\forall s_j \in N_i, a_j - \sum_{s_p \in N_j : d(s_p,s_i) < d(s_j,s_i)} b_{p,j} \leq 1 - (a_i - \sum_{s_q \in N_i} b_{q,i}) \tag{4.17}$$

If node s_i is elected as a cluster-head, then the right side of inequality 4.17 becomes 0, which gives the following equation:

$$\forall s_j \in N_i, a_j - \sum_{s_p \in N_j : d(s_p,s_i) < d(s_j,s_i)} b_{p,j} = 0 \tag{4.18}$$

Two cases are considered here:

Case 1. $a_j = 1$, which means that node s_j is selected. Therefore, equation 4.18 gives:

$$\sum_{s_p \in N_j : d(s_p,s_i) < d(s_j,s_i)} b_{p,j} = 1 \tag{4.19}$$

If the distance between s_j and s_i is the smallest among all the neighbors of node s_i (as shown in Figure 4.3), then:

$$\nexists s_p \in N_j : d(s_p, s_i) < d(s_j, s_i), b_{p,j} = 1 \tag{4.20}$$

Condition 4.16 remains true only for $j = j_n$.

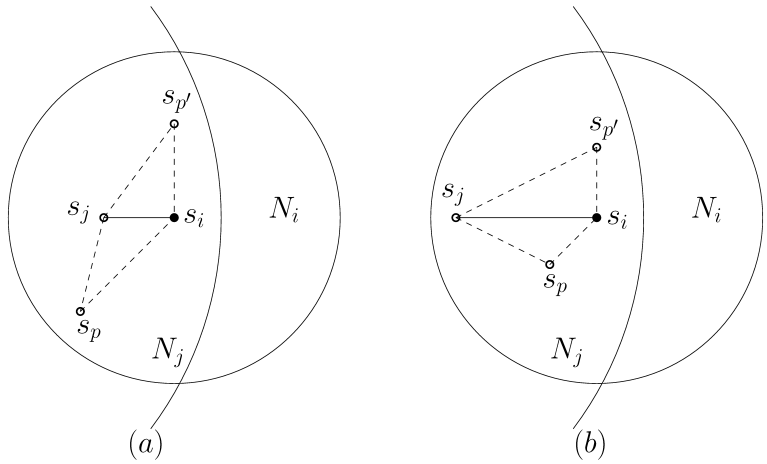


Figure 4.3: Illustration of the base case.

Case 2. $a_j = 0$, which means that node s_j is not selected for the current round. Hence, equation 4.18 gives:

$$\sum_{s_p \in N_j : d(s_p, s_i) < d(s_j, s_i)} b_{p,j} = 0 \tag{4.21}$$

This is coherent with the constraint 4.5. If node s_i is not elected as a cluster-head, then inequality 4.17 becomes:

$$\forall s_j \in N_i, a_j - \sum_{s_p \in N_j : d(s_p, s_i) < d(s_j, s_i)} b_{p,j} \leq 1 \tag{4.22}$$

The condition 4.22 is a general case, which is valid all the time.

Recursion Case. Suppose that the constraint 4.9 leads to condition 4.16 for $n-1$. We have to demonstrate this implication for n (as illustrated in Figure 4.4). As a hypothesis for $n-1$, we have the following assertion:

$$\exists s_{j_2} \in \boldsymbol{F}_{j_1,i}, \exists s_{j_3} \in \boldsymbol{F}_{j_2,i}, \ldots, \exists s_{j_n} \in \boldsymbol{F}_{j_{n-1},i} : b_{j_2,j_1} = 1, \ldots, b_{j_n,i} = 1 \tag{4.23}$$

We need to prove that $\exists s_{j_1} \in \boldsymbol{F}_{j,i}, b_{j_1,j} = 1$?

Since the constraint 4.9 is verified for n and according to Proposition 1, node s_j is a relay or a leaf node [3], i.e.

$$\exists s_{j_1} \in N_j : b_{j_1,j} = 1$$

According to the recursion hypothesis, we have the following condition:

[3]The left side of the implication is verified for n, we have to prove the right side.

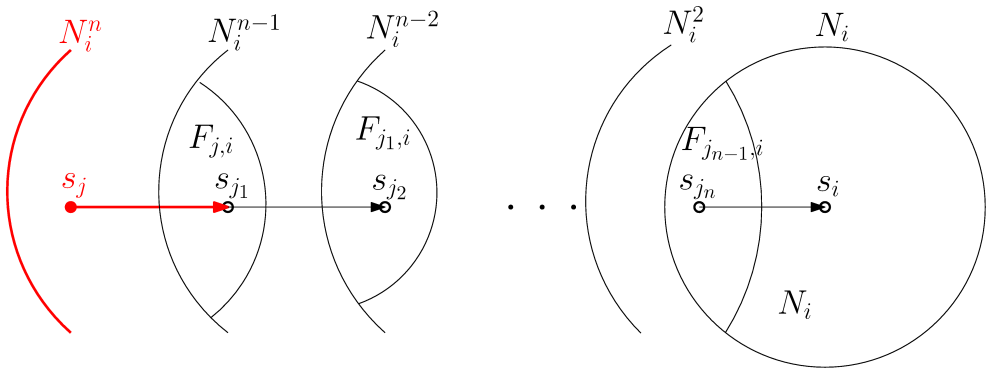


Figure 4.4: Illustration of the recursion case.

$$\forall s_j \in N_i^n, a_j - \sum_{s_p \in \boldsymbol{F}_{j,i}} b_{p,j} \leq 1 - (a_i - \sum_{s_q \in N_i} b_{q,i}) \tag{4.24}$$

Given a *selected* node $s_j \in N_i^n - N_i^{n-1}$ that verifies this last condition (4.24), we have:

$$\sum_{s_p \in \boldsymbol{F}_{j,i}} b_{p,j} = 1$$

This means that $\exists s_{j_1} \in \boldsymbol{F}_{j,i} : b_{j_1,j} = 1 \Leftrightarrow d(s_{j_1}, s_i) < d(s_j, s_i)$. □

4.3 Model Validation

In this section, we demonstrate the validity of our BILP model for the EMDP problem. We implement the model in GMPL (*Gnu MathProg Language*) using the GLPK toolkit and we present the exact solution of EDMP for a small instance ($m = 12$, $k = 1$, $n = 3$). We also investigate the complexity of EMDP and we show its NP-Completeness.

We implement our BILP model using GMPL[4]. We define four sets for: nodes, rounds, hops and neighborhoods. Next, we specify the parameters of the model: the network size m, the maximum depth of trees n, the number of rounds k, the sensitivity parameter τ, the energy parameter α, the reference value v_0, the amounts of energy consumption e_s, e_r, e_p and e_t, the neighbors of the nodes L, the energy of the nodes E and the readings of the nodes v.

The decisions variables are specified using *binary* keyword and each constraint starts with *s.t.* keyword. Each line corresponds to a constraint. The objective function is specified using *maximize* (or max.) keyword.

[4]Exhaustive description of the GMPL formalism is given in Appendix B.

The exact solution for a small instance of $m = 12$ nodes deployed in $100 \times 100m^2$ area is given in Appendix B. The maximum depth of data collection trees is set to $n = 3$ and the number of rounds is set to $k = 1$. A complete listing of the output of our BILP implementation is also given in Appendix B.

4.4 Conclusion

In this chapter, we formulated the problem of energy optimization in the context of correlated data collection in WSN. We proposed a novel BILP model to help characterizing this problem. We validated our BILP model by demonstrating formally certain crucial characteristics and implementing the model using GLPK toolkit. Given the complexity of EMDP, we argue that it requires heuristic solutions which constitutes the subject of the next chapter.

Chapter 5

Correlation-Based Adaptive Dynamic Clustering Algorithm for Data Collection

WIRELESS SENSOR NETWORKS are not just like any other mobile ad-hoc network or wireless communication system; they are *sources of spatial-temporal correlated data* that can be used to optimize the network operations [31], [113].

In Chapter 4, we proposed a BILP model for the EMDP problem to find the exact solution in the general context of data collection applications. The proposed approach takes profit from the correlation that may exist between data and exploits it to reduce the energy consumption by selecting the most appropriate nodes. We showed that EMDP is NP-Complete and requires heuristic solutions for big instances of the problem.

In this chapter, we consider the same EMDP problem but using an approximative approach. We propose a heuristic solution called CORAD (*Correlation-Based Adaptive Dynamic Clustering*) that constructs dynamic clusters based on the collected data. It aims at reducing the energy consumption of the network by selecting the nodes with maximum energy resources that contribute the most in enhancing the quality of data.

As we showed in Chapter 1 and Chapter 2, the sensor nodes are equipped with small, limited-capacity and non-rechargeable batteries. The optimization effort should be concentrated on the software layers by enabling the interaction between the communication sub-system and the sensing sub-system. We also showed that the correlations between the sensory data can be exploited to: (1) optimize the energy consumption during the data collection process by adapting the network structure to the dynamics of the events, and (2) preserve the data accuracy by reconfiguring the network each time these events change.

A *standard* way to reduce the energy consumption and extend the network lifetime is by turning-off all the nodes of the network (or putting them in a low-power operating mode) except a selected subset of them that must stay active, i.e. the nodes that may *probably* provide the most *useful* information to the end-user application according to its accuracy requirements.

In our approach, we investigate the possibilities of exploiting such a spatial-temporal data correlation aspect to reduce the energy consumption in a dense network structured as clusters[1]. We propose CORAD as an algorithm of constructing and reconfiguring the structure of the network according to the dynamic of the current observed event. Hence, with CORAD, an activation schedule based on the residual energy of the nodes and their potential of detection of the events is established to trade between the energy consumption and the data accuracy.

Many algorithms based on the techniques of adaptive dynamic clustering have been proposed in the literature. However, few of them exploit the aspect of data correlations to setup the network topology and adapt it to meet the application requirements in terms of data accuracy [31]. One of the most important algorithms among them is ASAP [114]. It is a *periodic* distributed sensing-driven cluster construction algorithm in which the reconfiguration of the clusters is based on the infrequent network-wide samplings called *forced samplings*, to discover newly established correlations.

In [115], an ASAP-like protocol is proposed where a *dissimilarity measure* between two nodes different from that proposed in ASAP is defined. The sampling and the data reporting tasks are done by some randomly selected nodes, and the cluster maintenance is the same as in EEDC protocol [116]. Nonetheless, the randomness of the node selection cannot guarantee the quality of data.

At the opposite, EEDC [116] constructs the clusters according to the pairwise dissimilarity of the nodes and the intra-cluster dissimilarity threshold. Multiple sensors are randomly scheduled for sampling and the temporal correlation is used to reduce the amount of the transmitted data.

When we focus on data correlations, the effectiveness of prediction is unavoidable. The framework proposed in [117] deals with this issue and proposes to predict the samplings of the nodes based on the desired error bound and the sensor data predictability.

Although AR (Auto Regressive) model is simplistic, it is frequently used by several algorithms to capture the spatial correlation such as in ELink [118]. Elink is a parameterized clustering algorithm that measures the distance between the *features* of the nodes. It constructs the clusters using a *sentinel* structure while ensuring the *δ-compactness* of each constructed cluster.

When a data reporting tree is set, it is necessary to adapt the reporting period

[1]For scalability purpose.

to eliminate unnecessary sending such as in [119]. The nodes having current readings different from the previous ones, do not send their data to the sink and update their period of data reporting using an *exponential aging method* based on their residual energy and the user-defined level of data accuracy.

In a data collection process, some aspects are related to the nature of the data correlations. According to [120], the data collection streaming mode is well-suited for static environments in the presence of strong spatial-temporal correlation, while the interactive data collection mode is suited for dynamic environments in the presence of strong spatial correlation only. For joining the clusters, the nodes in CAG algorithm [120] verify a certain condition about their readings. And for adjusting the clusters, the nodes check periodically their readings: if they do not satisfy the above condition then they should be evicted from their current cluster (migrate towards a near cluster or form a new cluster with themselves as cluster-heads).

In this context, neighborhood information is important to define the correlation region. In YEAST [121] and EAST [122], the nodes that belong to the same correlation region, whose definition depends on the application requirements and the nature of the events, tend to detect similar values. Thus, the energy optimization in this context is based on eliminating useless notification of sampling nodes using correlation region adjustment.

Each method of all the above-described ones considers only one aspect of the data collection process. i.e. the energy or the data sources or the detection, etc. None of them considers all the aspects. In CORAD, we define the models for all the above aspects. We couple all these models in a single dynamic data collection scheme that is presented in the next sections.

5.1 Basic Idea & Objectives

The objective of CORAD is to extend the network lifetime by reducing the energy consumption of the nodes while preserving an *acceptable* level of the accuracy of the generated data. To illustrate the benefits of such approach, we first recall some characteristics about the WSN operating modes.

According to [31], WSN operate generally in two main modes:

- **Monitoring mode:** where nodes report periodically their sensed data about a physical phenomenon to the base station for post-processing. It functions in a query-driven or time-driven fashion.

- **Reactive mode:** where nodes do nothing until they detect an event by themselves (this is also called event-driven mode). After that, only detecting nodes collaborate between each others to continuously report the data about the event until it despairs.

It is obvious that the second mode is more energy-efficient and suitable for region-localized and data-correlated event detection and tracking. However, it is more challenging given the additional temporal and quality constraints that it should satisfy. We start by studying the first mode for simple case to understand the underlying mechanisms that will serve for the second mode.

For system scalability, we use a dynamic clustering protocol whose objective is twofold:

- Setting-up a local data fusion (or aggregation) center instead of a global one to reduce the length of routing paths and decrease the communication overhead due to the network reconfigurations.
- Adapting the network topology based on the spatial-temporal data correlation generated by the sensors.

Since communications are the most predominant cause of energy depletion of the nodes, the initial election of the cluster-heads is based on their *proximity* to their neighbors and their residual energy. That is what we call static clustering. The correlations are then integrated in CORAD to adapt the network topology via two complementary mechanisms:

1. **Topology reconfiguration**: upon each major change of the event dynamics detected by the current cluster, the most appropriate nodes having the most *accurate* readings and that are close to the event focus, could selected for sampling and reporting their readings. The leader in this case specifies the participation of the nodes in the data collection process by establishing a schedule in terms active/inactive cycles. This mechanism reduces the energy consumption by selective activation of the nodes. However, in order to be efficient, it should take into account several parameters such as: event dynamics (in terms of computed correlation), required accuracy (in terms of user-defined accuracy), residual energy of the nodes, etc.
2. **Nodes' behavior adaptation**: according to the variation of measurements in time, the data reporting periods are adjusted based on the residual energy of the nodes and the data accuracy requirements.

Before presenting CORAD, we first describe the models used throughout our approach.

5.1.1 Network Model

We represent a WSN by an undirected graph $G(V, E)$: with $V = \{s_1, s_2, \ldots, s_m\}$ representing the set of nodes ($|V| = m$ is the size of the network) and E representing the set of edges. An edge $e_{ij} \in E$ exists if and only if a bidirectional link

between node s_i and node s_j exists. We suppose that the nodes are uniformly deployed over a 2D square area A. All the nodes are symmetric and homogeneous in terms of radio communication range R_c^i, sensing range R_s^i and initial energy budget e_{s_i}, i.e:

$$\forall s_i \in V, R_c^i = R_c, R_s^i = R_s, e_{s_i} = E_{max}$$

In addition, each node knows its position and those of its direct neighbors. For coverage and connectivity issues, we also suppose that:

$$R_c \geq 2R_s$$

5.1.2 Energy Model

We use the generic energy model proposed in [123] and [124]. In this model, the energy consumption in transmission and reception depends on the radio signal propagation model. We consider two models: the *free-space* model and the *multi-path* model. The nodes that transmit data spend e_{tx} Joules to send l bits over a distance d, and spend e_{rx} Joules to receive l bits independently from the distance. Equations 5.1 and 5.2 gives these two values:

$$e_{tx} = kd^{\alpha} + c \tag{5.1}$$

$$e_{rx} = c \tag{5.2}$$

Parameter α is the shadowing factor: it depends on the signal propagation model ($\alpha = 2$ for the free space model and $\alpha = 4$ for the multi-path model). If we assume that all the exchanged messages are identical in size then $c = le$ is constant. Moreover, the energy consumption in the sampling operations depends on the sampling duration and the hardware. In our case, we suppose that the nodes consume $e_{sp} = \tau$ (Joules) in each sampling operation[2].

5.1.3 Data Source Model

Uniformly distributed data over the monitoring area are often not correlated nor in time neither in space. Instead, spatial-temporal correlated data are generally modeled as *Random Gaussian Field* (RGF). For simulation purposes, we use the framework proposed in [125] to generate synthetic data by reproducing the signals gathered from real deployed WSN. To capture the spatial auto-correlation of the real signals, the framework assumes that the auto-correlation function is separable into temporal and spatial correlation:

$$\rho(\mathbf{p}_1, t_1, \mathbf{p}_2, t_2) = \rho_S(\mathbf{p}_1, \mathbf{p}_2)\rho_T(t_1, t_2)$$

[2]We can also consider the energy consumption in the data processing operations and the state transitions of the nodes, but in most cases, these amounts are insignificant.

Where $\mathbf{p}_1, \mathbf{p}_2 \in \mathcal{D}$ and $t_1, t_2 \in \mathcal{T}$, and $\mathcal{D}$ and $\mathcal{T}$ are space and time domain, respectively. The framework defines the spatial auto-correlation function by:

$$\forall t \in \mathcal{T}, \rho_S(\mathbf{p}_1, \mathbf{p}_2) = \frac{cov(z(\mathbf{p}_1, t), z(\mathbf{p}_2, t))}{\sigma_z(\mathbf{p}_1, t)\sigma_z(\mathbf{p}_1, t)}$$

It also uses two models to capture the spatial auto-correlation in the real signals, namely: the Power Exponential model and the Rational Quadratic model defined respectively by the equations 5.3 and 5.4:

$$\tilde{\rho}_{S_{PE}}(d) = e^{-(d/\zeta)^\nu} \tag{5.3}$$

$$\tilde{\rho}_{S_{RQ}}(d) = \frac{1}{1 + (d/\zeta)^\nu S_\nu} \tag{5.4}$$

Both of the models depend on two parameters ζ and ν which are respectively: the length and the order of the auto-correlation function.
Similarly, the framework defines the temporal auto-correlation function by:

$$\forall \mathbf{p} \in \mathcal{D}, \rho_T(t_1, t_2) = \frac{cov(z(\mathbf{p}, t_1), z(\mathbf{p}, t_2))}{\sigma_z(\mathbf{p}, t_1)\sigma_z(\mathbf{p}, t_2)}$$

At the end, the framework generates the synthetic data using a four-step algorithm described in [125].

5.1.4 Detection Model

We use the Eifes Sensing Model as a detection model. According to [126], the detection probability of an event at distance d by a node s is:

$$p_s(d) = \begin{cases} 0 & \text{if } d \geq R_s \\ e^{-\lambda d} & \text{if } 0 \leq d < R_s \end{cases} \tag{5.5}$$

Where R_s is the maximum sensing range of node s. Constant λ depends on the sensor hardware.

5.2 Static Clustering Scheme

Using the network and the energy models, we first design a static clustering algorithm based on two rules: the *election* rule and the *adhesion* rule. This static clustering scheme serves as a building block of CORAD.

Definition 3 (Election rule)**.** *The nearest node to the barycenter of the set of all the n-hop neighbors of some node s, with the maximum residual energy is elected as the cluster-head of the cluster to which node s belongs.*

Definition 4 (Adhesion rule)**.** *The parent of a member node s in the data collection tree is the one with the maximum energy resources that forwards the messages as further as possible toward its cluster-head.*

Since each node knows its cluster-head and the positions of its direct neighbors, it computes its *forwarding set* and selects the node u with the minimum ratio:

$$\frac{dist(u, ch)}{e_u}$$

In Figure 5.7, the forwarding set of node u is $F(u) = \{s_1, s_2\}$. As $e_2 > e_1$ and $dist(s_2, ch) \approx dist(s_1, ch)$, then $parent(u, ch) = s_2$. If the forwarding set is empty, then the member node is simply *removed* from the cluster.

Algorithm 1 is the centralized version of our static clustering algorithm. It is a greedy algorithm, i.e.: the while loop between Line 2 and Line 11 is for electing the cluster-heads. For each node in the set of remaining nodes (lines between 3 and 7), the algorithm computes its centrality (Line 6) then it selects node s with the minimum *centrality/energy* ratio from all the n-hop neighbors (Line 8). Node s and its n-hop neighbors are then removed from the global set of nodes (Line 9) and the same process is repeated with the remaining nodes (Line 10).

After cluster-heads election, Algorithm 1 constructs the data reporting trees: the for loop between Line 12 and Line 25 is the portion of code responsible for this. For each node in the current cluster (lines between 13 and 24): the algorithm checks if it is a direct neighbor of the cluster-head. If so, then its parent is the cluster-head itself. Otherwise, it computes its forwarding set (Line 17) and selects the parent based on the minimum ratio distance/energy (Line 21).

It is worth noting that Algorithm 1 cannot guarantee neither optimal clustering nor size-balanced clusters (see Figure 5.2 and Figure 5.3). Nonetheless, at the end of the clustering process, each node knows its cluster-head to which it reports its readings (in Figure 5.2 and Figure 5.3 the cluster-heads are represented by red points). However, in one-hop clustering case, this issue is not raised since member nodes use massive MAC-level broadcasts to transmit their readings and they do not need the *gateways* to aggregate and forward the sampling data to their local fusion centers contrary to multi-hop clustering scheme (Figure 5.1).

5.3 CORAD: Correlation-Based Adaptive Dynamic Clustering Scheme

The static scheme constructs a network topology based only on the location information and the residual energy of the nodes. This can lead to wasting precious energy resources of the network when arbitrary reporting nodes are selected. A natural approach to overcome this problem is to adapt the WSN structure to the

Algorithm 1 Static Clustering.

Require: : $\mathbf{V} \leftarrow \{s_1, s_2, \ldots, s_m\}$, $\mathbf{L} \leftarrow \{e_1, e_2, \ldots, e_m\}$, $n \leftarrow \#hopes$

{cluster-heads election}

1: $\mathbf{CH} \leftarrow \varnothing$
2: **while** $\mathbf{V} \neq \varnothing$ **do**
3: **for** $\forall s \in \mathbf{V}$ **do**
4: compute $\mathbf{V}_n(s)$
5: $g_{s,n} \leftarrow$barycenter($\mathbf{V}_n(s)$)
6: $ce_n(s) \leftarrow dist(s, g_{s,n})$
7: **end for**
8: $ch \leftarrow \arg\min_{s \in \mathbf{V}} \{ce_n(s)/e_s\}$
9: $\mathbf{V} \leftarrow \mathbf{V} - \mathbf{V}_n(ch)$
10: $\mathbf{CH} \leftarrow \mathbf{CH} \cup \{ch\}$
11: **end while**

{members adhesion}

12: **for** $\forall ch \in \mathbf{CH}$ **do**
13: **for** $\forall u \in \mathbf{V}_n(ch)$ **do**
14: **if** $u \in \mathbf{V}_1(ch)$ **then**
15: $u.parent \leftarrow ch$
16: **else**
17: $\mathbf{F}(u) = \{s \in \mathbf{S}_1(u), dist(u, ch) > dist(s, ch)\}$
18: **if** $\mathbf{F}(u) \neq \varnothing$ **then**
19: $u.parent \leftarrow \arg\min_{s \in \mathbf{F}(u)} \{dist(s, ch)/e_s\}$
20: **else**
21: $\mathbf{V}_n(ch) \leftarrow \mathbf{V}_n(ch) - \{u\}$ {*failure! empty forwarding set.*}
22: **end if**
23: **end if**
24: **end for**
25: **end for**

dynamics of the events. Our clustering algorithm should take profit from the correlation that may exist between the collected data and thus, it can reduce the energy consumption and improve the data accuracy by selecting only the nodes with useful data.

By doing so, the dynamic clustering algorithm can be tunned based on the accuracy level required by the end-user application, the intrinsic nature of the event and the residual energy of the nodes. This dynamic behavior can be implemented via two complementary mechanisms: the adaptation of the data reporting period and the reconfiguration of the network topology. The former operates on the *node level* to optimize data collections by exploiting the temporal correlation, and the

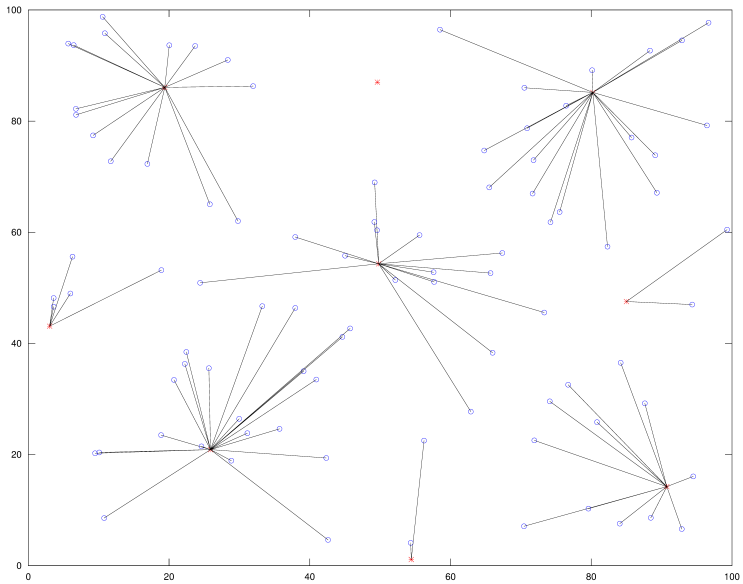


Figure 5.1: Static clustering with $m = 100$, $A = 100m \times 100m$, $R_c = 30$ and $n = 1$. Red points represent the cluster-heads and black (blue) points represent the members. The line segments represent direct communication links.

later operates on the *network level* to extend the network lifetime by exploiting the spatial correlation.

Correlation between data can be very high when the variation of data is minimal. Thus, the nodes should not waste their valuable energy resources to report them because they will not improve the quality of data. Generally, the absolute difference between two consecutive data values is compared to a *threshold* subject to update according to the residual energy of the nodes. If it exceeds this threshold then the nodes report their data. Otherwise, they do nothing but just adjusting their reporting periods.

What we presented above is the first mechanism. The second mechanism defines the way by which the clusters are constructed. It depends mainly on the data correlation capture model and the clustering rules.

5.3.1 Data Correlation Capture Model

The example of Figure 5.4 shows a region-based event distribution in a 2D surveillance area. We have six geographic regions: R_1, R_2, R_3, R_4, R_5 and R_6. In each region R_i, a *representative* value $\tilde{v}_i$ is associated with the set of the real signal values X_i verifying the following property:

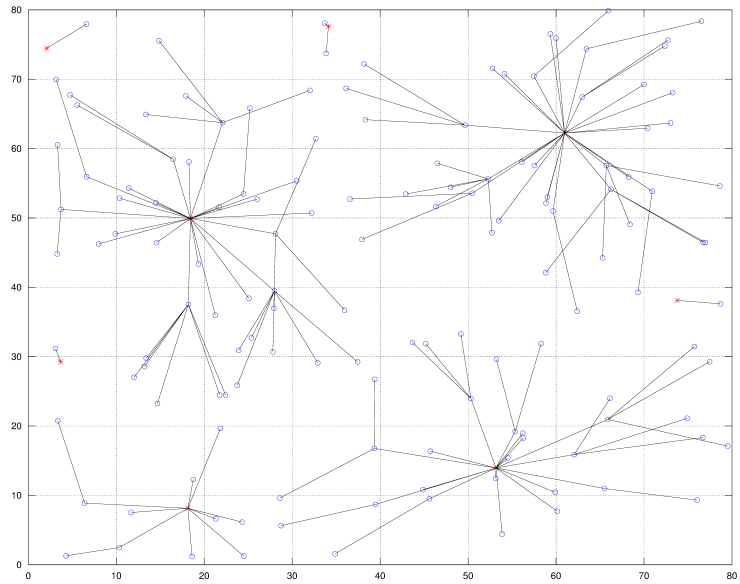


Figure 5.2: Static clustering with $m = 150$, $A = 80m \times 80m$, $R_c = 15$ and $n = 2$. Red points represent the cluster-heads and black (blue) points represent the members. The line segments represent direct communication links.

$$\forall v \in X_i : |v - \tilde{v}_i| < \delta \tag{5.6}$$

δ is the maximum variation on real signal values between two points.
For each set of values X_i, we associate a corresponding set of detecting nodes $\mathbf{S}_i$ called the *clustered-nodes* that depends on the data correlation capture model. In our case, we propose a *Min-Max* model:

The nodes lookup for the minimum and the maximum values in their sampled raw data, then they decide about their effective detected values based of the number of the occurrences of each minimum/maximum values which *should* exceed a threshold **Thd**.

As shown in Figure 5.5(a), each node s_i executes the sampling operation for a period of time T_{sens} and generates M values: $\hat{v}_i^1, \hat{v}_i^2, \ldots, \hat{v}_i^M$ in each point $p_i^1, p_i^j, \ldots, p_i^M$, respectively using the probabilistic detection model (Section 5.1.4):

$$\hat{v}_i^j = v_i^j e^{-\lambda d_i^j}$$

where d_i^j is the distance between node s_i and point p_i^j, and v_i^j is the real value of the signal at point p_i^j.
After that, node s_i looks for the number of the occurrences of its maximum and

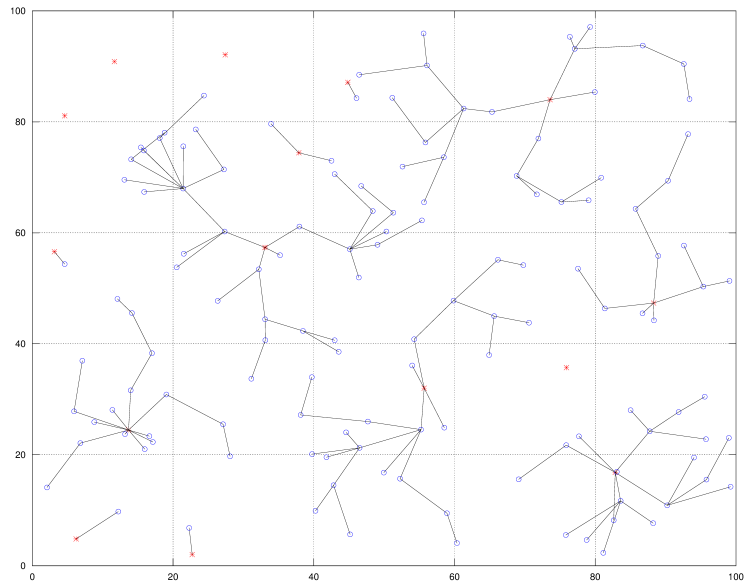


Figure 5.3: Static clustering with $m = 150$, $A = 100m \times 100m$, $R_c = 10$ and $n = 4$. Red points represent the cluster-heads and black (blue) points represent the members. The line segments represent direct communication links.

minimum values satisfying the following inequalities, respectively:

$$\hat{v}_i^j \leq V_{min} \tag{5.7}$$

$$\hat{v}_i^j \geq V_{max} \tag{5.8}$$

where V_{min} and V_{max} are the thresholds on the minimum and the maximum values, respectively. We denote by W and W' the sets of the values satisfying equations 5.7 and 5.8, respectively.

If $|W| \geq \mathbf{Thd}$ or $|W'| \geq \mathbf{Thd}$ then the detected value of a node s_i is $\hat{v}_i = min(W)$ or $\hat{v}_i = max(W')$, respectively. Otherwise, node s_i does not generate any significant value (s_i in not a detecting node).

Note that we use the same equation 5.6 to characterize two *correlated-value* nodes, i.e: two nodes s_i and s_j are correlated-value nodes if and only if: $|\hat{v}_i - \hat{v}_j| \leq \epsilon$, where ϵ is the maximum variation between the effective values of nodes s_i and s_j.

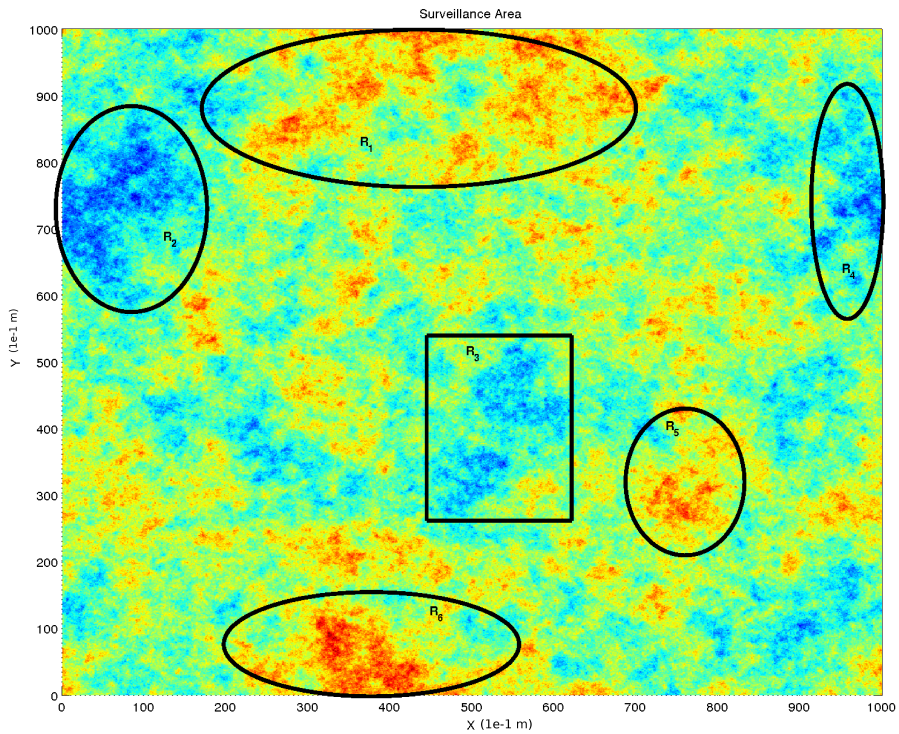


Figure 5.4: Example of event regions.

5.3.2 Dynamic Clustering Rules

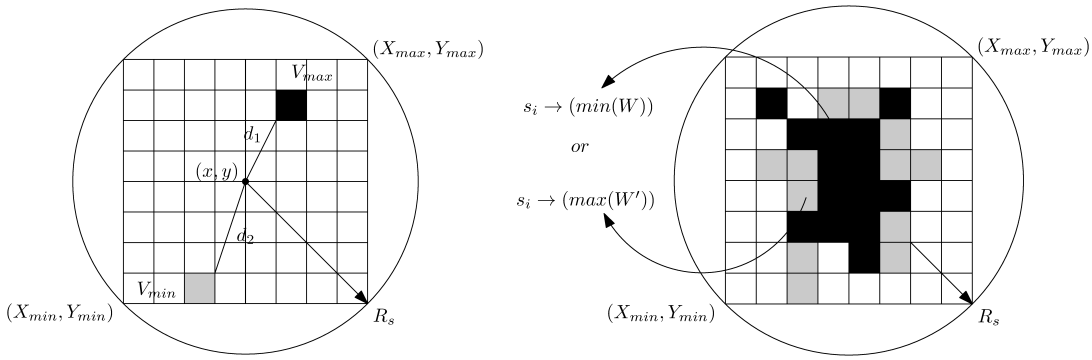
Our adaptive dynamic clustering algorithm runs in two phases: the initialization phase and the reconfiguration phase. It uses the above models to initialize the clusters and the adaptive reconfiguration rule to reconfigure them.

Initialization Phase

In the initialization phase, the algorithm takes three parameters as inputs:

1. The direct neighboring information.
2. The residual energy levels of the nodes.
3. The list of the correlated-value nodes.

As outputs, it determines the clusters that should be constructed according to the nature of the phenomenon being monitored or tracked.



(a) Min-Max Capture (b) Aggregation Model

Figure 5.5: Data correlation model.

Note that the nodes with weak-correlated values are not included in the cluster even if they are direct neighbors. At the opposite, the nodes with strong-correlated values can be included as cluster members when they belong to the same neighborhood of other nodes with strong-correlated values. Otherwise, they represent a new cluster or even they are not clustered at all. Therefore, the set of *clustered-nodes* consists of n-hop neighboring nodes with correlated-values depending on the data correlation models explained above.

The clustering rule is defined as follows[3]:

$$\left.\begin{array}{r} |\hat{v}_i - \hat{v}_j| < \epsilon \\ \wedge \\ H(s_i, s_j) \leq n \end{array}\right\} \Rightarrow s_i \text{ and } s_j \text{ belongs to the same cluster.}$$

In Figure 5.6, s_1 and s_4 are neighbor nodes and correlated-value nodes. However, each node has a different set of correlated-values nodes: s_8 is neighbor to s_4 and have a correlated-value with s_1, and s_9 is neighbor to s_1 and have a correlated-node with s_4. The set of *clustered-nodes* is defined by all the connected neighbors with correlated-values: in Figure 5.6, it is the set $\mathbf{S} = \{s_3, s_1, s_4, s_5, s_6, s_7\}$.

Algorithm 2 describes the initialization procedure. It is a greedy algorithm that runs in successive rounds: in each round, *only one* node is elected as a cluster-head within its set of clustered-nodes determined by the *clustering rule procedure* (Line 4). The repeat loop between Line 3 and Line 11 elects the cluster-heads based on the the minimum size of the set of clustered nodes $\mathcal{S}_{min}$ The algorithm ends when no set of clustered-nodes $\mathbf{S}_i$ with the minimum required size $\mathcal{S}_{min}$ remains (Line 5).

The $clustering_rule(\mathbf{V}, \mathbf{L}, n, \epsilon, V_{min}, V_{max})$ procedure computes all the sets of clustered-nodes $\mathbf{S}_i$ using the rules described in subsection 5.3.2. Then, it elects

[3]In our simulations, we set $\epsilon = 0.001$.

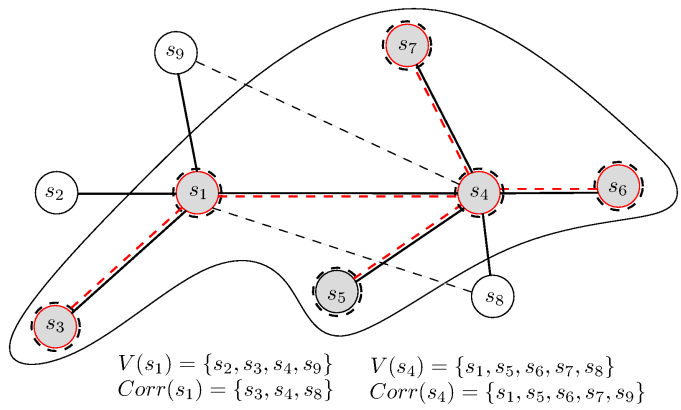


Figure 5.6: Determination of the clusters based on data correlations and neighboring information.

Algorithm 2 Initialization Procedure.

Require: : $\mathbf{V} \leftarrow \{s_1, s_2, \ldots, s_m\}$, $\mathbf{L} \leftarrow \{e_1, e_2, \ldots, e_m\}$, $\mathcal{S}_{min}$, $n \leftarrow \#hops$, ϵ, V_{min}, V_{max}

1: $\mathbf{S} \leftarrow \varnothing$ {*set of clustered nodes*}
2: $\mathbf{CH} \leftarrow \varnothing$ {*set of cluster-heads*}
3: **repeat**
4: $\{ch, \mathbf{S}\} \leftarrow clustering_rule(\mathbf{V}, \mathbf{L}, n, \epsilon, V_{min}, V_{max})$ {*Refer to subsection 5.3.2*}
5: **if** $(size(\mathbf{S}) < \mathcal{S}_{min})$ **then**
6: $break$
7: **end if**
8: $\check{v}_{ch} \leftarrow \sum_{i \in \mathbf{S}} \hat{v}_i / |\mathbf{S}|$ {*Aggregated value at the cluster-head*}
9: $\mathbf{CH} \leftarrow \mathbf{CH} \cup \{ch\}$
10: $\mathbf{V} \leftarrow \mathbf{V} - \mathbf{S}$
11: **until** $\mathbf{V} = \varnothing$
12: $construct_clusters(\mathbf{CH})$

the cluster-head ch with the minimum ratio centrality/energy (as in the static case) multiplied by the coefficient of variation $\frac{\sigma_i}{|\bar{v}_i|}$:

$$\forall u, \quad ch = \arg\min_{i \in \mathbf{S}_u} \{ \frac{ce(i)}{e_i} \times \frac{\sigma_i}{|\bar{v}_i|} \} \tag{5.9}$$

The aggregated value at the elected cluster-head ch is the mean value of the all received readings from cluster members (denoted by $\check{v}_{ch}$ in Algorithm 2, Line 8).

The centrality of node u, $ce(u)$ is defined as the same way as in the static case except it is computed over the set of clustered-nodes $\mathbf{S}_u$ instead of the set of n-hop neighbors $\mathbf{V}_n(u)$.

It is worth to note here that $\mathbf{V}_n(s)$ denotes the set of n-hop neighborhood of node s (including the neighbors from 1 to n hops and the node s itself). The

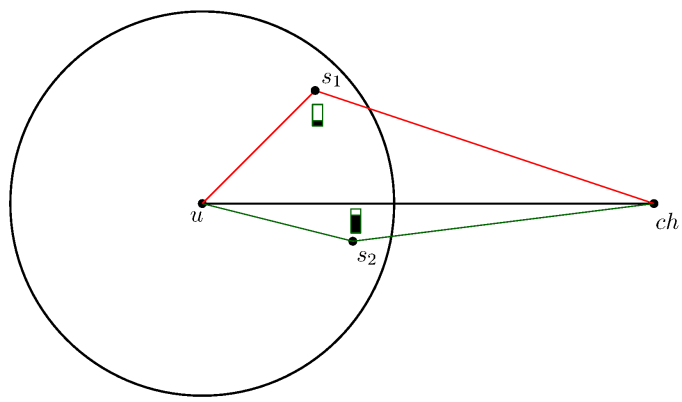
barycenter of $\mathbf{V}_n(s)$, $g_{s,n}(\overline{x}_{s,n}, \overline{y}_{s,n})$ is given by:

$$\overline{x}_{s,n} = \frac{\sum_{u \in \mathbf{V}_n(s)} x_u}{|\mathbf{V}_n(s)|}; \qquad \overline{y}_{s,n} = \frac{\sum_{u \in \mathbf{V}_n(s)} y_u}{|\mathbf{V}_n(s)|}$$

We define the *centrality* of node s in $\mathbf{V}_n(s)$ as:

$$ce_n(s) = dist(s, g_{s,n})$$

The procedure of construction of the clusters $construct_clusters(\mathbf{CH})$ uses the same adhesion rule as in the static case (Definition 4). Figure 5.7 shows this rule.



$F(u) = \{s_1, s_2\}$
$dist(s_1, ch)/e_{s_1} > dist(s_2, ch)/e_{s_2} \Rightarrow Parent(u, ch) = s_2$

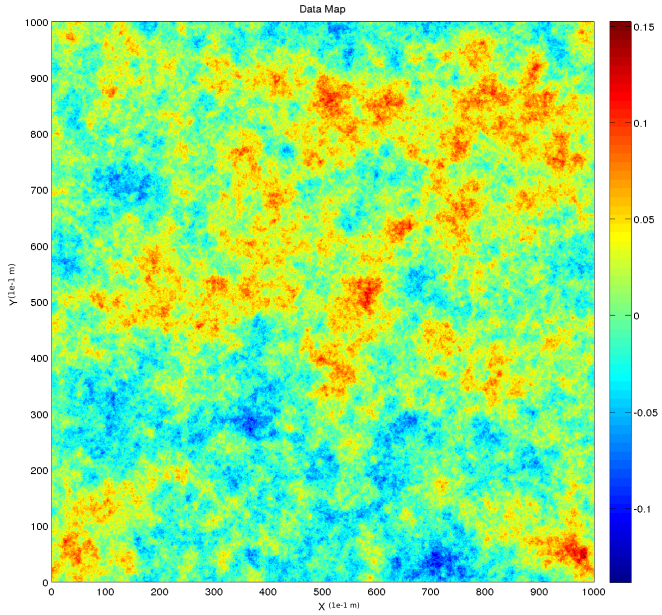
Figure 5.7: Example of parent selection.

To illustrate the initialization phase of CORAD, we run a Matlab simulation of 450 randomly deployed nodes in a $100 \times 100\ m^2$ square area. All the nodes are identical with communication range $R_c = 10m$ and sensing range $R_s = 5m$ and sensing parameter $\lambda = 0.03$. The clusters are constructed within 4-hops neighbors of the set of correlated-values nodes with the following parameters: $S_{min} = 10$, $V_{min} = -0.08$, $V_{max} = +0.08$, $\mathbf{Thd} = 5$ and $\epsilon = 0.001$.

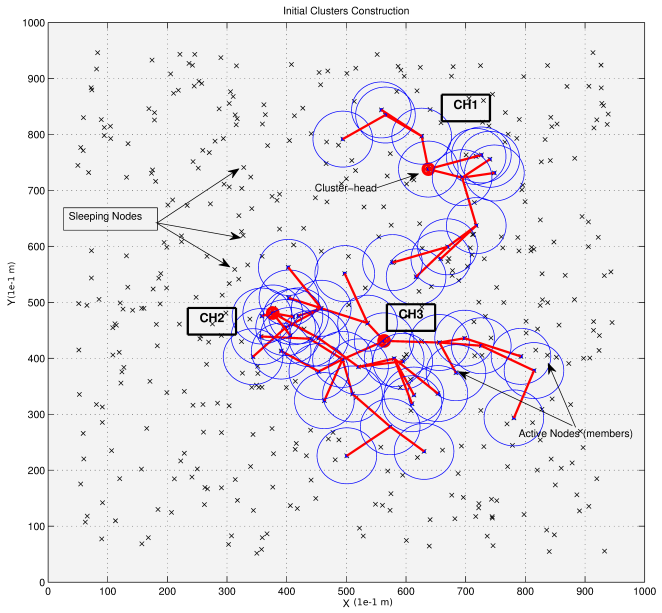
Figure 5.8(b) shows three clusters created in the initialization phase: CH1 with 15 nodes, CH2 with 11 nodes and CH3 with 26 nodes. We can see that the clusters are surrounding the event focus (see Figure 5.8(a)) and only a small fraction of nodes are clustered according the physical phenomenon (luminosity) and the cluster sizes are close to each others (balanced).

Reconfiguration Phase

Recall that the data collection process is run in successive rounds and in each round the member nodes report their detected values to the cluster-head. Thus, the reconfiguration procedure depends on two parameters: (i) the number of



(a) Map of synthetic data about luminosity



(b) Clusters created by CORAD

Figure 5.8: Initialization phase of CORAD.

clustered nodes at round t denoted by k_t, and (ii) the Mean Square Error (MSE) at round t denoted by Δ_t:

$$\Delta_t = \sqrt{\frac{\sum_{i=1}^{k_t}(\check{v}_i - v_i)^2}{k_t}}$$

The objective here is to determine the set of participating nodes in the sampling operations at round $t+1$, denoted by $\mathbf{CR}_{t+1}$.

If the number of clustered nodes is greater than threshold T_1 and the MSE is greater than threshold T_2, then some of the k created clusters at round t should be *shrunk* for round $t+1$ and the parameters V_{min} , V_{max}, $\mathcal{S}_{min}$ and **Thd** should be increased. At the opposite, if the number of clustered nodes is smaller than T_1 and the MSE is smaller than T_2, then the clusters should be *stretched* and the parameters V_{min} , V_{max}, $\mathcal{S}_{min}$ and **Thd** should be decreased. In all the cases, a set of non-participating nodes that are randomly selected based on some percentage (probability) p are added to $\mathbf{CR}_{t+1}$. Based on this observation, the definition of the dynamic reconfiguration rule is presented below:

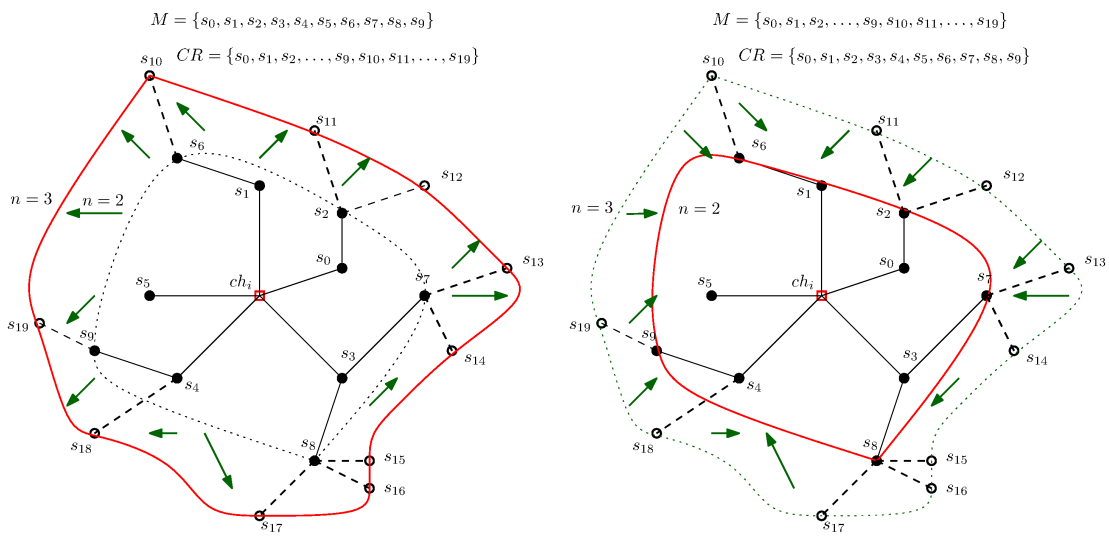
Definition 5 (Dynamic reconfiguration rule)**.** *The clusters with large size and poor quality of data are shrunk and the clusters with small size and good quality of data are stretched.*

To stretch a cluster, we simply *enlarge* it by the 1-hop neighbors of the $n-$hop neighboring nodes of the cluster-head. Similarly, to shrink a cluster we reduce it by the 1-hop neighbors of the $n-$hop neighboring nodes of the cluster-head (see example of Figure 5.9).

Algorithm 3 describes the dynamic reconfiguration procedure: it first checks the size of the set of clustered nodes (Line 3) then the MSE (Line 4). The shrinking case is detailed between lines 5 and 10, and the stretching case is detailed between lines 14 and 19. Note that the parameters V_{min}, V_{max}, **Thd** and $\mathcal{S}_{min}$ are simply updated by adding or subtracting the current value of MSE to V_{min} and V_{max}, adding or subtracting the mean size of the set of 1-hop neighbors to $\mathcal{S}_{min}$ and adding or subtracting 1 to **Thd** (Lines 7, 8, 9, 10, 16, 17, 18 and 19).

5.4 Simulation Results & Analysis

The goal of the simulations is to show that CORAD helps reducing the energy consumption and improving the network lifetime while ensuring the quality of the collected data. To achieve this goal, we run centralized simulation on Matlab. We define the simulation scenario as follows: m identical nodes (with characteristics R_c, R_s, E_{max}, λ, τ, l, α, k and c detailed in Table 5.1) are uniformly deployed in a 2D square of size $A \times A$, where a physical phenomenon (luminosity) evolves.



(a) Cluster Stretching (b) Cluster Shrinking

Figure 5.9: Cluster reconfiguration.

The end-user application requests values greater than V_{max} and/or smaller than V_{min}. In the initialization phase of CORAD, all the nodes execute the sampling operation to generate detections that are reported to the cluster-heads for r times (the value of r is specified in Table 5.1). After the initialization phase, CORAD is run for 100 rounds: in each round, the clusters are reconfigured based on the CORAD reconfiguration procedure described in Algorithm 3.

To show the CORAD performances, we define several parameters:

- The *number of created clusters* and the *average size of the clusters* to show the effectiveness of the clustering rules of CORAD.
- The *mean residual energy of the nodes*, the *standard deviation of the residual energy* and the *ratio of dead nodes* due to the energy depletion measured at the end of the simulation, to show the energy-efficiency of CORAD.
- The *mean square error* of the aggregated values in the cluster-heads, the average *number of lost messages* and the average *ratio of lost events* due to energy depletion of the nodes, to show the quality of data of CORAD.

All these parameters are averaged over 20 simulation runs taking as input 20 sets of synthetic data generated using the model presented in Section 5.1.3. The error bars on the figures represent the standard deviation of the measured values.

Nevertheless, before comparing CORAD to other algorithms, we analyze the above-stated performance parameters with respect to $\mathcal{S}_{min}$, n, **Thd** and ϵ in order to determine the CORAD optimal behavior.

Algorithm 3 Dynamic Reconfiguration Procedure.

Require: : $\mathbf{CH}_t \leftarrow \{ch_1, ch_2, \ldots, ch_{k_t}\}$, Δ_t, T_1, T_2, p

$M_t \leftarrow \bigcup_{i=1}^{k_t} \mathbf{S}_{ch_i}^n$ {$1 \rightarrow n$-*hops selected nodes*}
$L_t \leftarrow \bigcup_{i=1}^{k_t} \mathbf{S}_{ch_i}^1$ {1-*hop selected nodes*}
if $|M_t| > T_1$ **then**
  if $\Delta_t > T_2$ **then**
    $K_t \leftarrow \bigcup_{i=1}^{k_t} \mathbf{V}_{n-1}^*(ch_i)$ {$(n-1)$-*hops neighbors of cluster-heads*}
    $\mathbf{CR}_{t+1} \leftarrow M_t - \{\bigcup_{j \in K_t} \mathbf{S}_j^1\}$ {*eliminating nodes*}
    $V_{min} \leftarrow V_{min} - \Delta_t$
    $V_{max} \leftarrow V_{max} + \Delta_t$
    $\mathcal{S}_{min} \leftarrow \mathcal{S}_{min} + |L_t|/k_t$ {*add mean size of* 1-*hop neighboring set*}
    $\mathbf{Thd} \leftarrow \mathbf{Thd} + 1$
  end if
else
  if $\Delta_t \leq T_2$ **then**
    $K_t' \leftarrow \bigcup_{i=1}^{k_t} \mathbf{V}_n^*(ch_i)$
    $\mathbf{CR}_{t+1} \leftarrow M_t \cup \{\bigcup_{j \in K_t'} \mathbf{S}_j^1\}$ {*eliminating nodes*}
    $V_{min} \leftarrow V_{min} + \Delta_t$
    $V_{max} \leftarrow V_{max} - \Delta_t$
    $\mathcal{S}_{min} \leftarrow \mathcal{S}_{min} - |L_t|/k_t$ {*subtract mean size of* 1-*hop neighboring set*}
    $\mathbf{Thd} \leftarrow \mathbf{Thd} - 1$
  end if
end if
$\mathbf{R_p} \leftarrow random_select(p)$
$\mathbf{CR}_{t+1} \leftarrow \mathbf{CR}_{t+1} \cup \mathbf{R_p}$

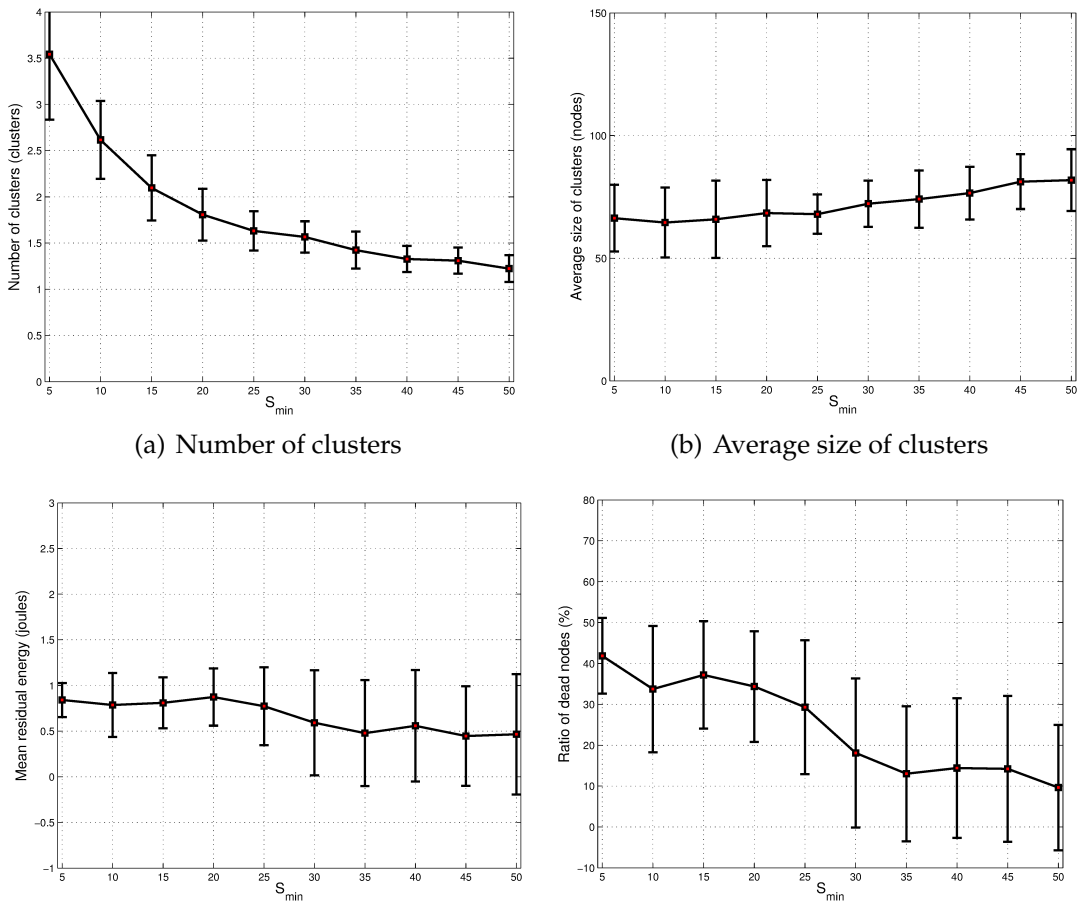
Table 5.1: Simulation parameters.

Parameters	Values
m	500 (nodes)
R_c	10 (m)
R_s	5 (m)
E_{max}	2 (Joules)
λ	0.03
τ	3.0×10^{-6}
l	500 (bytes)
α	2
k	5.0×10^{-9}
c	25.0×10^{-6}
A	100 (m)
r	2000 (periods)
V_{min}	-0.05
V_{max}	$+0.05$

Analysis & Discussion with Respect to $\mathcal{S}_{min}$

Parameter $\mathcal{S}_{min}$ means that the CORAD clustering process is not triggered until the minimum number of the nodes with correlated-values within n-hop neighborhood becomes greater that $\mathcal{S}_{min}$.

To analyze this parameter, we set the constant parameters as follows: $n = 5$, $\mathbf{Thd} = 5$, $\epsilon = 0.01$, and we vary the parameter $\mathcal{S}_{min}$ in $\{5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}$. The obtained results are shown in Figure 5.10.



(a) Number of clusters (b) Average size of clusters

(c) Mean residual energy (d) Ratio of dead nodes

Figure 5.10: CORAD performance vs. $\mathcal{S}_{min}$.

- Given a dense network (as in our case $m = 500$ and $A = 100$) where the events are spatially correlated and setting high values to $\mathcal{S}_{min}$, this means that the end-user application needs to aggregate the *multiple* detected values about a *very localized* event. This, of course, enhances the data accuracy but increases the energy consumption of the whole network because lot of nodes are selected to meet this constraint as shown in sub-figure 5.10(c).

- On the other hand, setting small values to $\mathcal{S}_{min}$ may create many clusters with small sizes, especially when the event is not extent as we can see in sub-figure 5.10(a) and sub-figure 5.10(b).
- It is easy to observe that the average size of the clusters increases with $\mathcal{S}_{min}$ (sub-figure 5.10(b)) because of the spatial correlation of the events.
- Sub-figure 5.10(d) shows the ratio of dead nodes due to the energy depletion. We can see that this ratio decreases with $\mathcal{S}_{min}$ because the created clusters have a large size, which balances the load of the data reporting process. Thus, many nodes participate in relaying data at the opposite of the case where $\mathcal{S}_{min}$ is small. In this case few nodes relay the same volume of data. Recall here that the main cause of nodes' death is the energy depletion during the process of data collection.

From the above discussion, we deduce that $\mathcal{S}_{min}$ should be set to medium value (in our case $\mathcal{S}_{min} = 25$) in order to trade between the number of the clusters (and their average size) and the residual energy of the nodes (which is maximum when $\mathcal{S}_{min} = 25$).

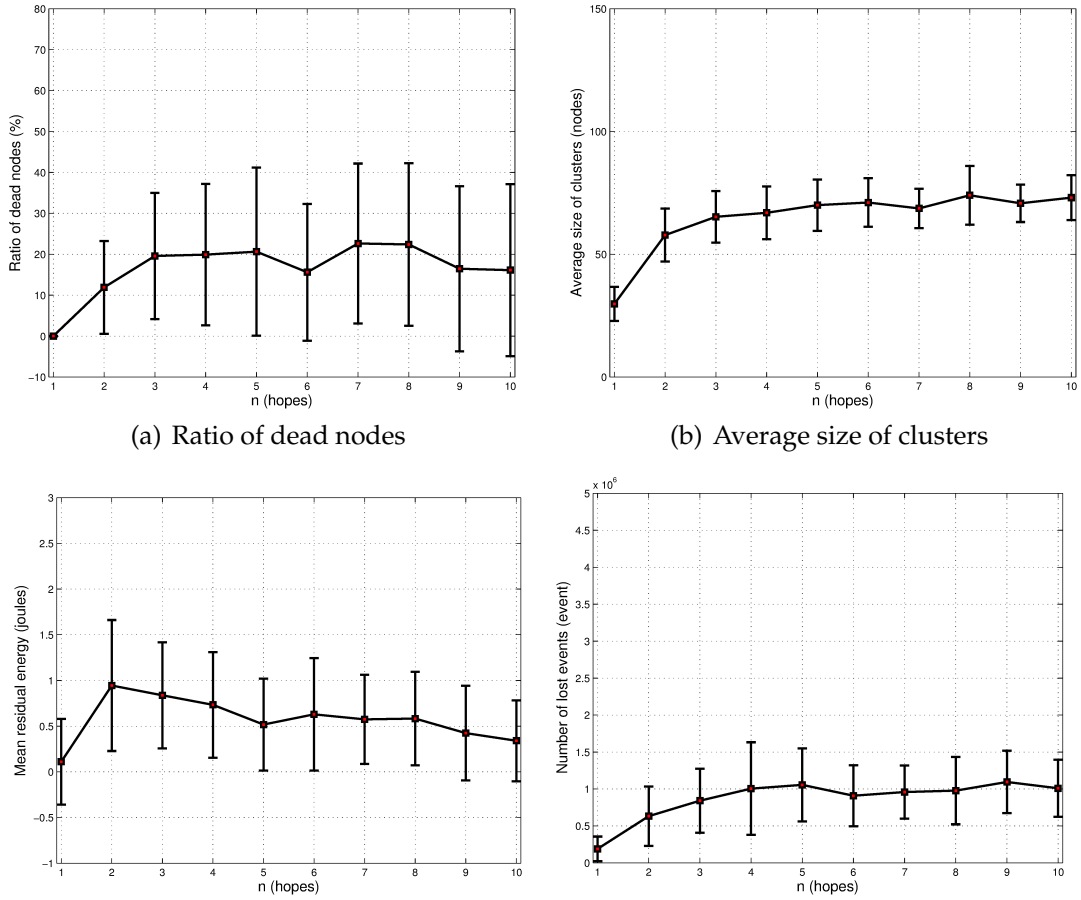
Analysis & Discussion with Respect to n

The parameter n reflects the extent of the clustering. When the event is *localized* then n should have small values to reduce the energy consumption in constructing the clusters and reporting the data. Otherwise, when the event is *network-wide* with strong spatial-temporal correlation, then n should be large.

To analyze this parameter, we set the constant parameters as follows: $\mathcal{S}_{min} = 25$, $\mathbf{Thd} = 5$, $\epsilon = 0.001$, and we vary n in $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. Figure 5.11 illustrates the obtained results.

- When n has large values, this can lead to excessive energy consumption because the length of the routes of data collection may increase with spread events as shown in sub-figure 5.11(a) and sub-figure 5.11(d); The ratio of dead nodes increases from 0% to 30% when n varies from 1 to 10. The same behavior with the number of lost events.

  Note here that an event is declared lost when a participating node is scheduled to do sampling but it cannot do that because of the energy depletion.
- As shown in sub-figure 5.11(b), the average size of the clusters is quasi-constant for the different values of n (it increases slightly with n). This means that even within a dense network, when the physical phenomenon being monitored presents weak spatial-temporal data correlation, increasing n does not improve the quality of data. Hence, n depends on the *degree* of correlation of the physical phenomenon.



(a) Ratio of dead nodes (b) Average size of clusters

(c) Mean residual energy (d) Number of lost events

Figure 5.11: CORAD performance vs. n.

- The mean residual energy decreases for high and small values of n and increases for medium values $n = 4$ or $n = 5$ (sub-figure 5.11(c)).

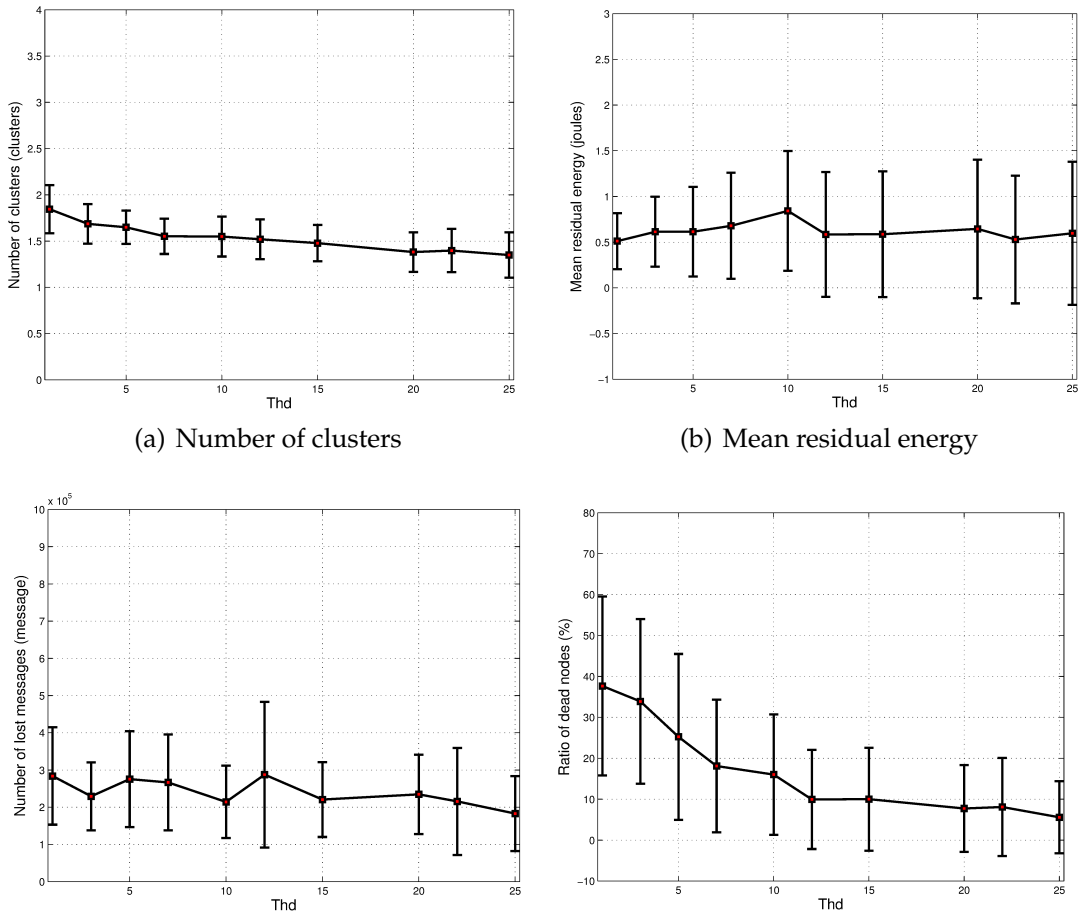
From the above discussion, we set $n = 5$ for optimal CORAD performance.

Analysis & Discussion with Respect to Thd

The parameter **Thd** determines the accuracy of the sampling operations. It depends on the nature of the physical phenomenon and the level of the quality of data required by the end-user application.

To analyze this parameter, we set the constant parameters as follows: $\mathcal{S}_{min} = 25$, $n = 5$, $\epsilon = 0.001$, and we vary **Thd** in $\{1, 3, 5, 7, 10, 12, 15, 20, 22, 25\}$. The obtained results are shown in Figure 5.12.

- From sub-figure 5.12(a), we can see that high values of **Thd** reduce the set



(a) Number of clusters (b) Mean residual energy

(c) Number of lost messages (d) Ratio of dead nodes

Figure 5.12: CORAD performance vs. **Thd**.

of detected values that satisfy the min-max model (subsection 5.3.1). When the physical phenomenon does not have strong-correlated function then the set of detecting nodes will be small. Thus, the number of clustered-nodes is reduced which decreases the number of created clusters.

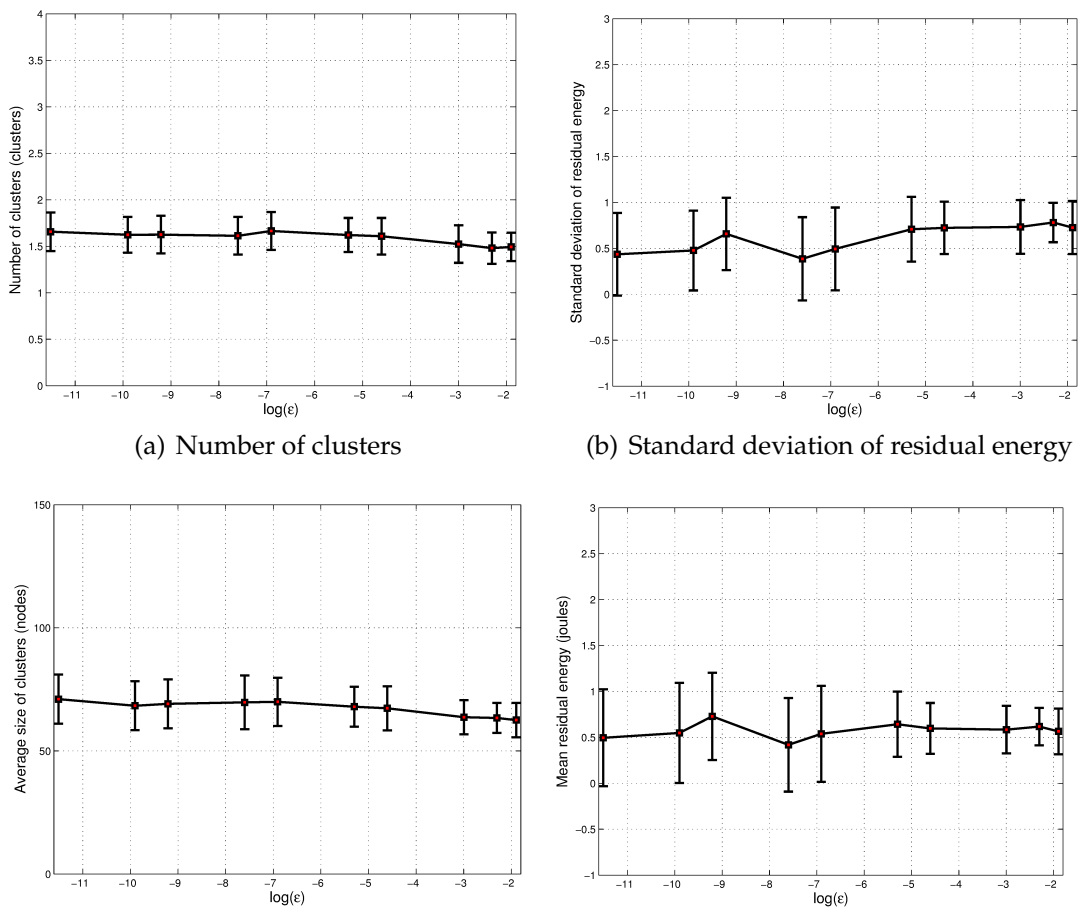
- However, as shown in sub-figure 5.12(b), the mean residual energy is kept quasi-constant because parameter **Thd** does not influence on the clustering structure of CORAD and the length of the paths of data collection, but it is directly related to the data accuracy.
- We can see on sub-figure 5.12(d) that the ratio of dead nodes decreases with **Thd** because few nodes participate in the data reporting operation when **Thd** is high.
- Also, the number of lost messages is quasi-constant (Figure 5.12(c)) because

it is *only* related to the mean residual energy of the nodes.

Analysis & Discussion with Respect to ϵ

The parameter ϵ specifies the *sharpness* of the correlation model. With strong-correlated data, very small values of ϵ do not influence on the number of the nodes with correlated-values. However, in the case of weak-correlated data, ϵ has a strong impact on the data correlation capture model.

To analyze this parameter, we set the constant parameters as follows: $\mathcal{S}_{min} = 25$, $n = 5$, $\mathbf{Thd} = 5$, and we vary ϵ in $\{10^{-5}, 5.10^{-5}, 10^{-4}, 5.10^{-4}, 10^{-3}, 5.10^{-3}, 10^{-2}, 5.10^{-2}, 0.1, 0.15\}$. Figure 5.13 show the obtained results.



(a) Number of clusters (b) Standard deviation of residual energy

(c) Average size of clusters (d) Mean residual energy

Figure 5.13: CORAD performance vs. ϵ.

- The number of the clusters and their average sizes are *insensitive* regarding ϵ because of the values V_{min} and V_{max} of the Min-Max model are relatively small (see sub-figure 5.13(a) and sub-figure 5.13(c)).

- The mean residual energy and the standard deviation of the residual energy graphs (sub-figure 5.13(b) and sub-figure 5.13(d)) have the same shape. They vary *slightly* with respect to ϵ. This is related to the tracked phenomenon for which V_{max} and V_{min} are relatively small.

Note here that ϵ is logarithmically scaled for presentation convenience.

5.5 Conclusion

Exploiting sensory data to optimize the energy in WSN is a shortfall that we have used in this chapter. We explored this track and proposed a dynamic clustering algorithm for data collection that takes profit from the data correlations to minimize the energy consumption.

Our algorithm CORAD maintains a certain level of quality of data while minimizing the energy consumption by: (i) smart selection of the nodes to be clustered, (ii) optimized election of the cluster-heads and the nodes that report data over the data collection tree, and (iii) reconfiguration of the clusters each time the physical phenomenon changes.

Certainly, CORAD is a model-depend algorithm but the results we obtained can be generalized to any other position-aware geographic-based clustering and data collection algorithm.

All the proposed and tested models are generic and can be applied to monitor or track any physical phenomenon with scaled-value detections.

We have proposed CORAD as a heuristic solution to the EMDP problem which is NP-Complete. It is not expected that CORAD provides the optimal solution but the obtained results show that it behaves well in many different configurations. Nonetheless, a comparison with other heuristic solutions is necessary to measure the *near-optimality* of CORAD.

Chapter 6

Conclusions & Future Work

THE RECENT ADVANCES in MEMS technology enable WSN applications in many domains such as: military, urban, health-care, agriculture, etc. However, the energy problem as a technological bottleneck slows down the development of these applications and thereby remains an open issue.

In our thesis, we approached this problem in the context of collaborative applications using some conceptual and mathematical tools, namely: the prediction-based schemes for target tracking using limited-range sensors, the correlation-based adaptive clustering for data collection, and the node selection based on energy-efficiency.

Our goal was to design energy-efficient mechanisms for data collection and tracking that take profit from the intrinsic characteristics of WSN such as: the limited ranges of sensing and communication, the density of the network and the correlated sensory data.

As it is shown, the severe hardware limitation of the sensors and the growing demand for data processing and communication involve new paradigms for the design of energy-efficient algorithms in WSN, including the optimization in the software layers of the protocol stack.

To undertake these challenges, we first studied the recent schemes proposed in the literature to extract the recommendations for possible research tracks. The elements that inspired us in designing our approaches are:

- The conception of the network as a sensing subsystem and a communication subsystem related by a prediction algorithm;
- The perception of WSN as a source of data that can used to optimize the network operations, and;
- The exploitation of the data correlations in the optimization effort of the energy consumption.

6.1 Contributions

Given these inspiring elements, we proposed four contributions, each of which was presented in a whole chapter:

1. A survey of recent proposed target tracking schemes based on predictions.
2. A target tracking scheme that aims to reduce the energy consumption by using the distributed Kalman Filter coupled with a dynamic clustering algorithm, called DKF_DC.
3. A mathematical model for the problem of energy optimization in the context of correlated data collection under data precision constraints, called EMDP.
4. A heuristic solution for the EMDP problem called CORAD which is a correlation based adaptive clustering algorithm for data collection that exploits data correlations.

We validated these contributions via mathematical proofs and/or implementations for simulation. We confirmed via the obtained results our proposed ideas and hypothesis for the design of energy-efficient algorithms in collaborative applications. We showed that the trade-off between the quality of the collected data and the energy efficiency is difficult to achieve, and depends on the application requirements.

Moreover, we argued that EMDP problem is *enough* complex that requires heuristic-based approximate solutions. Our greedy-based heuristic solution CORAD is one of them. We showed that it minimizes the energy consumption while it maintains an acceptable level of data accuracy.

6.2 Future Work

In view of these results, two research tracks, at least, can be released for future work:

6.2.1 Characterization of EMDP

Our BILP model for EMDP is a first step in its characterization in terms of complexity. As BILP models are a special case of ILP (Integer Linear Programming) and 0-1 IP (0-1 Integer Programming) models which are proved as NP-Complete [127], it gives us insight about the complexity of EMDP. However, it is necessary to formally define a reduction function that reduces, for example, ILP or 0-1 IP to EMDP, i.e.

$$\mathsf{ILP} \leq \mathsf{EMDP}$$

It is also necessary to prove that this reduction function is polynomial, i.e. its computation is bounded by a polynomial function of the input size, in order to prove the NP-Completeness of EMDP.

Nonetheless, this function may not exist; In this case, we have to prove its non-existence or to design a *polynomial algorithm* that solves EMDP to demonstrate that it is *not* NP-Complete, i.e. *polynomial*.

The characterization of EMDP may be a complex process because we have to explore all the possibilities to find the complexity class to which it belongs. We consider that this is a promising track for research.

6.2.2 Meta-Heuristics

In the case of the NP-Completeness of EMDP, the problem is intractable and the exact solutions are infeasible for medium and big instances. Thus, designing approximate solutions is the only possible approach of the problem.

Like our proposed CORAD solution, heuristics and meta-Heuristics are commonly applied in such context. The usual optimization methods such as: ant-colony, taboo search, simulated annealing, genetic programming, etc. can be used to design approximate solutions. Each heuristic-based solution can be evaluated in terms of its approximation of the exact solution. Then, the different solutions can be compared to find the trade-off between time-space computation cost and the *quality* of solution.

Bibliography

[1] Joanna Mcouat. *Wireless Sensor Networks: Principles, Design and Applications*. Springer Science & Business Media, 2013.

[2] Pedro José Marron, Stamatis Karnouskos, and Daniel Minder. *Research Roadmap on Cooperating Objects*. Office for Official Publications of the European Communities, 2009.

[3] Jukka Suhonen, Mikko Kohvakka, Ville Kaseva, Timo D. Hämäläinen, and Marko Hännikäinen. *Low-Power Wireless Sensor Networks: Protocols, Services and Applications* . Springer Science+Business Media, 2012.

[4] Ivan Stojmenovic. *Handbokk of Sensor Networks: Algorithms and Architecture*. Wiley Series on Parallel and Distributed Computing. John Wiley & Sons, Inc., Hoboken, New Jersey, 2005.

[5] Meyer Sound. *http://www.meyersound.com/sites/default/files/mica_ds.pdf*.

[6] Crossbow. *http://www.eol.ucar.edu/isf/facilities/isa/internal/CrossBow/DataSheets/mica2.pdf*.

[7] Memsic. *http://www.memsic.com/userfiles/files/DataSheets/WSN/micaz_datasheet-t.pdf*.

[8] Moteiv Corporation. *http://www4.ncsu.edu/ kkolla/CSC714/datasheet.pdf*.

[9] Crossbow. *http://www.willow.co.uk/TelosB_Datasheet.pdf*.

[10] Intel Corporation. *http://wsn.cse.wustl.edu/images/c/cb/Imote2-ds-rev2_2.pdf*.

[11] Kristopher S.J. Pister. Smart Dust. Technical report, UC Berkeley, 497, Cory Hall Berkeley CA 94720, 1997.

[12] Jeremy Blum. *Exploring Arduino: Tools and Techniques for Engineering Wizardry*. John Wiley & Sons, August 2013.

[13] Karel Heurtefeux. *Protocoles localisés pour réseaux de capteurs*. PhD thesis, L'Institut National des Sciences Appliquées de Lyon, 2009.

[14] Admar Ajith Kumar Somappa, Knut Øvsthus, and Lars Michael Kristensen. An Industrial Perspective on Wireless Sensor Networks; A Survey of Requirements, Protocols, and Challenges. *Communications Surveys Tutorials, IEEE*, 16(3):1391–1412, Third 2014.

[15] Kamrul Islam, Weiming Shen, and Xianbin Wang. Wireless Sensor Network Reliability and Security in Factory Automation: A Survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 42(6):1243–1256, Nov 2012.

[16] Rumen Kyusakov, Jens Eliasson, Jerker Delsing, Jan van Deventer, and Jonas Gustafsson. Integration of Wireless Sensor and Actuator Nodes With IT Infrastructure Using Service-Oriented Architecture. *Industrial Informatics, IEEE Transactions on*, 9(1):43–51, Feb 2013.

[17] Seong-Eun Yoo, Poh Kit Chong, Daeyoung Kim, Yoonmee Doh, Minh-Long Pham, Eunchang Choi, and Jaedoo Huh. Guaranteeing Real-Time Services for Industrial Wireless Sensor Networks With IEEE 802.15.4. *Industrial Electronics, IEEE Transactions on*, 57(11):3868–3876, Nov 2010.

[18] William Merrill. Where is the return on investment in wireless sensor networks? *Wireless Communications, IEEE*, 17(1):4–6, February 2010.

[19] Miguel A. Labrador and Pedro M. Wightman. *Topology Control in Wireless Sensor Networks* . Springer Science + Business Media B.V, 2009.

[20] T. Kalaivani, A. Allirani, and P. Priya. A survey on Zigbee based wireless sensor networks in agriculture. In *Trendz in Information Sciences and Computing (TISC), 2011 3rd International Conference on*, pages 85–89, Dec 2011.

[21] Mounib Khanafer, Mouhcine Guennoun, and Hussein T. Mouftah. WSN Architectures for Intelligent Transportation Systems. In *New Technologies, Mobility and Security (NTMS), 2009 3rd International Conference on*, pages 1–8, Dec 2009.

[22] A. Pascale, M. Nicoli, F. Deflorio, B. Dalla Chiara, and U. Spagnolini. Wireless sensor networks for traffic management and road safety. *Intelligent Transport Systems, IET*, 6(1):67–77, March 2012.

[23] Jin Zhou, Chiouguey L Phillip Chen, Long Chen, and Wei Zhao. A User-Customizable Urban Traffic Information Collection Method Based on Wireless Sensor Networks. *Intelligent Transportation Systems, IEEE Transactions on*, 14(3):1119–1128, Sept 2013.

[24] Geoffery Werner-Allen, Konrad Lorincz, Matt Welsh, Omar Marcillo, Jeff Johnson, Mario Ruiz, and Jonathan Lees. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, 10(2):18–25, 2006.

[25] Chunlei An, Andras Timm-Giel, and Carmelita Goerg. Virtual Sensor Network Lifeline for Communications in Fire Fighting Rescue Scenarios. In *Vehicular Technology Conference Fall (VTC 2009-Fall), 2009 IEEE 70th*, pages 1–5, Sept 2009.

[26] Yuan Zhang, Limin Sun, Houbing Song, and Xiaojun Cao. Ubiquitous WSN for Healthcare: Recent Advances and Future Prospects. *Internet of Things Journal, IEEE*, 1(4):311–318, Aug 2014.

[27] Ahmet Burak Gokbayrak, Sinan Divarci, and Oguzhan Urhan. Wireless sensor network gateway design for home automation applications. In *Signal Processing and Communications Applications Conference (SIU), 2014 22nd*, pages 1770–1773, April 2014.

[28] Debraj Basu, Giovanni Moretti, Gourab Sen Gupta, and Stephan Marsland. Wireless sensor network based smart home: Sensor selection, deployment and monitoring. In *Sensors Applications Symposium (SAS), 2013 IEEE*, pages 49–54, Feb 2013.

[29] Bhaskar Krishnamachari, Deborah Estrin, and Stephen Wicker. Modelling data-centric routing in wireless sensor networks . In *INFOCOM 2002* , June 2002.

[30] Waltenegus Dargie and Christian Poellabauer. *Fundamentals of Wireless Sensor Networks: Theory and Practice*. A John Wiley and Sons, Ltd., Publication, 2010.

[31] Ian.F. Akyildiz, Weilian Su, Yogi Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422, 2002.

[32] Giuseppe Anastasi, Marco Conti, Mario Di Francesco, and Andrea Passarella. Energy conservation in wireless sensor networks: A survey. *Ad Hoc Networks*, 7(3):537–568, 2009.

[33] Nikolaos A. Pantazis and Demitrios D. Vergados. A survey on power control issues in wireless sensor networks. *Communications Surveys & Tutorials, IEEE*, 9(4):86–107, 2007.

[34] Azrin Abd Aziz, Y. Ahmed Sekercioglu, Paul Fitzpatrick, and Milosh Ivanovich. A Survey on Distributed Topology Control Techniques for Extending the Lifetime of Battery Powered Wireless Sensor Networks. *Communications Surveys & Tutorials, IEEE*, PP(99):1–24, 2012.

[35] Volkan Isler and Ruzena Bajcsy. The Sensor Selection Problem for Uncertainty Sensing Models. *Transactions on Automation Science and Engeneering*, 3(4):372–381, October 2006.

[36] Wendong Xiao, Chen Khong Tham, and Sajal K. Das. Collaborative Sensing to Improve Information Quality for Target Tracking in Wireless Sensor Networks. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, pages 99–104. IEEE, 2010.

[37] Abdelmalik Bachir, Mischa Dohler, Thomas Watteyne, and Kin K. Leung. MAC essentials for wireless sensor networks. *Communications Surveys & Tutorials, IEEE*, 12(2):222–248, 2010.

[38] O. Demigha, W.-K. Hidouci, and T. Ahmed. On Energy Efficiency in Collaborative Target Tracking in Wireless Sensor Network: A Review. *Communications Surveys Tutorials, IEEE*, 15(3):1210–1222, Third 2013.

[39] Greg Welch and Gary Bishop. An introduction to the Kalman filter. *University of North Carolina at Chapel Hill, Chapel Hill, NC*, 1995.

[40] Eric A. Wan and Rudolph Van Der Merwe. The unscented Kalman filter for nonlinear estimation. In *The IEEE 2000 Symposium On Adaptive Systems for Signal Processing, Communications, and Control 2000. AS-SPCC.*, pages 153–158. IEEE, 2002.

[41] Reza Olfati-Saber and Nils F. Sandell. Distributed Tracking in Sensor Networks with Limited Sensing Range. In *American Control Conference, 2008*, pages 3157–3162. IEEE, June 2008.

[42] James Kennedy and Russel Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948. IEEE, 1995.

[43] Jing Teng, Hichem Snoussi, and Cedric Richard. Decentralized Variational Filtering for Target Tracking in Binary Sensor Networks. *IEEE Transactions on Mobile Computing*, 9(10):1465–1477, October 2010.

[44] Richard J. Meinhold and Nozer D. Singpurwalla. Understanding the Kalman filter. *American Statistician*, 37(2):123–127, 1983.

[45] James V. Candy. Model-based signal processing. *The Journal of the Acoustical Society of America*, 119:2553, 2006.

[46] Ramesh Rajagopalan and Pramod K. Varshney. Data-aggregation techniques in sensor networks: a survey. *Communications Surveys & Tutorials, IEEE*, 8(4):48–63, 2006.

[47] Feng Zhao, Jaewon Shin, and James Reich. Information-Driven Dynamic Sensor Collaboration. *IEEE Signal Processing Magazine*, 19(2):61–72, March 2002.

[48] Djamel Benferhat and Jean Frédéric Myoupo. A Physical CPT and Regional CSP-Based Hybrid Algorithm for Energy Efficiency in Target Tracking in Wireless Sensor Networks. In *2010 Ninth International Conference on Networks*, pages 127–132, 2010.

[49] Hanbiao Wang, Greg Pottie, Kung Yao, and Deborah Estrin. Entropy-based Sensor Selection Heuristic for Target Tracking. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, pages 36–45. ACM, April 2004.

[50] Emre Ertin, John W. Fisher, and Lee C. Potter. Maximum mutual information principle for dynamic sensor query problems. In *Information Processing in Sensor Networks*, pages 558–558. Springer, 2003.

[51] Wei Zhao, Ying Han, Hai Wu, and Lei Zhang. Weighted Distance Based Sensor Selection for Target Tracking in Wireless Sensor Networks. *Signal Processing Letters, IEEE*, 16(08):647–650, August 2009.

[52] Reza Olfati-Saber. Distributed Tracking for Mobile Sensor Networks with Information-Driven Mobility. In *Proceedings of the 2007 American Control Conference*, pages 4606–4612, July 2007.

[53] Jean Michel Passerieux and Dominique Van Cappel. Optimal observer maneuver for bearings-only tracking. *Aerospace and Electronic Systems, IEEE Transactions on*, 34(3):777–788, 1998.

[54] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34. ACM, 1987.

[55] Reza Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *Automatic Control, IEEE Transactions on*, 51(3):401–420, 2006.

[56] James E. Weimer, Bruno Sinopoli, and Bruce H. Krogh. Relaxation Approach to Dynamic Sensor Selection in Large-Scale Wireless Networks. *28th International Confrence on Distributed Computing Systems Workshops*, pages 501–506, 2008.

[57] Jianyong Lin, Wendong Xiao, Frank L. Lewis, and Lihua Xie. Energy-Efficient Distributed Adaptive Multisensor Scheduling for Target Tracking in Wireless Sensor Networks. *Instrumentation and Measurement, IEEE Transactions on*, 58(06):1886–1896, June 2009.

[58] XIAO Wen-Dong, WU Jian-Kang, XIE Li-Hua, and DONG Liang. Sensor Scheduling for Target Tracking in Networks of Active Sensors. *ACTA Automatica Sinica*, 32(6):922–928, November 2006.

[59] Bin Liu, Fengyuan Ren, Chuang Lin, and Xin Jiang. Performance Analysis of Sleep Scheduling Schemes in Sensor Networks Using Stochastic Petri Net. pages 4278–4283, 2008.

[60] Ling Shi, Agostino Capponi, Karl H. Johansson, and Richard M. Murray. Resource optimisation in a wireless sensor network with guaranteed estimator performance. *IET Control Theory and Applications*, 04(05):710– 723, 2010.

[61] Yousef E. M. Hamouda and Chris Phillips. Metadata-Based Adaptive Sampling for Energy-Efficient Collaborative Target Tracking in Wireless Sensor Networks. In *10th IEEE International Conference on Computer and Information Technology*, pages 313–320, 2010.

[62] Shucheng Dai, Changjie Tang, Shaojie Qiao, Yue Wang, Hongjun Li, and Chuan Li. An Energy-Efficient Tracking Algorithm based on Gene Expression Programming in Wireless Sensor Networks. In *The 1st International Conference on Information Science and Engineering*, pages 774–777, 2009.

[63] Gyula. Simon, Miklos Molar, Laszlo Gonczy, and Bernard Cousin. Dependable k-coverage algorithms for sensor networks. In *Instrumentation and Measurement Technology Conference Proceedings, 2007. IMTC 2007. IEEE*, pages 1–6. IEEE, 2007.

[64] Gabor Bergmann, M Molnár, L. Gönczy, and Bernard Cousin. Optimal Period Length for the CGS Sensor Network Scheduling Algorithm. In *Sixth International Conference on Networking and Services*, pages 192–199. IEEE, 2010.

[65] Di Tian and Nocilas D. Georganas. Location and calculation-free node-scheduling schemes in large wireless sensor networks. *Ad Hoc Networks*, 2(1):65–85, 2004.

[66] X. Rong Li and Vesslin P. Jilkov. Survey of Maneuvering Target Tracking. Part III: Motion Models of Ballistic and Space Targets. *IEEE Transactions on Aerospace and Electronic Systems*, 46(01):96–119, January 2010.

[67] Xu Ji, Yi-Ying Zhang, Sajjad Hussain, Dong-Xu Jin, Eun-Mook Lee, and Myong-Soon Park. FOTP: Face-based Object Tracking Protocol in Wireless Sensor Network. In *Fourth International Conference on Computer Sciences and Convergence Information Technology*, 2009.

[68] Md. Zakirul Alam Bhuiyan, Guojun Wang, and Jie Wu. Polygon-Based Tracking Framework in Surveillance Wireless Sensor Networks. In *15th International Conference on Parallel and Distributed Systems*, pages 174–181, 2009.

[69] Akond Ashfaque Ur Rahman, Mahmuda Naznin, and Md. Atiqul Islam Mollah. Energy-efficient Multiple Targets Tracking Using Target Kinematics in Wireless Sensor Networks. In *Fourth International Conference on Sensor Technologies and Applications*, pages 275–280, 2010.

[70] Flavia Delicato, Fabio Protti, Luci Pirmez, and Jose Ferreira de Rezenda. An Efficient Heuristic for Selecting Active Nodes in Wireless Sensor Networks. *Computer Networks*, 50(18):3701–3720, 2006.

[71] Li Liu, Hao Li, Junling Wang, Lian Li, and Caihong Li. Heuristics for Mobile Object Tracking Problem in Wireless Sensor Networks. *Frontiers in Algorithmics*, pages 251–260, 2009.

[72] Manish Bhardwaj and Annatha P. Chandrakasan. Bounding the Lifetime of Sensor Networks via optimal Role Assignments. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1587–1596. IEEE, 2002.

[73] Jianyong Lin, Lihua Xie, and Wendong Xiao. State-Centric Multi-Sensor Scheduling for Target Tracking in Wireless Sensor Networks. In *Information, Communications and Signal Processing, 2007 6th International Conference on*, pages 1–4. IEEE, 2008.

[74] Volodymyr Pryyma, Damla Turgut, and Ladislau Bölöni. Active time scheduling for rechargeable sensor networks. *Computer Networks*, 54(4):631–640, 2010.

[75] Chao-Chun Chen, Jenq-Muh Hsu, and Chien-Han Liao. HAMA: A Three-Layered Architecture for Integrating Object Tracking and Location Management in Wireless Sensor Networks. In *Third International Conference on Multimedia and Ubiquitous Engineering*, pages 268–275, 2009.

[76] Qing Ling, Yinfei Fu, and Zhi Tian. Localized sensor management for multi-target tracking in wireless sensor networks. *Information Fusion*, pages 194–201, 2011.

[77] Kiam H. Ang, Gregory Chong, and Yun Li. PID control system analysis, design, and technology. *Control Systems Technology, IEEE Transactions on*, 13(4):559–576, 2005.

[78] Tom Vercauteren and Xiaodong Wang. Decentralized sigma-point information filters for target tracking in collaborative sensor networks. *IEEE Transactions on Signal processing*, 53:2997–3009, 2005.

[79] Stephan Olariu, Mohamed Eltoweissy, and Mohamed Younis. ANSWER: AutoNomouS netWorked sEnsoR system. *Journal of Parallel and Distributed Computing*, 67(1):111–124, 2007.

[80] Aysegul Alaybeyoglu, Orhan Dagdeviren, Aylin Kantarci, and Kayhan Erciyes. A Distributed Wakening Based Target Tracking Protocol for Wireless Sensor Networks. In *Ninth International Symposium on Parallel and Distributed Computing*, pages 165–172, 2010.

[81] Oscar Garcia, Alejandro Quintero, and Samuel Pierre. A global profile-based algorithm for energy minimization in object tracking sensor networks. *Computer Communications*, 33:736–744, 2010.

[82] WangChang Yang, Zhen Fu, JungHwan. Kim, and MyongSoon. Park. An adaptive dynamic cluster-based protocol for target tracking in wireless sensor networks. In *Proceedings of the joint 9th Asia-Pacific web and 8th international conference on web-age information management conference on Advances in data and web management*, pages 157–167. Springer-Verlag, 2007.

[83] Loredana Arienzo and Maurizio Longo. Energy-Efficient Target Tracking in Sensor Networks. *Ad Hoc Networks*, pages 249–264, 2010.

[84] Xiaofei Xing, Guojun Wang, and Jie Wu. Herd-Based Target Tracking Protocol in Wireless Sensor Networks. *Wireless Algorithms, Systems, and Applications*, pages 135–148, 2009.

[85] Zhibo Wang, Wei Lou, Zhi Wang, Junchoa Ma, and Honglong Chen. A novel mobility management scheme for target tracking in cluster-based sensor networks. *Distributed Computing in Sensor Systems*, pages 172–186, 2010.

[86] Kil-Woong Jang. Location Tracking for Wireless Sensor Networks. *Next Generation Teletraffic and Wired/Wireless Advanced Networking*, pages 306–315, 2007.

[87] Dong-Xu Jin, Sajjad Hussain Chauhdary, Xu Ji, Yi-Ying Zhang, Dong-Hyun Lee, and Myong-Soon Park. Energy-Efficiency Continuous Object Tracking

Via Automatically Adjusting Sensing Range in Wireless Sensor Network. In *Fourth International Conference on Computer Sciences and Convergence Information Technology*, 2009.

[88] Jun-Hwan Kim, Kee-Bum Kim, Chauhdry S. Hussain, Min-Woo Cui, and Myong-Soon Park. Energy-Efficient Tracking of Continuous Objects in Wireless Sensor Networks. *Ubiquitous Intelligence and Computing*, pages 323–337, 2010.

[89] Cheng Zhong and Michel Worboys. Energy-Efficient Continuous Boundary Monitoring in Sensor Networks. Technical report, Technical Report, 2007. Available: http://ilab1. korea. ac. kr/papers/ref2. pdf, 2007.

[90] Wang-Rong Chang, Hui-Tang Lin, and Zong-Zhi Cheng. CODA: A Continuous Object Detection and Tracking Algorithm for Wireless Ad Hoc Sensor Networks. In *Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE*, pages 168–174. IEEE, 2008.

[91] Bernard Chazelle. Approximation and decomposition of shapes. *Advances in Robotics 1: Algorithmic and Geometric Aspects of Robotics,*, 1:145–185, 1985.

[92] Kuei-Ping Shih, Sheng-Shih Wang, Hung-Chang Chen, and Pao-Hwa Yang. CollECT: Collaborative event detection and tracking in wireless heterogeneous sensor networks. *Computer Communications*, 31:3124–3136, 2008.

[93] Javed Aslam, Zack Butler, Florin Constantin, Vatentino Crespi, George Cybenko, and Daniela Rus. Tracking a Moving Object with a Binary Sensor Network. In *Proceeding of SenSys'03*, pages 150–161. ACM, November 2003.

[94] Donglei Cao, Beihong Jin, Sajal K. Das, and Jiannong Cao. On collaborative tracking of a target group using binary proximity sensors. *Journal of Parallel and Distributed Computing*, 70:825–838, 2010.

[95] Reza Olfati-Saber. Kalman-Consensus Filter : Optimality, Stability, and Performance. In *Proceedings of the Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*, pages 7036–7042. IEEE Computer Society, December 2009.

[96] Usman A. Khan and Joze M. F. Moura. Distributed Kalman Filters in Sensor Networks: Bipartite Fusion Graph. *SSP*, 2007.

[97] Soummya Kar and José M.F. Moura. Gossip and distributed Kalman filtering: Weak consensus under weak detectability. *IEEE Transactions on Signal Processing*, 99, 2010.

[98] Alejandro Ribeiro, Georgios B. Giannakis, and Stergios I. Roumeliotis. SOI-KF: Distributed Kalman Filtering With Low-Cost Communications Using the Sign of Innovations. *IEEE Transactions on Signal processing*, 54(12):4782–4795, December 2006.

[99] Reza Olfati-Saber. Distributed Kalman Filter with Embedded Consensus Filters. In *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference*, pages 8179–8184. IEEE Computer Society, December 2005.

[100] Demetri P. Spanos, Reza Olfati-Saber, and Richard M. Murray. Dynamic consensus on mobile networks. In *The 16th IFAC World Congress, Prague, Czech*, 2005.

[101] Reza Olfati-Saber, J. Alex Fax, and Richard M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[102] Oualid Demigha, Hamza Ould Slimane, Abderahim Bouziani, and Walid-Khaled Hidouci. Energy efficient target tracking in wireless sensor networks with limited sensing range. In *ICSNC 2011, The Sixth International Conference on Systems and Networks Communications*. IARIA, October 181-187.

[103] Henry Medeiros, Johnny Park, and Avinash C. Kak. Distributed Object Tracking Using a Cluster-Based Kalman Filter in Wireless Camera Networks. *IEEE Journal of Selected Topics in Signal Processing*, 02(04):448–463, August 2008.

[104] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. TOSSIM: accurate and scalable simulation of entire TinyOS applications. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 126–137. ACM, 2003.

[105] Yujie Zhu, Karthikeyan Sundaresan, and Raghupathy Sivakumar. Practical limits on achievable energy improvements and useable delay tolerance in correlation aware data gathering in wireless sensor networks. In *Sensor and Ad Hoc Communications and Networks, 2005. IEEE SECON 2005. 2005 Second Annual IEEE Communications Society Conference on*, pages 328–339, 2005.

[106] Ugo Monaco, Francesca Cuomo, Tommaso Melodia, Fabio Ricciato, and Marco Borghini. Understanding optimal data gathering in the energy and latency domains of a wireless sensor network. *Computer Networks*, 50(18):3564–3584, 2006.

[107] Razvan Cristescu, Baltasar Beferull-Lozano, and Martin Vetterli. On network correlated data gathering. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2571–2582 vol.4, 2004.

[108] Jamal N. Al-Karaki and Ahmed E. Kamal. On the correlated data gathering problem in wireless sensor networks. In *Computers and Communications, 2004. Proceedings. ISCC 2004. Ninth International Symposium on*, volume 1, pages 226–231 Vol.1, 2004.

[109] Wenjuan Liu, Dongmei Zhao, and Gang Zhu. Association Schemes in a Wireless Sensor Network with a Cluster Tree Topology. In *Vehicular Technology Conference Fall (VTC 2010-Fall), 2010 IEEE 72nd*, pages 1–5, Sept 2010.

[110] Liansheng Tan, Yanlin Gong, and Gong Chen. A Balanced Parallel Clustering Protocol for Wireless Sensor Networks Using K-Means Techniques. In *Sensor Technologies and Applications, 2008. SENSORCOMM '08. Second International Conference on*, pages 300–305, 2008.

[111] Zhao Qun, Shen Gang, and Jin Shan. Energy Consumption Optimization for Data Collection with Precision Constraints in Wireless Sensor Networks. In *Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*, pages 1–5, April 2009.

[112] Oualid Demigha, Walid-Khaled Hidouci, and Toufik Ahmed. A Novel BILP Model for Energy Optimization Under Data Precision Constraints in Wireless Sensor Networks. *Communications Letters, IEEE*, 18(12):2185–2188, Dec 2014.

[113] Mehmet C. Vuran, Özgur B. Akan, and Ian F. Akyildiz. Spatio-temporal correlation: theory and applications for wireless sensor networks. *Computer Networks*, 45(3):245–259, 2004.

[114] Bugra Gedik, Ling Liu, and Philip S. Yu. ASAP: an adaptive sampling approach to data collection in sensor networks. *Parallel and Distributed Systems, IEEE Transactions on*, 18(12):1766–1783, 2007.

[115] Trong Duc Le, Ngoc Duy Pham, and Hyunseung Choo. Towards a distributed clustering scheme based on spatial correlation in wsns. In *Wireless Communications and Mobile Computing Conference, 2008. IWCMC'08. International*, pages 529–534. Ieee, 2008.

[116] Chong Liu, Kui Wu, and Jian Pei. An energy-efficient data collection framework for wireless sensor networks by exploiting spatiotemporal correlation. *Parallel and Distributed Systems, IEEE Transactions on*, 18(7):1010–1023, 2007.

[117] Hongbo Jiang, Shudong Jin, and Chonggang Wang. Prediction or not? An energy-efficient framework for clustering-based data collection in wireless sensor networks. *Parallel and Distributed Systems, IEEE Transactions on*, 22(6):1064–1071, 2011.

[118] Anand Meka and Ambuj Singh. Distributed spatial clustering in sensor networks. *Advances in Database Technology-EDBT 2006*, pages 980–1000, 2006.

[119] Xueyan Tang and Jianliang Xu. Adaptive data collection strategies for lifetime-constrained wireless sensor networks. *Parallel and Distributed Systems, IEEE Transactions on*, 19(6):721–734, 2008.

[120] Sunhee Yoon and Cyrus Shahabi. The Clustered AGgregation (CAG) technique leveraging spatial and temporal correlations in wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 3(1):3, 2007.

[121] Leandro A. Villas, Azzedine Boukerche, Horacio A.B.F. de Oliveira, Regina B. de Araujo, and Antonio A.F. Loureiro. A spatial correlation aware algorithm to perform efficient data collection in wireless sensor networks. *Ad Hoc Networks*, pages 69–85, January 2014.

[122] Leandro A. Villas, Azzedine Boukerche, Daniel L. Guidoni, Horacio A.B.F. de Oliveira, Regina B. de Araujo, and Antonio A.F. Loureiro. An Energy-Aware Spatio-Temporal Correlation Mechanism to Perform Efficient Data Collection in Wireless Sensor Networks. *Computer Communications*, 2012.

[123] Wei Ye, John Heidemann, and Deborah Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *Networking, IEEE/ACM Transactions on*, 12(3):493–506, 2004.

[124] Chao Sha, Ruchuan Wang, Haiping Huang, and Lijuan Sun. Energy efficient clustering algorithm for data aggregation in wireless sensor networks. *The Journal of China Universities of Posts and Telecommunications*, 17:104–109, 2010.

[125] Davide Zordan, Giorgio Quer, Michele Zorzi, and Michele Rossi. Modeling and Generation of Space-Time Correlated Signals for Sensor Network Fields. In *Proceedings of IEEE Globecom*, pages 5–9, 2011.

[126] Anwar Hossain, Prabir Kumar Biswas, and Saswat Chakrabarti. Sensing models and its impact on network coverage in wireless sensor network. In *Industrial and Information Systems, 2008. ICIIS 2008. IEEE Region 10 and the Third international Conference on*, pages 1–5. IEEE, 2008.

[127] Richard M. Karp. Reducibility Among Combinatorial Problems. *Complexity of Computer Computations*, pages 85–103, 1972.

Appendix A

The Kalman Filter

The Kalman Filter (KF) is a mathematical power tool that is applied to many important sensing based systems such as computer graphics, simulators of music instruments, target tracking, fitting spline surfaces, etc. It is the optimal estimator for a large class of problems, and its advantage is its easy usage.

The KF was first proposed in a famous paper published in 1960 by *Rudolf Kalman*: "*A New Approach to Linear Filtering and Prediction Problems*", Transaction ASME, Journal of Basic Engineering., vol 18, pages 34-45. R. Kalman based the construction of the state estimation filter on the properties of the Gaussian random variables. He proposed to minimize the *state vector covariance norm*, which yield to a classical recursion: the new state is deduced from the previous estimation by adding a correction term proportional to the prediction error.

A.1 Preliminaries

Te explain the essence of the KF, some preliminary concepts about the theory of probability and random variables are necessary.

A.1.1 Probability

Formally, the probability of the occurrence of a discrete event A is given by:

$$p(A) = \frac{possible_outcomes_favoring_event_A}{total_number_of_all_possible_outcomes}$$

The probability of an outcome favoring either A or B is given by:

$$p(A \cup B) = p(A) + p(B)$$

If two outcome events are independent then the probability of both occurring is the product of their individual probabilities:

$$p(A \cap B) = p(A)p(B)$$

Finally, the probability of outcome event A given the occurrence of outcome event B is called *conditional probability*, and is given by:

$$p(A|B) = \frac{p(A \cap B)}{p(B)}$$

A.1.2 Random Variables

At the opposite to discrete events, in the case of tracking or motion capture for example, the occurrence of the events is *continuous*. So, their probabilities are represented by continuous random variables. A random variable is typically a function that maps all the points of the sample space to real numbers. For example, if $X(t)$ maps time to position, then it would tell us at any instant, the expected position.

In the case of continuous random variables, the probability of a *single* event A is in fact 0. Instead, we can only evaluate the probabilities within intervals. A common function representing the probability of some random variable is the *cumulative distribution function*:

$$F_X(x) = p(-\infty, x]$$

This function represents the sum (integral) of all the probabilities of random variable X for all the events up to and including x. It has the following properties:

$$F_X(x) \to 0 \text{ as } x \to -\infty$$

$$F_X(x) \to 1 \text{ as } x \to +\infty$$

$$F_X(x) \text{ is non-decreasing}$$

The *probability density function* is the derivative of the cumulative distribution function:

$$f_X(x) = \frac{d}{dx} F_X(x)$$

Following the above properties of the cumulative probability function, the probability density function has the following properties:

$$f_X(x) \text{ is non-negative}$$

$$\int_{-\infty}^{+\infty} f_X(x)dx = 1$$

Finally, the probability over an interval $[a, b]$ is given by:

$$p_X[a, b] = \int_a^b f_X(x)dx = F_X(b) - F_X(a)$$

A.1.3 Mean, Variance & Covariance

The expected value of a discrete random variable X is given by:

$$E(X) = \sum_{i=1}^{n} p_i x_i$$

for n possible outcomes $x_1, x_2, \ldots, x_n$ and their corresponding probabilities $p_1, p_2, \ldots, p_n$. Similarly, for a continuous random variable, the expected value is defined as:

$$E(X) = \int_{-\infty}^{+\infty} x f_X(x) dx$$

The expected value of a random variable is also known as the *first statistical moment*. We can compute the expected value of variable X^k to obtain the kth statistical moments, by:

$$E(X^k) = \int_{-\infty}^{+\infty} x^k f_X(x) dx$$

A particular interest is put on $k = 2$, for the *second moment* given by:

$$E(X^2) = \int_{-\infty}^{+\infty} x^2 f_X(x) dx$$

The variance is defined as:

$$Var(X) = E(X - E(X^2)) = E(X^2) - E(X)^2$$

It is an important statistical property for random signals because it gives a sense how much jitter or noise is in the signal. The square root of the variance is the *standard deviation*, which is also useful because while it is always positive, it has the same units as the original signal.

The covariance between two jointly random variables with *finite moments* is defined as:

$$\sigma(X, Y) = E[(X - E[X])(Y - E[Y])]$$

By using the linearity property of the expectation, the covariance can be simplified to:

$$\sigma(X, Y) = E[XY] - E[X]E[Y]$$

If X is a random vector with covariance matrix $\Sigma(X)$, and A a matrix that can act on X, the covariance matrix of the vector AX is:

$$\Sigma(AX) = A\Sigma(X)A^T$$

A.1.4 Gaussian Distribution

Many random processes occurring in nature actually appear to be normally distributed, or very close. Under some moderate conditions, it can be proved that a sum of random variables with *any* distribution tends toward a normal distribution. This is stated in the theorem called the *central limit theorem*.

Given a random process $X \sim N(\mu, \sigma^2)$, i.e. a continuous random variable X that is normally distributed with mean μ and variance σ^2, the probability density function of X is given by:

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Any linear function of a normally distributed random process is also normally distributed. If $Y = aX + b$, then $Y \sim N(a\mu + b, a^2\sigma^2)$. The probability density function for Y is then given by:

$$f_Y(y) = \frac{1}{\sqrt{2a\pi\sigma^2}} e^{-\frac{(y-(a\mu+b))^2}{2a^2\sigma^2}}$$

If X_1 and X_2 are independent, and $X_1 \sim N(\mu_1, \sigma_1^2)$ and $X_2 \sim N(\mu_2, \sigma_2^2)$, then:

$$X_1 + X_2 \sim N(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$$

The corresponding probability density function becomes:

$$f_X(x_1 + x_2) = \frac{1}{\sqrt{2a\pi(\sigma_1^2 + \sigma_2^2)}} e^{-\frac{(x-(\mu_1+\mu_2))^2}{2(\sigma_1^2+\sigma_2^2)}}$$

The *Bayes rule* follows from the conditional probability definition. It offers a way to specify the probability density of a random variable X given a variable Y. It is given as:

$$f_{X|Y}(x) = \frac{f_{Y|X}(y) f_X(x)}{f_Y(y)}$$

Finally, the discrete probability mass function of a discrete process X given a continuous process Y conditioned on $Y = y$, is given by:

$$p_X(x|Y = y) = \frac{f_Y(y|X = x) p_X(x)}{\sum_z f_Y(y|X = z) p_X(z)}$$

A.2 Discrete Kalman Filter

The KF is used for stochastic estimation from noisy measurements. It is composed of a set of mathematical equations that implement predictor-corrector type estimator that minimizes the estimated error covariance.

The KF addresses the problem of trying to estimate the state $x \in \Re^n$ of a discrete-time controlled process that is governed by the linear stochastic difference equation:

$$x_k = Ax_{k-1} + Bu_k + w_{k-1}$$

With a measurement $z \in \Re^n$ that is:

$$z_k = Hx_k + v_k$$

The random variables w_k and v_k represent the process and the measurement noises, respectively. They are assumed to be independent, white and with normal probability distribution:

$$w \sim N(0, Q)$$

$$v \sim N(0, R)$$

In practice, the *process noise covariance* Q and the *measurement noise covariance* R matrices might change each time step or measurement, however we assume that they are constant.

The matrix A of size $n \times n$ relates the state at the previous time step $k-1$ to the state at the current time step k, in the absence of a driving function or process noise. Matrix A may also change in the time, but we suppose it is constant. As for matrix B of size $n \times l$, it relates the optional control input $u \in \Re^l$ to the process state x. Matrix H of size $m \times n$ relates the state to the measurement z_k. It may change in each time step or measurement, but for simplicity reasons, we assume it is constant.

A.2.1 Computational Origins of KF

We define $\hat{x}_k^- \in \Re^n$ (super minus) to be the *a priori* state estimate at step k given knowledge of the process at step k, and $\hat{x}_k \in \Re^n$ to be the *a posteriori* state estimate at step k given measurement z_k. We can then define the a priori and the a posteriori estimate errors as:

$$e_k^- \equiv x_k - \hat{x}_k^-$$

$$e_k \equiv x_k - \hat{x}_k$$

The a priori estimate error covariance is then:

$$P_k^- = E[e_k^- e_k^{-T}]$$

and the a posteriori estimate error covariance is:

$$P_k = E[e_k e_k^T]$$

In deriving the equations of the KF, we begin with the goal of finding an equation that computes an a posteriori state estimate $\hat{x}^k$ as a linear combination of an a priori state estimate $\hat{x}_k^-$, and a weighted difference between an actual measurement z_k and a measurement prediction $H\hat{x}_k^-$ as shown in the following equation:

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H\hat{x}_k^-)$$

The difference in the above equation is called the *measurement innovation*, or the *residual*. It reflects the discrepancy between the predicted measurement $H\hat{x}_k^-$ and the actual measurement z_k. A residual of zero means that the two are in complete agreement.

The justification of the above equation is rooted in the probability of the a priori estimate $\hat{x}_k^-$ conditioned on all prior measurements z_k (Bayes' rule). Hence, it is clear that the Kalman Filter maintains the first two moments:

$$E[x_k] = \hat{x}_k$$

$$E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T] = P_k$$

The a priori state estimate reflects the first moment (the mean) and the a posteriori state estimate reflects the second moment (the variance).

The matrix K in the above equation, of size $n \times m$ is chosen to be the *gain* or the *blending factor* that minimizes the a posteriori error covariance. This minimization can be achieved by substituting this last equation in the definitions of e_k and P_k, performing the indicated expectation, taking the derivative of the trace of the result with respect to K, and setting that result equals to zero. Solving the resulting equation for K gives:

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} = \frac{P_k^- H^T}{H P_k^- H^T + R}$$

From this last equation, when the error covariance matrix R approaches zero, the gain K weights the residual more heavily:

$$\lim_{R_k \to 0} K_k = H^{-1}$$

On the other hand, as the a priori estimate error covariance P_k^- approaches zero, the gain K weights the residual less heavily:

$$\lim_{P_k^- \to 0} K_k = 0$$

Another way of thinking about the weighting by K is that as the measurement error covariance R approaches zero, the actual measurement z_k is *trusted* more and more, while the predicted measurement $H\hat{x}_k^-$ is trusted less and less. On the other hand, as the a priori estimate error covariance P_k^- approaches zero, the actual measurement z_k is trusted less and less, while the predicted measurement $H\hat{x}_k^-$ is trusted more and more.

A.2.2 High-Level Operation of KF

The KF estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of noisy measurements. As such, the equations for the Kalman Filter fall into two groups: *time update* equations and *measurement update* equations.

The time update equations are responsible for projecting forward (in time) the current state and the error covariance estimates to obtain a priori estimates for the next time step. The measurement update equations are responsible for the feedback, i.e. for incorporating a new measurement into the a priori estimate to obtain an improved a posteriori estimate.

The time update equations can also be thought of as *predictor* equations, while the measurement update equations can be thought of as *corrector* equations. In final, the estimation algorithm resembles that of a *predictor-corrector* algorithm for solving numerical problems.

The specific equations for the time update are presented below:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$$

$$P_k^- = AP_{k-1}A^T + Q$$

The measurement update are as follows:

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

$$P_k = (I - K_k H)P_k^-$$

The first task during the measurement update is to compute the Kalman gain, K_k. The next step is to measure the process to obtain z_k, and then to generate an a posteriori state estimate by incorporating the measurement. The final step is to obtain an a posteriori error covariance estimate.

After each time and measurement update pair, the process is repeated with the previous a posteriori estimates used to project or predict the new a priori estimates. This recursive nature is one of the very appealing features of the Kalman Filter. It conditions the current estimate on all the past measurements.

Another way to present the KF equations is as follows.
The state evolution is given by:

$$X_{t+1} = A_t X_t + b_t + w_t$$

Where X is the state vector we intend to estimate, A_t is the square transition matrix of the process. The control b_t is given, and there is a zero mean process noise w_t with known covariance r_t^w. The noise w_t is independent of X_t.

The measured vector y_t is given by the measurement equation:

$$y_t = H_t X_t + v_t$$

H_t is a rectangular measurement matrix, v_t is a zero-mean measurement noise of known covariance matrix r_t^v. The noise v_t is independent from X_t. The dimension of w_t is the dimension of x_t and the dimension of v_t is the dimension of y_t.

The covariance of the state vector X_t is:

$$P_t = E[(X_t - E[X_t])(X_t^T - E[X_t^T])]$$

Where X^T is the transpose of X. The goal of the Kalman Filter is to deduce from y_t the vector X_t whose covariance matrix has the lowest norm. The steps of the estimation are:

- Prediction of the state X_t:

$$X_{t+1|t} = A_t X_t + b_t$$

- Intermediate update of the state covariance matrix that takes into account the evolution given by the process transition:

$$P_{t+1|t} = A_t P_t A_t^T + r_t^w$$

- Computation of the optimal gain:

$$K_{t+1} = P_{t+1|t} H_{t+1}^T (H_{t+1} P_{t+1|t} H_{t+1}^T + r_{t+1}^v)^{-1}$$

  This optimal gain depends on the statistical characteristics of the measurement noise, but it does not take the measures into account: it may be computed a priori.

- Update of the state covariance matrix:

$$P_{t+1} = P_{t+1|t} - P_{t+1|t} H_{t+1}^T (H_{t+1} P_{t+1|t} H_{t+1}^T + r_{t+1}^v)^{-1} H_{t+1} P_{t+1|t}$$

  Or expressed as a function of K_{t+1}:

$$P_{t+1} = [I - K_{t+1} H_{t+1}] P_{t+1|t}$$

- Computation of the new estimate of the state:

$$X_{t+1} = X_{t+1|t} + K_{t+1}[y_{t+1} - H_{t+1} X_{t+1|t}]$$

A.3 Extended Kalman Filter

The Kalman Filter addresses the general problem of trying to estimate the state $x \in \Re^n$ of a discrete-time controlled process that is governed by a *linear* stochastic difference equation. If the process to be estimated or the measurement relationship to the process are not linear, the KF can be adapted to this situation by linearizing the current mean and covariance. That is referred to the *Extended Kalman Filter* (EKF).

To linearize the estimation around the current estimate, partial derivative of the process and the measurement functions are used. Let us consider that the process state $x \in \Re^n$ is governed by a non-linear stochastic difference equation:

$$x_k = f(x_{k-1}, u_{k-1}, w_{k-1})$$

with measurement $z \in \Re^m$ that is:

$$z_k = h(x_k, v_k)$$

with random variables w_k and v_k that represent the process and the measurement noises. f and h are a non-linear functions. The state and the measurement vectors can be approximated without the noises w_k and v_k as follows:

$$\tilde{x}_k = f(\hat{x}_{k-1}, u_{k-1}, 0)$$

$$\tilde{z}_k = h(\tilde{x}_k, 0)$$

$\hat{x}_k$ is the a posteriori state estimate.
It is important to note that a fundamental flaw of EKF is that the distributions of the different random variables are no longer normal after undergoing their respective non-linear transformations. EKF is simply an *ad hoc* state estimator that only approximates the optimality of Bayes' rule by linearization.

A.3.1 Computational Origins of EKF

The governing equations that linearizes the estimates are:

$$x_k \approx \tilde{x}_k + A(x_{k-1} - \hat{x}_{k-1}) + Ww_{k-1}$$

$$z_k \approx \tilde{z}_k + H(x_k - \tilde{x}_k) + Vv_k$$

With:

- x_k and z_k are the actual state and measurement vectors.
- $\tilde{x}_k$ and $\tilde{z}_k$ are the approximate state and measurement vectors.

- $\hat{x}_k$ is an a posteriori estimate at step k.
- w_k and v_k represent the process and the measurement noises.
- A is the Jacobian matrix of partial derivatives of f with respect to x, that is:

$$A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\hat{x}_{k-1}, u_{k-1}, 0)$$

- W is the Jacobian matrix of partial derivatives of f with respect to w, that is:

$$W_{[i,j]} = \frac{\partial f_{[i]}}{\partial w_{[j]}}(\hat{x}_{k-1}, u_{k-1}, 0)$$

- H is the Jacobian matrix of partial derivatives of h with respect to x, that is:

$$H_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}}(\tilde{x}_k, 0)$$

- V is the Jacobian matrix of partial derivatives of h with respect to v, that is:

$$V_{[i,j]} = \frac{\partial h_{[i]}}{\partial v_{[j]}}(\tilde{x}_k, 0)$$

The matrices A, W, H and V can be different at each time step. However, they are represented without index k for simplicity reasons.

The prediction error is defined as:

$$\tilde{e}_{x_k} = x_k - \tilde{x}_k$$

and the measurement residual is:

$$\tilde{e}_{z_k} = z_k - \tilde{z}_k$$

In practice, we have access to z_k which is the actual measurement, but we do not have access to x_k which is the actual state vector to be estimated. Thus, The *error process* can be expressed using the following approximations:

$$\tilde{e}_{x_k} \approx A(x_{k-1} - \hat{x}_{k-1}) + \epsilon_k$$

$$\tilde{e_{z_k}} \approx H\tilde{e_{x_k}} + \eta_k$$

with ϵ_k and η_k are new independent random variables having zero mean and covariance matrices WQW^T and VRV^T, respectively (R and Q are the same matrices defined in Section A.2).

These two last equations are linear and closely resemble to the difference and measurement equations of the discrete Kalman Filer. The actual measurement residual $\tilde{e}_{z_k}$ can be used to estimate the prediction error $\tilde{e}_{x_k}$. This estimate is called $\hat{e}_k$, and could then be used to obtain the a posteriori state estimate for the original non-linear process as:

$$\hat{x}_k = \tilde{x}_k + \hat{e}_k$$

The random variables $\tilde{e}_{x_k}, \epsilon_k$ and η_k have the approximative probability distributions:

$$\tilde{e}_{x_k} \sim N(0, E[\tilde{e}_{x_k}\tilde{e}_{x_k}^T])$$
$$\epsilon_k \sim N(0WQ_kW^T)$$
$$\eta_k \sim N(0, VR_kV^T)$$

Given these approximations and letting the predicted value of $\hat{e}_k$ be zero, the KF equation used to estimate $\hat{e}_k$ is:

$$\hat{e}_k = K_k\tilde{e}_{z_k}$$

By substituting this equation back into the a posteriori state estimate, we get:

$$\hat{x}_k = \tilde{x}_k + K_k\tilde{e}_{z_k} = \tilde{x}_k + K_k(z_k - \tilde{z}_k)$$

This equation can now be used for the measurement update in EKF.

The EKF time update equations are shown below:

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1}, 0)$$
$$P_k^- = A_kP_{k-1}A_k^T + W_kQ_{k-1}W_k^T$$

Similarly, the EKF measurement update equations are:

$$K_k = P_k^-H_k^T(H_kP_k^-H_k^T + V_kR_kV_k^T)^{-1}$$
$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-, 0))$$
$$P_k = (I - K_kH_k)P_k^-$$

As shown by these equations, the basic high-level operation of the EKF is the same as the KF. An important feature of the EKF is that the Jacobian H_k in the equation for the Kalman gain K_k serves to correctly propagate or magnify only the relevant component of the measurement information. For example, if there is not a one-to-one mapping between the measurement z_k and the state via h, the Jacobian H_k affects the Kalman gain so that it only magnifies the portion of the residual $z_k - h(\hat{x}_k, 0)$ that does affect the state. If over *all* the measurements, there is *not* a one-to-one mapping between the measurement z_k and the state via h, then as it might be expected, the filter will quickly diverge. In this case, the process is *unobservable*.

Appendix B

Gnu Linear Programming Kit

The Gnu Linear Programming Kit (GLPK) is an efficient tool for solving linear optimization problems with continuous and mixed variables (discrete and continuous). This kit is composed of a modeling language: the **Gnu MathProg Language** and a library of C functions used by the **glpsol** solver. The advantage of this toolkit is being *open source* and free for access, and its installation and usage are easy.

GLPK is a library of C functions that use the usual methods for solving the optimization problems such as: simplex, branch-and-bound, interior points methods, etc. In addition, GLPK provides a set of tools for objects creation, problem solving, output display, etc.

GLPK is not a program as it does not have a *main()* function. It is the user to write the program that calls the library functions. However, GLPK has a default solver, glpsol, that solves the generated program.

B.1 Gnu MathProg Language

To use the functions of GLPK, it is necessary to transform the problem in question into a formalism defined by the solver using the Gnu MathProg Modeling Language (GMLP). Through GMLP, the user writes the data file, defines the problem by a near-mathematical expressions, and the modeling language handles the model transformation and establishes the link to the solver.

GMLP is a subset of the AMPL language which is a powerful modeling language that can be coupled with many different solvers such as: CPLEX, EXPRESS, MOSEK, etc. When it is coupled with the suitable solver, AMPL can easily be used to solve complex problems such as: non-linear optimization problems, convex optimization, etc.

The goal from the use of GLPK is to model linear optimization problems, pure or mixed, of the form:

$$
\begin{aligned}
\min_x(\max_x) \quad & c_1x_1 + c_2x_2 + \cdots + c_nx_n \\
s.t. \quad & \\
& l_1 \leq a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq u_1 \\
& l_2 \leq a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq u_2 \\
& \cdots \\
& l_m \leq a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq u_m
\end{aligned}
$$

The bounds l_i (resp. u_i) may have infinite value $+\infty$ (resp. $-\infty$). In the case of pure linear problems, the variables x_i are reals. In the case of mixed linear problems, some variables x_i may be integers. The quantities c_i, a_{ij}, l_i and u_i are the input data given by the user.

A model is constructed based on elementary objects which are: *parameters, sets, variables, constraints and objective*. The objects *parameter* and *set* define the data of the problem. The object *variable* defines the variables of the problem. The object *constraint* defines the constraints of the problem, and the object *objective* defines the objective function of the problem. Each elementary object is composed of expressions defined in Section B.1.3.

B.1.1 Coding Rules

To model a problem in GLPK, a user should define two parts:

- The **model** section that contains all the declarations, the computed parameters, the definitions of the constraints and the objective.
- The **data** section that contains all the constant data such as: the values of the parameters, the content of the sets, etc.

The two sections may be introduced in the same file as illustrated in Listing B.1 or in two different files as illustrated in Listing B.2 and Listing B.3. In one-file declaration, it is necessary to separate the data section by **data;** and **end;**. The file may be saved with extension `.mod`.

In the case of two separate files declaration, the model file *must* be saved under the extension `.mod` and the data file *must* be saved under the extension **.dat**. Any quantity used in the *dat* file should be declared in the *mod* file. In practice, it is strongly recommended to separate the model and the data sections in order to have a *generic* model that accept a multiple sets of data.

```
statement
...
data;
data_block
...
data_block
end;
```

Listing B.1: One-file declaration.

```
statement
statement
statement
statement
...
statement
end;
```

Listing B.2: Model file.

```
data;
data_block
data_block
data_block
...
data_block
end;
```

Listing B.3: Data file.

Symbols & Keywords

To describe a model, we use ASCII coded characters:

- Alphabetical characters: A, B, C, . . . , Z, a, b, c, . . . , z (case sensitive);
- Numerical characters: 0, 1, 2, . . . , 9;
- Special characters: " # & ' ( ) * + , - . / : ; <> = [ ] {} |

The spaces are effective only in the comments and the strings.

The symbolic names are composed of alphabetical and numerical characters; the first must be alphabetical. All the symbolic names must be unique. They are used by the user to identify the different objects: parameters, sets, variables, constraints and objective.

The numbers are wrote in the format $xx(Esyy)$, where xx is a real number with an eventual decimal part separated by a point, s is the sign - or + and yy is the exponent. The letter E can also be replaced by e. The strings are delimited by " or '.

The keywords are a sequence of alphabetical and numerical characters that are recognizable by GLPK. We distinguish two classes: the reserved words having fixed meaning, which can not be used to define symbolic names, and the non-reserved words which are recognizable according to the context.

The reserved words are presented in Table B.1.

Table B.1: Reserved keywords of GMLP.

and	**diff**	**if**	**less**	**or**	**union**
by	**div**	**in**	**mod**	**symdiff**	**within**
cross	**else**	**inter**	**not**	**then**	

The delimiters are simple or double special characters, which are presented in Table B.2.

It is also possible to include some comments on only one line using special character #, or on multiple lines using spacial characters /* and */.

Table B.2: Delimiters in GMLP.

+	^	==	!	:	)
-	&	>=	&&	;	[
*	<	>	||	:=	]
/	<=	<>	.	..	{
**	=	!=	,	(	}

B.1.2 Compilation

If we suppose that all the declarations are put in one file **model.mod**, the call to the solver is done as follows:

```
$glpsol --model model.mod
```

In the general case, where the model and the data are in separate files **model.mod** and **data.dat**, the call is done as follows:

```
$glpsol --model model.mod --data data.dat
```

Many options can be added to this command line. They are presented in Table B.3.

Table B.3: Main options for compilation using glpsol.

General form	Function
--display *filename*	Writes all the display of the model in *filename*
--output *filename*	Writes the solution of the problem in *filename*
--tmlim *nnn*	Limits the time of execution to *nnn* seconds
--memlim *nnn*	Limits the size of memory to *nnn* MO
--check	Checks only the correctness of the model
--simplex	Uses the simplex method (default)
--interior	Uses the Interior Points method
--glp (resp. **--wglp** *filename*)	Read (resp. Write) the problem in Gnu LP format
--mps (resp. **--wmps** *filename*)	Read (resp. Write) the problem in MPS format
--freemps (resp. **--wfreemps** *filename*)	Read (resp. Write) the problem in free MPS format
--cpxlp (resp. **--wcpxlp** *filename*)	Read (resp. Write) the problem in CPLEX format
--math	Read the problem in Gnu MathProg format
--wtxt *filename*	Writes the problem in plain text file *filename*

B.1.3 Expressions

The expressions serve to obtain the constructions used to define the objects of the model. The general form of an expression is:

primary_expr operator primary_expr operator...operator primary_expr

There are two types of expressions: numerical expressions and symbolic expressions.

Numerical Expressions

A numerical expression is used to define or compute a numerical value. It is generally composed of at least two primary expressions and an arithmetic operator. For example, **2*5.5** represents a number or **param s:=5;** that represents a constant. We can also represent multi-dimensional parameters by:

$$a[i_1, i_2, \ldots, i_n]$$

Where indices $i_1, i_2 \ldots, i_n$ may be numerical or symbolic expressions. For example, we can declare:

param A := 1 2 3 4 5 6 7;

The element **A[3]*3** exists and its value is 12.

GMLP provides a set of pre-defined functions that can be applied to the numerical expressions. Table B.4 gives a brief description of these functions. The user can also define its own functions, such as:

var fraction {x in X, y in Y} := x/y;

Iteration Expressions

There are four operators of iteration that can be used in numerical expressions. They are summarized in Table B.5. For example, when we write:

sum{i in 1..2, j in 1..2} x[i,j];

It returns: **x[1,1]+x[1,2]+x[2,1]+x[2,2]**.

Conditional Expressions

There two syntax forms of the conditional expressions:

if *condition* **then** *solution*1 **else** *solution*2. Or,
if *condition* **then** *solution*1

Where *condition* is a logical expression, and *solution*1 and *solution*2 are numerical expressions.

The arithmetic operators presented in Table B.6 can be used with the numerical expressions.

x and y are primary numerical expressions. If the numerical expression include more than one operator, then all the operators are computed from left to right (except the power operator which is calculated form right to left) according to the hierarchy of operators presented in Table B.7. It is worth to note here that any operation put between parenthesis is executed at first.

Table B.4: List of the available functions in GLPK.

Function	Role
abs(x)	Absolute value of x
atan(x)	Arc tang. of x (radians)
atan(y,x)	Arc tang. of y/x (radians)
ceil(x)	Smallest integer close to x
cos(x)	$\cos(x)$
floor(x)	Greatest integer close to x
exp(x)	$\exp(x)$
log(x)	$\log(x)$
log10(x)	$\log_{10}(x)$
max($x_1, x_2, \ldots, x_n$**)**	Maximum value among $x_1, x_2, \ldots, x_n$
min($x_1, x_2, \ldots, x_n$**)**	Minimum value among $x_1, x_2, \ldots, x_n$
round(x)	Round x to the nearest value
round(x,n)	Round x with n decimals
sin(x)	$\sin(x)$
sqrt(x)	$\sqrt{x}$
trunc(x)	Truncate x to the nearest value
trunc(x,n)	Truncate x with n decimals
Irand224(x)	Return a random number in interval $[0, 2^{24})$
Uniform01(x)	Return a random number in interval $[0, 1)$
Uniform(a,b)	Return a random number in interval $[a, b)$
Normal01(x)	Normal function with mean 0 and deviation 1
Normal(m,s)	Normal function with mean m and deviation s

Table B.5: List of the iteration operators provided by GLPK.

Operator	Role
sum$\{set\}$	Sum: $\sum_{(i_1,i_2,\ldots,i_n)\in\Delta} x(i_1, i_2, \ldots, i_n)$
prod$\{set\}$	Product: $\prod_{(i_1,i_2,\ldots,i_n)\in\Delta} x(i_1, i_2, \ldots, i_n)$
min$\{set\}$	Minimum: $\min_{(i_1,i_2,\ldots,i_n)\in\Delta} x(i_1, i_2, \ldots, i_n)$
max$\{set\}$	Maximum: $\max_{(i_1,i_2,\ldots,i_n)\in\Delta} x(i_1, i_2, \ldots, i_n)$

Symbolic Expressions

The symbolic expressions serve to execute operations on the strings. The primary symbolic expression is a string, an index, or a conditional symbolic expression. It is accepted in GLPK to use a numerical expression as a symbolic expression but in this case, the resulting value is converted to symbolic. In GLPK, there is only one symbolic operator which is *concatenation* (&): x & y It concatenates two symbolic expressions into one string.

Table B.6: List of the arithmetic operators.

Operator	Role
`+x`	Plus 1: $x = x + 1$
`-x`	Minus 1: $x = x - 1$
`x+y`	Addition
`x-y`	Subtraction
`x less y`	Positive difference: if $x < y$ then 0, else $x - y$
`x*y`	Multiplication
`x/y`	Division
`x div y`	Quotient of the exact division
`x mod y`	Rest of the exact division
`x **y,` x^y	Power

Table B.7: Hierarchy of the arithmetic operators.

Operator	Hierarchy
Evaluation of functions (`abs, log,...`)	1st
Power (**)	2nd
Plus 1, Minus 1 (+,-)	3rd
Multiplication and division (`*,/,div,mod`)	4th
Iteration operator (`sum,prod,min,max`)	5th
Addition (`+,-,diff`)	6th
Conditional (`if ... then ... else ...`)	7th

Expressions on Indices

An expression on the indices is a construction that allows defining the definition set of a variable or a constraint, or specify the set of the iteration operators. There are two forms of such expressions:

$$\{entry_1, entry_2, \ldots, entry_m\}. \text{ Or}$$
$$\{entry_1, entry_2, \ldots, entry_m : predicate\}.$$

Where $entry_i$ is an expression specifying the set to which belongs index i, and $predicate$ is a logical expression specifying some conditions about the indices. The index in not necessary when it is not used in the definition of the object, and the predicate must be at the end and may contain all the indices. For example:

`sum`{`i in 1..3, j in 1..3: i==j`} `x[i,j];`
It returns `x[1,1]+x[2,2]+x[3,3]`

Some rules:

- An $entry_i$ may have 3 forms: `t in S`, or `(`$t_1, t_2, \ldots, t_n$`) in S`, or `S`, where $t_1, t_2, \ldots, t_n$ are indices and S is a set.

- When we write **`object(t in S)`**, the variable i is local and does not exist outside. It is auto-declared and does not require general declaration.
- If we have three sets: **`A =`** {**`4,7`**}, **`B =`** {**`(1, Jan), (1, Feb), (1, Mar)`**}, **`C =`** {**`a, b, c`**}. The following declaration: {**`i in A, (j,k) in B, l in C`**} is equivalent to the following algorithmic declaration:

$$\forall a \in A, \forall (j,k) \in B, \forall l \in C : action(a, j, k, l);$$

Set Expressions

A set expression is a rule for computing a set, i.e. a collection of n-tuples whose components are numerical and/or symbolic quantities. The different types of the set expressions are:

- **Literal formulation:** it has two forms: $(e_1, e_2, \ldots, e_m)$ or $\{(e_{11}, \ldots, e_{1n}), (e_{21,\ldots,e_{2n}}), \ldots, (e_{m1}, \ldots, e_{mn})\}$
- **Indexed formulation:** the primary expression may be a simple set or a multi-dimensional set (ex. **`S[i-1,j+1]`**).
- **Arithmetic formulation:** the primary expression can be written as: $t_0..t_f$ **`by`** δ_t or $t_0..t_f$. t_0, t_f and δ_t are numerical expressions. The resulting set is: $\{t_0 + \delta_t, t_0 + 2\delta_t, \ldots, t_f\}$.
- **Iterated formulation:** it has the following form:

  `setof` *indexing_expression* *integrand*

  Where *indexing_expression* is a set of indices specifying the number of repetition to run, and *integrand* is a number, a symbolic expression, or a list of numbers or symbolic expressions separated with commas and closed with parenthesis. For example, we can write:

  `setof`{**`i in 1..2, j in 1..3`**}**`(i,j+1)`**, which creates the set: {**`(1,2), (1,3), (1,4), (2,2), (2,3), (2,4)`**}

- **Conditional formulation:** the primary expression may be a conditional expression that attributes a set or another, according to a logical expression. The syntax is:

  `if b then X else Y`

  With X and Y are the sets and b is a logical expression. In addition, GLPK provides five set operator, presented in Table B.8.

Table B.8: Set operators.

Formulation	Function
`X union Y`	Union of sets X and Y: $X \cup Y$
`X diff Y`	Difference between sets X and Y: $X \setminus Y$
`X symdiff Y`	Symmetric difference between X and Y: $X \bigoplus Y$
`X inter Y`	Intersection of X and Y: $X \cap Y$
`X cross Y`	Cartesian product of X and Y: $X \times Y$

Logical Expressions

A logical expression is a rule that verifies if a test is true or false. The different types of primary logical expressions are:

- **Numerical:** if the primary logical expression is numerical then the result is true if the value is non-null. False, otherwise.

- **Relational:** the different relational operators used in the logical expressions are presented in Table B.9. $x_1, x_2, \ldots, x_n$ and y are numeric or symbolic,

Table B.9: Logic operators.

Formulation	Corresponding Test
$x < y$	Test if $x < y$
$x <= y$	Test if $x <= y$
$x = y$, $x == y$	Test if $x = y$
$x >= y$	Test if $x >= y$
$x <> y$, $x! = y$	Test if $x \neq y$
`x in Y`	Test if $x \in Y$
$(x_1, x_2, \ldots, x_n)$ `in Y`	Test if $(x_1, x_2, \ldots, x_n) \in Y$
`x not in Y, x !in Y`	Test if $x \notin Y$
$(x_1, x_2, \ldots, x_n)$ `not in Y,` $(x_1, x_2, \ldots, x_n)$ `!in Y`	Test if $(x_1, x_2, \ldots, x_n) \notin Y$
`X not within Y`	Test if $X \subseteq Y$
`X not within Y, X !within Y`	Test if $X \subsetneq Y$

and X and Y are the sets. If x and y are symbolic expressions then only operators `=, ==,` $<>$ and `!=` can be used.

- **Iteration:** an iterated expression is a primary expression having the following form:

  `iterated_operator indexing_expression integrand`

  with `iterated_operator` is an iteration operator, `indexing_expression` is a set of indices that indicate the number of iteration to be executed, and

integrand is a logical expression to be iterated. GLPK provides two operators of iteration:

1. **forall**: which is equivalent to the mathematical operator $\forall$. The resulting test is true if the test is true for all the elements generated by **indexing_expression**.
2. **exists**: which is equivalent to the mathematical operator $\exists$. The resulting test is true if the test is true at least for one element generated by **indexing_expression**.

Linear Expression

A linear expression is a rule to compute linear mathematical formulas. A linear expression is composed of:

- **Variables:** a primary expression is based on simple variables (ex. **T**) or indexed variables **x[i,j]**.
- **Iterations:** a linear expression can be computed using iteration operators that conserve the linearity. The only operator available is:

  sum *indexing_expression* **integrand**

  with *indexing_expression* is the set of indices to which the sum is applied, and **integrand** is the linear expression to be iterated.
- **Conditional:** we can compute a linear expression using conditional expression, of the form:

  if b then f else g, Or
  if b then f

  with b, f and g are linear expressions.

B.1.4 Modeling Objects

The modeling objects are the keystones of constructing a model. In GLPK, there are five modeling objects, which are: the sets, the parameters, the variables, the constraints and the objective. The different expressions defined in Section B.1.3 can be used to parametrize these objects.

Sets

`set` in the model section declares the sets yet defined in the data section, or defines them using the expressions. The general form of a set declaration is:

`set` *name* {**`domain`**}**`, attrib, ..., attrib`**

With:

- *name* is the symbolic name of the set.
- *domain* is optional, and it defines the dimension and/or the definition set.
- *attrib*, ..., *attrib* is a list of optional attributes that define different specifications:
  - **`dimen`** *n*, specifies the dimension of the n-tuples of the set.
  - **`within`** *expression*, forces the elements of the set to be within a larger set.
  - **`:=`** *expression*, assigns a fixed or a computed value to the set.
  - **`default expression`**, specifies a default value to the set or to one of its elements when no values are provided.

Examples: hereafter the declaration of some sets:

```
set nodes;
set arc (nodes cross nodes);
```

Parameters

`parameter` in the model section declares the parameters defined in the data section, or defines computed parameters using analytical or logical expressions (in which the variables can not be used). The general form of a parameter declaration is:

`param` *name* {**`domain`**}**`, attrib, ..., attrib`**

With:

- *name* is the symbolic name of the parameter.
- *domain* is optional, and it defines the dimension and/or the definition set of the parameter.
- *attrib*, ..., *attrib* is list of attributes that define different specifications:

- **integer**, specifies that the parameter is integer.
- **binary**, specifies that the parameter is binary.
- **symbolic**, specifies that the parameter is symbolic.
- *relation* with $<$ $<=$ $=$ $==$ $>=$ $>$ $<>$ $!=$, forces the parameter to verify the formulated conditions.
- **in** *expression*, forces the parameter to be within a certain set.
- **:= expression**, assigns a fixed or a computed value to the parameter.
- **default expression**, specifies a default value to the parameter when no values are provided.

Examples: hereafter the declaration of some parameters:

```
param I := 3;
param units(I);
param N := 60, integer, >= 0, <= 1000;
param A {n in 0..N, k in 0..N} := if k=0 or k=n then 1
else A[n-1,k-1]+A[n-1,k];
```

Variables

variable defines the variables of the optimization problem and its conditions. The general form of a variable declaration is:

var *name* {**domain**}, **attrib**, ..., **attrib**

With:

- *name* is the symbolic name of the variable.
- *domain* is optional, and it defines the dimension and/or the definition set of the variable.
- $attrib, \ldots, attrib$ is a list of attributes that define different specifications:
  - **integer**, forces the variable to be integer.
  - **binary**, forces the variable to be binary.
  - **<= expression**, specifies the upper bound of the variable.
  - **>= expression**, specifies the lower bound of the variable.
  - **= (or ==) expression**, specifies a fixed value of the variable.

Examples: hereafter the declaration of some variables:

```
var x = 0;
var y{I,J};
var An in I, integer, >= b[n], <= c[n];
```

Constraints

constraint defines the constraints of the optimization problem. Its general form is as follows:

```
subject to name {domain} :  expression, = expression;
subject to name {domain} :  expression, <= expression;
subject to name {domain} :  expression, >= expression;
subject to name {domain} :  expression, <= expression, >= expression;
subject to name {domain} :  expression, >= expression, <= expression;
```

With:

- *name* is the symbolic name of the constraint.
- *domain* is optional, and it defines the number of the constraints of this type.
- *expressions* are the linear expressions to compute the components of the constraint.

The keyword **subject to** can be replaced by **s.t.**, or even can be omitted. If *domain* is not specified then the constraint is simple. Otherwise, the number of the constraints follows the number of the elements in the domain. The constraints should be linear.

Examples: hereafter the declaration of some constraints:

```
subject to C1 :  x + y + z >= 0;
s.t.  C2 t in 1..T, i in 1..I: x[t] + y[t] <= sqrt(2)*i;
```

Objective

objective defines the objective of the optimization problem. Its general form is:

```
minimize name {domain} :  expression;
maximize name {domain} :  expression;
```

If the *domain* is not specified then the objective function is a simple scalar. Otherwise, the number of the elements in *domain* determines the dimension of the objective. Only one objective can be defined in a problem. If many objectives are defined, GLPK considers only the first one; the others are simply ignored.

Examples: hereafter the declaration of some objective functions:

```
minimize obj:  x + 1.5 * (y + z);
maximize total_gain:  sumg in 1..G gain[g] * products[g];
```

B.2 GLPK Implementation for EMDP

Listing B.4 shows the implementation of our BILP model for the EMDP problem using GLPK, and Listing B.5 shows the output of the **glpsol** solver for a small instance of the problem with the parameters specified in Table B.10.

Table B.10: Parameters of the GLPK implementation.

Parameter	Value
Network Size (Nodes)	12
Max. Energy (J)	5
Radio Propagation Coefficient (α)	2
Trees Depth (n)	3
Number of Rounds (k)	1
Sensitivity Parameter (τ)	0.01
Reference Value (v_0)	0.7
Sampling Energy (e_s, J)	5×10^{-6}
Receiving Energy (e_r, J)	25^{-6}
Processing Energy (e_p, J)	10^{-9}

Listing B.4: BILP implementation using GLPK.

```
/*  Sets  */
set V; /* Nodes */
set K; /* Rounds */
set H; /* Hops */
set N{V,H}; /* Neighborhood */

/*  Parameters */
param m, integer, >= 1, <= 1000; /* Network Size */
param n, integer, >= 1, <= 10; /* Tree Depth */
param k, integer, >= 1, <= 100;  /* Rounds */
param tau, > 0; /* Data Sensitivity */
param alpha; /* Energy Param. */
param v0; /* Reference Value */
param es; /* Energy Consumption in Sensing */
param er; /* Energy Consumption in receiving */
param ep;  /* Energy Consumption in Data Processing */
param L{V,V}; /* Neighborhood */
param et{i in V, j in N[i,1]} := 25e-6+5e-9*L[i,j]^alpha; /* Energy
    Consumption in transmission */
param E{K,V}; /* Energy of Nodes */
param v{K,V};  /* Readings of Nodes */

/*  Variables   */
var a{K,V}, binary; /* a_i^t */

var b{K, i in V, j in V}, binary; /* b_{i,j}^t */

/*  Constraints   */
s.t. cluster_one{t in K}: sum{i in V}a[t,i]-sum{i in V, j in N[i,1]}b[t
    ,j,i] >= 1;

s.t. cluster_child{t in K, i in V}: sum{j in N[i,1]}b[t,i,j] <= a[t,i
    ]*(card(N[i,1])-1);

s.t. cluster_child_bis{t in K, i in V}: sum{j in N[i,1]}b[t,j,i] <= a[t
    ,i];

s.t. data_selection1{t in K}: sum{i in V}a[t,i]*v[t,i] >= m*(1-tau)*v0;

s.t. data_selection2{t in K}: sum{i in V}a[t,i]*v[t,i] <= m*(1+tau)*v0;

s.t. energy{t in K, i in V, j in N[i,n]}: (a[t,i]-sum{l in N[i,1]}b[t,l
    ,i])*E[t,i] >= (a[t,i]-sum{l in N[i,1]}b[t,l,i]-1+a[t,j])*E[t,j];

s.t. data_forwarding{t in K, i in V, n1 in 2..n, j in (N[i,n1] diff N[i
    ,n1-1])}: a[t,j]-sum{p in (N[j,1] inter N[i,n1-1])}b[t,p,j] <= 1-(a
    [t,i]-sum{q in N[i,1]}b[t,q,i]);

s.t. energy_update{t in K, i in V}: E[t+1,i] = E[t,i]-(a[t,i]*es+(er+ep
```

```
    )*sum{j in N[i,1]}b[t,i,j]+sum{j in N[i,1]}b[t,j,i]*et[i,j]);

/* Objective Function */
minimize energy_min: sum{t in K}((es*sum{i in V}a[t,i])+((er+ep)*(sum{i
     in V, j in N[i,1]}b[t,i,j]))+(sum{i in V, j in N[i,1]}(b[t,j,i]*et
    [i,j]))));
end;
```

Listing B.5: Output of glpsol solver.

```
Problem:    emdp
Rows:       226
Columns:    66 (66 integer, 66 binary)
Non-zeros:  1647
Status:     INTEGER OPTIMAL
Objective:  energy_min = 0.00061011355 (MINimum)

No.  Row name            Activity    Lower bound  Upper bound
---  ---------           --------    -----------  -----------
1    cluster_one[1]         1                         1
2    cluster_child[1,1]    -2                        -0
3    cluster_child[1,2]    -4                        -0
4    cluster_child[1,3]    -4                        -0
5    cluster_child[1,4]    -3                        -0
6    cluster_child[1,5]    -4                        -0
7    cluster_child[1,6]    -2                        -0
8    cluster_child[1,7]    -4                        -0
9    cluster_child[1,8]    -3                        -0
10   cluster_child[1,9]    -2                        -0
11   cluster_child[1,10]   -2                        -0
12   cluster_child[1,11]   -1                        -0
13   cluster_child[1,12]    0                        -0
14   cluster_child_bis[1,1]   -1                     -0
15   cluster_child_bis[1,2]    0                     -0
16   cluster_child_bis[1,3]    0                     -0
17   cluster_child_bis[1,4]    0                     -0
18   cluster_child_bis[1,5]    0                     -0
19   cluster_child_bis[1,6]    0                     -0
20   cluster_child_bis[1,7]    0                     -0
21   cluster_child_bis[1,8]    0                     -0
22   cluster_child_bis[1,9]    0                     -0
23   cluster_child_bis[1,10]   0                     -0
24   cluster_child_bis[1,11]   0                     -0
25   cluster_child_bis[1,12]   0                     -0
26   data_selection1[1]        8.57      8.4942
27   data_selection2[1]        8.57                    8.6658
28   energy[1,1,2]              2          -4
29   energy[1,1,3]              2          -4
30   energy[1,1,4]              4          -3
31   energy[1,1,5]              0          -5
32   energy[1,1,6]              0          -5
33   energy[1,1,7]             -2          -6
34   energy[1,1,8]              6          -2
35   energy[1,1,9]              4          -3
36   energy[1,1,10]            -6          -8
37   energy[1,1,11]             0          -5
38   energy[1,1,12]             6          -2
39   energy[1,2,1]            -10         -10
40   energy[1,2,3]             -4          -4
41   energy[1,2,4]             -3          -3
42   energy[1,2,5]             -5          -5
43   energy[1,2,6]             -5          -5
44   energy[1,2,7]             -6          -6
45   energy[1,2,8]             -2          -2
46   energy[1,2,9]             -3          -3
47   energy[1,2,10]            -8          -8
48   energy[1,2,11]            -5          -5
49   energy[1,2,12]            -2          -2
50   energy[1,3,1]            -10         -10
51   energy[1,3,2]             -4          -4
52   energy[1,3,4]             -3          -3
53   energy[1,3,5]             -5          -5
54   energy[1,3,6]             -5          -5
55   energy[1,3,7]             -6          -6
56   energy[1,3,8]             -2          -2
57   energy[1,3,9]             -3          -3
58   energy[1,3,10]            -8          -8
59   energy[1,3,11]            -5          -5
60   energy[1,3,12]            -2          -2
61   energy[1,4,1]            -10         -10
62   energy[1,4,2]             -4          -4
63   energy[1,4,3]             -4          -4
64   energy[1,4,5]             -5          -5
65   energy[1,4,6]             -5          -5
66   energy[1,4,7]             -6          -6
67   energy[1,4,8]             -2          -2
68   energy[1,4,9]             -3          -3
69   energy[1,4,10]            -8          -8
70   energy[1,4,11]            -5          -5
71   energy[1,4,12]            -2          -2
72   energy[1,5,1]            -10         -10
73   energy[1,5,2]             -4          -4
74   energy[1,5,3]             -4          -4
75   energy[1,5,4]             -3          -3
76   energy[1,5,6]             -5          -5
77   energy[1,5,7]             -6          -6
78   energy[1,5,8]             -2          -2
79   energy[1,5,9]             -3          -3
80   energy[1,5,10]            -8          -8
81   energy[1,5,11]            -5          -5
82   energy[1,5,12]            -2          -2
83   energy[1,6,1]            -10         -10
84   energy[1,6,2]             -4          -4
85   energy[1,6,3]             -4          -4
86   energy[1,6,4]             -3          -3
87   energy[1,6,5]             -5          -5
88   energy[1,6,7]             -6          -6
89   energy[1,6,8]             -2          -2
90   energy[1,6,9]             -3          -3
91   energy[1,6,10]            -8          -8
92   energy[1,6,11]            -5          -5
93   energy[1,6,12]            -2          -2
94   energy[1,7,1]            -10         -10
95   energy[1,7,2]             -4          -4
96   energy[1,7,3]             -4          -4
97   energy[1,7,4]             -3          -3
98   energy[1,7,5]             -5          -5
99   energy[1,7,6]             -5          -5
100  energy[1,7,8]             -2          -2
101  energy[1,7,9]             -3          -3
102  energy[1,7,10]            -8          -8
103  energy[1,7,11]            -5          -5
104  energy[1,7,12]            -2          -2
105  energy[1,8,1]            -10         -10
106  energy[1,8,2]             -4          -4
107  energy[1,8,3]             -4          -4
108  energy[1,8,4]             -3          -3
109  energy[1,8,5]             -5          -5
110  energy[1,8,6]             -5          -5
111  energy[1,8,7]             -6          -6
112  energy[1,8,9]             -3          -3
113  energy[1,8,10]            -8          -8
114  energy[1,8,11]            -5          -5
115  energy[1,8,12]            -2          -2
116  energy[1,9,1]            -10         -10
117  energy[1,9,2]             -4          -4
118  energy[1,9,3]             -4          -4
119  energy[1,9,4]             -3          -3
120  energy[1,9,5]             -5          -5
121  energy[1,9,6]             -5          -5
122  energy[1,9,7]             -6          -6
123  energy[1,9,8]             -2          -2
124  energy[1,9,10]            -8          -8
125  energy[1,9,12]            -2          -2
126  energy[1,10,1]           -10         -10
127  energy[1,10,2]            -4          -4
128  energy[1,10,3]            -4          -4
129  energy[1,10,4]            -3          -3
130  energy[1,10,5]            -5          -5
131  energy[1,10,6]            -5          -5
132  energy[1,10,7]            -6          -6
133  energy[1,10,8]            -2          -2
134  energy[1,10,9]            -3          -3
135  energy[1,11,1]           -10         -10
136  energy[1,11,2]            -4          -4
137  energy[1,11,3]            -4          -4
138  energy[1,11,4]            -3          -3
139  energy[1,11,5]            -5          -5
140  energy[1,11,6]            -5          -5
141  energy[1,11,7]            -6          -6
142  energy[1,11,8]            -2          -2
143  energy[1,11,12]           -2          -2
144  energy[1,12,1]           -10         -10
145  energy[1,12,2]            -4          -4
146  energy[1,12,3]            -4          -4
147  energy[1,12,4]            -3          -3
148  energy[1,12,5]            -5          -5
149  energy[1,12,6]            -5          -5
150  energy[1,12,7]            -6          -6
151  energy[1,12,8]            -2          -2
152  energy[1,12,9]            -3          -3
153  energy[1,12,11]           -5          -5
154  data_forwarding[1,1,2,9]   1                     1
155  data_forwarding[1,1,2,10]  1                     1
156  data_forwarding[1,1,2,11]  1                     1
157  data_forwarding[1,1,2,12]  1                     1
158  data_forwarding[1,2,2,4]   0                     1
159  data_forwarding[1,2,2,8]   0                     1
160  data_forwarding[1,2,2,10]  0                     1
161  data_forwarding[1,2,2,12]  1                     1
162  data_forwarding[1,2,3,11]  0                     1
```

163	data_forwarding[1,3,2,5]	1	1
164	data_forwarding[1,3,2,8]	0	1
165	data_forwarding[1,3,2,9]	0	1
166	data_forwarding[1,3,2,10]	0	1
167	data_forwarding[1,3,2,11]	0	1
168	data_forwarding[1,3,3,12]	0	1
169	data_forwarding[1,4,2,2]	0	1
170	data_forwarding[1,4,2,6]	1	1
171	data_forwarding[1,4,2,7]	1	1
172	data_forwarding[1,4,2,12]	0	1
173	data_forwarding[1,4,3,9]	0	1
174	data_forwarding[1,4,3,10]	0	1
175	data_forwarding[1,5,2,3]	0	1
176	data_forwarding[1,5,2,6]	0	1
177	data_forwarding[1,5,2,7]	1	1
178	data_forwarding[1,5,2,9]	1	1
179	data_forwarding[1,5,2,11]	0	1
180	data_forwarding[1,5,3,10]	0	1
181	data_forwarding[1,6,2,4]	0	1
182	data_forwarding[1,6,2,5]	1	1
183	data_forwarding[1,6,2,8]	0	1
184	data_forwarding[1,6,3,11]	0	1
185	data_forwarding[1,6,3,12]	0	1
186	data_forwarding[1,7,2,4]	0	1
187	data_forwarding[1,7,2,5]	1	1
188	data_forwarding[1,7,2,8]	0	1
189	data_forwarding[1,7,2,9]	0	1
190	data_forwarding[1,7,3,11]	0	1
191	data_forwarding[1,7,3,12]	0	1
192	data_forwarding[1,8,2,2]	0	1
193	data_forwarding[1,8,2,3]	0	1
194	data_forwarding[1,8,2,6]	1	1
195	data_forwarding[1,8,2,7]	1	1
196	data_forwarding[1,8,3,9]	0	1
197	data_forwarding[1,8,3,10]	0	1
198	data_forwarding[1,9,2,1]	1	1
199	data_forwarding[1,9,2,3]	1	1
200	data_forwarding[1,9,2,5]	1	1
201	data_forwarding[1,9,2,7]	0	1
202	data_forwarding[1,9,3,4]	0	1
203	data_forwarding[1,9,3,8]	0	1
204	data_forwarding[1,9,3,12]	1	1
205	data_forwarding[1,10,2,1]	1	1
206	data_forwarding[1,10,2,2]	1	1
207	data_forwarding[1,10,2,3]	1	1
208	data_forwarding[1,10,3,4]	0	1
209	data_forwarding[1,10,3,5]	1	1
210	data_forwarding[1,10,3,8]	0	1
211	data_forwarding[1,11,2,1]	1	1
212	data_forwarding[1,11,2,3]	1	1
213	data_forwarding[1,11,2,5]	1	1
214	data_forwarding[1,11,2,12]	0	1
215	data_forwarding[1,11,3,2]	0	1
216	data_forwarding[1,11,3,6]	1	1
217	data_forwarding[1,11,3,7]	1	1
218	data_forwarding[1,12,2,1]	1	1
219	data_forwarding[1,12,2,2]	1	1
220	data_forwarding[1,12,2,4]	1	1
221	data_forwarding[1,12,2,11]	1	1
222	data_forwarding[1,12,3,3]	0	1
223	data_forwarding[1,12,3,6]	0	1
224	data_forwarding[1,12,3,7]	1	1
225	data_forwarding[1,12,3,9]	1	1
226	energy_min	0.000610114	

No.	Column name	Activity	Lower bound	Upper bound
1	a[1,1]	* 1	0	1
2	a[1,2]	* 1	0	1
3	a[1,3]	* 1	0	1
4	a[1,4]	* 1	0	1
5	a[1,5]	* 1	0	1
6	a[1,6]	* 1	0	1
7	a[1,7]	* 1	0	1
8	a[1,8]	* 1	0	1
9	a[1,9]	* 1	0	1
10	a[1,10]	* 1	0	1
11	a[1,11]	* 1	0	1
12	a[1,12]	* 1	0	1
13	b[1,2,1]	* 0	0	1
14	b[1,3,1]	* 0	0	1
15	b[1,4,1]	* 0	0	1
16	b[1,5,1]	* 0	0	1
17	b[1,6,1]	* 0	0	1
18	b[1,7,1]	* 0	0	1
19	b[1,8,1]	* 0	0	1
20	b[1,1,2]	* 1	0	1
21	b[1,3,2]	* 0	0	1
22	b[1,5,2]	* 0	0	1
23	b[1,6,2]	* 0	0	1
24	b[1,7,2]	* 0	0	1
25	b[1,9,2]	* 0	0	1
26	b[1,1,3]	* 1	0	1
27	b[1,2,3]	* 0	0	1
28	b[1,4,3]	* 0	0	1
29	b[1,6,3]	* 0	0	1
30	b[1,7,3]	* 0	0	1
31	b[1,1,4]	* 1	0	1
32	b[1,3,4]	* 0	0	1
33	b[1,5,4]	* 0	0	1
34	b[1,8,4]	* 0	0	1
35	b[1,11,4]	* 0	0	1
36	b[1,1,5]	* 0	0	1
37	b[1,2,5]	* 0	0	1
38	b[1,4,5]	* 0	0	1
39	b[1,8,5]	* 0	0	1
40	b[1,12,5]	* 1	0	1
41	b[1,1,6]	* 0	0	1
42	b[1,2,6]	* 1	0	1
43	b[1,3,6]	* 0	0	1
44	b[1,7,6]	* 0	0	1
45	b[1,9,6]	* 0	0	1
46	b[1,10,6]	* 0	0	1
47	b[1,1,7]	* 0	0	1
48	b[1,2,7]	* 0	0	1
49	b[1,3,7]	* 0	0	1
50	b[1,6,7]	* 1	0	1
51	b[1,10,7]	* 0	0	1
52	b[1,1,8]	* 1	0	1
53	b[1,4,8]	* 0	0	1
54	b[1,5,8]	* 0	0	1
55	b[1,11,8]	* 0	0	1
56	b[1,12,8]	* 0	0	1
57	b[1,2,9]	* 0	0	1
58	b[1,6,9]	* 1	0	1
59	b[1,10,9]	* 0	0	1
60	b[1,6,10]	* 1	0	1
61	b[1,7,10]	* 0	0	1
62	b[1,9,10]	* 0	0	1
63	b[1,4,11]	* 1	0	1
64	b[1,8,11]	* 0	0	1
65	b[1,5,12]	* 0	0	1
66	b[1,8,12]	* 1	0	1

```
Integer feasibility conditions:

KKT.PE: max.abs.err = 0.00e+00 on row 0
        max.rel.err = 0.00e+00 on row 0
        High quality

KKT.PB: max.abs.err = 0.00e+00 on row 0
        max.rel.err = 0.00e+00 on row 0
        High quality

End of output
```