A Sensor Network Querying Framework for Target Tracking

by

Francisco de la Parra

A thesis submitted to the
School of Computing
in conformity with the requirements for
the degree of Master of Science

Queen's University
Kingston, Ontario, Canada
March 2009

Abstract

Successful tracking of a mobile target with a sensor network requires effective answers to the challenges of uncertainty in the measured data, small latency in acquiring and reporting the tracking information, and compliance with the stringent constraints imposed by the scarce resources available on each sensor node: limited available power, restricted availability of the inter-node communication links, relatively moderate computational power.

This thesis introduces the architecture of a hierarchical, self-organizing, two-tier, mission-specific sensor network, composed of sensors and routers, to track the trajectory and velocity of a single mobile target in a two-dimensional convex sensor field. A query-driven approach is proposed to input configuration parameters to the network, which allow sensors to self-configure into regions, and routers into tree-like structures, with the common goal of sensing and tracking the target in an energy-aware manner, and communicating this tracking data to a base station node incurring low-overhead responses, respectively.

The proposed algorithms to define and organize the sensor regions, establish the data routing scheme, and create the data stream representing the real-time location/velocity of a target, are heuristic, distributed, and represent localized node collaborations. Node behaviours have been modeled using state diagrams and inter-node collaborations have been designed using straightforward messaging schemes.

This work has attempted to establish that by using a query-driven approach to track a target, high-level knowledge can be injected to the sensor network self-organization processes and its following operation, which allows the implementation of an energy-efficient, low-overhead tracking scheme. The resulting system, although built upon simple components and interactions, is complex in extension, and not directly available for exact evaluation. However, it provides intuitively advantageous behaviours.

Acknowledgments

I would like to thank my thesis supervisor, Dr. Selim Akl, for his support, ideas, guidance and assistance in completing this thesis. I would also like to thank the members of my thesis examination committee: Dr. Hossam Hassanein, Dr. Juergen Dingel, and Dr. Mohamed Ibnkahla, for their helpful suggestions.

I would like to dedicate this thesis to my wife Joanne for her love, inspiration and encouragement to undertake this extraordinary assignment, and to my children Lauren and Daniel who have been an endless source of motivation.

Contents

List of Tables

List of Figures

List of Acronyms

ALUC	Alternate Links Usage Configuration
AoA	Angle of Arrival
BLSD	Biased Sensor Leader Designation
BS-node	Base Station Node
DSB	Data Stream Builder
GPS	Global Positioning System
HCRRF	Hop Count Restricted Region Formation
HFAT	Hierarchical Focus of Attention Tracking
LEACH	Low Energy Adaptive Clustering Hierarchy
LOB	Line of Bearing
LS-node	Leader Sensor Node
MAC	Medium Access Control
NAPR	Neighbour Angular Partition Routing
QoS	Quality of Service
RF	Radio Frequency
R-node	Router Node
RS-node	Region Sensor Node
RSS	Region Sensing Scope
S-node	Sensor Node
SNR	Signal-to-Noise Ratio
SOA	Self-Organization Algorithm
ToA	Time of Arrival

List of Symbols

$\cup$	Union of sets
$\cap$	Intersection of sets
$\{\}$	Set
Φ	Empty set
$\lfloor x \rfloor$	$floor(x)$
$Min()$	Minimum value in a set
$Max()$	Maximum value in a set
$(\dot{x}, \dot{y})$	Velocity vector with components in direction x and y
$\hat{x}$	Estimated value of variable x
//	Comment line
|	*Such that* delimiter in set definition
>>	*Much greater than* operator
$[x \mid y]$	Either x or y
$X \backsim f(x)$	X follows probability distribution $f(x)$
$N(\mu, \sigma^2)$	Normal probability distribution. μ = mean, σ^2 = variance

Chapter 1

Introduction

Advances in technology have made possible the manufacturing of small, low-cost, low-power, communication-enabled, programmable micro-sensors [1, 3, 24] which can be deployed and configured into large and dense wireless sensor networks [15, 40]. These large-scale networks allow the collection of vast amounts of very granular spatio-temporal data in applications monitoring environments that are geographically dispersed or present accessibility challenges (wildlife monitoring, chemical processes, battlefield tracking, area surveyance, etc.). Additionally, the sensor nodes in these networks can be programmed to support changes in communication/sensing patterns and multi-mode operation (i.e. standby, active node, router node, etc.), allowing the implementation of relatively powerful adaptation functionality for fault tolerance, energy conservation and improved data collection, aggregation and reporting [15, 16, 38, 40, 46]. These properties make sensor networks a resourceful tool with a great potential for dramatically improving the sensing quality in environments characterized by continuous and rapid state changes. The tracking of mobile targets in enclosed geographical areas, for example: vehicles in a metropolitan area, warfare elements in a battlefield, displacement of a chemical concentration in an area of interest, etc., is one type of application that can take advantage of this distributed sensing approach [8, 28].

This thesis elaborates on a model for tracking the trajectory of a single mobile target, moving within the limits of a convex *sensor field*, using a query-driven approach to configure a large and dense sensor network, having two types of nodes: sensors and routers, for a low-latency response in the data collection task and minimized energy usage during the tracking operation.

1.1 A Sensor Network for Target Tracking

Different aspects of the approach of tracking a mobile target with a large and dense sensor network have been studied in detail during recent years because of the potential advantages, in comparison to previous systems such as radar or sensor arrays, that can be realized in monitoring irregular terrains: reduced signal-to-noise ratios, increased accessibility of the sensed environment, and improved quality of the measurements. The networked nature and large number of interacting components, as well as the limited node energy, processing capacity, and communication capability normally found on the component nodes of this type of system, bring up the difficult challenge of managing potentially high volumes of data traffic with fairly reduced resources. In principle, it becomes apparent that high level data routing, aggregation and classification decisions, rather than low-level data transport optimizations, provide the conditions to efficiently manage the high-volume and rapid-response requirements of the target-tracking operation. Furthermore, a hierarchical architecture of heterogeneous components for this type of network becomes a natural choice.

This thesis introduces a *large and dense* sensor network, *strongly connected* [35] (i.e. there exists a connection path between any two nodes of the network), with a two-tier architecture of sensor (S) and router (R) nodes, for the tracking of a single mobile target. This work develops the strategies and algorithms for extracting to a base station, in an efficient and accurate manner, the tracking data corresponding to the trajectory of a mobile target in a

format ready for rendering. It uses the following heuristics principles throughout the design process:

- *Heterogeneous network* - Separate into layers the *data sensing* and *data routing* functions, required to reproduce the trajectory of the target on the base station node, and assign to sensor and router nodes, respectively. Design R-nodes with more energy, processing, and communication capabilities than S-nodes.

- *Hierarchical delegation of control functions* - Assign higher-level control functions to the nodes equipped with more resources. Allocate only low-level control functions to S-nodes: sense mobile target, report to a leader sensor within a group, local data aggregation for reporting to the base station, become a standby or active node. Allocate higher-level control functions to R-nodes: extended-group data aggregation and classification, sensor group configuration, sensor group activation/deactivation, fault tolerant data routing.

- *Hierarchical delegation of decisions* - Assign higher-level decisions to R-nodes closer to the base station.

- *Even distribution of the tracking load* - Distribute the tracking processing load (sensing, data aggregation, data routing, control and decision functions) in a well-balanced manner across the entire network, from sensor nodes in the vicinity of the target all the way to the base station.

- *Low complexity operations* - Minimize or eliminate the use of computing-intensive operations with high message complexity.

- *Low overhead tracking coordination* - Coordinate the tracking operation for minimal energy usage, improved availability and low-overhead fault tolerance by only activating the nodes close to the current location of the target and using a collaborative geographical data routing over the R-nodes.

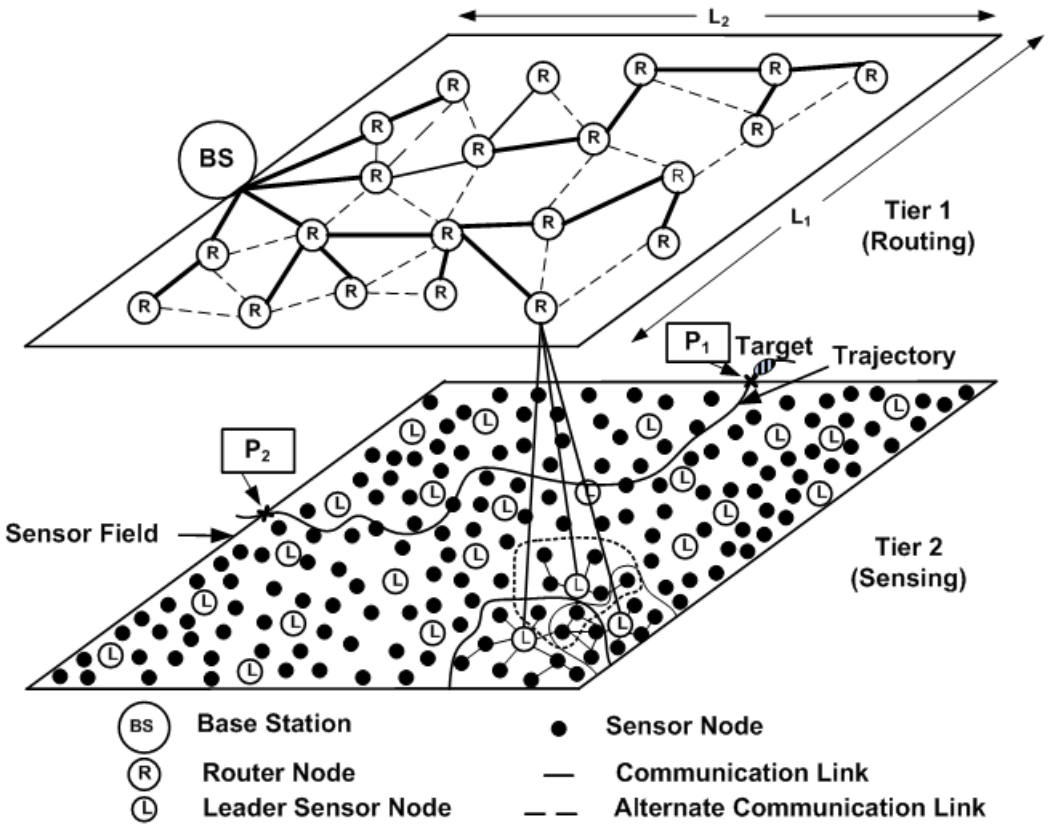


Figure 1.1: Two-Tier Sensor Network Architecture for Tracking a Single Mobile Target

In this study, the *sensor field* is defined as a two-dimensional rectagular area of length L_1 and width L_2, where S and R-nodes are deployed in a random manner with an approximately uniform density. Although both types of nodes share the same deployment area, for communication purposes they can be seen as a two separate layers of *strongly connected* nodes having S-R inter-layer communication links at specific points. The communication radii defined for the S and R-nodes, as well as their relative physical locations, establish a node connectivity of the assumed deployment characterized by the following:

- There exists a connection path between any two S-nodes.
- There exists a connection path between any two R-nodes.

- Each R-node is directly connected to at least one S-node.

Figure 1.1 shows the two-tier architecture of the sensor network.

S and R-nodes are modeled as state machines which are updated by the running of data routing, data collection, power management and fault tolerance algorithms. In general, the behaviour of the entire network is modeled as a set of asynchronous distributed processes, running on the S and R-nodes, communicating with each other by exchanging messages over the inter-node communication links.

S-nodes are defined as having omnidirectional sensing and communication capabilities which are modeled as circles, centered at each node, representing their scope of action. R-nodes have only omnidirectional communication capability also modeled as node-centered circles. Both types of nodes can operate in one of the two following power states: *active* and *standby*. During normal operation, nodes in the standby mode are returned to the active mode by receiving an *activation message* from an active node. However, in specific situations they can also re-activate themselves after a defined elapsed time.

All S-nodes have the same sensing radius S_s and communication radius S_c, and the same single sensing modality (i.e. acoustic, radio-signal based, etc.) for detecting the presence of a target located at a certain range. Based on target tracking conditions pre-established for the network, some of them assume the more complex role of a *leader sensor* (LS). The sensing and communication radii in the standby mode are S_{s_0} and S_{c_0} respectively, where $S_s >> S_{s_0}$ and $S_c > S_{c_0}$ (S_{c_0} is defined based on minimal sensor connectivity requirements). Each S-node can detect the range (i.e. euclidean distance to an object), with a given margin of error, of a point-like target currently moving within the scope of its sensing radius. All S-nodes are deployed with approximately the same energy reserve E_s.

All R-nodes have the same communication radius R_c, and execute the same distributed algorithms for data routing, data collection, power management and fault tolerance. The communication radius in the standby mode is R_{c_0}, where $R_c >> R_{c_0}$. All R-nodes are deployed with approximately the same energy reserve E_r. By design, $R_c >> S_c$.

1.2 A Query-Driven Approach to Configure the Network

This work uses a querying scheme to track the trajectory of a single target entering a planar sensor field, as shown in figure 1.2, where it is assumed that a user of the system issues a declarative query to inform the sensor network about the expected initial state and general dynamic behaviour of the target, as well as to indicate the frequency with which the state of the target will be sampled and the duration of the tracking operation. This action establishes a *persistent tracking query* whose output is a continuous stream of sampled values for the state of the target, represented by the 4-tuple $(x(t), y(t), \dot{x}(t), \dot{y}(t))$, where the pairs $(x(t), y(t))$ and $(\dot{x}(t), \dot{y}(t))$ represent the target's current location and velocity, respectively.

The nodes in the network (BS, R, and S) process the *parameter values* provided by the user in the tracking query to set up, in a distributed manner, the *routing scheme* to extract the current state of the target to the BS-node. These parameters are also used to organize the S-layer into regions of S-nodes, of variable size, for carrying out the target tracking in an energy-efficient manner. The central ideas are: to configure the network (S and R-nodes) for minimal paths, approximately uniform energy depletion among the different nodes of the network, reduced energy usage, reasonable fault-tolerance for failed communication links, and improved sensor collaboration for accurate and energy-aware tracking.

In the general scheme, the BS-node finishes processing the tracking query and contacts the R-nodes which are its immediate neighbours, requesting from each one of them to send

its (x, y) coordinates and reading of its energy reserve. Next, the BS-node selects a sub-set of these nodes – the ones that would produce a reduced overhead and energy consumption tracking response – and makes them its *primary children*. The links between the BS-node and its primary children become the *primary links* from the R-nodes to the BS-node. The R-nodes not selected by the BS-node become its *secondary children* and the links from these nodes to the BS-node become *secondary links*. The latter links are used to provide fault tolerance in the case of a link failure. The BS-node informs the secondary children that it is their *secondary parent*, allowing the children nodes to establish a link-priority scheme when communicating with the BS-node.

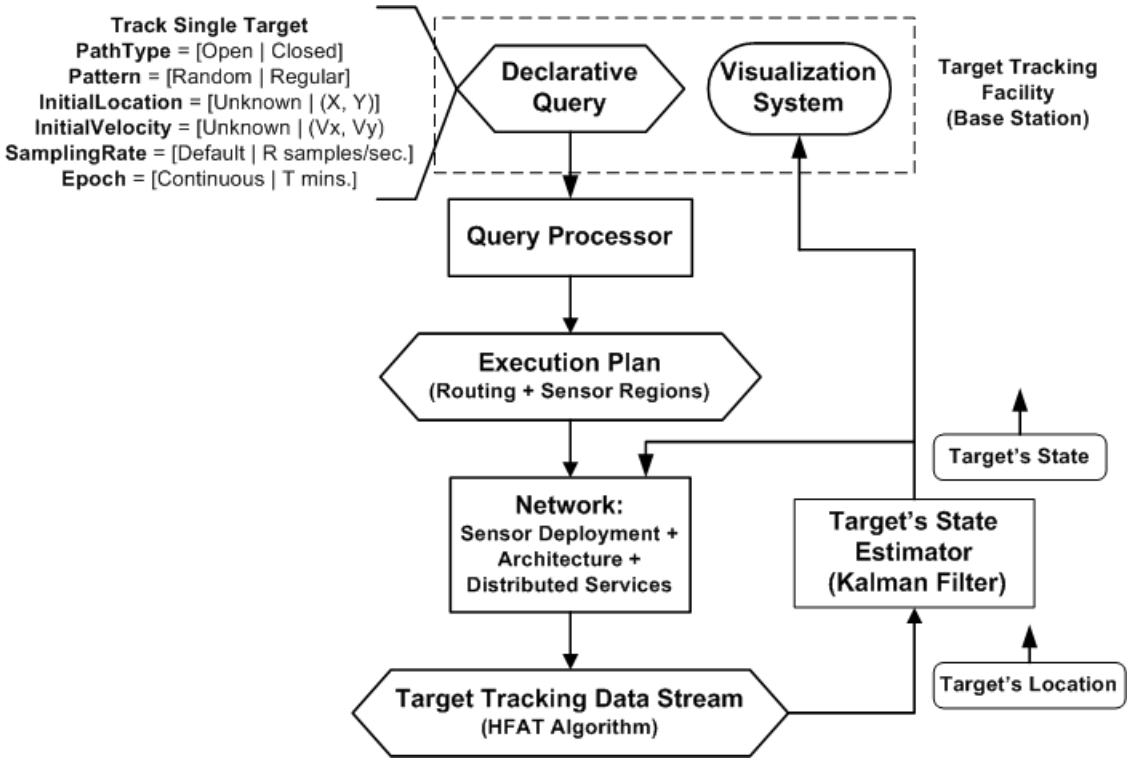


Figure 1.2: Querying Scheme for a Low-Overhead Target Tracking Response

In completing the configuration scheme, each of the primary children R-nodes of the BS-node configures the sensor regions in its vicinity and then repeats the process described

in the previous paragraph for the BS-node. This recursive process stops at R-nodes not having any children R-nodes. The entire process stops when all the primary and secondary R-nodes and links are determined, and all the S-nodes in the deployment are structured into regions having connection paths to a local R-node. The primary nodes and links will form a rooted tree with the BS-node as the root. The selection of secondary nodes and links, one at a time, will form alternate rooted trees, all having the BS-node as a root.

In each sensor region, the R-node associated with the region *designates a temporary leader sensor* (LS), based on its information about the dynamic behaviour of the target, the shape of the region, and the energy distribution among the nodes in the region.

The tracking data is extracted from the area where the target is currently located via the LS-nodes, routed to the closest R- node, and from this point, routed to the BS-node by the nodes in the router layer, having each R-node send the received data to its parent. If the link from an R-node to its parent fails, the R-node will use the link to the secondary parent having the next priority.

1.3 Hierarchical Focus of Attention Tracking

The Hierarchical Focus of Attention Tracking (HFAT) algorithm proposed in this thesis is an energy-aware, low-overhead approach to track the dynamic behaviour (i.e. two-dimensional location and velocity) of a single target in a convex sensor field. The following paragraphs describe its main steps:

After the routing is configured for the entire network, the BS-node initiates a *continuous target detection* mode consisting of periodically polling the different LS-nodes, in a given sequence, and prompting them for a positive single target detection. The LS-node polling mechanism is propagated over the layer of R-nodes, having each R-node prompt only its

immediate neighbour R-nodes. The shape and progression of these detection waves [3] is determined by the parameters defined in the tracking query.

The first sensor region to detect a target informs the BS-node about the event by sending the current location of the target to this node. At this point, the BS-node stops the target-detection polling and switches to the *target-tracking* mode. The BS-node broadcasts the initiation of the tracking mode to all the R-nodes, sends an *activate region* message to the R-node associated with the sensor region which detected the target, making it the *active region* for target tracking, and broadcasts a *deactivate region* message to the R-nodes associated with all the other regions making them *inactive regions* for target tracking.

The sensor network remains in the target-tracking mode as long as the target is located in the sensor field and its (x, y) location can be sensed and reported to the BS-node. At time t_n $(n = 1, 2, ...)$, the BS-node sends a *locate target* message to the R-node associated with the active sensor region, to start a new target detection/tracking cycle. During this steady operation, the S-nodes in the active sensor region collaborate to determine the target's location, at a rate determined in the tracking query, by sensing and reporting *target range* measurements to the LS-node in the region. Also, each node in the region reports to the LS-node, after a specified number of sampling periods, a measurement of its energy reserve. The LS-node processes the range readings from the sensors, along with its own reading, and calculates a *best estimate* of the *current* location of the target using a *trilateration* algorithm. The LS-node reports the calculated target's location to its associated R-node, including a time stamp for this value. After a speficied number of sampling periods, the LS-node reports the energy readings received from the sensors in its region, as well as its own, to the associated R-node.

The R-node associated with the active region routes the target's location to its parent R-node. This process is repeated by each parent R-node, upon receiving a target location

message, until this information reaches the BS-node. Also, after a specified number of sampling periods, the R-node reports a reading of its energy reserve to its parent node (R or BS-node).

The BS-node builds up the target's state data stream $(x(t_n), y(t_n), \dot{x}(t_n), \dot{y_n}(t))$ $(n = 1, 2, ...)$, with the $(x(t_n), y(t_n))$ target location data received from the R-nodes, using a Kalman filter [9], which provides good quality interpolation of location data and estimation of the target's velocity $(\dot{x}(t), \dot{y}(t))$ based on a *linear model with random walk* [6] for the trajectory of the target.

Also, using the filtered estimate for the target's state, the BS-node calculates an *a priori estimate* for the target's location at time t_{n+1} (i.e. $(x(t_{n+1}), y(t_{n+1}))$, and re-initiates a *continuous target detection* mode, if this latter estimate indicates that the target has left the sensor field. At the same time, it sends a *deactivate regions* message to the R-node which originated the last target location. Otherwise, the BS-node determines whether the target has left all the sensor regions associated with its child R-node which last reported a target location, and if this has occurred, it determines the new set of sensor regions and associated child R-node where the target will be found next. Then, it sends a *deactivate regions* message to the R-node which originated the last target location and a *locate region* to the child R-node having associated the new region where the target will be found. If no *region set* change has occurred, the BS-node sends a *focused* message to the child R-node which lat reported a target's location. This processing wave continues until it reaches the R-node in the immediate vicinity of the current estimated location of the target. This latter R-node processes the same *region change* algorithm, implemented between parent and child R-nodes, with its associated LS-nodes. At this point, the algorithm has determined whether the target will remain or not in the same sensor region, and only the region containing the target is *active*.

This last operation completes the processing cycle corresponding to time t_n. The BS-node will initiate the next processing cycle by issuing a *locate target* message at time $t_{n+1} = t_n + T$ and repeating the process previously described.

If the associated R-node receives a negative *focused* message from the LS-node, it determines, based on the dynamic behaviour of the target and the geometry of sensor regions under its control, whether another region is capable of sensing the next state of the target. If the answer is affirmative, it sends a *deactivate sensors* message to the LS-node of the current active region and an *activate sensors* message to the LS-node of the newly-selected region. Then, it routes the state of the target towards the BS-node as explained in the previous paragraph.

If, upon receiving a negative *focused* message from the LS-node, the associated R-node determines that none of the regions under its control can sense the next state of the target, it sends a negative *focused* message and the current state of the target to its parent R-node, and a *deactivate sensors* message to the originating LS-node. This process is repeated until an R-node, or the BS-node, finds a region capable of sensing the next state of the target, at which point, it sends an *activate sensors* message to the LS-node of the newly selected region. Thereafter, the current state of the target is routed to the BS-node completing the current target-tracking cycle. If this processing wave [3] reaches the BS-node and a region capable of sensing the next state of the target is not found, the BS-node initiates a *continuous target detection* processing wave, and this operation mode remains in effect until the target is detected again.

Finally, the following steps in the target-tracking algorithm implement energy-aware and fault tolerant behaviours:

- After the completion of a target-tracking cycle, each R-node, having associated sensor

regions, determines, based on a preferred node energy depletion pattern, whether to keep the same LS-nodes or *designate* new ones.

- R-nodes having sensor regions under their control report their energy reserves to the BS-node. This latter node will trigger a network re-configuration wave if the depletion of the energy on an R-node causes a loss of sensing coverage.

- The BS-node will trigger a network re-configuration wave after the target is lost, estimated as located in the sensor field, and not recovered after some pre-determined elapsed time.

- R-nodes will use alternate links to report the state of the target to their parent R-nodes when the primary links are not available.

1.4 Contributions

The contributions of this thesis are:

1. The design of a two-tier network architecture, with homogeneous layers of sensor (S) and router (R) nodes, to track a single mobile target incurring reduced-overhead responses and energy consumption, in a manner compatible with energy efficiencies that can be attained by the supporting layers of the communication stack.

2. A query-driven approach to configure an efficient routing scheme on the proposed network based on building preferred paths on the R-node layer, as well as creating a flexible scheme of sensor regions and leader nodes within regions to provide a light approach to track a single target. Embedding of high-level knowledge, through the use of parameters, about the dynamic behaviour of a target in a tracking query allows organizing the network for improved light behaviour over systems based on network self-organization only.

3. The *Neighbour Angular Partition Routing* (NAPR) algorithm to configure the routing of data, from the sensor network to a base station node and viceversa, in an energy aware fashion. Algorithm NAPR can create a scheme of energy-aware shorter paths to the BS-node, adapted to a given target dynamics, to provide improved data extraction performance. It shows a lighter behaviour than routing schemes requiring dynamic re-calculation of routes to the BS-node.

4. The algorithms to define the organization and shape of dynamic sensor regions in the network – Biased Sensor Leader Designation (BSLD) and Hop Count Restricted Region Formation (HCRRF) – aiming to facilitate light, energy-aware tracking of a single target by completely avoiding high-overhead continuous leader election and region formation schemes around the current location of the target.

5. The *Hierarchical Focus of Attention Tracking* (HFAT) algorithm for single-target tracking. This distributed algorithm strives to provide a reasonably accurate target tracking response with low latency, energy savings, and a degree of fault-tolerance. It allocates the more computing intensive tasks, in tracking a single target, to the BS-node, implementing a light data fusion scheme at the LS-node level, and extracting to the former node very small data sets. It requests fewer resources from the network than schemes maintaining target trajectories on sensors or relaying nodes.

6. Algorithms to support a light localization, detection, and tracking of a single target, as well as reporting its state (i.e. location and velocity) to a user of the system. The *Region Sensing Scope* (RSS) algorithm obtains the graph corresponding to the polygonal perimeter of a sensor regions's sensing scope, providing a light tool to localize a target within a given sensor region. The *adaptive trilateration* algorithm provides the current coordinates of a target, out of a target-range readings set, in a light manner, with *best effort* estimation for missing data. The *target's state estimator* provides a reasonably good estimation of both, the target's location and velocity. The

combined use of these algorithms provides good estimates of the target's state with lower resource utilization than schemes attempting to obtain all measurements (i.e. location and velocity) directly from individual sensors.

1.5 Organization of this Thesis

This thesis is organized as follows:

- Chapter 1 shows an overview of the general set-up used to track a mobile target, with a proposed sensor network architecture and algorithms, incurring low-overhead in the tracking operation.
- Chapter 2 introduces the the fundamental elements, collaboration techniques, models and requirements considered in tracking a mobile target with a sensor network.
- Chapter 3 describes the architectural details of the network proposed in this thesis to track a single mobile target.
- Chapter 4 describes the distributed algorithms designed to configure the network, based on a query issued by the user of the system, for low-overhead, energy-aware tracking with acceptable accuracy.
- Chapter 5 summarizes the work completed, contains the conclusions about the modeling of the proposed sensor network, the associated algorithms, and discusses future work.

Chapter 2

Background

Expanding on the ideas comprising the main contribution of this thesis – the architecture of a hierarchical sensor network, the algorithms to configure it and allow it to track a single target with a low-overhead response requiring reduced energy consumption, based on parameters embedded in a query issued by a system user – this chapter provides the background on the central aspects considered in designing a sensor network for the specific application of target tracking.

Section 2.1 presents the general model for tracking a single target with a sensor network, the network architectures proposed in the literature to address the problem, the sensor collaboration and tracking schemes for reduced latency and energy consumption. Section 2.2 describes models of target-dynamic behaviour, target detection and localization techniques used in sensor networks, trajectory estimation and prediction approaches. Section 2.3 provides an overview of different programming models for sensor nodes used in addressing the target-tracking problem. Section 2.4 outlines efficient approaches to processing and executing queries issued to a sensor network from a base station under low-overhead and energy-efficiency requirements.

2.1 Sensor Networks for Single-Target Tracking

Different aspects of tracking single mobile targets with a sensor network have been described in the literature during recent years: energy-aware target detection and tracking [4, 19, 33, 39, 49], sensor collaboration schemes [7, 8, 5, 53, 55, 56], efficient signal processing approaches [28, 33, 40, 50], etc., mainly in the context of an underlying architecture of equal sensor nodes [16]. Although the advantages of using hierarchical sensor network architectures in target-tracking applications have been referenced in research reports [15, 25], there is still a great deal to investigate on the subject. Both architectures show some strengths and weaknesses that deserve a concise analysis to provide background clarity to design decisions that will be introduced later on as part of this thesis. This section introduces the single target tracking scenario, and describes these two main architectures: flat and hierarchical, which have been applied in providing answers to this problem.

2.1.1 General Deployment Model

Providing a formal definition of the single-target tracking problem on a plane is a crucial step in establishing the boundaries for the behaviour and performance of the network architecture and algorithms to address the problem. The framework of definitions used in this work can be stated as follows:

Given a two-dimensional convex field $\mathcal{F}$ (such as a battlefield, a wildlife conservation area, a manufacturing facility, etc.) that can be modeled as an enclosed flat surface, a requirement has been established to monitor the dynamic behaviour, namely, the trajectory $\vec{\mathcal{P}}(t) = (x(t), y(t))$ and velocity $\vec{\mathcal{V}}(t) = (\dot{x}(t), \dot{y}(t))$, of a *point-like* single target $\mathcal{T}$ that has entered in the area. A set $S = \{S_1, S_2, ..., S_n\}$ of sensors has been deployed on field $\mathcal{F}$ using a certain positioning rule $\mathcal{PR}$ (i.e. sensors positioned in pre-determined locations, or under a specific regular topological arrangement, or randomly positioned with approximately a

uniform density, etc.) to survey field $\mathcal{F}$ and report the instantaneous velocity and location of target $\mathcal{T}$ to a base station node $\mathcal{BS}$. Each sensor $S_i(i = 1, ..., n)$ has a sensing radius R_{s_i}, a communication radius R_{c_i}, an energy reserve E_i, and a behaviour $\mathfrak{B}_i$ corresponding to that of an asynchronous automaton. Sensor positioning rule $\mathcal{PR}$ produces a *connected graph* under the given communication radii R_{c_i}, and *complete sensing coverage* of field $\mathcal{F}$ under the given sensing radii R_{s_i}.

The *single-target tracking operation* consists of the visualization of the data reported to node $\mathcal{BS}$ from the sensor field by the set S as a result of performing the following operations: target detection, target-tracking (activation/deactivation of sensors, trajectory estimation), routing of the target's state (4-tuple $(x(t), y(t), \dot{x}(t), \dot{y}(t))$) to node $\mathcal{BS}$. With some small variations, this is the general model over which different solutions to the single-target problem have been researched. The emphasis has been on creating tracking schemes and protocols to capture and report the statio-temporal data corresponding to the target's movement in an energy-efficient way, in most situations relegating to a second priority other requirements such as low-overhead response and tracking fidelity. To provide a general background on the practical issues and the challenges that this type of system brings about, the next sections introduce the main characteristics of these networks as they relate to the target tracking problem.

2.1.2 Network Architectures

The overall goal in tracking a mobile target with a sensor network is to reproduce on a visualization display (located at a base station), as accurately and efficiently as possible, the spatio-temporal data generated by sensing the trajectory of this dynamic entity with the component nodes of the network. This apparently straightforward operation requires the completion of multiple functionality stages such as: sensing the current location of the target at a given rate, consolidating generated time series and reporting suitable fused

values, routing those fused values from the area around the current position of the target to the base station, estimating its velocity, controlling the network for minimal energy usage, managing the network for efficient self-configuration in exception situations (i.e. links failures, node failures, lost targets, etc.). The *flat* and *hierarchical* architectures have been proposed in the literature as possible solutions to distribute, organize and coordinate the functions previously described.

Flat Networks – Their architecture consists of equal sensor nodes with basically the same sensing, data fusion, data routing, general processing power, and communication capabilities. Query generation can potentially occur at any node of the network [16, 56]; however, in the case of target tracking, it is normally a function executed at a base station with expected results at the same location. Sensor nodes are usually assigned identifiers relating them to their current geographical locations in the sensor field. They assume temporary roles (i.e. active node, leader sensor, member node, data router, etc.) or combinations of roles based on the current location of the target detected by the network, and they manage the requirements of low energy consumption, improved communication paths and fault tolerance in a highly distributed manner. Multi-hop communication is prevalent, although it is a serious drawback when meeting stringent requirements of low-overhead tracking and extended network lifetime under different target trajectories. Temporary leader election schemes localized around the target's current position have been a common approach in the literature [8, 55] to implement sensor collaboration for the tasks of signal processing and data fusion. Most of these schemes require elaborate node-synchronization approaches, construction of real-time data structures with the nodes and communication links currently involved in the tracking (i.e. tree structures with the leader node as the root, sequential leader chain, etc.), and relatively high message complexity to accomplish the previous tasks.

Hierarchical Networks – These networks normally have a layered architecture consisting of two or more physical or logical tiers of nodes with dissimilar roles [25]. Networks with different *physical tiers* consist of deployments where some sensor nodes have more processing power, communication capability and energy reserve than others, allowing for the assignment of computing, communication and sensing tasks in approximately direct relation to the per-node available capacity. Networks with different *logical tiers* represent deployments of equal sensor nodes where some nodes, on a temporary basis, assume higher level clustering roles for the purpose of data fusion and routing tasks, as well as network configuration and power management functions. Network architectures implementing a combination of physical and logical tiers, such as the network proposed in this thesis, allow a more adjustable allocation of tasks to resources for tracking spatio-temporal data. Query generation almost always occurs at a base station node, with query dissemination schemes that are hierarchical in nature, yet fast and efficient in execution.

2.1.3 Distributed and Collaborative Tracking

The general approach proposed in the literature to track the state $\vec{S}(t) = (x(t), y(t), \dot{x}(t), \dot{y}(t))$ of a moving target with a sensor network consists of allowing a sub-set $A = \{S_1, ..., S_k\}$ of *active* sensors, located in the vicinity of the current target's position $\vec{\mathcal{P}}(t) = (x(t), y(t))$, to collaborate with each other and possibly with some of the heterogeneous general processing (i.e. data fusion, selection, and routing logic) nodes of a set $G = \{G_1, ..., G_k\}$, by exchanging data and control messages, in order to perfom the tasks required to complete this operation. The sensor network acquires samples of the state of the target at discrete times $\{t_1, ..., t_n\}$, usually separated by a constant period T, and processes them under the assumption that the target behaves as a discrete dynamic system.

Target Detection – Point-like targets can be classified as *passive* - those not emitting any kind of signal and just acting as reflectors of signals emitted by the sensors in their

proximity area - and *active* or *instrumented* - those defined as emitting sources of some kind of acoustic, seismic or electromagnetic signal that will be detected by the sensors in their surroundings. Attempting to determine with acceptable accuracy the current location of either type of target, using a single sensor, would require complex signal processing and considerable energy usage on that battery-operated, low-computational power device, thus depleting its limited resources very rapidly and dramatically shortening its service life. By identifying a group of sensors to detect the target's current position, one can assign simpler and more energy efficient detection tasks. By having the sensors in the identified group collaborate with each other, it is possible to build a more accurate and reliable estimation of the target's current location.

- Target *ranging* is a detection technique that lends itself to a simplified implementation on a single sensor; it is commonly used with passive targets and provides a measurement of the distance between a sensor and the target, without any indication of direction [5]. A sensor sends a signal (i.e. radio frequency (RF) or ultrasound signal) to the target and measures the *time-of-arrival* (ToA) of the reflected signal to estimate the distance to the target.

- *Line of bearing* (LOB) estimation is another technique used with active targets, although more computationally intensive for implementation on single sensors; it is commonly used in detecting targets emitting acoustic signals and it is based on having a sensor filter the signal incoming direction producing the best *signal-to-noise ratio* (SNR) [50] to provide an estimation of the direction (i.e. angle relative to a reference direction), relative to itself, where it is most likely the target is currently located, without estimating its distance to the target.

Single sensors computing partial estimates of a target's location, by applying one of the techniques previously described, can now collaborate to provide a group estimate of a specific

(x, y) coordinate set for the target, using one of the approaches described in section 2.2.2.

Signal Processing – A typical scenario in tracking a mobile target with a sensor network is one where a large number n of nodes generate a time series $P_j = \{p_{1j}, ..., p_{mj}, p_{(m+1)j}, ...\}$ $(i = 1, ..., m, m+1, ...; j = 1, ..., n)$ corresponding to filtered sampled values of a physical property $\mathcal{P}$ (i.e. sound intensity, energy detected on reflected radiation, etc.) detected as generated from the target. The values in each time series P_j are usually produced at a high rate, generating some degree of redundancy in the measurements, which, although useful to produce a better estimate for the current value of physical property $\mathcal{P}$, represents high volumes of data to be communicated to, for example, nodes producing consolidated estimates of this property . Furthermore, the time series P_j generated by a node j is used to estimate a physical signal that provides only a partial estimation of the current target's location. Under these circumstances, sensor collaboration for signal acquisition and filtering, provides improved conditions to carry out these tasks in an energy efficient manner (refer to section 2.1.6). In this work, it is assumed that each sensor node is built as a low-power, single-chip hardware device [3] implementing all the following functions:

- *Sense* the physical signal irradiated by the target and convert it to electrical signals of appropriate levels.
- *Sample* the converted electrical signal to produce a time series of digitized discrete values with a resolution of n bits.
- *Perform spectral analysis* on the digitized signal to determine the occurrence of specific *events* such as the presence of a nearby target. This analysis also allows the sensor node to decide whether to operate or remain in standby mode.
- *Communicate* a subset of values extracted from the digitized signal to other nodes in the network by using a wireless transmission module.

- *Control* the operation of the sensor, analog-to-digital converter, spectral analysis, and communication modules by executing a stored program.

2.1.4 Collaborative Data Fusion

To track a target, a processing node in a sensor network needs to establish and maintain in its memory an *estimation* of the current state $\vec{S}(t) = (x(t), y(t), \dot{x}(t), \dot{y}(t))$ of the target, along with relevant state history and temporal predictions of future states, all tasks collectively known as establishing a *track*. A track for a target can be initiated at the *base station* or at any other, usually high capacity, node. The collaboration scheme in this case is characterized by a group of sensors capturing different measurements of the target's current state and communicating them to a *data fusion node* which maintains a *partial* or *complete* track [30]. Variations of this scheme represent the following different architectures for data fusion in a multi-sensor environment:

- *Centralized*– All sensors communicate their measurements to a central node which maintains a complete track.

- *Hierarchical*– Sensors communicate their measurements to local nodes having partial tracks, which in turn communicate their tracks to higher level nodes. This process can repeat through many different levels before reaching the hierarchy's root which maintains a complete track. *Hierarchical data fusion with feedback* represents a slight variation of this architecture and is characterized by maintaining complete tracks at nodes other than the root.

- *Distributed*– Sensors communicate their measurements to local nodes, which may or may not maintain tracks, for data routing, estimation or fusion. *Routing-only nodes* just relay measurements to either estimation or fusion nodes. *Estimation-only nodes* maintain partial tracks and report their tracks to data fusion nodes. *Fusion nodes* maintain partial or complete tracks.

This thesis proposes a variation of the distributed scheme where localized data fusion, around the current location of the target, occurs in a centralized manner on a node temporarily designated as the *local tracker*. This node maintains a complete track of the target for the time it is located within the limits of the *active-for-detection* sensor node region. The network nodes with the capacity to establish a complete track collaborate to determine whether the target remains in the active region or moves to another region. When the target moves to a new active sensor region, the current tracker node transfers the complete track to the new data fusion node associated with the latter region. The data fusion nodes report to the base station only the current values for the state vector.

2.1.5 Data Routing

Although a sensor network with a base station node can potentially offer better target detection and tracking than a high capacity single-sensor or an array-sensor system (i.e. radar), it introduces the additional challenge of constantly and efficiently communicating the target's state to the base location *using the structure and functionality of the network*. In this scenario, a *heterogenous network* with high-capacity nodes dedicated to *long-distance* routing and more complex processing, and lower-capacity nodes dedicated to sensing functions, can offer better performance, scalability, and an extended lifetime over a *homogeneous network* of equal-capacity nodes. The hierarchical routing [2] typical in these heterogeneous networks is characterized by the clustering of data on the high-capacity nodes, and the efficient relaying of fused data towards the base station. Energy-efficient target tracking, due to the generation of spatio-temporal state data, requires the continuous selection of temporary clustering nodes located in areas close to the current location of the target, and routing approaches that use minimal paths and are *proactive* (i.e. all routes are computed well in advance). The nodes of the hierarchical network assume *temporary allowed roles* and collaborate with each other to implement these efficient clustering and routing approaches.

A number of network-level hierarchical routing schemes for sensor networks have been proposed in recent years, having the common approach of selecting temporary or permanent packet-clustering nodes as a result of node collaborations whose goal is to minimize the overall energy consumption on the network while extracting usually aggregated data to destination nodes (i.e. base station). For example, the Low Energy Adaptive Clustering Hierarchy (LEACH) protocol [23] randomly selects temporary clustering nodes and aggregates data on those nodes for transmission to the base station. The work presented in [46] describes a Self-Organization Algorithm (SOA) in the context of a generic architecture of sensor, aggregator, router, and sink nodes; sensor nodes send their data to nodes designated as routers, which form the communication backbone for the entire network and act as data disseminators.

This thesis proposes a version for query-driven target tracking of the hierarchical architecture supporting the SOA described in [46]. It is also composed of sensor and router nodes, with the following additional capabilities: sensors form groups around a given router node and have a designated leader; only the leader sensor in each group communicates with a router node; local routing is performed between members of the local group; router nodes also make higher level decisions such as: designate leader sensors in its surrounding area, set a sensor group in standby or normal operation mode, maintain a complete track of a target and transfer it to subsequent router nodes when required. Communication between the different nodes of the network and the base station is carried out using *tree-based routing* (refer to section 2.4), set up when a tracking query issued by a user is disseminated to the network.

2.1.6 Energy-Aware Low-Overhead Tracking

Limited battery-based energy reserve is the basic limitation found on the nodes of sensor network deployments, creating the requirement of energy-saving schemes that take advantage of the specific characteristics of the sensed phenomenon and the network operation under those circumstances. In the case of a mobile target, these schemes can build up their efficiency on the *localized nature* of the spatio-temporal data generated by the dynamic behaviour of the target, requiring that only the nodes in the target's proximity be fully functional to implement the tracking operation. If the network is to be designed for extended lifetime operation, it is essential that sensor nodes be able to function in multiple energy consumption modes: an *active* mode defined by normal energy level usages and full communication/processing capabilities, as well as one or more *standby* modes characterized by disabled communication and low processing, and therefore very low energy consumption. Taking advantage of the geographic dispersion and distributed processing existing in a sensor network, these nodes capable of operating at different energy levels collaborate and self-organize in a distributed manner to maintain *active* a selected group of them around the current position of the target. Once this active group is formed, a *leader* node is either *elected* by the members of the group or *designated* by some higher level node.

Both activities - *sensor grouping* and *leader election* - have as a main purpose the implementation of low-energy schemes for target detection and tracking by collaborating in signal processing, data fusion and communication tasks. In this scheme, signal processing algorithms are implemented on low-power hardware existing on sensor nodes, and allow the efficient estimating and reporting of suitable values to other members of the active sensor group or higher-level nodes. Data fusion at the sensor group level minimizes the amount of data reported to the base station. Multi-hop communication and efficient routing techniques minimize the energy consumed in reporting the state of the target, generated by the active sensor group, to the base station.

Although this idea of sensor group activation/deactivation and collaboration within the group is ubiquitous in target tracking research, very little has been said about the required number and configuration scheme of the sensors forming the active group, the syncronization of the active region with the target's current location, and the effective mechanisms for handing over the sensor region *activation token*. The *Hierarchical Focus of Attention Tracking* algorithm, proposed in this thesis, addresses these latter issues.

2.2 Target Dynamics and Tracking

The common approach in tracking a mobile target with a sensor network is to model it as a point moving in a convex sensor field on a *smooth trajectory* on the average, with constant velocity or acceleration, subject to random changes in these parameters within allowed limits. Sampling the state of the target at a given rate allows the modeling of its dynamic behaviour as a discrete, linear system, affected by additive noise [4, 6]. These constraints set the boundaries for the quality of service (i.e. tracking fidelity, response latency) that can be expected from a given tracking scenario and provide useful guidelines for modeling, designing and implementing a system which performs well most of the time. The central idea is to design a sensor network which handles well situations that are absolutely prevalent, with additional capabilities to perform acceptably well in extreme situations (i.e. target moving with steep acceleration or random acceleration) that occur within acceptable limits, and finally provide exception handling capabilities when components fail or limit conditions are exceeded. In this thesis, it is assumed that the state of the target will be sampled at a constant and sufficiently high rate, with a time period T, to estimate the velocity of the target as a *constant value* between sampling times t_{n-1} and $t_n = t_{n-1} + T$.

2.2.1 Models of Target Dynamics

When sampling the state of a target moving with a constant velocity, the *deterministic* behaviour exposed in this scenario can be represented and analyzed in straightforwad terms by the following discrete linear system of equations, known as the *system dynamics model* [6], relating the states at times $n-1$ and n ($n = 1, 2, ...$):

$$\vec{S}_n = \Phi \vec{S}_{n-1} \tag{2.1}$$

where: $\Phi = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ (State transition matrix), T: Sampling period

and: $\vec{S}_n = \begin{bmatrix} x_n \\ y_n \\ \dot{x}_n \\ \dot{y}_n \end{bmatrix}$ (State at time t_n), $\vec{S}_{n-1} = \begin{bmatrix} x_{n-1} \\ y_{n-1} \\ \dot{x}_{n-1} \\ \dot{y}_{n-1} \end{bmatrix}$ (State at time t_{n-1})

However, in more realistic situations, the target will define a trajectory affected by some degree of *uncertainty*, as it will likely experience unforeseen dynamic changes in velocity or acceleration not anticipated in a constant velocity model. In this case, a more suitable approach to describe its behaviour is to use a *constant-velocity trajectory* model with a *random-walk velocity*. In this model, the velocity of the target is considered constant, during the elapsed time T between two consecutive sampling times t_{n-1} and t_n, and equal to the value estimated at t_{n-1}, which is updated by a random increment $\vec{V}_{n-1}$ at time t_{n-1} as shown in the following equation:

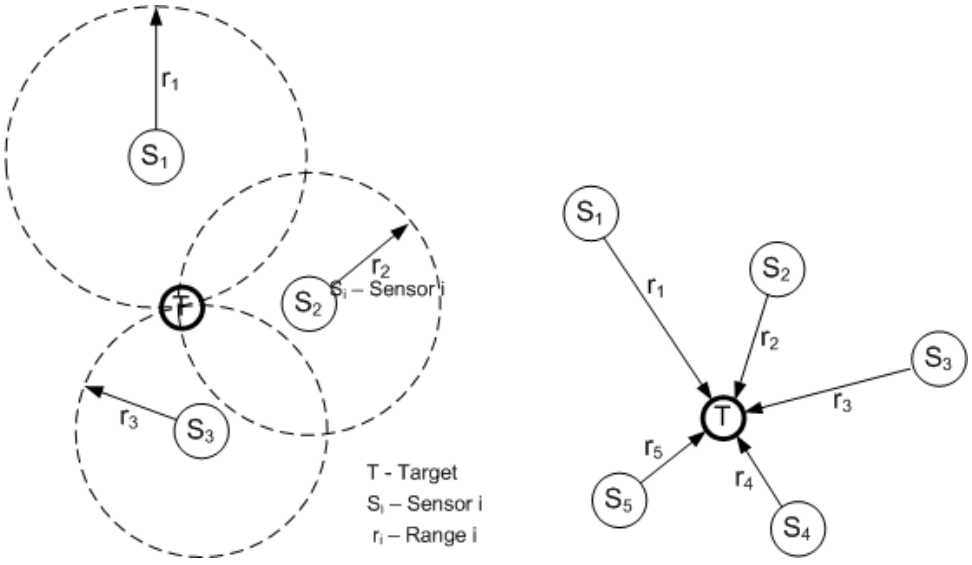
$$\vec{S}_n = \Phi \vec{S}_{n-1} + \vec{V}_{n-1} \tag{2.2}$$

where: $\vec{V}_{n-1} = \begin{bmatrix} 0 \\ 0 \\ \nu_{x_{n-1}} \\ \nu_{y_{n-1}} \end{bmatrix}$ (Dynamic model driving noise vector)
represents a random change in velocity at time t_{n-1}.

Other system dynamic models have been studied in the literature, such as the constant acceleration model [6]. However, they do not represent the main stream of target dynamic situations. Therefore, this thesis uses the constant-velocity with random-walk velocity model as the base for the analysis and design of the query-based, sensor network-based target tracking.

2.2.2 Collaborative Target Detection and Localization

Ranging sensors represent a mature, cost-effective, and efficient technology to detect targets, making them a natural choice for use in sensor networks. Basically, they measure the distance to a target without giving an indication of the direction where it can be found, or they set an indication of the presence of a target within a certain range. In the opposite end, *Angle-Of-Arrival* (AoA) sensors detect the *Line of Bearing* (LOB) and provide a measurement of the angle formed by the direction where the target is found and a reference direction, suggesting that perhaps a combination of ranging and AoA sensors would produce all the values needed to determine the position of a target. However, AoA sensors require very complex hardware implementations, usually involving multiple antennae, making them an ineffective and costly solution for wireless sensor networks. In this situation, the natural alternative is the use of ranging sensors only, having multiple sensors detect the same target and collaborate with designated sensors to completely resolve the localization of a target.



(a) Hyperbolic Trilateration (b) Multilateration

Figure 2.1: Target Localization Using Measurements from Multiple Sensors

Hyperbolic Trilateration – In this target localization approach, at least three sensors can detect the target simultaneously, and can communicate their range measurements r_i and own location coordinates (x_i, y_i) to a processing node. The latter node computes the intersection of three circles (represented by the system of equations 2.3), each one centered at one of the (x_i, y_i) sensor locations, to determine the coordinates (x_t, y_t) of the target. The computing node can use an adaptive algorithm to select the best three ranging measurements in a situation where these estimates are received simultaneously from more than three sensors. It can also implement some degree of interpolation in some instances where measurements from less than three sensors are received, but a complete set of at least three rangings has been received in the previous instance.

$$\begin{aligned} (x - x_1)^2 + (y - y_1)^2 &= r_1^2 \\ (x - x_2)^2 + (y - y_2)^2 &= r_2^2 \\ (x - x_3)^2 + (y - y_2)^2 &= r_3^2 \end{aligned} \tag{2.3}$$

Multilateration – Applied to target tracking, this technique can be considered a generalization of the trilateration localization method, again with n sensors ($n \geq 3$) sending their range measurements r_i to a processing node. However, in this case the processing node maintains range estimates $\hat{r}_i$ from the current location of the target to the sensors providing range measurements, and estimates the new coordinates (x_u, y_u) for the target as those resulting from minimizing the coordinate differences of target locations computed from measured and estimated ranges respectively (expression 2.4), for all n measured ranges incorporated into the solution.

$$min\sqrt{x_i - x_u)^2 + (y_i - y_u)^2} \qquad \forall \hat{r}_i \; ; \quad i = 0, ..., n \tag{2.4}$$

This thesis uses hyperbolic trilateration for target localization algorithms, based on its much lighter computing requirements compared to multilateration.

2.2.3 Trajectory Estimation and Prediction

As described in section 2.1.4, a sensor network implements the target tracking by building a *track*, consisting of discretized values corresponding to a history of measurements, current measurements, and future estimates of the state of the target. Theoretically, a track could be established on any high performance node of the network. However, the existence of energy-restricted nodes in a sensor network makes its implementation on an energy-unrestricted node (i.e. base station) a more viable solution, due to the more computing and communication intensive operations required to maintain it. Modeling the dynamic of the target's state as a discrete linear system (section 2.2.1) subject to noise allows the use of a signal smoothing and interpolation *mathematical filter* for estimating current and predicting future states. Among these tools, the Kalman filter is one of the most widely used for its efficiency, stability, and its ability to track past state history and estimate present/future states in a recursive manner (i.e. estimation of t_n-values based only on t_{n-1}-values) [6, 9].

This work uses a *discrete Kalman filter*, implemented on the base station, to maintain a track for a single target, which is updated with target location data measurements extracted/fused by sensor and router nodes respectively. The filter provides a dynamic overall *estimate* of the next state of the target $\vec{\hat{S}}(t_n) = (\hat{x}(t_n), \hat{y}(t_n), \hat{\dot{x}}(t_n), \hat{\dot{y}}(t_n))$ using a two-step process:

- *Predict* $\vec{\tilde{S}}(t_n)$ using the overall value estimated for $\vec{\hat{S}}(t_{n-1})$ and the *system dynamics equation* (equation 2.2). This is called the *a priori* state estimate.
- *Correct* $\tilde{\hat{S}}(t_n)$ using the target location measurement at time t_n and a system of *filtering* equations (table 2.2) to produce the overall *a posteriori* state estimate $\vec{\hat{S}}(t_n)$ for time t_n.

In the final step for the current iteration, the base station communicates the location portion of the overall estimate $\vec{\hat{S}}(t_n)$ back to the active sensor group currently providing target position measurements. This sensor group uses this item to implement the group activation/deactivation steps of the Hierarchical Focus of Attention tracking algorithm proposed in this work, and to decide whether it is necessary to send a new target location measurement to the base station for time t_{n+1} in the case where *measured* and *estimated* values disagree.

Tracking with the Kalman Filter – To apply a discrete version of the Kalman filter in a scenario where a target moves in the two-dimensional convex plane of a sensor network, with a dynamic behaviour as described by equation 2.2, it is assumed that the network detects the target's location (x, y) at discrete times t_n $(n = 1, 2, ...)$, generating values for these coordinates that are subject to measurement error. Additionally, it is assumed that the base station *estimates* the velocity of the target directly from the (x, y) coordinate values provided by the network, and that the following *Observation System Equation* describes the

relation between the measured values and the value of the state at time t_n:

$$\vec{Z}_n = H\vec{S}_n + \vec{N}_n \tag{2.5}$$

where:
$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$ (Observation matrix),
and:
$\vec{Z}_n = \begin{bmatrix} z_{x_n} \\ z_{y_n} \end{bmatrix}$ (Measurement vector), $\vec{N}_n = \begin{bmatrix} \eta_{x_n} \\ \eta_{y_n} \end{bmatrix}$ (Observation error vector)

Using a discrete Kalman filter in target tracking provides a recursive solution to estimate the state of the target at any time t_n, based on the previous state estimated at time t_{n-1}, the estimated error covariance of the state at time t_{n-1}, and the target location measured at time t_n. The equations defining the filter (tables 2.1 and 2.2) are recursively evaluated to implement the following two-step algorithm for estimating the value of the state at any given sampling time t_n:

1. Provide an *a priori prediction* $\vec{\tilde{S}}_n$ for the value of the state at t_n, by obtaining it from the system dynamics equation (equation 2.6). Also, obtain an a priori evaluation of the *estimation error covariance* matrix $\tilde{P}_n$ (equation 2.7) from an a posteriori evaluation of the same covariance matrix and the *system dynamics noise covariance* matrix $Q = \mathbb{E}[\vec{V}_n\vec{V}_n^T]$ at time t_n, where $\nu_{\dot{x}_n}$ and $\nu_{\dot{x}_n}$ are *Gaussian* noise variables with zero mean and variance σ^2 (i.e. $\nu_{x_n} \backsim N(0, {\sigma_{\nu_x}}^2)$ and $\nu_{y_n} \backsim N(0, {\sigma_{\nu_y}}^2)$); $\mathbb{E}[\vec{V}_n\vec{V}_n^T]$ is the expectation of matrix $[\vec{V}_n\vec{V}_n^T]$.

2. *Correct the a priori prediction* $\vec{\tilde{S}}_n$ to obtain an *a posteriori estimation* $\vec{\hat{S}}_n$ of the state at time t_n by evaluating the filtering equation 2.9, based on the measurement vector $\vec{Z}_n$ obtained from the sensor network and the Kalman Gain matrix K_n calculated from equation 2.8. In this latter equation, H is the *observation* matrix (equation 2.5),

$R = \mathbb{E}[\vec{N}_n \vec{N}_n^T]$ is the *measurement noise covariance* matrix, where $\eta_{x_n} \backsim N(0, {\sigma_{\eta_x}}^2)$, and $\eta_{y_n} \backsim N(0, {\sigma_{\eta_y}}^2)$ (equation 2.5); $\mathbb{E}[\vec{N}_n \vec{N}_n^T]$ is the expectation of matrix $[\vec{N}_n \vec{N}_n^T]$. Finally, obtain an a posteriori evaluation of the error covariance matrix $\hat{P}_n$ for time t_n (equation 2.10; I is the identity matrix). $\tilde{P}_n$ is the estimation error covariance matrix calculated in step 1.

Table 2.1: Time Update Equations – *Prediction* Step (time t_n)

A priori state estimate:

$$\vec{\tilde{S}}_n = \Phi \vec{\hat{S}}_{n-1} \tag{2.6}$$

A priori estimation error covariance matrix:

$$\tilde{P}_n = \Phi \hat{P}_{n-1} \Phi^T + Q \tag{2.7}$$

where: $\vec{e}_n = \vec{Z}_n - \vec{\tilde{S}}_n$ (Estimation error); $\hat{P}_n = \mathbb{E}[\vec{e}_n \vec{e}_n^T]$ (Expectation of matrix $[\vec{e}_n \vec{e}_n^T]$) and: $\vec{e}_n \backsim N(0, {\sigma_e}^2)$

Table 2.2: Measurement Update Equations – *Correction* Step (time t_n)

Kalman Gain:

$$K_n = \tilde{P}_n H^T (H \tilde{P}_n H^T + R)^{-1} \tag{2.8}$$

A posteriori state estimation (Kalman Filter equation):

$$\vec{\hat{S}}_n = \vec{\tilde{S}}_n + K_n (\vec{Z}_n - H \vec{\tilde{S}}_n) \tag{2.9}$$

A posteriori estimation error covariance matrix:

$$\hat{P}_n = (I - K_n H) \tilde{P}_n \tag{2.10}$$

2.3 Sensor Network Programming Models

The spatial and logical distribution, as well as the limited energy and computational resources present in the nodes of a sensor network, make programming this type of system a very challenging task. In this environment, the following basic requirements must be taken into consideration when designing efficient algorithms and programming approaches [22, 47] to extend the availability and lifetime of the network:

- *Energy-efficient* data extraction and node management approaches requiring minimal energy consumption.
- *Scalable* communication and node collaboration schemes to allow the cooperative work of large numbers of sensors without noticeable degradation.
- *Fault-tolerant* behaviour of the network nodes to support high availability of the services provided by the network even in the presence of communication link or node failures.
- *Collaborative* data collection and information processing schemes to reduce data loads to be communicated throughout the network and the overall energy usage.

Additionally, it is common to find very important application-specific *Quality of Service* (QoS) and operational requirements, such as the ones defined for the target tracking sensor network proposed in this work.

Under these circumstances, it is difficult for a single programming model to handle such a wide variety of requirements efficiently, hence a number of different models, operating at different levels of abstraction [43], have been proposed as more focused solutions. At each level of abstraction, these models define a number of component elements, a messaging system to allow the communication between these components, and a programming language

composed of a set of relatively simple primitives that allow designers to build more complex applications. The key idea is to build services at a lower level of abstraction that would simplify the sensor network programming at a higher level of abstraction. In line with this modeling approach, this thesis addresses the single-target tracking problem using a very high-level approach based on a declarative query language, similar to the ones used in the *database approach* for sensor networks, with lower-level algorithms for target detection and tracking, data fusion and routing, and reduced energy consumption.

The following sections present a brief view of these models, based on a taxonomy introduced in [47] to survey the more visible programming models found in the literature.

2.3.1 Node-Level Models

Node-level models provide abstractions of components that can interact directly with the hardware of a sensor node or implement functions that are more node-specific.

A first type of these models is the combination of the sensor node's operating system and a low-level language providing programming primitives to access specific hardware devices with the goal of building interacting *modules*. The TinyOS [27] operating system is a typical example of this class with its component-based structure and a language (i.e. nesC) that allows modular programming. In TinyOS, higher-level components issue *commands* to interact with lower-level components, and the latter components trigger *events* to interact with their higher-level counterparts.

Virtual machines is a slightly higher level version of node programming characterized by the implementation of a platform-independent programming environment oriented to specific types of sensor applications [26]. These type of models support the dynamic loading and execution of new programming code while nodes are in operation, as well as the development of platform-independent execution models.

2.3.2 Group-Level Models

Group-level models provide a more abstract macrolanguage [37] and a *set of operations* to allow programming the behaviour of a group of nodes denominated *abstract regions* [51]. The group can be formed based on physical proximity: nodes within N hops of each other, nodes in a spanning tree, or nodes having a common property: sensing the same variable, using the same sensing mode, etc.. The abstract operators trigger some behaviour of the group and return some quality metric or a value when they finish their execution. *Neighbour discovery*, *node enumeration*, *multiple node data sharing*, *data reduction*, *data fusion*, and other application-specific *node collaborations* are examples of these operators. These operators aim to hide from the programmer the sometimes complex details of executing the operations previously described, allowing at the same time some flexibility and ease-of-use to build sensor network applications.

2.3.3 Network-Level Models

Network-level models present a very high level view of the sensor system where instructions, in the form of a very high level macrolanguage or declarative query, are issued to the network in order to obtain some specific types of results. The user of the system specifies the nature and desired quality of the results to be obtained from the sensor network, without having to indicate any details of how this system will generate these results. The underlying assumptions are that: 1) the *declarative language* used to assign the required tasks to the sensor network is *complete* in the sense that it allows users to obtain the results they need, and 2) the network has been programmed accordingly with the appropriate *degree of adaptability and performance* to provide those results. The *database view* of the sensor network and the macroprogramming environment are two main programming paradigms based on network-level models.

The database approach visualizes the network as a relational database composed of data tables that can be queried using a declarative query language like SQL (Structured Query Language); TinyDB [32] is the typical example of this approach. The network is programmed to support the implementation of the operators normally found in a relational query language: *Selection*, *Projection*, *Aggregation*, etc., and to provide the data structures and data flows that would satisfy the requirements of any valid query.

Macroprogramming at the network level is mainly used to extend or complement the sensor network functionality that is available to users through declaratives languages in more flexible and complete ways. This approach allows the programming of specialized global network behaviours by reusing lower-level pre-coded functionality and making available to the programmer a set of high level primitives for specifying network behaviour [36].

2.4 Query Processing and Execution

As described in section 2.3.3, using a querying approach to collect data from a sensor network establishes an environment where users issue declarative statements describing the logical nature of the data sets requested and the high-level operations to perform on them rather than specify the algorithms to execute on the different nodes of the network to satisfy those requests. In a typical scenario, users start the operational cycle by issuing a *declarative* query at the base station specifying: the nature of the data extraction (i.e. *Track Target*, *Select Data*, etc.), the physical properties sensed and measured, any data aggregation operations, any data tranformation operations, the duration of the query, and the *sampling period* or *epoch* for the sensed data [18, 52].

This thesis introduces a framework and the distributed algorithms necessary to allow a user to issue a persistent query to track the velocity and location of a single mobile

target, with an assumed *dynamics*, generating low-overhead query responses in an energy-aware manner. The base station compiles the query and disseminates it to the network by triggering a distributed *wave processing* algorithm [42] that configures the sensor network, which is composed of router and sensor nodes, for improved stream data extraction. Router-router, router-sensor, and sensor-sensor collaboration algorithms build and report to the base station the data stream representing the real-time state of the target, effecting small delays and low energy-consumption.

Query Compilation – As in standard database systems, this step involves parsing the query and checking its validity, but more important, it involves determining a good *execution plan* for the query and the basic data structures that will support it. In sensor networks, given their distributed nature and the limited resources of their nodes, this query optimization task is usually completed at the base station. In order to complete this task successfully, the base station must have direct or indirect knowledge about the architecture, current topology, and current state of the network. The nodes of the network can implement further optimizations of the execution plan after they receive their *local version* of the execution plan, based on application-specific rules. They can also dynamically adjust the plan based on local conditions.

Query Dissemination – Typically, the base station compiles the query and *disseminates* to the different nodes of the network a set of translated lower-level instructions indicating the data needed from them, the software modules they have to execute and the roles they have to assume. During this query dissemination stage, it is a common approach to establish the communication topology for the execution of the query as a *tree-based* routing [18] structure. In this scheme, the base station is the root of the tree, and for data dissemination, nodes closer to the root, named *ancestors*, communicate with their immediate *descendants*, which are one more step away from the root. Communication from descendant to ancestor nodes

implements the data flow from the network to the base station.

Query Execution – The next stage, *query execution*, starts immediately after the instructions for the query are disseminated to all the network nodes that should be involved in building a continuous *data stream* representing the query result values. The query executes in a *persistent* mode, returning sampled data, usually at a constant rate, until its expiration time occurs or until it is terminated by the user. In the case of a *target-tracking query*, the data stream will return data only if a target is detected in the sensor field. However, the query will persist until its expiration time or cancellation.

Chapter 3

Sensor Network Architecture

This chapter describes the architecture, components and major tasks of the hierarchical sensor network proposed in this thesis (figure 1.1).

3.1 Core Assumptions

Tracking a mobile target with a wireless sensor network in an efficient and accurate way requires setting up certain basic properties of the nodes of this system such as location, connectivity, and sensing coverage, in a way that can provide a low-overhead detection response, uninterrupted operation and extended network lifetime. Given the complexity and versatility of this system, the solution space for this problem can be very large, and the search for absolute optimal solutions can be a daunting if not intractable task. A clearly more effective approach in this situation is to use some heuristic rules (section 1.1) to establish a set of basic assumptions about the deployment, structure, and node behaviour of the network proposed as a solution. This section expands the initial assumptions provided in section 1.1 (i.e. two-tier architecture, equal sensor nodes (S-nodes), omnidirectional sensors, equal router nodes (R-nodes), nodes can operate in active and stand-by mode, etc.).

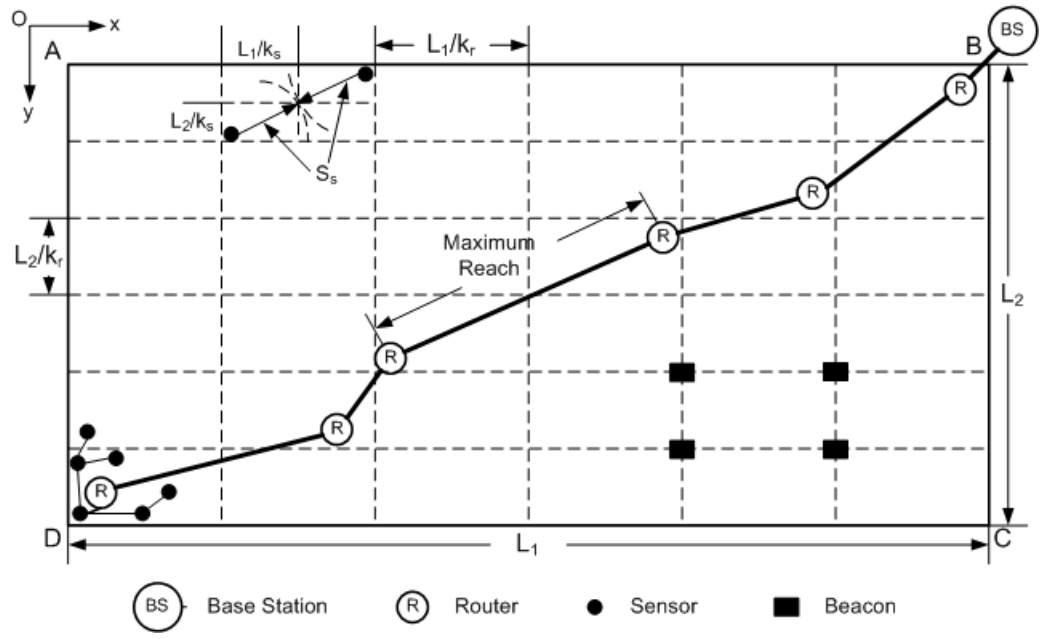


Figure 3.1: Node Deployment Grid on Sensor Field

3.1.1 Deployment Assumptions

For node deployment purposes, this work assumes that the rectangular *sensor field* $\mathcal{F}$ with lenght L_1 and width L_2 (section 1.1) can be subdivided into an imaginary grid of $N = k^2$ cells $SC_{i,j}$ of size $(L_1/k)(L_2/k)$ $(i, j, k \in \mathbb{N},\ k >> 1;\ r \in \mathbb{R})$, where k, the number of subdivisions per dimension, is a parameter that can be adjusted for each *node type* and desired *node density* for the network. It is also assumed that given the physical dimensions L_1 and L_2 of the sensor field, there are no restrictions on the total numbers of S-nodes T_S and R-nodes T_R available for deployment. With these assumptions, the deployment rules for sensor and router nodes can be established as follows:

Sensor Node Deployment Rule – *At least one S-node* is deployed anywhere in the area corresponding to a cell $SC_{i,j}$, for all (i, j) pairs $(i, j = 1, ..., N)$, generated with a k_s number of grid subdivisions per dimension. Since all S-nodes have the same sensing radius

S_s, in order to provide complete sensing coverage of field $\mathcal{F}$, S_s must be greater or equal than the diagonal distance $d = \sqrt{(L_1/k_s)^2 + (L_2/k_s)^2}$ in cell $SC_{i,j}$ (figure 3.1). Therefore, for a given set of S_s, L_1, and L_2 design parameters:

$$k_s \geq \frac{\sqrt{L_1^2 + L_2^2}}{S_s} \tag{3.1}$$

In order to avoid unnecessary overlapping of the sensing task it is assumed that, by design, $S_c \geq S_s$, where S_c is the S-node's communication radius. Given the previously specified rule for sensor deployment, this design condition ensures that the network will be a *connected graph* from the communication perspective. It also ensures a minimum sensor density of 1 $[sensor/cell]$, to provide lower bounds in sensing coverage and node connectivity, and at the same time allow some degree of randomness in deploying sensors.

Router Node Deployment Rule – Similar to sensor deployment, *at least one R-node* is deployed anywhere in the area corresponding to cell $RC_{i,j}$, for all (i, j) pairs $(i, j = 1, ..., M)$, generated with a k_r number of grid subdivisions per dimension. However, in this case the selection of k_r takes into consideration the location of the base station (BS-node). The longest path along a *routing tree* between an R-node, linked to an S-node group detecting a target, and the BS-node, hence, the path with potentially highest latency and communication power consumption, occurs when the latter node is located on one of the corners of the sensor field, and the former one is close to the opposite corner (figure 3.1). In this situation, the connecting path whose trace is closest to the S-node field's diagonal line, consists of the least number of hops and is the most efficient in reducing the latency of the tracking response. Therefore, given the communication radius R_c of an R-node, an initial heuristic approximation is to set k_r to a value which would ensure that two R-nodes deployed along this path in opposite corners of two diagonally connected cells are *immediate neighbours* (i.e. Maximum Reach in figure 3.1). This condition is met when

$R_c \geq 2\sqrt{(L_1/k_r)^2 + (L_2/k_r)^2}$, which determines k_r as:

$$k_r \geq 2\frac{\sqrt{L_1^2 + L_2^2}}{R_c} \tag{3.2}$$

In addition, this work assumes the following exponential model for the energy cost of transmitting data over the link connecting two adjacent R-nodes:

$$E_c = \beta * d^{\gamma} \tag{3.3}$$

where: β=overhead per bit, γ=path-loss exponent ($2 \leq \gamma \leq 4$)

Based on this exponential model, the deployment rule for R-nodes indirectly establishes an approximate upper bound for the energy consumed in communicating data between a target detection focal point and the base station, as paths involving a slightly bigger number of hops between the same data *source* and destination nodes will use less energy overall.

Node Localization Assumptions – In order to obtain an origin for a local system of coordinates, it is assumed that the geographical location of the BS-node in reference to a more universal inertial system (i.e. Global Positioning System (GPS)) is known, and this node is situated, for example, over side $\overline{AB}$ of the sensor field (figure 3.1), at distances d_1 and d_2 of vertices A and B, respectively. With this information, the BS-node can determine the geographical location for the origin O of a local system of coordinates located on vertex A, with x, y axis over sides $\overline{AB}$ and $\overline{AC}$, respectively. Finally, the BS-node can determine its own coordinates in this local system of coordinates as (x_{bs},y_{bs}).

All the other nodes in the network determine their coordinates in reference to the local system. The localization scheme proposed in [10] is in place. *Beacon nodes*, with omnidirectional transmission capability, communication radius $B_c \gg \sqrt{(L_1/k_r)^2 + (L_2/k_r)^2}$ (to provide communication coverage over multiple cells of a router grid), and known unique identification keys B_p ($p = 1, ..., k_r^2$) are anchored at each intersection of the grid defined for

the deployment of routers (figure 3.1). These k_r^2 nodes broadcast their location coordinates (x_p, y_p) every T_b time units, using a collision avoidance schedule, to all the nodes within their transmission range. The m R-nodes and n S-nodes deployed listen to these messages for a period of time T_l ($T_l \gg T_b$), during which, each node receives $N_{rec}(p)$ of the total $N_{sent}(p)$ messages sent by beacon node p. Each node builds a set $C_i = \{B_q | q = 1, ...N_i\}$ $(i = 1, ..., (R+T))$ of *good reference nodes* by including in it only the beacon nodes for which $\frac{N_{rec}(p)}{N_{sent}(p)} > \beta$ (for example $\beta = 0.9$). Finally, each node i localizes itself to the *estimated location* $(\hat{X}_i, \hat{Y}_i)$ corresponding to the centroid calculated from the coordinates of all nodes in C_i:

$$(\hat{X}_i, \hat{Y}_i) = \left(\frac{1}{N_i} \sum_{q=1}^{N_i} X_q, \frac{1}{N_i} \sum_{q=1}^{N_i} Y_q \right) \tag{3.4}$$

3.1.2 Hardware and Operating System Assumptions

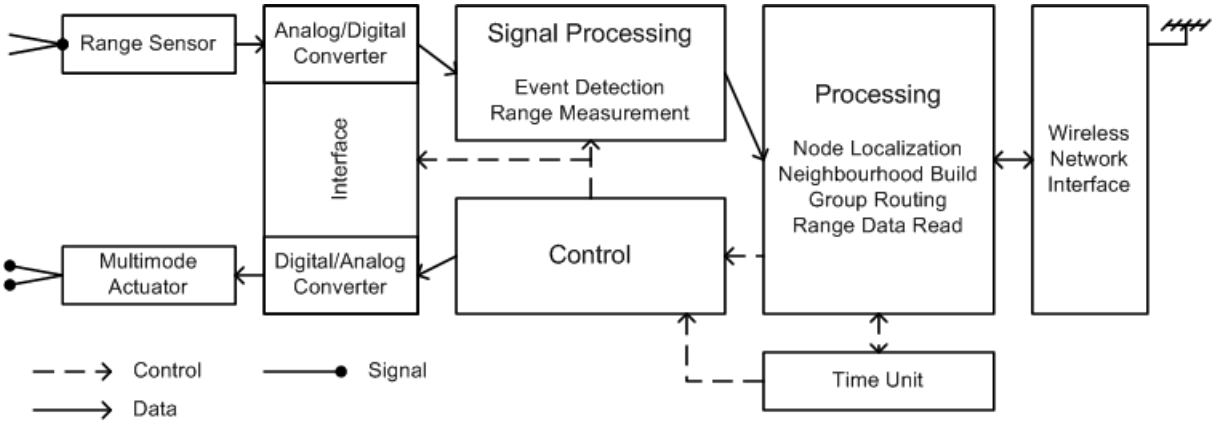


Figure 3.2: Sensor Node Hardware Architecture

Sensor Nodes – The hardware architecture of each S-node S_i $(i = 1, ..., S)$ (figure 3.2) supports the detection of the *range* $d_i(t_n)$ (i.e. euclidean distance between the sensor and a target at time t_n) for a *passive target* in its vicinity. The S-node emits a pulsating signal that is reflected by the target, and then detected (it is assumed omnidirectional sensing

modeled as a circle centered on the sensor), transduced and digitized by the *range sensor* and *analog/digital converter* units. Next, the *digital processing* unit analyzes the digital signal and transfers to the *processing* unit an *event detection* bit and the measured range for the detected target. If the event detection bit is set, the processing unit assembles a data message containing the 4-tuple $(x_i(t_n), y_i(t_n), d_i(t_n), t_n)$ ((x_i, y_i): sensor's position; d_i: range (i.e. sensor-to-target distance); t_n: time stamp attached to a range measurement) and loads it onto the *wireless network interface* for transmission to a node designated as a *leader sensor* (LS-node) in its vicinity, which, in turn, computes the target's location (x, y) (data fusion) and routes to the closest R-node. During normal operation, each sensor node continuously executes this sequence with a time period T_t, specified by the system designer, if enabled to complete this task by the *Control Unit*. All sensors keep the time on their *time unit* synchronized to a universal time server. All S-nodes support the execution of a standard sensor operating system (i.e. TinyOS [27]) which isolates the hardware from high-level programming tasks and provides the services for the execution of algorithms at a very abstract level.

Router Nodes – In comparison to an S-node, an R-node R_j $(j = 1, ..., R)$ has a reduced architecture composed of the processing, time, and wireless network interface units only (figure 3.2), as it does not perform target detection tasks. However, the processing unit, in addition to node localization, performs higher level processing tasks such as network routing, sensor group formation, designation of LS-nodes, and execution of the Hierarchical Focus of Attention tracking algorithm. Also, the wireles network interface has a communication radius $R_c >> S_c$ as determined in section 3.1.1, and $E_r >> E_s$ (E_r: router energy reserve; E_s: sensor energy reserve). R-nodes play a relaying-only role in the tasks of fusing and extracting target location data to the BS-node, receiving fused target range data from LS-nodes, represented by 3-tuples of type $(x(t_n), y(t_n), t_n)$ ((x, y): target's location; t_n: time stamp attached to a target location measurement), and routing them to this destination

without changing their content.

Base Station – This node is a server grade computer supporting a high performance operating system and an input/output graphical facility to allow user interaction.

3.1.3 Communications Assumptions

The BS-node, R-nodes, and S-nodes use the *multi-hop* communication approach, where two nodes that are not immediate neighbours communicate with each other over a path composed of *chained point-to-point links*. This work assumes that the communication protocol stack to support the error-free point-to-point transfer of *application messages* is a service available on all nodes of the network (i.e. implemented in firmware), and will include in its scope only *higher level interactions* between network nodes such as data routing and data fusion.

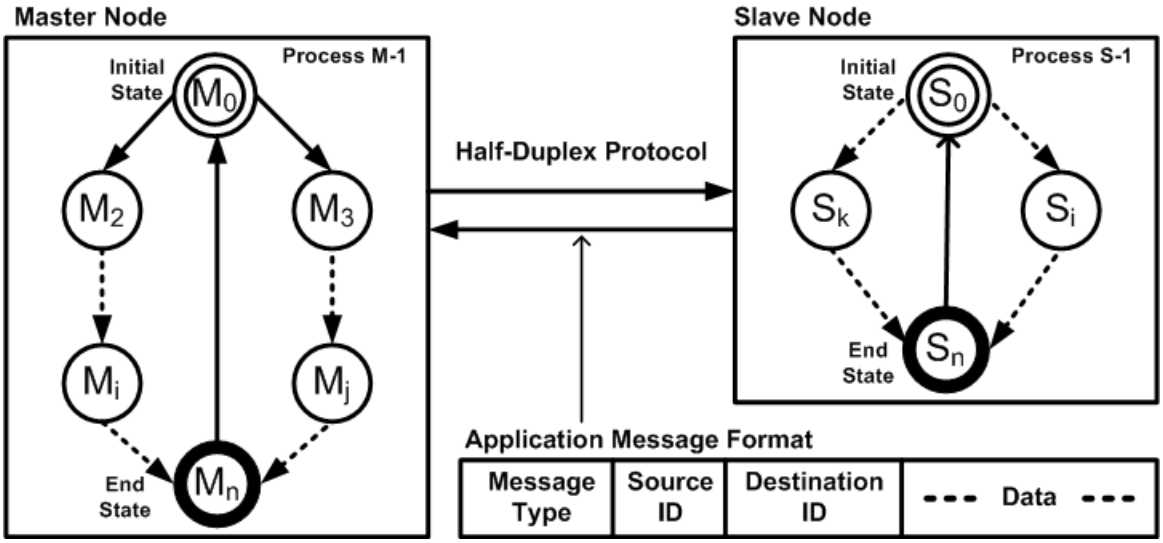


Figure 3.3: Node Collaboration General Model

Each interaction I_k is established by *cooperative processes* [31, 48], running on different nodes of the network, where each process is modeled as a *finite state automaton* with

initial and end states (figure 3.3). Each interaction follows the *master-slave* scheme, it is always initiated by the node acting as a master and involves a finite number of message exchanges in *half-duplex* mode. A set of sequential interactions I_k $(k = 1, ..., n)$ constitutes a collaboration $\mathbb{C}$ between two or more nodes. Table 3.1 describes the formatting and content of the messages exchanged in a collaboration.

Field	**Format**	**Values**
MessageType	1 bit	0 – control
		1 – data
SourceID, DestinationID	NodeType + (x, y)	NodeType: 0 – router, 1 – sensor, 2 – base station
		(x, y): node coordinates
DataField (MessageType = 0)	DataType + Data	DataType = 0 – target range
		DataType = 1 – target location
		DataType = 2 – router or sensor location
DataField (MessageType = 1)	ControlCode (indicates action requested)	0 – request node coordinates
		1 – assign leader sensor role
		2 – switch to stand-by mode
		3 – switch to operation mode

Table 3.1: Application Message Formatting

3.2 Sensor Network Model

The assumptions specified in previous sections established the physical and communication topology of a hierarchical network, deployed as a combination of R-nodes and S-nodes positioned, with a controlled degree of randomness (i.e. approximately uniform density), in a plane convex field (figure 1.1). Building upon this deployment, as well as the hardware and software abstractions specified for each node type, and concentrating on the specific requirements of the target tracking application, the following sections describe the main architectural elements and algorithms proposed in this thesis.

Applying the heuristics rules defined in section 1.1, this work proposes the general architecture depicted in figure 3.4 as a solution for the single-target tracking problem. The

network behaves as a set of distributed cooperative processes executing on the BS-node, R-nodes, LS-nodes, and S-nodes to extract the spatio-temporal data corresponding to the target's trajectory.

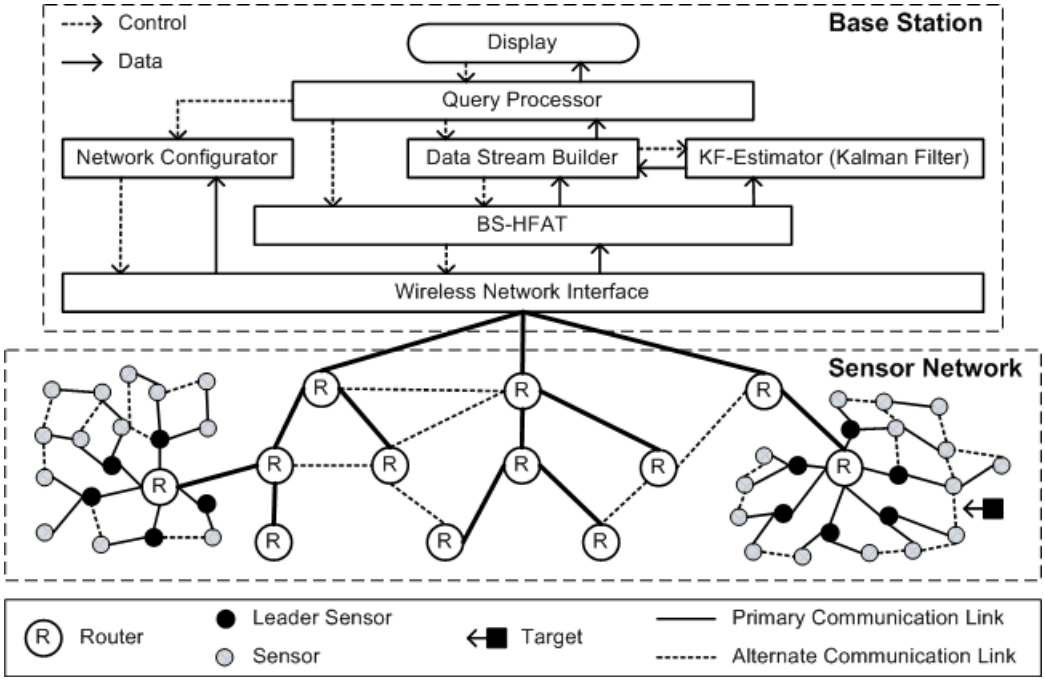


Figure 3.4: Target Tracking Network General Architecture

A user issues a tracking query on an interactive terminal at the BS-node to initiate all tracking operations. The *query processor* module accepts the query and triggers the execution of a *network configurator*, which in turn has the BS-node contact a given subset of the R-nodes detected as immediate neighbours. In general, R-nodes perform tasks that are triggered by the BS-node, or other R-node one link closer the BS-node. They configure their communication links and routing structures, using a *wave* configuration algorithm based on message exchanges, so that all communications towards the BS-node, from a specific area on the sensor field, are relayed over the branches of a tree structure rooted at this node (figure 3.4). When R-nodes are *terminal nodes* of this tree, they configure

the S-nodes located in their vicinity and report the completion of this process to its parent R-node. An R-node having all its children R-nodes configured, proceeds to configure the S-nodes in its vicinity and report the completion of this process to its parent R-node. This configuration wave continues until all the immediate neighbour R-nodes selected by the BS-node report a *configuration complete* message to this latter node, at which point the network is completely configured to start tracking a target.

In configuration mode, an R-node determines a subset of its immediate neighbours and designates them as LS-nodes based on remaining energy and relative location. Then, it initiates a *sensor configuration wave*, starting on each LS-node and based on message exchanges, which propagates up to k steps ($k = 1, 2, ...$) away from its initial point. At each step of the wave an S-node configures itself to belong to the group whose LS-node initiated the configuration wave, and identifies the S-node from whom it received the configuration wave as the designated *relay sensor* for communicating with its LS-node. Then, it finds all its immediate neighbour S-nodes, and selects one, based on remaining energy and relative location, as the *next node* in the k-step configuration wave if itself is not a k-step node. It designates its remaining neighbourss as *alternate* relay sensors. Finally, it propagates the configuration wave to the next node, and reports a configuration-complete message to its LS-node.

3.3 Target Tracking Query System

The querying scheme (section 1.2) makes use of a single command whose syntax allows entering specification parameters for tracking a single target. This data streaming query injects configuration parameters to the network, initiates a configuration wave of the entire network, and retrieves estimated data corresponding to the sampled state of the target: $(\hat{x}(t_n), \hat{y}(t_n), \hat{\dot{x}}(t_n), \hat{\dot{y}}(t_n))$. Its syntax is as follows:

Track Target

PathType = $[OpenPath|ClosedPath]$

Pattern = $[Random|Line(m,n)|Ellipse(x_0,y_0,a,b)]$

InitialLocation = $[Unknown|(x_0,y_0)]$

InitialVelocity = $[Unknown|(\dot{x}_0,\dot{y}_0)]$

SamplingRate = R

Epoch = $[Continuous|T_r]$

where:

- **PathType** – *Expected trajectory* of a target in the sensor field. Default value is *OpenPath*.
  - *OpenPath*: the target enters and exits the the sensor field at two different points, moving along a path that does not intersect with itself.
  - *ClosedPath*: the target enters the field and remains in it, moving along a loop-like path.
- **Pattern** – *Expected dynamic behaviour* of the target. Default value is *Random*.
  - *Random*: the target moves with *variable velocity* emulating a *random walk* (i.e. a *velocity change* Δv at time t_n is a *Gaussian* random variable such that: $\Delta v \backsim N(0,\sigma^2)$.
  - $Line(m,n)$: the target moves in a straight line whose equation is $y = mx + n$ in reference to the origin of coordinates for the sensor field.
  - $Ellipse(x_0,y_0,a,b)$: the target moves along an ellipse centered at (x_0,y_0) whose equation is:

$$\frac{(x-x_0)^2}{a^2} + \frac{(y-y_0)^2}{b^2} = 1 \tag{3.5}$$

- **InitialLocation** – *Expected location* where the target will be initially detected. Default value is *Unknown*.

  - *Unknown*: the initial position of the target is unknown and will be detected by the sensor network.
  - (x_0, y_0): initial location of the target.

- **InitialVelocity** – *Expected velocity* for the target when it is initially detected. Default value is *Unknown*.
  - *Unknown*: the initial velocity of the target is unknown and will be detected by the sensor network.
  - $(\dot{x}_0, \dot{y}_0)$: initial velocity of the target.

- **SamplingRate** – Number of samples per unit of time R taken of the target's state. The sampling period is $T = 1/R$. Default value is 1 $[sample/second]$.

- **Epoch** – Total running time specified for the query. Default value is *Continuous*.
  - *Continuous*: the query will run continuously until it is aborted.
  - T_r: running time specified for the query.

The *query processor* module assumes the default value for any parameter not specified in the query. Therefore, a **Track Target** query, without any parameters, expects to track a target moving along an *open path* which is a *random walk*, whose initial location and velocity are unknown and will be detected by the sensor network, and the target's state will be sampled continuously every 1 second.

Query Processing – It is assumed in this work that all queries issued to the network have a correct syntax and follow the semantic rules described in this section for the querying parameters. Therefore, query parsing and syntax checking are considered outside of this scope, and the query processing emphasis is placed on producing a *query execution plan* that

affects the network configuration process, the HFAT tracking algorithm, and the target's state estimator (table 3.2).

[a] Mapping of parameters to processes affected by query set up values.

Parameter	**Process Affected**		
	Network Configuration	**HFAT Algorithm**	**State Estimator**
PathType	routing tree, region shape	region activation	noise covariance matrices
Pattern	routing tree, region shape	region activation	noise covariance matrices
InitialLocation	N/A	region activation	track initialization
InitialVelocity	N/A	region activation	track initialization
SamplingRate	network re-configuration	sampling threshold	noise covariance matrices
Epoch	network re-configuration	sampling planning	N/A

Table 3.2: Parameter-Module Mapping in the Query Execution Plan

Section 4.2 describe the details of how the parameter values affect the items tabulated on table 3.2.

Query Execution – This stage involves the completion of main tasks executed in sequence:

1. *Network configuration* – The following items: data routing structures, LS-node designations, sensor regions formation, and state estimator configuration, are configured to obtain a low-overhead response with reduced energy usage in tracking a single target. The BS-node triggers a *configuration wave* that reaches the appropriate R-nodes and S-nodes. This wave is executed synchronously in the sense that it is started and controlled by the BS-node with the orderly participation of the network nodes. Overall, it represents the network setup for an *expected target dynamics* as specified by a user of the system.

2. *Scanning of the sensor field* for the presence of a mobile target for the specified running time of the query.

3. Tracking of a target, if one has been detected, and production of a *data stream* corresponding to its the sampled state. This output represents the answer to the query and the actual dynamic behaviour of the target.

Under certain exceptional situations, such as loosing track of a target, or detected imbalances in energy usage among the network nodes, or an excessive number of interrupted node-to-node links, the query will reset its operation to steps 1 or 2.

3.4 Base Station Architecture

The BS-node provides the display facility to visualize the estimated path of a mobile object in the monitored sensor field. It is assumed to be a multitasking wireless node with unlimited energy reserve and high-end processing capabilities. The following sections describe the architecture and operation of the modules supporting the single-target tracking operations at this node.

3.4.1 Query Processor

The *Query Processor* module plays the *supervisor* role for the entire target tracking operation, sequencing, triggering, and coordinating the execution of the other modules at the BS-node and the network nodes. Figure 3.5 and table 3.3 describe the global state diagam for this module.

It is assumed that this module operates in the context of an operating system and user interaction facility, and that, it initiates its operation at state Q_0 the first time it receives a user-issued tracking query from this environment. Thereafter, it cycles over the states as indicated in figure 3.5, returning the dynamic state $(x(t_n), y(t_n), \dot{x}(t_n), \dot{y}(t_n))$, estimated at time t_n, of a target in the sensor field, whenever one is detected.

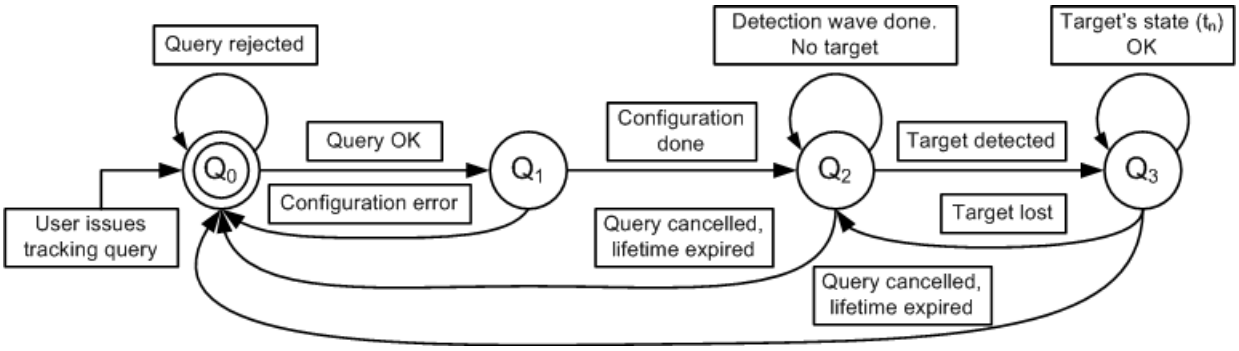


Figure 3.5: Query Processor – State Diagram

State Name	Description	Previous State	Transition Event
Q_0	**Tracking query input:** BS-node waits for user to issue a tracking query. Then compiles, processes, and triggers execution of query.	N/A	User issues tracking query.
		Q_0	Query rejected.
		Q_1	Network configuration error.
		Q_2	Query cancelled, or lifetime expired.
		Q_3	Query cancelled, or lifetime expired.
Q_1	**Sensor network configuration:** BS-node initiates and completes network configuration wave.	Q_0	Tracking query compiled succesfully.
Q_2	**Target detection:** Network is running a target detection processing wave.	Q_1	Network configuration completed normally.
		Q_2	Target detection processing wave completed. No target detected.
		Q_3	Sensor network has stopped detecting the target.
Q_3	**Target tracking:** BS-node continuously samples the sensor network to track a target.	Q_2	Sensor network detected a target.
		Q_3	Sensor network completed one detection/tracking cycle.

Table 3.3: Query Processor – States Description

3.4.2 Network Configurator - Base Station Processing

The *Network Configurator* module is called by the *Query Processor*, while this latter module is in state Q_1, to select a sub-set of the R-nodes that are immediate neighbours (i.e. *one-hop-router*) of the BS-node, based on distance and remaining energy (section 4.1.1), and supervises the routing configuration on those nodes for relaying tracking data. This module also supervises the configuration of regions occurring in the vicinity of selected

R-nodes. As stated in section 3.2, an R-node reports a *configuration complete* only after itself has designated the applicable LS-nodes, the regions in its vicinity are completely configured, and all the children R-nodes that it has selected have reported a *configuration complete* status. Starting at the BS-node, and continuing with their one-hop-routers and their descendants, this processing wave expands recursively until it reaches *terminal* R-nodes (i.e. R-nodes without children R-nodes). From this point, the *configuration complete* status progresses back through the R-nodes until it reaches the BS-node. At each step of this backtracking wave, each R-node reports to its parent R-node the coordinates of the vertices corresponding to the sensor region coverage polygon (figure 4.4) generated by the sensors under its supervision. Figure 3.6 and table 3.4 describe the global state diagram for this module, section 4.1.1 provides the algorithmic details.

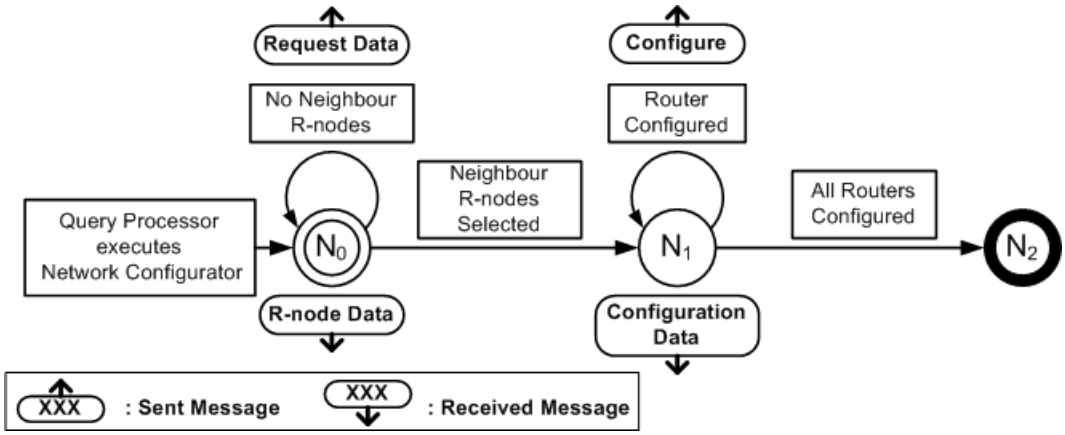


Figure 3.6: BS-node Network Configurator – State Diagram

State Name	Description	Previous State	Transition Event	Message
N_0	**Neighbour R-node search:** BS-node broadcasts *Request Data* message to all one-hop-routers.	N/A	Query Processor executes Network Configurator.	N/A
		N_0	No one-hop-router has responded to the *Request Data* broadcast message.	Send: *Request Data*
N_1	**Selected R-node configuration:** BS-node has received data from one-hop-routers, selects a subset of them, and sends them a *Configure* message to configure the routing and sensor regions.	N_0	BS-node selected a subset of one-hop-routers as children nodes.	Receive: *R-node Data*
		N_1	BS-node has received a *Configuration Data* message from a one-hop-routers.	Send: *Configure* Receive: *Configuration Data*
N_2	**Network configuration complete:** BS-node has received a *Configuration Data* message from all its one-hop-routers	N_1	All one-hop-routers configured.	N/A

Table 3.4: BS-node Network Configurator – States Description

3.4.3 HFAT Algorithm - Base Station Processing

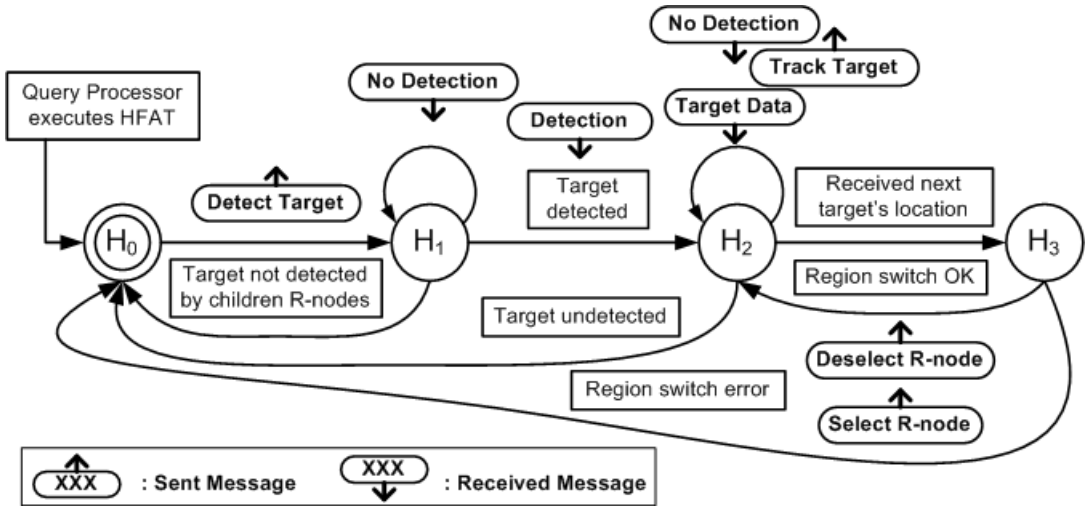


Figure 3.7: BS-node Hierarchical Focus of Attention Tracking – State Diagram

State Name	Description	Previous State	Transition Event	Message
H_0	**Undetected target:** BS-node has not received reports of a detected target from its children one-hop-routers.	N/A	Query Processor executes BS-HFAT.	N/A
		H_1	All one-hop-routers have responded with a *No Detection* message.	Receive: *No Detection*
		H_2	Target goes undetected during tracking.	Receive: *No Detection*
		H_3	Next tracking sensor region not determined or target out of sensor field.	N/A
H_1	**Detection wave:** BS-node has started a target detection processing wave.	H_0	BS-HFAT has broadcast a *Detect Target* message to all one-hop-routers	Send: *Detect Target*
		H_1	BS-HFAT has received a *No Detection* answer to the *Detect Target* broadcast.	Receive: *No Detection*
H_2	**Tracking:** BS-node is tracking the state data stream of a target detected in the sensor field.	H_1	BS-HFAT received a *Detection* message from a one-hop-router.	Receive: *Detection*
		H_2	1) BS-HFAT received a *Target Data* message from a one-hop-router. 2) BS-HFAT received a *Start Cycle* command from the Data Stream Builder.	Receive: *Target Data* Send: *Track Target*
		H_3	BS-HFAT has calculated a new active sensor region and re-selects one-hop-routers.	Send: *Select R-node* *Deselect R-node*
H_3	**Region switch:** BS-node determines the next active sensor region under a one-hop-router.	H_2	BS-HFAT received the *a priori target's location* for time t_{n+1} from the *Data Stream Builder*.	N/A

Table 3.5: BS-node Hierarchical Focus of Attention Tracking – States Description

The BS-HFAT (*BS-node – Hierarchical Focus of Attention Tracking* algorithm) module is called by the *Query Processor*, while this latter module is in state Q_2, to initiate a detection wave in the sensor field to locate a target, and once detected, maintain its associated data stream of state values $(x(t_n), y(t_n), \dot{x}(t_n), \dot{y}(t_n))$ estimated at discrete times t_n $(n = 1, 2, ...)$. This module broadcasts a *Detect Target* message to all its selected one-hop-routers immediately after it is initiated (state H_0), and waits for a detection response from them

(state H_1). If all of them respond with a *No Detection* message, the module switches back to its initial state and repeats the previous process. If one of them responds with a *Detection* message, the module sends back to it a *Select Router* message to make it the designated R-node to relay state data to the BS-node from the current location of the target, sends back to all the other one-hop-routers a *Deselect Router* message to restrain them from relaying state data to the BS-node, and switches to *tracking* the target (state H_2). If the module receives a *Target Data* message while being in state H_2, it relays the state data to the *KF-Estimator* module and remains in the same state.

3.4.4 Kalman Filter for Target's State

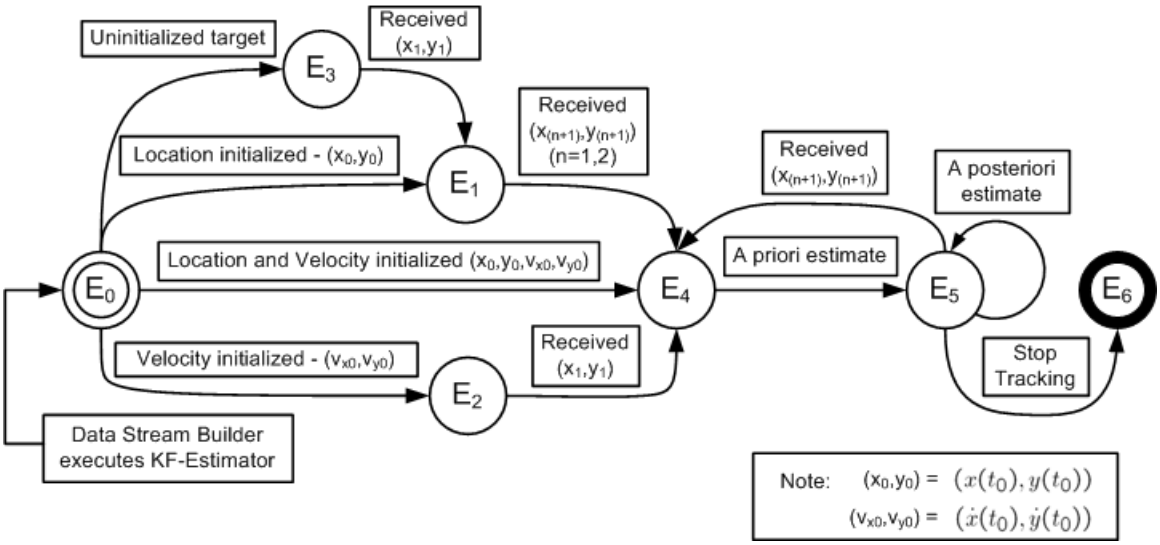


Figure 3.8: KF-Estimator of Target's State – State Diagram

The *Data Stream Builder* calls the *KF-Estimator* module when the *BS-HFAT* reports the *Tracking* state H_2. *KF-Estimator* implements a discrete Kalman Filter (section 2.2.3) to estimate the target's state $(x(t_n), y(t_n), \dot{x}(t_n), \dot{y}(t_n))$, where the target dynamics is assumed to be a *constant velocity* trajectory with random walk velocity changes at discrete times t_n

$(n = 1, 2, ...)$ (equation 2.2), and the sampling period T is a constant specified by the *Query Processor*.

State Name	Description	Prev. State	Transition Event	Processing
E_0	**Parameter input:** Read user-defined parameters.	N/A	Data Stream Builder executes KF-Estimator.	Read initial state of the target from Query Processor.
E_1	**Known target location:** Initial target's location is known either from the user or the network.	E_0	Initial user-defined location $(x(t_0), y(t_0)$ loaded.	N/A
		E_3	Initial network-detected location $(x(t_1), y(t_1))$ loaded.	N/A
E_2	**Known target velocity:** Initial target's velocity provided by the user.	E_0	Initial user-defined velocity $(\dot{x}(t_0), \dot{y}(t_0))$ loaded.	N/A
E_3	**Unknown target state:** Not initialized by the user	E_0	$(x(t_0), y(t_0)$ and $(\dot{x}(t_0), \dot{y}(t_0))$ not defined.	N/A
E_4	**Prediction:** A priori estimate of the target's state.	E_0	$(x(t_0), y(t_0)$ and $(\dot{x}(t_0), \dot{y}(t_0))$ defined.	N/A
		E_1	Network-detected location $(x(t_n), y(t_n))$ loaded. loaded.	$\dot{x}(t_n) = \frac{(x(t_n) - x(t_{n-1})}{T}$ $\dot{y}(t_n) = \frac{(y(t_n) - y(t_{n-1})}{T}$ $i = 1$ or $i = 2$
		E_2	Network-detected location $(x(t_1), y(t_1))$ loaded.	N/A
		E_5	Network-detected location $(x(t_{n+1}), y(t_{n+1}))$ loaded.	N/A
E_5	**Correction:** A posteriori estimate of the target's state.	E_4	KF-Estimator calculates a priori estimate of the target's state.	Equations in table 2.1
		E_5	KS-Estimator calculates A posteriori estimate of the target's state.	Equations in table 2.2
E_6	**Stop:** KS-Estimator is finished.	E_5	Data Stream Builder terminates KS-Estimator	N/A

Table 3.6: KF-Estimator – States Description

If the user provides, in the tracking query, initial values for the location and velocity of the target, the *KF-Estimator* module enters in the *filtering cycle* (states E_4 and E_5 in figure 3.8) for the target's state immediately, and waits for the next target's location $(x(t_{n+1}), y(t_{n+1}))$ to be supplied by the sensor network via the BS-HFAT module. If only an initial target

velocity is provided, the module waits until it has received a target location reading from the network (state E_2), entering the filtering cycle thereafter. If only an initial target location is provided (state E_1), after the first network reading, the module also estimates the initial velocity as specified in table 3.6 before entering the filtering cycle.

If no initial values for location and velocity are provided, the *KF-Estimator* module waits until it has received two consecutive target location readings from the network (states E_3, E_1), and estimates the initial velocity as indicated in table 3.6 before entering the filtering cycle. The *Data Stream Builder* stops and reinitializes this module whenever the tracking is interrupted either by a target going out of scope or undetected.

3.4.5 Base Station Data Stream Builder

The *Query Processor* calls module *Data Stream Builder* (DSB), after completing the network configuration phase or after an interruption of the data stream (i.e. module *Data Stream Builder* has timed out), to control the continuous flow of filtered target state values produced by module *KF-Estimator*.

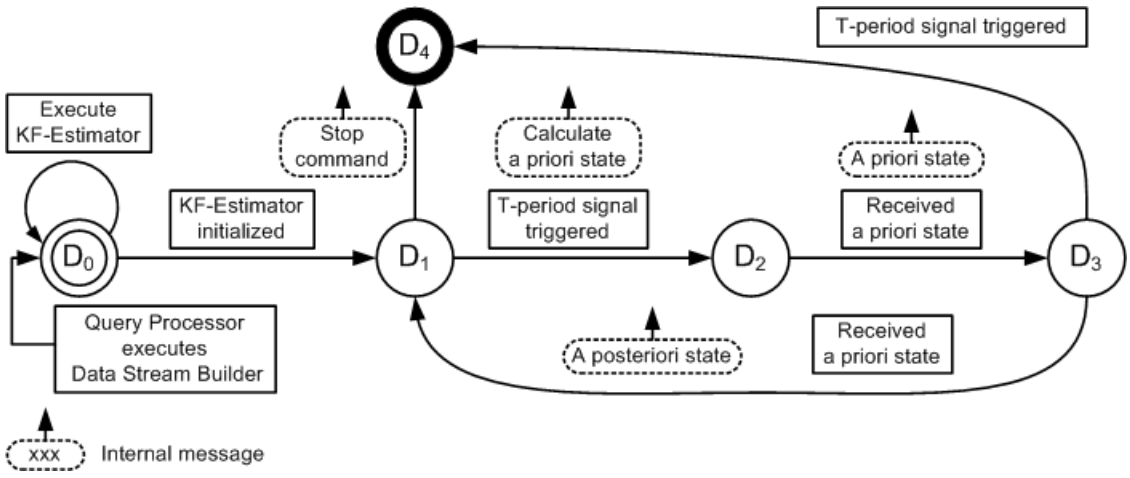


Figure 3.9: BS-node Data Stream Builder – State Diagram

State Name	Description	Previous State	Transition Event	Internal Message
D_0	**Initialization:** Read query parameters and execute KF-Estimator.	N/A	Query Processor executes Data Stream Builder (DSB)	N/A
		D_0	DSB executes KF-Estimator	N/A
D_1	**Tracking cycle start:** DSB waiting for *T-period signal* to start next tracking cycle.	D_0	*KF-Estimator* reported to DSB an *initialized* status.	Receive: *KF-Estimator initialized*
		D_3	DSB received estimate of the target's state from *KF-Estimator*.	Receive: *A posteriori estimate*
D_2	**A priori wait:** DSB is waiting for the *KF-Estimator* to send *a priori* estimate of the target's state.	D_1	*T-period signal* has been triggered.	Send: *A priori state request*
D_3	**A posteriori wait:** DSB is waiting for the *KF-Estimator* to send *a posteriori* estimate of the target's state.	D_2	*KF-Estimator* responds to DSB's request to estimate the *a priori* state of the target. DSB forwards estimate to BS-HFAT.	Receive: *A priori estimate* Send: *A posteriori state request*
D_4	**Stop:** DSB has stopped execution.	D_1	*Query Processor* terminates DSB.	Receive: *Stop* DSB
		D_3	*T-period signal* has been triggered. DSB reports timeout to *Query Processor*	Send: *Timeout*

Table 3.7: BS-node Data Stream Builder – States Description

Module DSB executes module *KF-Estimator* (state D_0) and waits until the latter module has processed the initial state $(x(t_0), y(t_0), y(\dot{t}_0), y(\dot{t}_0))$ of a detected target, at which point it receives a *KF-Estimator initialized* message. This event signals the initiation of a *track* for the target, and module DSB enters a *tracking loop* delimited by states D_1, D_2, and D_3, in which, each cycle is started when it receives a *T-period signal* from a timer internal to the BS-node.

In each tracking cycle, module DSB receives an *a priori* estimate of the target's state correponding to sampling time t_{n+1}, which feeds back to module BS-HFAT to support the activation/deactivation of sensor regions in the distributed HFAT algorithm, and an *a*

posteriori one, which transfers to the *Query Processor* as the end response to the tracking query issued by the user. Figure 3.9 and table 3.7 describe this module's state diagram.

3.5 Router Architecture

The architecture of the R-node (figure 3.10) provides structure to network configuration, data stream building, and HFAT algorithm functions similar to the ones defined for the BS-node, with additional support for the routing of data to and from the BS-node and the calculation of the target's location: $(x(t_n), y(t_n))$, at each sampling time t_n. The primary design goal for R-nodes is the support of an energy-aware communication and computation backbone which would extend the network's lifetime, therefore, they neither calculate the complete state of the target: $(x(t_n), y(t_n), \dot{x}(t_n), \dot{y}(t_n))$, nor filter measured data for improved estimation.

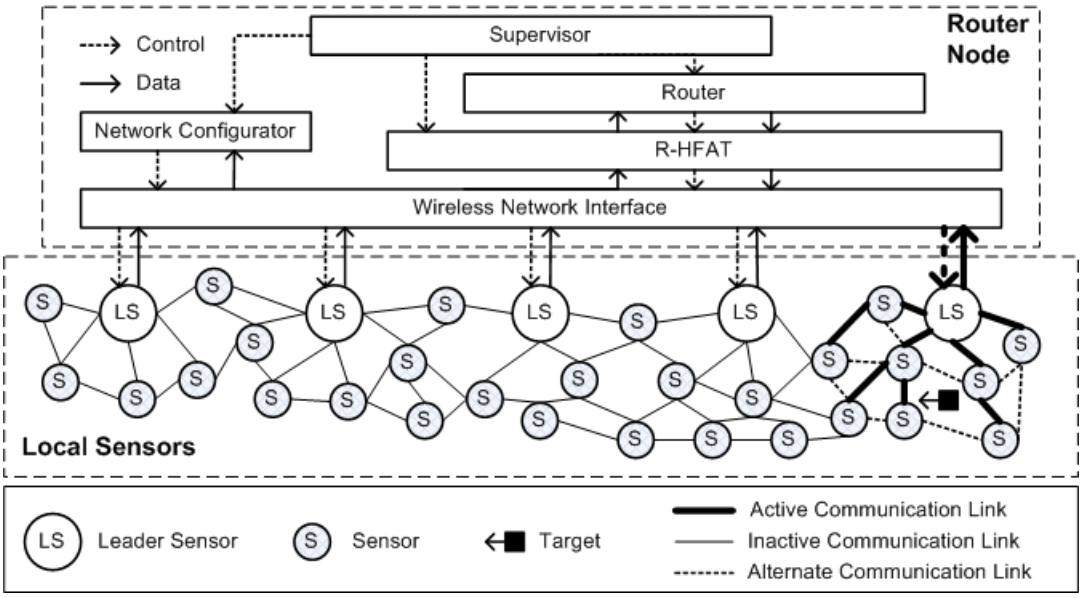


Figure 3.10: Router Node Architecture

3.5.1 Router Supervisor

The *Supervisor* on the R-node (figure 3.11 and table 3.8) is the analog of the *Query Processor* on the BS-node, sequentially triggering and synchronizing node initialization, network configuration, and target tracking tasks.

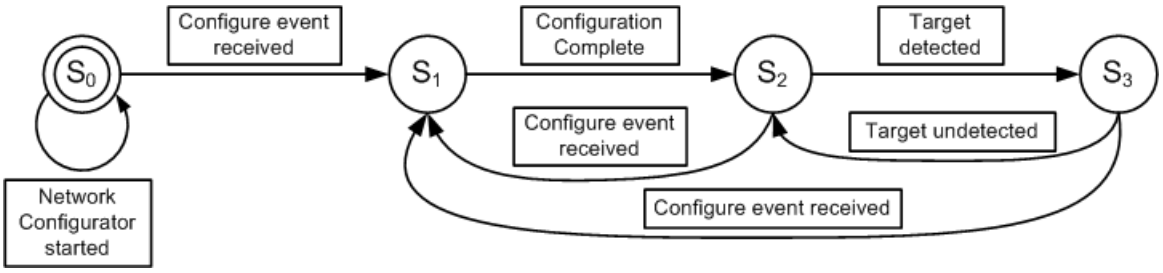


Figure 3.11: R-node Supervisor – State Diagram

State Name	Description	Previous State	Transition Event	Internal Message
S_0	**Initialization:** Waiting for *Configure* event.	N/A	R-node booted up.	N/A
		S_0	*Network Configurator* started.	N/A
S_1	**Configuration:** R-node received request to configure the sub-network containing its children nodes and it is waiting for the completion of this process.	S_0	*Network Configurator* reported received *Configure* message.	Receive: *Configure* event
		S_2	R-HFAT reported received *Configure* message.	Receive: *Configure* event
		S_3	R-HFAT reported received *Configure* message.	Receive: *Configure* event
S_2	**Detection:** R-node is participating in the target detection waves initiated by the BS-node.	S_1	*Network Configurator* reported received *Configured* message.	Receive: *Configured* event
		S_3	R-HFAT reported received *No Detection* message.	Receive: *No Detection* event
S_3	**Tracking:** R-node is routing tracking data towards the BS-node.	S_2	R-HFAT reported received *Detection* message.	Receive: *Detection* event

Table 3.8: R-node Supervisor – States Description

If the R-node is participating as a network node for the first time, it starts the *Network Configurator* and places it in *listening for configuration* mode, where the latter module will not perform any configuration tasks until it receives a *Configure* message from another R-node as part of a configuration wave initiated at the BS-node.

When this last event occurs, the *Network Configurator* designates, using algorithm **BLSD** (section 4.1.2), the LS-nodes in the R-node's surrounding area, and asks them to form sensor regions around them. When the regionalization process is complete, it contacts a subset of the R-node's one-hop-routers, selected based on their energy reserve and relative distance, and asks them to configure the sensor regions around them. When all the R-nodes in the selected subset report their configuration as complete, and the local sensor regions are configured, it sends a configuration complete status to the current R-node's parent node and reports this event to the *Supervisor*.

Next, the *Supervisor* deactivates the *Network Configurator*, starts the R-HFAT module and places it in a *listening for detection* mode, in which, it will not perform any detection tasks until it receives a *Detect Target* message from a parent R-node or the BS-node. Upon receiving this message, R-HFAT commands the leader sensors and sensor regions in the surroundings of the current R-node to detect a target. If one is detected, it transfers the detection status to the parent R-node for relaying to the BS-node and notifies the *Supervisor* about this event. The *Supervisor* switches R-HFAT to *tracking* mode and starts the *Router* module. If a target goes undetected while being tracked by the current R-node's sensor regions, R-HFAT communicates this event to the parent R-node for relaying to the BS-node.

If R-HFAT receives a *Configure* message, it reports this event to the *Supervisor*. This latter module halts modules R-HFAT, R-DSB, and *Router*, and re-starts module *Network Configurator* to reconfigure sensor regions and child R-nodes.

3.5.2 Network Configurator - Router Level

Upon receiving a *Configure* message, the *Network Configurator* (figure 3.12 and table 3.9) broadcasts a *Request Data* message, containing the current R-node's location coordinates, to all its one-hop-sensors (i.e. sensors within the R-node's communication range). Each sensor, with a communication radius S_c, receiving the *Configure* message, calculates its distance D_{sr} to the broadcasting R-node; if $D_{sr} \leq S_c$, the sensor responds with an *S-node Data* message, containing its coordinates and energy reserve value. If no sensor nodes respond to the *Request Data* broadcast, the module switches immediately to configure the R-node's one- hop-routers. Otherwise, the *Network Configurator* uses the *Biased Leader Sensors Designation* algorithm (section 4.1.2) to designate a subset of the responding sensors as LS-nodes. Next, it sends a *Configure* message to each LS-node, containing the implicit request to form a sensor region around it. Each LS-node replies with *Configuration Data* message when its sensor region is configured.

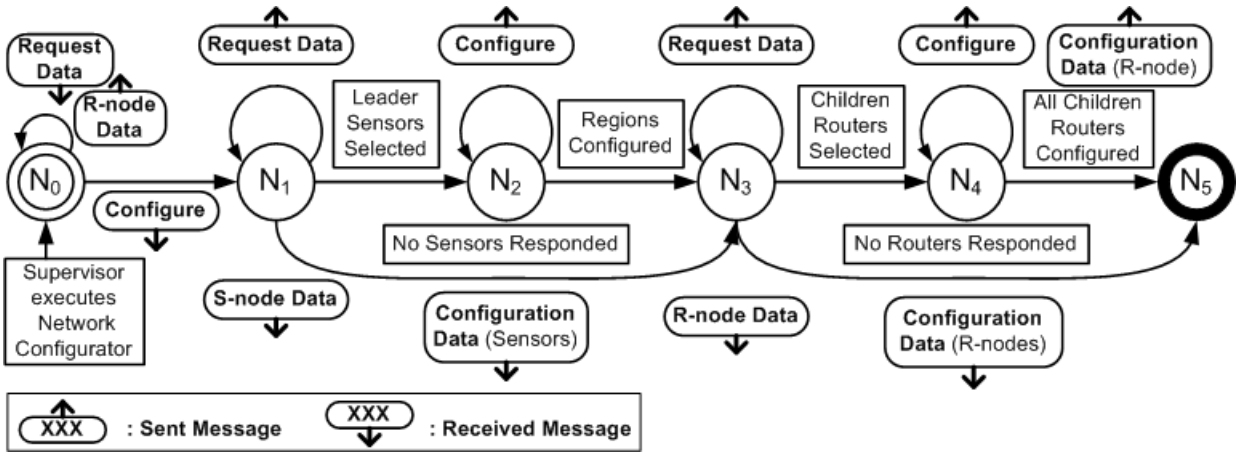


Figure 3.12: R-node Network Configurator – State Diagram

Once all designated LS-nodes reply to the configuration request, the *Network Configurator* flags the condition of local regions configured for the current R-node and proceeds to request the configuration of selected one-hop-routers; it broadcasts a *Request Data* message to all

the current R-node's one-hop-routers. Each of the receiving R-nodes not having all its local sensor regions configured responds with a *Configuration Data* message, containing its coordinates and energy reserve value.

State Name	Description	Prev. State	Transition Event	Message
N_0	**Initialization:** Waiting for *Configure* message.	N/A	*Supervisor* calls *Network Configurator*.	N/A
N_1	**Sensor Search:** Requesting data from one-hop-sensors.	N_0	Received *Configure* message from parent router.	Receive: *Configure*
		N_1	Receiving *S-node Data* messages from sensors.	Send: *Request Data* Receive: *S-node Data*
N_2	**Sensor Configuration:** Requesting formation of sensor regions.	N_1	*Network Configurator* selected leader sensors.	N/A
		N_2	Receiving sensor *Configuration Data* messages.	Send: *Configure* Receive: *Configuration Data*
N_3	**Router Search:** Requesting data from one-hop-routers.	N_1	No sensors replied to *Data Request* broadcast.	Report: *No Sensor Regions*
		N_2	All sensor regions around parent router configured.	Report: *Sensor Regions Configured*
		N_3	Receiving *R-node Data* messages from routers.	Send: *Request Data* Receive: *R-node Data*
N_4	**Router Configuration:** Requesting configuration of selected one-hop-routers.	N_3	*Network Configurator* selected one-hop-routers.	N/A
		N_4	Receiving router *Configuration Data* messages.	Send: *Configure* Receive: *Configuration Data*
N_5	**Termination:** *Network Configurator* reports *configured* status to *Supervisor* and/or parent node.	N_3	No routers replied to *Data Request* broadcast.	Report: *Terminal Router*
		N_4	All selected one-hop-routers configured.	Report: *Configured* Send: *Configuration Data*

Table 3.9: R-node Network Configurator – States Description

The *Network Configurator* uses the *Neighbour Angular Partition Routing* (NAPR) algoritm (section 4.1.1) to select a subset $\mathcal{R}_{dr}$ of the R-node's one-hop-routers as its *data relaying* children R-nodes. If no one-hop-routers respond to the *Request Data* broadcast, the module reports this event to the Supervisor and terminates immediately. Otherwise, it

selects each R-node in $\mathcal{R}_{dr}$, in the same order provided by algorithm NAPR, and sends it a *Configure* message to request the configuration of sensor regions and data relaying R-nodes under its scope. Finally, after all R-nodes in $\mathcal{R}_{dr}$ respond with a *Configuration Data* message, it sends its own *Configured Data* message to the node that requested the configuration of the current R-node, and reports a configuration complete event to the *Supervisor*.

3.5.3 HFAT - Router Level

The R-HFAT (R-node – *Hierarchical Focus of Attention Tracking* algorithm) module (figure 3.13 and table 3.10) is called by the *Router Supervisor*, while this latter module is in state S_2 (*Detection*). It is initialized to start listening for a *Detect* message from the current R-node's communication parent R-node, sent as a component of the target detection wave initiated by the BS-node. Upon receiving this message, it broadcasts it to all its associated leader sensors. If it receives a *Detection* message from any of its leader sensors, it relays this latter message to the current R-node's parent R-node and switches to the *active tracking* mode. If it does not receive a *Detection* message from any of its leader sensors, it switches to the *passive tracking* mode and broadcasts a *Detect* message to all the one-hop-routers configured as the current R-node's children by algorithm NAPR and waits for them to send either *Detection* or *No Detection* messages. If it receives a *Detection* from any of the children R-nodes, it relays the message to the parent R-node and switches to the *routing* mode. If it receives *No Detection* messages from all the children R-nodes, it relays to the parent R-node just one *No Detection* message.

If R-HFAT receives a *Track Target* message containing the estimated next target location, while in the *active tracking* mode, it computes the sensor region under its control that will be able to track the target (i.e *active region*). Thereafter, it relays the *Track Target* message to the LS-node of the active region and waits for this node to reply with a *Target Data* message containing the target's location αT $(0 < \alpha < 1)$. If the LS-node does not reply,

within an elapsed time αT $(0 < \alpha < 1)$, with the expected message, or it replies with a *No Detection* message, R-HFAT sends a *No Detection* message to the parent R-node and reports this event to the *Supervisor*. If it does receive the *Target Data* reply, it forwards the message to the *Router* module, which in turn will determine to which parent R-node to forward the message and will pass it back to R-HFAT for transmission over a network link.

If R-HFAT receives a *Track Target* message containing the estimated next target location, while in the *passive tracking* mode, it determines the child R-node having an associated sensor region that will be able to track the target. If the computed child R-node is different from the last one selected, it sends a *Select R-node* message to the former R-node and a *Deselect R-node* message to the latter one. Thereafter, it relays the *Track Target* message to the selected child R-node.

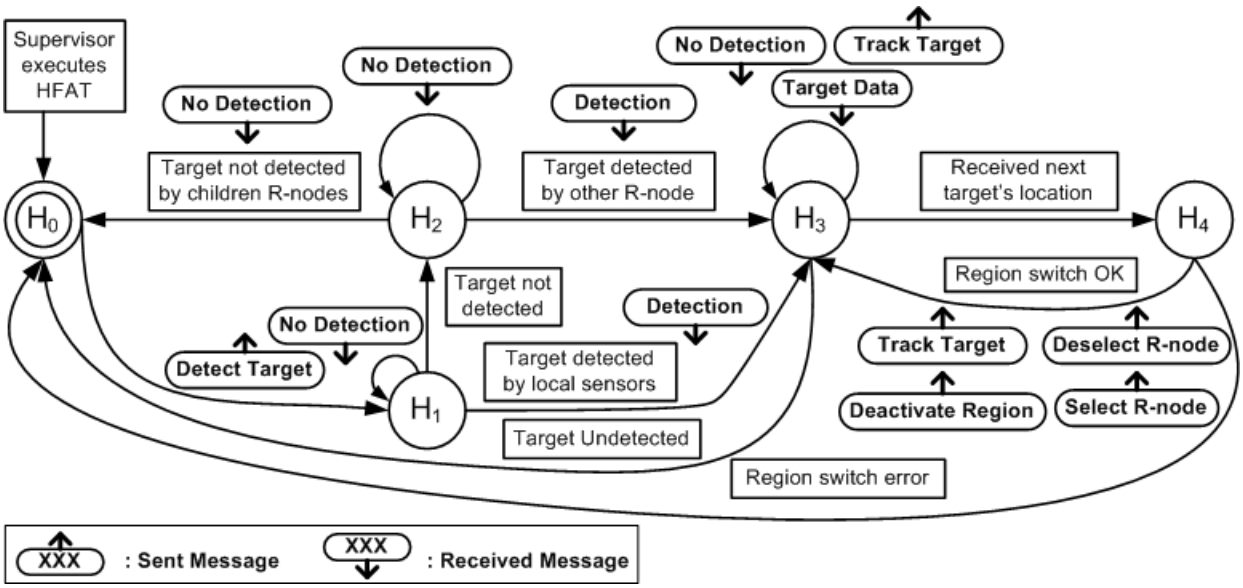


Figure 3.13: R-node Hierarchical Focus of Attention Tracking – State Diagram

State Name	Description	Prev. State	Transition Event	Message
H_0	**Undetected target:** R-node has not received reports of a detected target from neither sensors nor its children one-hop-routers.	N/A	Supervisor executes R-HFAT.	
		H_2	All one-hop-routers replied with a *No Detection* message.	Receive: *No Detection*
		H_3	Target goes undetected during tracking.	Receive: *No Detection*
		H_4	Next tracking sensor region not determined or target out of sensor field.	N/A
H_1	**Sensor detection wave:** R-node has broadcast a *Detect Target* message to local sensors.	H_0	R-HFAT received a *Detect Target* message from a parent R-node.	Send: *Detect Target*
		H_1	R-HFAT has received a *No Detection* reply to *Detect Target* broadcast.	Receive: *No Detection*
H_2	**Router detection wave:** R-node attempts to detect target over sensors local to children R-nodes.	H_1	R-HFAT received *No Detection* messages from all LS-nodes.	Receive: *No Detection*
		H_2	R-HFAT received *No Detection* message from a child R-node.	Receive: *No Detection*
H_3	**Tracking:** R-node tracks the state of a target detected by neighbour sensors or sensors close to its children R-nodes.	H_1	R-HFAT receives a *Detection* message from an LS-node.	Receive: *Detection*
		H_2	A child R-node replied with a Detection message.	Receive: *Detection*
		H_3	R-HFAT receives a *Target Data* message from an LS-node.	Receive: *Target Data* *No Detection* Send: *Track Target*
		H_4	R-HFAT determined next active sensor region.	Send: *Select R-node* *Deselect R-node* *Track Target* *Deactivate Region*
H_4	**Region switch:** R-node determines next active sensor region under children R-nodes.	H_3	R-HFAT receives next target's location from parent R-node.	N/A

Table 3.10: R-node Hierarchical Focus of Attention Tracking – States Description

If R-HFAT receives a *Target Data* message from a child R-node, while in the *passive tracking* mode, it forwards it to the *Router* module for determination of which parent R-node that should receive it next. Upon receiving the updated *Target Data* message back,

it transmits it to the next parent R-node.

If R-HFAT is in the passive tracking mode and receives a *Select R-node* message, it checks whether the estimated next location of the target falls within a sensor region in its local scope or not. If the answer is positive, R-HFAT switches to the *active tracking* mode and sends a *Track Target* message to the leader sensor in the sensor region to activate.

If R-HFAT is in the active tracking mode and receives a *Deselect R-node* message, it switches to the *passive tracking* mode and sends a *Deactivate Region* message to the leader sensor in its active sensor region.

3.5.4 Router Module

The *Supervisor* calls the *Router* module (figure 3.11, table 3.11) when reaching state S_3 (Tracking), and keeps it running until it switches back to state S_1 (Configuration). While active, the *Router* remains in permanent contact with module R-HFAT, and whenever the latter module receives a *Tracking Data*, *No Detection*, or *Track Target* message with a destination other than the one corresponding to the current R-node, it transfers it to the former module, which looks up the routing table, configured according to algorithm NAPR (section 4.1.1), to find the next-relay node in the path to the message's final destination. After attaching this information to the received message, the *Router* returns it back to R-HFAT for transmission to the next relay R-node.

If after transmitting the message, R-HFAT receives a timely *Confirmation* message from the next relay R-node, it forwards it to the *Router*, which returns to its original state R_0 (Idle) (figure 3.14). However, if it does not receive a *Confirmation* message within a specified time period, it times out and provides an update of this status to the *Router*, which switches to state S_1 (Routing) and attempts to find an alternate next-relay node, based of the priorities determined by algorithm ALUC (section 4.1.5).

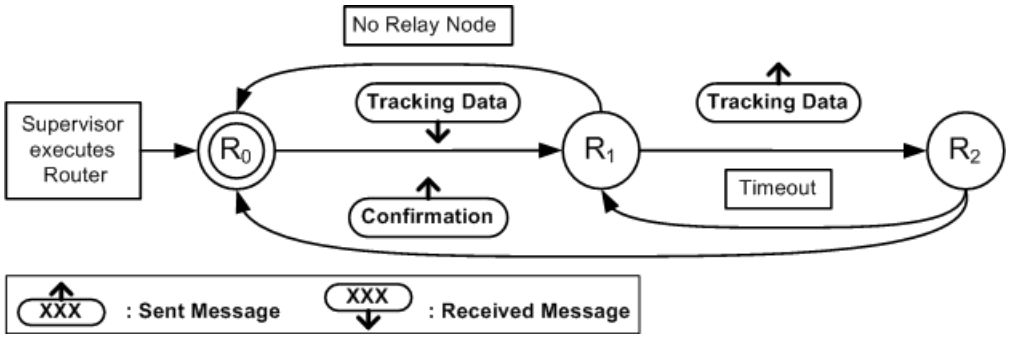


Figure 3.14: R-node Router – State Diagram

The *Router* and R-HFAT modules repeat the previous cycle until a successful transmission completes, or until the former module does not find a next alternate relay node for the intended destination, in which case, it informs R-HFAT about this situation and switches back to state R_0. Upon receiving this last update, R-HFAT switches to the *passive tracking* mode if currently active.

State Name	**Description**	**Previous State**	**Transition Event**	**Message**
R_0	**Idle:** Waiting for tracking data to be routed to the BS-node.	N/A	*Supervisor* executes *Router*.	N/A
		R_1	No relay R-node found.	N/A
		R_2	*Router* has received confirmation of tracking data received by parent R-node.	Receive: *Confirmation*
R_1	**Routing:** *Router* module computing next relay R-node for tracking data.	R_0	*Router* has confirmed reception of tracking data from a child R-node.	Receive: *Tracking Data* Send: *Confirmation*
		R_2	R-node has not received *Confirmation* message from parent R-node.	N/A
R_2	**Confirming:** Waiting for confirmation from parent R-node of received tracking data.	R_1	*Router* relays tracking data to selected parent R-node.	Send: *Tracking Data*

Table 3.11: BS-node Network Configurator – States Description

3.6 Sensor Architecture

The sensor network model proposed in this work assumes that all sensors have identical sensing hardware, as described in section 3.1.2, which is able to sense and sample the sensor's current distance to a mobile target (i.e *range*), set a detection event bit on when a target is within the sensing scope, and, when this event occurs, report a sampled measurement of the target's range to the sensor's *processing modules*. These modules (figure 3.15) implement the necessary logic to supervise the operations on each sensor, control its data acquisition process, handle the communication tasks required to extract the sampled data to the network with destination to the BS-node, generate and interpret the message exchamges required to support sensor collaborations tasks such as region formation and exception handling, and finally, keep track of the energy reserve on the sensor. A general sensor operation model from the standpoint of the processing modules, from now on denominated *processor*, can be described as follows:

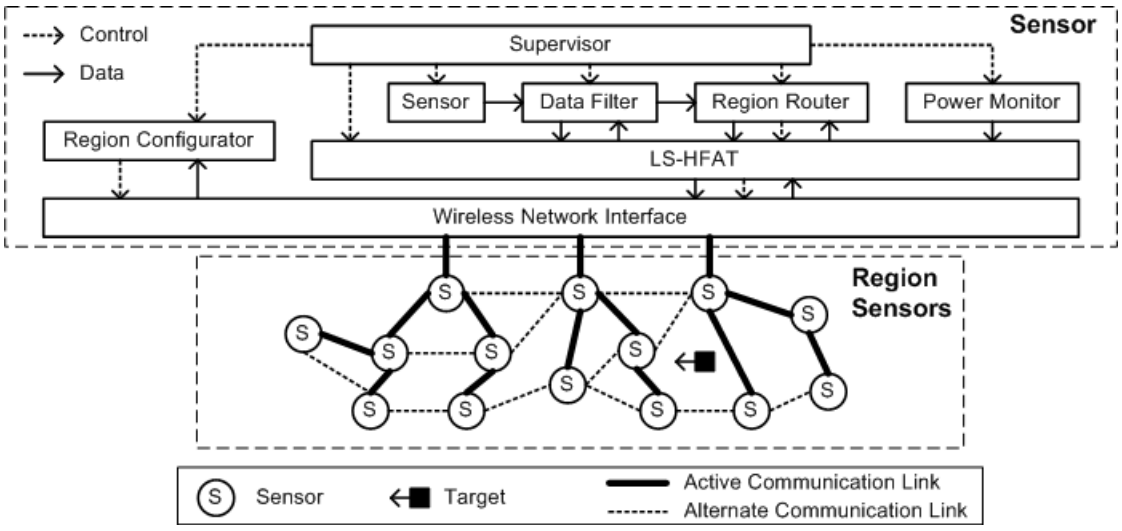


Figure 3.15: Sensor Node Architecture

- Processor can operate in the *active* and *standby* modes. In the standby mode, it does

not perform any sensing or data transmission tasks (it can only receive messages) until it receives an activation message from a *parent* node. In the active mode, it can perform all the tasks listed next.

- Processor receives, a time series $L = \{r_i(t_i) \mid i = 1, 2, ...\}$ of local range values produced by its own hardware, whenever it is capable of detecting a target.
- Processor can receive one or more time series $R_k = \{r_{j_k}(t_{j_k}) \mid j_k = 1, 2, ...\}$ of range values produced by neighbouring sensors.
- Processor can operate in the *leader sensor* and *region sensor* modes.
- In the leader sensor mode, processor analyzes the local L and remote R_k time series, and using an adaptive trilateration algorithm (section 4.2.3), it computes a best effort estimate of the target's current location.
- In the region sensor mode, processor relays the local L and remote R_k time series, using the routing scheme configured by algorithm HCRRF (section 4.1.3), to the leader sensor in the region.

3.6.1 Sensor Supervisor

The *Supervisor* on an S-node (leader or region sensor; figure 3.16, table 3.12) behaves in an almost identical way as its counterpart on an R-node. However, since by network design, sensors do not initiate operations in an asynchronous manner, instead they wait for an R-node to trigger synchronous configuration and target tracking operations, the *Supervisor* in this case reflects this behaviour, that is: sensors can be reconfigured while being idle or tracking, sensors provide target location or range values only under a command (i.e. message) received from a higher level node.

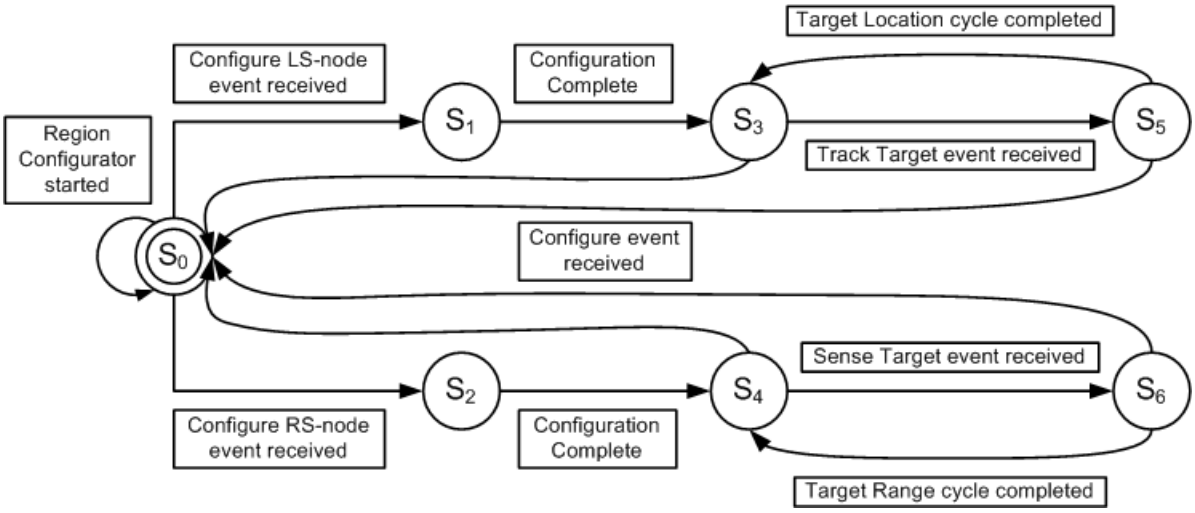


Figure 3.16: S-node Supervisor – State Diagram

If the *Supervisor* receives a *Configure* event, it starts the *Region Configurator* module in a specific modality to either configure an LS-node or an RS-node, depending on whether the command originated on an R-node or another RS-node, respectively. In the LS-node mode, the *Region Configurator* sets up the sensor to process local and remote target range data, and compute a target locations using the trilateration approach. In the RS-mode, it sets up the sensor to process local target range data and relay to an LS-node the remote data by creating the proper routing table.

The *Supervisor* stops the *Region Configurator* module when the sensor is completely configured and starts the LS-HFAT (LS-node) or the RS-HFAT (RS-node) module. LS-HFAT consolidates local and remote target range data streams to compute a stream of target location values. RS-HFAT builds a local data stream of target range values for relaying to an LS-node.

State Name	Description	Prev. State	Transition Event	Internal Message
S_0	**Initialization:** Waiting for *Configure* event originated by R-node (S_0, S_3, S_5) or S-node (S_0, S_4, S_6).	S_0	*Region Configurator* started.	N/A
		S_3	*Configure* event received	*Configure* event
		S_4	*Configure* event received.	*Configure* event
		S_5	*Configure* event received.	*Configure* event
		S_6	*Configure* event received.	*Configure* event
S_1	**LS-node Configuration:** Sensor configured as LS-node.	S_0	*Region Configurator* received message to configure LS-node.	*Configure* event
S_2	**RS-node Configuration:** Sensor configured as RS-node.	S_0	*RegionConfigurator* received message to configure RS-node.	*Configure* event
S_3	**Ready-To-Track:** Ready to process local and remote target range data, and compute target location.	S_1	LS-node configuration complete.	*Configured* event
		S_5	Completed target location computation.	*Target Location* event
S_4	**Ready-To-Sense:** Ready to process local target range data stream and provide best reading.	S_2	RS-node configuration complete.	*Configured* event
		S_6	Provided target range value.	*Target Range* event
S_5	**Tracking:** Computing location of the target.	S_3	*Track Target* event received.	*Track Target* event
S_6	**Sensing:** Selecting best target range reading.	S_4	*Sense Target* event received.	*Sense Target* event

Table 3.12: S-node Supervisor – States Description

The *Supervisor* also starts modules *Sensor*, *Data Filter*, *Region Router*, and *Power Monitor*, keeping them running until it receives a *Configure* event from the LS-HFAT (LS-node) or the RS-HFAT (RS-node) module.

3.6.2 Region Configurator

The *Region Configurator* module (figure 3.17, table 3.13) builds up the routing table, using algorithm HCRRF (section 4.1.3), on each sensor and adds an RS-node to a region if the maximum region size is not exceeded (i.e. *number of links to* LS-node $\leq$ $MaxHopCount$).

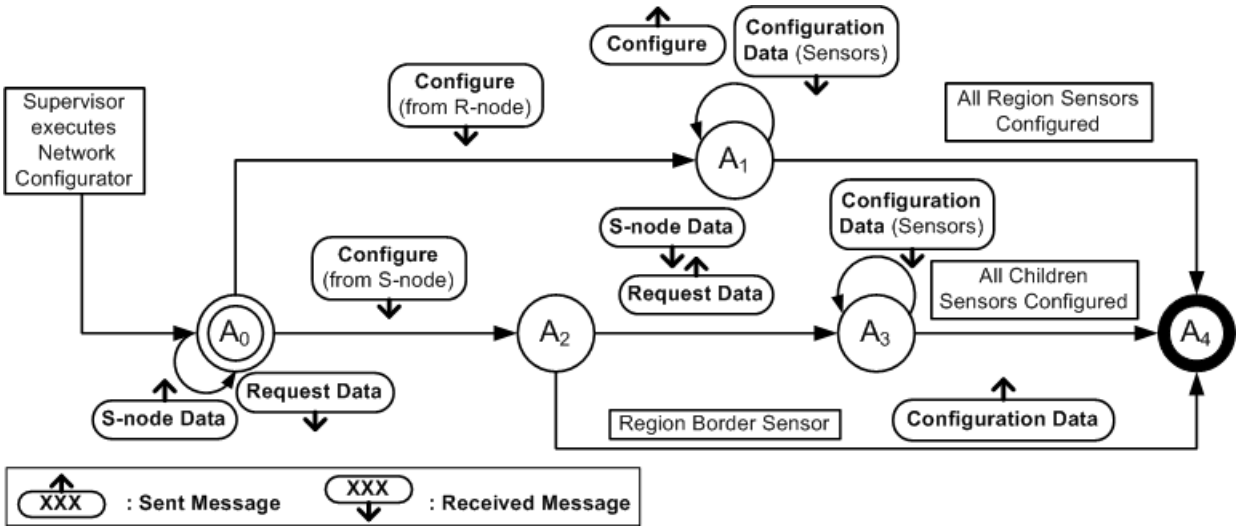


Figure 3.17: S-node Region Configurator – State Diagram

State Name	Description	Prev. State	Transition Event	Message
A_0	**Initialization:** Waiting for *Configure* message.	N/A	*Supervisor* calls *Region Configurator*.	N/A
		A_0	Request data from neighbour sensors.	Send: *Request Data* Receive: *S-node Data*
A_1	**Region Configuration:** LS-node triggered a region formation wave.	A_0	Received *Configure* message from R-node.	Receive: *Configure*
		A_1	Received configuration confirmation from RS-node.	Receive: *Configuration Data*
A_2	**Routing Configuration:** Configure routing to LS-node.	A_0	Received *Configure* message from RS-node.	Receive: *Configure*
A_3	**Sub-region Configuration:** RS-node triggered sub-region formation wave.	A_2	Configured routing to LS-node.	N/A
		A_3	Received configuration confirmation from RS-node.	Receive: *Configuration Data*
A_4	**Termination:** *Region Configurator* reports region configuration complete status to *Supervisor*.	A_1	Complete region configured.	N/A
		A_2	Region border sensor detected.	Send: *Configuration Data*
		A_3	RS-node's sub-region configuration complete.	N/A

Table 3.13: S-node Region Configurator – States Description

Upon receiving a *Configure* message from an R-node, the *Region Configurator* module configures the sensor as an LS-node, hence, capable of processing local and remote (from its region) target range data and compute target locations by using a trilateration algorithm (section 4.2.3). Immediately after, it commands the LS-node to trigger a region configuration wave, including sensors located up to $MaxHopCount$ links away, by discovering its neighbours, via an exchange of *Request Data* (broadcast) and *S-node Data* messages, and sending them *Configure* messages. Once the configuration wave, over the RS-nodes, reaches the region's maximum size, it returns back to the LS-node via succesive configuration confirmations represented by *Configuration Data* messages. At this point the region is completely configured and the *Region Configurator* module terminates on the LS-node, reporting a configuration complete status to the *Supervisor*.

Upon receiving a *Configure* message from an LS-node, or another RS-node closer to the LS-node, the *Region Configurator* module configures the sensor as an RS-node, hence, capable of processing local and relaying remote target range data to the LS-node. If it detects that the current RS-node is $MaxHopCount$ links away from the LS-node, it sends a *Configuration Data* message to the RS-node (or LS-node) from whom it received a *Configure* message, and terminates its execution and reports a configuration complete event to the *Supervisor*. Otherwise, it discovers its neighbour RS-nodes (*Request Data* (broadcast) v/s *S-node Data message exchanges* and, if any, it sends them a *Configure* message. When all its neighbours confirm their configuration by responding with a *Configuration Data* message, it sends its own *Configuration Data* message to the RS-node which addressed this current R-node, and reports a configuration complete to the *Supervisor*.

Region Routing – Having a sensor region established, the behaviour of the *Region Router* module is identical to its counterpart implemented on the R-nodes (section 3.5.4), with the same state diagram (figure 3.14) and description table (table 3.11). However, the

routing table it uses has been constructed by an algorithm whose goal is to provide reasonable sensing coverage by a sensor region (algorithm HCRRF), in contrast with algorithm NAPR whose goal is to build a routing table that supports approximately even communication energy consumption among the R-nodes. The *Region Router* relays data within a region only, and one of the end points in all communication must be the region's LS-node.

3.6.3 Sensor Data Streams

Local Target Range Stream – For simplicity, this work assumes that the *Control Unit*, in an S-node's sensing hardware (section 3.1.2), can accept a command from the *Processor* unit to initiate a short sampling cycle which would return one target range value r_l (and a detection bit), as a result of sampling and processing signals emanating from the target. The *Sensor* module handles the signal processing aspects of this cycle, and feeds value r_l to the *Data Filter* module. If the current node is an RS-node, this latter module passes this value on to the *Region Router* module, which attaches routing information to it and transfers it to the RS-HFAT module for transmission to the region's LS-node. If the current node is an LS-node, the *Data Filter* module keeps the r_l value, and uses it to compute the target's location in conjunction with target range values proceeding from other sensors in the region.

Target Location Stream – On an LS-node, the LS-HFAT module receives the target range values from the RS-nodes in the region and tranfers them to the *Data Filter* module, which gathers these values for only a time T_w (design parameter) and then computes the target's location using the algorithm described in section 4.2.3. The *Data Filter* transfers this value to the LS-HFAT module for relaying to the R-node with destination to the BS-node.

Power Monitor – It is also assumed that a sensor node's hardware is capable of measuring its energy reserve and providing a numeric value reflecting it. This value is attached to the target range data stream, after a given number of sensing cycles, for trasmission to the LS-node and the associated RS-node. The R-node will use these values to re-evaluate the LS-node designations and possibly the shape of its associated sensor regions.

Chapter 4

Low-Overhead Target Tracking

This chapter describes the distributed algorithms to configure the sensor network proposed in this thesis in order to extract to a BS-node the real-time location of a mobile target incurring low-overhead processing and reduced energy consumption. It also describes the node-collaboration-based algorithms implementing the functions (i.e. target detection and localization, data routing, state filtering and estimation) necessary to extract target location data from the sensor network to the BS-node, meeting the goals previously laid out and providing a reasonable degree of accuracy in tracking a target.

4.1 Network Configuration for Low-Overhead Data Extraction

Based on a tracking query issued at the BS-node (section 3.3) to input configuration parameters, the sensor network configures itself, in a distributed manner, attempting to settle into a topology that would advantageously use the values provided, under the following scheme:

1. The network has a hierarchical architecture, nodes collaborate with each other assuming hierarchical roles – leader sensor v/s region sensor, parent R-node v/s child R-node, BS-node v/s R-node – and the top of the hierarchy is the BS-node.

2. The network reconfigures the data routing scheme and node collaboration groupings (i.e. R-nodes tree structure and sensor regions) each time a new tracking query is issued at the base station.

3. The sensor nodes need to collaborate with each other, within regions of appropriate size and shape, to detect, localize and track a target, as they individually can only detect the target's *range* (i.e. distance from the target to a sensor) and not its location coordinates.

The following sections explain the details about configuring the data routing, assigning sensor roles, and forming sensor regions.

4.1.1 Neighbour Angular Partition Routing

Presenting the results of tracking a mobile target, whose dynamic behaviour can be modeled as constant velocity with random velocity changes at discrete times (section 2.2.1), on the BS-node, requires routing target location data from different areas of the sensor field to the same destination with approximately the same efficiency. Given a deployment of R-nodes, each one having the same communication radius R_c which establishes a *geometric communication spanner* $\mathcal{S}$ (i.e. connected graph including all the R-nodes in the network and links between nodes represent the euclidean distances between them) among them, the multi-hop communication from any point of the sensor field to the BS-node will be more efficient if it is performed over paths that are part of a *minimum spanning tree* $\mathcal{T}_{min}$, rooted at the BS-node, which is a sub-graph of $\mathcal{S}$.

Establishing $\mathcal{T}_{min}$ as the communication structure for delivering tracking data to the BS-node from any point of the sensor field would ensure the lowest latency possible given a specific R-node deployment. However, this approach selects preferential paths containing R-nodes which exhaust their energy budget at a higher rate than the average, and probably operate in border conditions in reference to remaining energy reserve. Therefore, in order to alleviate this problem, these paths should contain nodes with the highest energy reserve and the lowest inter-node distances possible, in order to use the total energy available in the R-nodes in an approximately uniform manner, to extend the R-node network's lifetime.

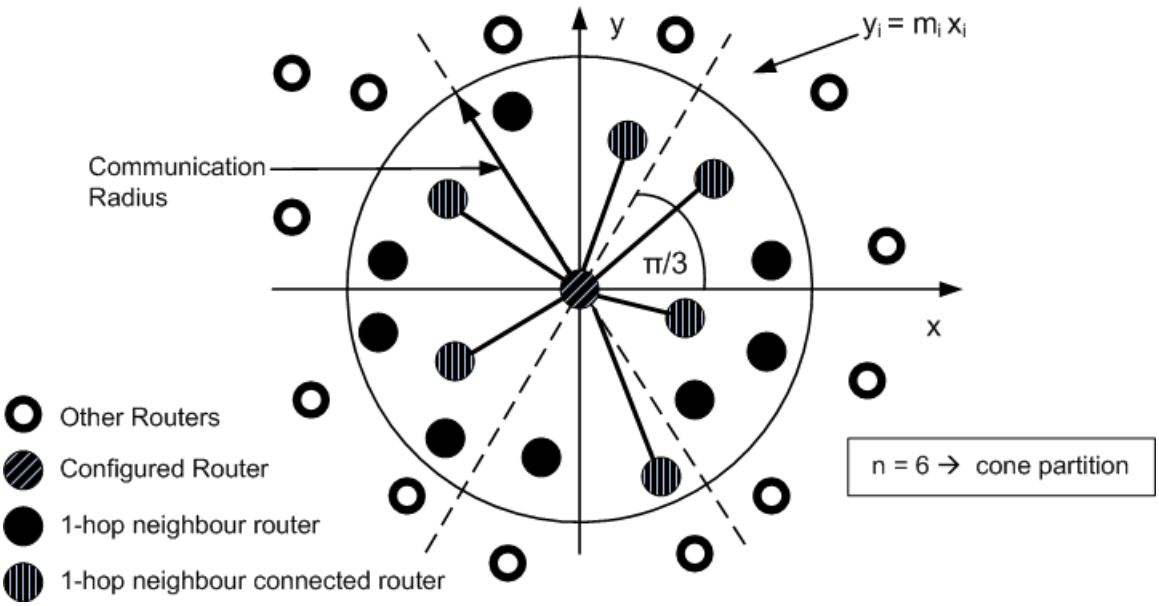


Figure 4.1: Neighbour Angular Partition Routing

Algorithm Neighbour Angular Partition Routing (NAPR) attempts to strike a balance between a low-overhead response and an extended R-node network lifetime when building the routing tree $\mathcal{T}_{min}$ for a given tracking query. NAPR is a heuristic distributed algorithm, which executes in the network configuration phase of a tracking query, as an expanding/contracting recursive *processing wave* [42] starting at the BS-node, to set up the

routing from the different areas of the sensor field to the BS-node over a communication tree $\mathcal{T}_{NAPR}$ having paths that are reasonably minimal and contain nodes with higher energy reserve levels. Function NAPR provides the pseudo-code for algorithm NAPR; beginning at the BS-node, it is executed on each R-node selected by the configuration wave created by the execution of the same function on previous nodes. It returns a *routing table*, for relaying information from the sensor network to the BS-node and vice versa, whose composite data type is *routingTable = (DestinationNode, NextNode)*. It takes the following parameters:

- P (*Partition Generator Number*), of type *integer*, $P = 1, 2, ...$; it is used to generate an angular partition, containing $2(2P + 1)$ cones (figure 4.1 shows partitioning for $P = 1$), of the R-nodes which are one-hop neighbours of the current R-node.

- *scanScope*, of type *integer*, whose domain is the set $\{0, 1, -1\}$, where, for a system of coordinates with origin on the current R-node, algorithm NAPR will check for children R-nodes only in the quadrants determined by this parameter, according to the following scheme:

  a) 0 – Quadrants I and II.

  b) 1 – Quadrants I and IV.

  c) -1 – Quadrants II and III.

- *R-nodeLocation*, of type *point* (representing an ordered pair (x, y)), which provides the location of the R-node currently executing the function.

- *EnergyGap*, of type *real*, which provides input for the maximum energy gap desired between the R-node with the highest energy in a grouping of these nodes and the remaining nodes in the group.

routingTable function **NAPR**(*int* P, *int* scanScope, *point* R-nodeLocation, *real* EnergyGap)

// Request location and energy reserve from one-hop R-nodes.

broadcast(**Request Data**, MaxHopCount = 1)

if **receivedMessage**(**R-node Data**) > 0

// One or more R-nodes answered with requested data.

// Partition the set of one-hop neighbour R-nodes.

$\mathcal{C}$ = **angularPartition**(P, R-nodeLocation)

// where: $\mathcal{C} = \{\mathcal{C}_0, ..., \mathcal{C}_{(2(2P+1)-1)}\}$

// and: $\mathcal{C}_i \cap \mathcal{C}_j = \phi, i \neq j$; $\{Circle\ of\ radius\ R_c\} = \{\mathcal{C}_0 \cup ... \cup \mathcal{C}_{(2(2P+1)-1)}\}$

if scanScope = 0 then

// This R-node is in parent's R-node's ScanAngle: $\frac{\pi}{3} < ScanAngle <= \frac{2}{3}\pi$

$\mathcal{R}_0 = \{\mathcal{C}_0, ..., \mathcal{C}_{2P}\}$

// Build the routing for scanScope = 0

routingTable = **buildRouting**($\mathcal{R}_0$, EnergyGap)

else if scanScope = 1 then

// This R-node is in parent's R-node's ScanAngle: $0 < ScanAngle <= \frac{\pi}{3}$

$\mathcal{R}_1 = \{\mathcal{C}_0, ..., \mathcal{C}_{\lfloor \frac{1}{2}(2P+1) \rfloor}\} \cup \{\mathcal{C}_{\lfloor \frac{3}{2}(2P+1) \rfloor}, ..., \mathcal{C}_{4P+1)}\}$

// Build the routing for scanScope = 1

routingTable = **buildRouting**($\mathcal{R}_1$, EnergyGap)

else if scanScope = -1 then

// This R-node is in parent's R-node's ScanAngle: $\frac{2}{3}\pi < ScanAngle <= \pi$

$\mathcal{R}_{-1} = \{\mathcal{C}_{\lfloor \frac{1}{2}(2P+1) \rfloor}, ..., \mathcal{C}_{\lfloor \frac{3}{2}(2P+1) \rfloor}\}$

// Build the routing for scanScope = -1

routingTable = **buildRouting**($\mathcal{R}_{-1}$, EnergyGap)

end if

// Send location of descendant R-nodes to main and alternate parent R-nodes.

Configuration Data=$\{R_i(Location) \mid R_i \in \text{RoutingTable} \wedge R_i(Relationship) = Child\}$

send(**Configuration Data**, *main parent*)

for each *alternate parent*

send(**Configuration Data**, *alternate parent*)

end for

else

// *Current R-node is leaf node. Send empty list to main and alternate parent R-nodes.*

Configuration Data=ϕ

send(**Configuration Data**, *main parent*)

for each *alternate parent*

send(**Configuration Data**, *alternate parent*)

end for

end if

return routingTable

end function

Functions "broadcast", "receivedMessage", and "send" are defined as follows:

- **broadcast**(*message*, *MaxHopCount*): sends *message* to all R-nodes located within *MaxHopCount* communication hops from the current R-node.
- **receivedMessage**(*message*, *source*): keeps a tally of the messages of *messageType* received by the current R-node from a *source* R-node and builds the list $\mathcal{N}$ of one-hop-neighbours. Each entry of $\mathcal{N}$ is a 3-tuple:
  $(R - nodeLocation, EnergyReserve, ConfiguredFlag) = ((x_r, y_r), E_{(x_r,y_r)}, Bit)$.
- **send**(*message*, *destination*): send a message of *messageType* to *destination* R-node.

R-nodePartition function **AngularPartition**(*int* P, *point* currentR-nodeLocation)

// Partition the one-hop neighbour R-nodes by dividing circle around current R-node

// into C_i $(i = 1, ..., 2(2P + 1))$ *cones of equal size* (figure 4.1).

// Calculate the gradients m_i *of lines* $y = m_i x$*, generating the cone partition.*

for $i = 0$ to $2(2P + 1) - 1$

$m_i = \tan(\frac{\pi i}{2P+1})$

end for

// Partition the one-hop-neighbour R-nodes.

for each R-node in $\mathcal{N}$

if $x = x_0$ then

if $y > y_0$ then

R-node $\in$ $C_{\lfloor \frac{2P+1}{2} \rfloor}$

else

R-node $\in$ $C_{(4P - \lfloor \frac{2P+1}{2} \rfloor + 1)}$

end if

else

$m = \frac{y-y_0}{x-x_0}$

for $k = 0$ to $(4P + 1)$

if $(m > \tan(\frac{\pi \lfloor \frac{2P+1}{2} \rfloor}{2P+1})$ or $m < \tan(\frac{\pi(2(2P+1) - \lfloor \frac{2P+1}{2} \rfloor)}{(2P+1)}))$

then

if $y > y_0$

R-node $\in$ $C_{(\pi \lfloor \frac{2P+1}{2} \rfloor)}$

else

R-node $\in$ $C_{(4P - \lfloor \frac{2P+1}{2} \rfloor + 1)}$

end if

else

if ($m \geq \tan(\frac{\pi k}{2P+1})$ and $m < \tan(\frac{\pi(k+1)}{2P+1})$)

R-node $\in$ C_k

exit for

end if

end if

end for

end if

end for

return R-nodePartition

end function

routingTable function **buildRouting**(*R-nodePartition* $\mathcal{R}$, *real* EnergyGap)

// Given $\mathcal{R} = \{\mathcal{C}_{i_o}, ..., \mathcal{C}_{i_n}\} \subset \mathcal{C} = \{\mathcal{C}_0, ..., \mathcal{C}_{(2(2P+1)-1)}\}$, *compute routing table*

// in an energy-aware fashion. Start with empty routing table.

routingTable=ϕ

// Add parent R-node as relay to BS-node. Routing table entries

// are 3-tuples with the format (Location, LinkPriority, Relationship)

routingTable=routingTable $\cup$ $\{((x_{parent}, y_{parent}), Main, Parent)\}$

for each C_{i_j} in $\mathcal{R}$

for each R-node in $\mathcal{C}_{i_j}$

if *ConfiguredFlag* = 0 then

// Select neighbour R-nodes whose routing table is not configured.

// Calculate euclidean distance to current R-node.

$d_r = \sqrt{(x_r - x_0)^2 + (y_r - y_0)^2}$

// Build list of location and energy values for each R-node in $\mathcal{C}_{i_j}$.

// Use a transmission energy consumption model which is $\propto \frac{1}{d^2}$.

Add 3-tuple $((x_r, y_r), E_{(x_r,y_r)}, \frac{E_{(x_r,y_r)}}{d_r^2})$ to list L_{i_j}

end if

end for

Sort L_{i_j} Descending By $E_{(x_r,y_r)}$ Into L_{i_j} ($m = m_{i_j}$ *items*).

// Select R-node with highest energy (first element).

$E_{max} = Energy(L_{i_j}(1))$

for $k = 2$ to m

if $E_{max} - Energy(L_{i_j}(k))) <$ EnergyGap

// Keep R-nodes with energy within EnergyGap *from* E_{max}.

Copy 3-tuple $((x_r, y_r), E_{(x_r,y_r)}, \frac{E_{(x_r,y_r)}}{d_r^2})$ to list R_{i_j}

end if

end for

Sort R_{i_j} Descending By $\frac{E_{(x_r,y_r)}}{d_r^2}$ Into R_{i_j} (n_{i_j} elements)

// Select R-node with highest $\frac{E_{(x_r,y_r)}}{d_r^2}$ ratio (first element)

// as the main relay R-node.

$R_{main} = R_{i_j}(1)$

// Add location of R_{main} *to routing table as main relay for* C_{i_j}

routingTable=routingTable $\cup$ $\{((x_{main}, y_{main}), Main, Child)\}$

if $n_{i_j} > 1$ then

// Add to routing table the locations of other R-nodes as alternates for C_{i_j}

for $k = 2$ to n_{i_j}

routingTable=routingTable $\cup$ $\{((x_k, y_k), Alternate, Child)\}$

end for

end if

end for

return routingTable

end function

4.1.2 Biased Leader Sensors Designation

Algorithm HFAT strives to track a single target in an energy-aware manner by involving in the process a single sensor region – with configurable organization, shape and size – located around the current target's position, and restricting all the other sensors in the network from spending energy in processing and communication tasks. For this purpose, in the configuration phase of processing a tracking query, R-nodes that are selected to form the communication tree $\mathcal{T}_{NAPR}$, routed on the BS-node, organize the sensors around them into regions and designate a leader sensor in each of them.

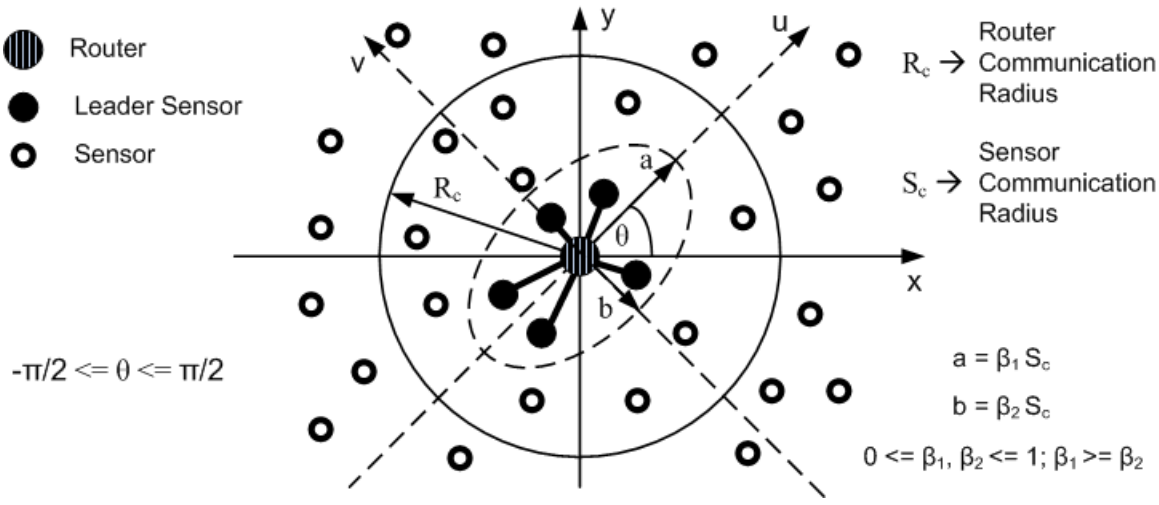


Figure 4.2: Biased Leader Sensors Designation

Heuristic algorithm *Biased Leader Sensors Designation* (BLSD), described in this section, executes on each R-node of $\mathcal{T}_{NAPR}$ and implements a procedure to designate specific sensors in its surrounding area as *leaders*, with the goal of having them grow sensor regions around them presenting less overlap in specific directions and locations of the sensor

field. Less overlapping of the sensor regions traversed by the target, in the direction of the intended target's trajectory, translates into a lower energy consumption, on the router and sensor nodes, employed in processing and communications tasks to activate/deactive regions around the current target's location.

$$\frac{u^2}{a^2} + \frac{v^2}{b^2} = 1 \tag{4.1}$$

$$\begin{aligned} u &= x\cos\theta + y\sin\theta \\ v &= -x\sin\theta + y\cos\theta \end{aligned} \tag{4.2}$$

$$\frac{(x\cos\theta + y\sin\theta)^2}{a^2} + \frac{(-x\sin\theta + \cos\theta)^2}{b^2} = 1 \tag{4.3}$$

$$y = \frac{y_s - y_r}{x_s - x_r}(x - x_s) + y_s \tag{4.4}$$

(x_r, y_r): Current R-node's coordinates; (x_s, y_s): Neighbour S-node's coordinates

Algorithm BLSD considers as candidates for leader nomination the sensors contained in an elliptical area centered around the current R-node (figure 4.2, equations 4.1, 4.2, and 4.3), whose major and minor radii (a and b respectively, $a \geq b$) have configurable sizes and orientation (i.e. angle θ with reference line crossing the BS-node), and makes a final selection based on their energy reserve and their orientation with reference to the R-node. This algorithm also uses the processing provided in functions "broadcast" and "receivedMessage" (defined in section 4.1.1) only that in this case the data returned form a list $\mathcal{S}$ of 3-tuples with the format:

$(S - nodeLocation, Distance, EnergyReserve) = ((x_s, y_s), d_s, E_{(x_s,y_s)})$,

where $d_s = \sqrt{(x_s - x_r)^2 + (y_s - y_r)^2}$ (distance from S-node to R-node).

leaderList function **BLSD**(*real* angle, *real* a-radius, *real* b-radius, *int* maxSensors)

// Request location and energy reserve from one-hop S-nodes.

// Start with empty list of Candidate Sensors $\mathcal{CS}$ to become leader sensors.

$\mathcal{CS} = \phi$

broadcast(**Request Data**, MaxHopCount = 1)

if **receivedMessage**(**S-node Data**) > 0

for each S-node in $\mathcal{S}$

if $d_s \leq$ b-radius

// This sensor is within the elliptical area.

$\mathcal{CS} = \mathcal{CS} \cup \{((x_s, y_s), E_{(x_s,y_s)})\}$

else

if $d_s \leq$ a-radius

// Determine intersection of ellipse ξ (equation 4.3) *and*

// line γ (equation 4.4), *point closer to sensor.*

$(x_i, y_i) = \xi \cap \gamma$

if $d_s \leq \sqrt{(x_i - x_r)^2 + (y_i - y_r)^2}$

$\mathcal{CS} = \mathcal{CS} \cup \{((x_s, y_s), d_s, E_{(x_s,y_s)})\}$

end if

end

end for

// So far, list $\mathcal{CS}$ *contains* m *candidate sensors.*

// Build the final list $\mathcal{LS}$ *of leader sensors.*

if $m \leq$ maxSensors then

// All candidates become leader sensors.

$\mathcal{LS} = \mathcal{CS}$

else

Sort $\mathcal{CS}$ Descending By $\frac{E_{(x_s,y_s)}}{d_s^2}$ Into $\mathcal{CS}$

// Select the first maxSensors *items in* $\mathcal{CS}$

$\mathcal{LS} = \{((x_s, y_s), d_s, E_{(x_s,y_s)}) \mid s = 1, ..., \text{maxSensors}\}$

end if

else

// *No sensors found in R-node's communication range.*

$\mathcal{LS} = \phi$

end if

return $\mathcal{LS}$

end function

4.1.3 Hop-Count-Restricted Region Formation

Once an R-node designates the leader sensors in its vicinity, it instructs each one of them, in a sequential manner, to form a sensor region by having it execute distributed recursive algorithm *Hop-Count-Restricted Region Formation* (HCRRF) in conjunction with its neigbouring sensors. The algorithm has a twofold purpose: determine the sensors that will form a region whose topology is energy-efficient, in size and shape, for low-overhead target tracking, and establish the data routing scheme from these sensors to the leader and vice versa. Overall, it behaves as a *wave algorithm* intiated at the leader sensor, propagating outwards up to maximum number of links from this node, and returning back to this initial node, at which point, it has configured the routing tables on all sensors in the region. The core ideas implemented in its execution can be summarized as follows:

- The leader sensor is the origin of an expanding sensor region, and initially executes algorithm HCRRF (described below) with a *HopCount*= 0. Therefore, it executes HCRRF(*null*, 0, *regionHopCount*). The region can grow up to a maximum size such that every sensor is within *regionHopCount* hops from the leader sensor (figure 4.3 shows an example for *regionHopCount*=3).

- All sensors that are not leaders have an initial *HopCount*= ∞. The leader sensor has

no predecessor in the region, condition that is expressed in setting this latter node to $null = (-1, -1)$.

- *regionHopCount* is chosen as a heuristic compromise between the desired region's sensing coverage, the energy reserve on the leader sensor, and the expected dynamic behaviour of the tracked target.

- After this initial call, the algorithm is recursively executed on sensors that are k ($k = 1, ...,$*regionHopCount*) hops away from the leader sensor, by having the execution on sensor at step $(k-1)$ recursively call HCRRF($(x_{(k+1)}, y_{(k+1)})$, k, *regionHopCount*) on sensor at step k.

- At each step of the recursion, a sensor determines its routing table for relaying data to the leader sensor (including alternate routes, since this is the preferential direction for data flows) and from the leader to the region sensors. The format of each entry in this table is $(nextSensor, relationship, priority)$, where $nextSensor = (x_s, y_s)$, $relationship = \{parent, child\}$, and $priority = \{main, alternate\}$.

This algorithm also uses functions "broadcast", "receivedMessage", and "send", with definitions as described in section 4.1.1.

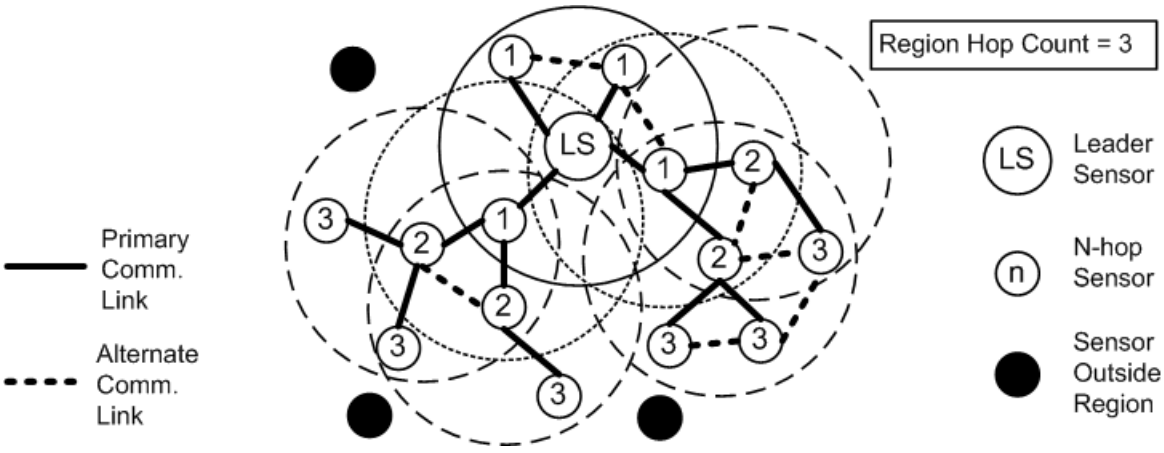


Figure 4.3: Hop Count Restricted Region Formation

routingTable function **HCRRF**(*point* sensor, *int* HopCount, *int* regionHopCount)

// *Form sensor region that expands up to* regionHopCount *links away from leader sensor.*

// *Determine the routing table* $\mathcal{RT}$ *for each sensor in the region. Start with empty table.*

$\mathcal{RT} = \phi$

if sensor $= \phi$ then

$\mathcal{RT} = \mathcal{RT} \cup \{(R - node, parent, main)\}$

else

$\mathcal{RT} = \mathcal{RT} \cup \{(sensor, parent, main)\}$

end if

if HopCount $<$ regionHopCount then

broadcast(**Request Data**, MaxHopCount = 1)

if **receivedMessage**(**S-node Data**) > 0

//*The one-hop neighbours in list* $\mathcal{NS}$ *answered the broadcast message.*

//*Inform them that they are at step* HopCount+1 *in this region's configuration.*

for each S-node in $\mathcal{NS}$

send((**HopCount + 1**), **S-node**)

end for

```
            for each S-node in $\mathcal{NS}$
                if HopCount < S-node(HopCount) then
                    // Trigger recursive execution of HCRRF on node at step $k+1$.
                    send(Execute HCRRF(S-node, HopCount + 1, regionHopCount), S-node)
                    // Wait for S-node to answer in less than maxWaitTime.
                    wait(ResponseTime) for receivedMessage(Configuration Data)
                    // Time out if S-node has not answered before maxWaitTime expires.
                    timeout(maxWaitTime)
                    if receivedMessage(Configuration Data)
                        $\mathcal{RT} = \mathcal{RT} \cup \{(sensor, parent, main)\}$
                    end if
                else
                        $\mathcal{RT} = \mathcal{RT} \cup \{(sensor, parent, alternate)\}$
                end if
                $\mathcal{RT} = \mathcal{RT} \cup \{$Configuration Data$\}$
            end for
        end if
    end if
    if sensor $\neq$ null then
        Configuration Data = $(x_k, y_x)$
        send(Configuration Data, sensor)
    end if
    return $\mathcal{RT}$
end function
```

4.1.4 Region Sensing Scope

After the execution of algorithms BLSD and HCRRF, R-nodes know the coordinates of all the the sensors in their associated regions, hence, they can determine their sensing coverage. Algorithm *Region Sensing Scope* (RSS) executes on the BS-node and R-nodes, to determine a graph *sensing border polygon* (figure 4.4) which represents a *good* approximation of the perimeter for the sensor region's sensing coverage. The following assumptions are the base of this algorithm:

- Each sensor provides equal omnidirectional sensing coverage, represented by a circle of radius r, centered on the sensor. This coverage is upper-bound approximated by a square, inscribed in the circle, with a diagonal of size $2r$ (figure 4.4) and an orientation such that its sides are parallel to the axes of cartesian system of coordinates with origin on the R-node. If the coordinates of a sensor are (x_s, y_s) in this reference system, the coordinates of the four points where the square touches the circle are, in counter-clockwise order, given by:

$$\begin{aligned}
(x_s^+, y_s^+) &= (x_s + \frac{r\sqrt{2}}{2}, y_s + \frac{r\sqrt{2}}{2}) \\
(x_s^-, y_s^+) &= (x_s - \frac{r\sqrt{2}}{2}, y_s + \frac{r\sqrt{2}}{2}) \\
(x_s^-, y_s^-) &= (x_s - \frac{r\sqrt{2}}{2}, y_s - \frac{r\sqrt{2}}{2}) \\
(x_s^+, y_s^-) &= (x_s + \frac{r\sqrt{2}}{2}, y_s - \frac{r\sqrt{2}}{2})
\end{aligned} \tag{4.5}$$

- The sensor regions have enough sensing overlapping, generated at configuration time, such that, for a given target dynamics, the network can provide continuous detection when neighbouring regions are activated/deactivated to provide energy-aware tracking.

- The sensor density is sufficiently high so each sensor region provides complete sensing coverage, without any interior gaps, of the area enclosed by the *sensing border polygon*.

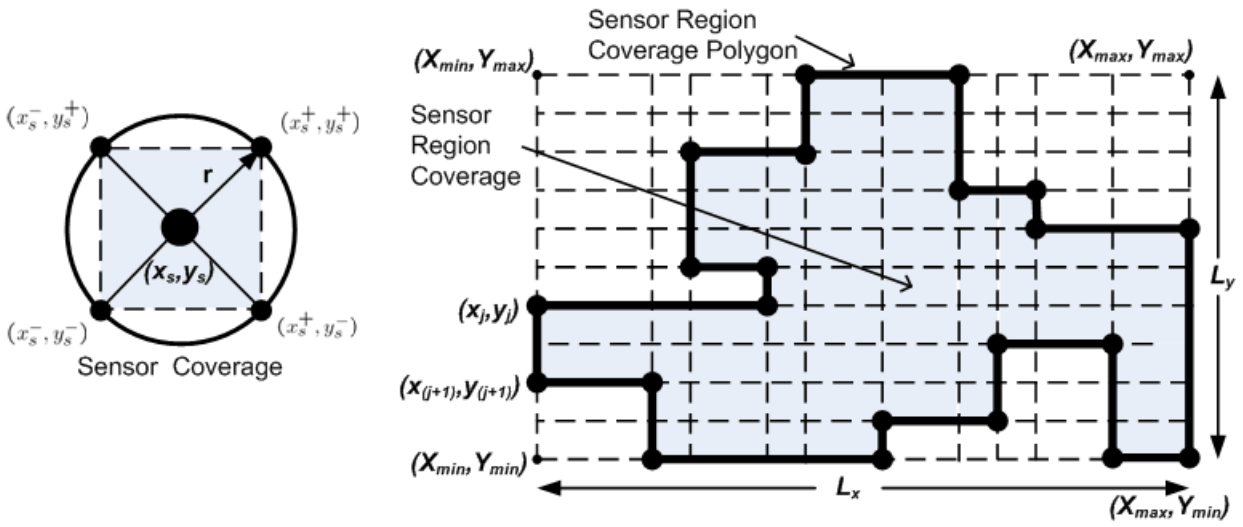


Figure 4.4: Sensor Region Polygonal Sensing Scope

Function RSS (appendix A) implements this algorithm, taking as input a non-empty set of points $\mathcal{S} = \{(x_i, y_i) \mid i = 1, ..., m\}$, representing the coordinates of each sensor in reference to an origin located on an R-node, and providing as output the data structure $\mathcal{P}$ of type *perimeterGraph*:

$\mathcal{P} = \{s_k = [(x_i, y_i),\ (x_j, y_j)] \mid \bigcup_{k=1}^{N} s_k$ is polygonal perimeter of region's sensing scope$\}$

representing a graph: $\mathcal{G} = \{\{v_k\},\ \{e_l\}\} \mid v_k \in \{vertices\},\ e_l \in \{edges\}\}$,

where: $v_k = (x_k, y_k)$, $e_l =$ line segment (x_i, y_i) to (x_j, y_j).

After executing algorithm RSS, the BS-node and the R-nodes have enough sensing coverage information, about the regions under their observation, to determine whether the target's next location (time t_{n+1}) remains within the same region or moves to a new one. Once the region's *polygonal* perimeter is known (figure 4.4), obtaining a positive answer to this question is reduced to finding two *x-direction* and two *y-direction* edges in $\mathcal{P}$ enclosing this location. Otherwise, the target's next location is outside the area covered by the region under consideration.

4.1.5 Fault Tolerance Configuration

Algorithms NAPR and HCRRF configure the R-node communication tree and the sensor region routing scheme, respectively, identifying main links between nodes for the proper relaying of tracking data to the BS-node. They also identify, for each main link connecting two nodes, a set of zero or more alternate links, without any specific usage priority after the main link. It is left to further processing on each affected node to decide about a priority scheme on using alternate links to provide a level of main link fault tolerance.

Given the hierarchical topology and architecture of the network proposed in this work, R-node failures closer to the BS-node affect the reporting of tracking activities of larger areas of S-nodes. Therefore, it is necessary to introduce, as a design parameter, some value(s) that would indicate to the network what proportion of it is tolerated to be out service from the tracking standpoint.

This section describes simple heuristic approaches proposed to address the two problems previuosly explained in an energy-aware fashion, starting with a scheme for using alternate links.

Alternate Links Usage Configuration– The assumption is that the *Network Configurator* and *Region Configurator* modules on R-nodes and S-nodes respectively, would configure the scheme of alternate links usage at the very end of their execution phase. The configuration process assigns an integer number k, representing an estimated and relative level of link's energy capacity (based on using a transmision energy model $\propto \frac{1}{d^2}$), to each alternate link. The idea is that during normal operation, starting with the link assigned the highest k and using them in a cyclical manner, a node will use any one of them at most k before switching to the one with the next lower k. Function ALUC (*Alternate Links Usage Configuration*) implements this configuration; it takes as input the routing table for a given

node and returns it with the k numbers appended to table entries representing alternate links.

routingTable function **ALUC**(*routingTable* $\mathcal{RT}$, *real* EnergyGap)

// Initialize link priority list to empty.

$\mathcal{PL} = \phi$

// Identify alternate one-hop parent nodes used by this node.

Select *Location* from $\mathcal{RT}$ where *LinkPriority=Alternate* into *AlternateNodes*

If *AlternateNodes* $\neq \phi$ then

for each *Location* in *AlternateNodes*

// Request the parent node's energy reserve and distance to this node.

send(**Request Data**, *Location*)

// if the parent answers add it to node list $\mathcal{NL}$.

if **receivedMessage**(**Energy Data**, *Location*) > 0 then

$E =$**Energy Data**(*EnergyReserve*)

$d =$**Energy Data**(*InterNodeDistance*)

$\mathcal{NL} = \mathcal{NL} \cup \{(Location, E, \frac{E}{d^2})\}$

end if

end for

Sort $\mathcal{NL}$ Ascending By $\frac{E}{d^2}$ into $\mathcal{NL}$

$k = 0 \quad (k \in \mathbb{Z})$

$prevEnergy = 0$

for each *Item* in $\mathcal{NL}$

$\Delta E = \mathcal{NL}(E) - prevEnergy$

if $|\Delta E| >$ EnergyGap then

$\Delta K = \lfloor \frac{|\Delta E|}{EnergyGap} \rfloor$

if $\Delta E \geq 0$ then

$k = k + \Delta K$

else

$k = k - \Delta K$

end if

end if

$\mathcal{PL} = \mathcal{PL} \cup \{(Location, k)\})$

$prevEnergy = \mathcal{NL}(E)$

end for

// Update alternate links in routing table with the k numbers.

for each $\mathcal{PL}$(Location) in $\mathcal{PL}$

Replace $\mathcal{RT}(Location, LinkPriority, Relationship)$

With $\mathcal{RT}(Location, LinkPriority, Relationship, k)$

Where $\mathcal{PL}$(Location) = $\mathcal{RT}(Location)$ And $\mathcal{RT}(LinkPriority) = Alternate$

end for

end if

return routingTable

end function

R-node Outage Tolerance– The goal is to limit the proportion of the network that will be disabled from carrying out the tracking function because of a complete outage of an R-node. Assuming that at configuration time the *Network Configurator* module on the BS-node and R-nodes inform their children R-nodes, in the **broadcast**(**Request Data**, MaxHopCount=1) message, about the number of hops required to reach the BS-node from its own location, it is possible to have all the R-nodes in the network know their distance d, in communication hops, to the BS-node. Also, in the same message, a *MaxDistance* to the BS-node design parameter, issued at the BS-node, can be propagated to all R-nodes.

Using this setup, a simple R-node outage tolerance can be established by having R-nodes in *passive tracking* mode (section 3.5.3) compare their own distance d to the BS-node with *MaxDistance* when one of their children R-nodes is detected as not responding, and declare itself *out of service until reconfigured* if $d < MaxDistance$. This action will cause an R-node disconnection wave that will eventually reach the BS-node, at which point this latter node will trigger a complete network re-configuration as it was executed when the tracking query was first processed by the network.

4.2 Hierarchical Focus of Attention Tracking

Chapter 3 described the architecture, messaging interactions, and behaviours of the modules executing on the BS-node, R-nodes, and S-nodes, to implement algorithm HFAT in a distributed manner. It provided detailed descriptions of the hierarchy of tasks, their timing and synchonization, from the standpoint of each individual node, considering the rest of the network as the node's *environment* and the interactions with it as the messaging complying with a established *protocol* in a chain of distributed events necessary to track a target. The following sections provide the complementary view of this approach; they describe HFAT from an overall system standpoint, as perceived by an external observer able to visualize the entire sensor network at any given time.

4.2.1 Initial Target Detection

This work assumes that the sensor field is a rectangular area of lenght L_1 and width L_2 (figure 1.1), where, in the general case, a point-like target enters the area at point P_1 and exits at point P_2, defining a trajectory which represents a *random walk* dynamic behaviour with constant velocity between sampling times (section 2.2.1). In order to track a single target in an energy-aware fashion, algorithm HFAT operates on a sensor network organized in regions enabling the target sensing only in the region currently containing it (*active*

region), and disabling all communication and processing tasks related to tracking on all other regions (*inactive regions*). This approach requires initial synchronization between the region that should be active and the target's detected entry point to the sensor field (P_1).

Since P_1 and the initial velocity of the target are not always known *a priori*, the sensor network must implement a mechanism to achieve this synchronization in an energy aware manner, without having all sensor regions activated at the same time. This work proposes the approach of using *detection waves*, whose profile is determined by the parameters **InitialLocation** and **InitialVelocity**, provided in a tracking query issued at the BS-node (section 3.3), to synchronize a target's location to an active region, as follows:

Unknown InitialLocation and InitialVelocity – This scenario requires the highest energy consumption for target detection, as it demands that most sections of the network be active for longer times to achieve succesful initial target detection. In this scheme the BS-node sets all its children R-nodes in the *active tracking* mode (section 3.5.3), in which case, these nodes will activate all their associated regions. Detection of the target by one or more of these regions is sent to the BS-node, at which point this node will set all its children R-nodes, except the R-node closest to the reported target's location, in the *passive tracking* mode, hence the associated sensor regions will be disabled for tracking a target. If no detection is reported within a design waiting time T_{w_1}, each children R-node repeats the operation previously explained with its own children R-nodes, remains in the active tracking mode for a design time T_{a_1} and then turns itself to the passive tracking mode. If no detection occurs at each R-node processing step, this wave expands until it reaches the perimeter of the sensor field. For each step k ($k = 1, 2, ...$), $T_{w_k} < T_{w_{(k+1)}}$ and $T_{a_k} < T_{a_{(k+1)}}$. The entire wave is repeated after a design time T_d, and it is interrupted, as previously explained, at any step k where the detection of a target occurs.

Unknown InitialLocation, Known InitialVelocity – The same wave detection scheme, used to handle the scenario of unknown initial target location and velocity, applies in this situation. However, in this case, for a target with an initial velocity $\vec{v} = (\dot{x}, \dot{y})$ and speed $s = \sqrt{\dot{x}^2 + \dot{y}^2}$, the times T_{w_k}, T_{a_k} $(k = 1, ..., n)$, and T_d are adjusted as follows:

- Higher speed $s \Rightarrow$ shorter T_{w_k}, longer T_{a_k}, shorter T_d.
- Lower speed $s \Rightarrow$ longer T_{w_k}, shorter T_{a_k}, longer T_d.

Known InitialLocation – Assuming that the target's speed s is less or equal than a *Maximum Speed*, which is considered a network operational specification, in this scenario the BS-node directly sets, by propagating a **Track Target** command over the proper descendant R-nodes (section 3.5.3), the R-node controlling the region containing the target to the *active tracking* mode and the HFAT algorithm starts tracking the target immnediately.

4.2.2 Target Tracking Approach

The tracking of the target starts immediately after a sensor region detects it for the first time. The BS-node has the overall control over tracking the target and, having acquired its initial location and identified the R-node controlling the sensor region currently containing it, it continually triggers a *tracking cycle*, with a period T, until the target leaves the sensor field. The following is the general structure of this cycle:

- The BS-node estimates the next location of the target for the current cycle (Kalman filter, section 3.4.4); based on this information it computes the sensor region that will have sensing scope next over the target and identifies the R-node controlling that region.
- If the sensor region and the R-node controlling it change, the BS-node sends, over the R-nodes, commands (sections 3.4.3, 3.5.3) to set the *passive tracking* mode on the R-node loosing sensing scope over the target, and the *active tracking* mode on the

R-node gaining sensing scope over the target. If an R-node switches to *active tracking*, it turns all its associated leader sensors (LS-nodes) to the *active* power mode. If an R-node switches to *passive tracking*, it turns all its associated LS-nodes to the *standby* power mode.

- The BS-node sends, over the R-nodes, a *Track Target* command to the R-node having sensing scope over the next location of the target.

- The addressed R-node identifies the sensor region that will have sensing scope over the next location of the target, and instructs its leader LS-nodes to report the target's location. If a region switch is necessary, the R-node instructs the LS-node of the region that reported the target's location in the previous cycle to turn all the sensors in its region to the *standby* mode; and intructs the LS-node of the new region to turn all its sensors to the *active* mode.

- The addressed LS-node instructs the sensors (S-nodes) in its region to report their *sensor range* readings immediately.

- The S-nodes in the region that can detect target (section 3.1.2) report their most current range measurements to the LS-node.

- The LS-node computes the current (x, y) location of the target using a trilateration algorithm (section 4.2.3), and reports its associated R-node for relaying to the BS-node.

- Starting with the *active tracking* R-node, the R-nodes forming path to the BS-node relay the target's location to the latter node.

- The BS-node, upon receiving the target's location from the sensor network, estimates its current velocity (Kalman filter, section 3.4.4) and its next location.

- If the target's next location is within the limits of the sensor field, the BS-node triggers a new cycle when the time T has elapsed. Otherwise, the BS-node starts a target detection wave.

4.2.3 Region Data Stream Builder Algorithm

Assuming that sensors can operate in a *normal* or a *standby* power mode, being the second one a state of low energy consumption (i.e. no sensing, reduced computation, receive-only communication), when tracking the target it is only necessary to have in the normal mode the sensors in the *active region*. Therefore, if a region switch has occurred with the start of a new tracking cycle, the LS-node in the new sensing region broadcasts a *Set Active* message to all its associated sensors to turn them to the *active* power mode. Similarly, the LS-node, in the region becoming inactive, broadcasts a *Set Standby* message to all its associated sensors setting them in the *stanby* power mode.

Upon receiving a *Track Target* from its associated R-node, the LS-node in the active region broadcasts a *Detect Target* message to all the sensors in the region. Upon receiving this latter message, the sensors in the region, which can currently detect the target, send immediately their last range measurement to the LS-node in **Range Data** message. With this information, the LS-node computes the current location of the target using the following algorithm, which takes as input a list of items of type *rangeMeasurement*=$(sensor, range, timestamp)$ (where $sensor = (x, y)$, $range$ is of type $real$, $timestamp$ is measurement time with a millisecond resolution):

locationPoint function **getTargetLocation**(*rangeMeasurement* RangeList)

// Count the number of measurements returned by the S-nodes to the LS-node

// and compute a target location accordingly.

if **Count**(RangeList) = 0 then

// No measurements returned by S-nodes. Inform the R-node about this event.

Send(No Detection, R-Node)

targetLocation = $\phi = (-1, -1)$

else if **Count**(RangeList) = 1 then

// Only one S-node returned a measurement. Assume target's location = S-node's location.

targetLocation = S-node(x, y)

else if **Count**(RangeList) = 2 then

// Only two S-nodes S_1 and S_2 *returned measurements.*

// Assume target's location is middle point of segment joining S_1 and S_2.

// Solve the following system of two equations for (x, y)*:*

1) $\sqrt{(x - x_1)^2 + (y - y_1)^2} = \sqrt{(x - x_2)^2 + (y - y_2)^2}$ *// (x,y) is equidistant from* S_1, S_2.

2) $\frac{y-y_1}{x-x_1} = \frac{y_2-y_1}{x_2-x_1}$ *// (x,y) is on line through* S_1 and S_2.

targetLocation = (x, y)

else

// Three or more sensors returned measurements.

// Apply trilateration on best three range measurements.

if **Count**(RangeList) > 3 then

Sort RangeList Ascending By *timestamp* Into RangeList *// $n > 3$ items.*

// Initialize TimeDifference *list of type* (timeDiff,$(x_k, y_k), (x_{(k+1)}, y_{(k+1)}), r_k, r_{k+1}$));

// where timeDiff = *elapsed time*, (x, y) = *sensor*, and r_k = $sensor_k$

TimeDifference = *empty*

for $k = 1$ to $(n - 1)$

timeDiff = RangeList(timestamp)$[k + 1]$ - RangeList(timestamp)$[k]$

TimeDifference = TimeDifference $\cup$ {(timeDiff,$(x_k, y_k), (x_{(k+1)}, y_{(k+1)}), r_k, r_{k+1}$)}

end for

Sort(TimeDifference) Ascending by timeDiff into TimeDifference

// Select first item of TimeDifference *into* $(\Delta T, (X_1, Y_1), (X_2, Y_2), r_1, r_2, k, (k + 1))$

// *and concatenate their indices. First 2 measurements selected*

$(\Delta T,(X_1, Y_1), (X_2, Y_2), r_1, r_2, k, (k+1)) = \text{TimeDifference}[1] \ \| \ k \ \| \ (k+1)$

// *Obtain the third best measurement.*

if $k = 0$ then

$(X_3, Y_3), range_3, timestamp) = \text{RangeList}[k+2]$

else if $k = n - 2$ then

$(X_3, Y_3), range_3, timestamp) = \text{RangeList}[k-1]$

else

if ((RangeList($timestamp$)$[k]$-RangeList($timestamp$)$[k-1]$) <

(RangeList($timestamp$)$[k+3]$-RangeList($timestamp$)$[k+2]$)) then

$((X_3, Y_3), r_3, timestamp)$=RangeList$[k-1]$

else

$((X_3, Y_3), r_3, timestamp) = \text{RangeList}[k+2]$

end if

end if

else

$((X_1, Y_1), r_1, timestamp) = \text{RangeList}[1]$

$((X_2, Y_2), r_2, timestamp) = \text{RangeList}[2]$

$((X_3, Y_3), r_3, timestamp) = \text{RangeList}[3]$

end if

// *Solve the following system of 3 equations to obtain the target's location* (x, y).

1) $(x - X_1)^2 + (y - Y_1)^2 = r_1^2$

2) $(x - X_2)^2 + (y - Y_2)^2 = r_2^2$

3) $(x - X_3)^2 + (y - Y_3)^2 = r_3^2$

targetLocation $= (x, y)$

else

targetLocation = $\phi = (-1, -1)$

end if

return targetLocation

end function

Next, the LS-node sends the target's computed location, in a *Target Data* message, to its associated R-node for relaying to the BS-node over the R-node communication layer. If the computed location is $(-1, -1)$, the LS-node sends a *No Detection* message instead.

4.2.4 Routing and Extraction of the Tracking Data

The steady-state tracking of a target, after one is first detected in the sensor field, built upon the sensor network architecture proposed in this work, is mostly based on the continuous and cyclical exchange of the *currently measured target location* between the R-node closest to the current location of the target and the BS-node, and *estimated next target location* in the other direction. The data routing in both directions is table-based and the routing tables are configured by algorithm NAPR (section 4.1.1), establishing an energy-aware communication tree $\mathcal{T}_{NAPR}$. Link fault tolerance is supported, and implemented by the routing table of each R-node, for data exchanges flowing in both directions – R-nodes to BS-node and vice versa – and alternate links, if required, are used in a cyclical manner based of the priorities assigned by algorithm ALUC (section 4.1.5).

The active sensor region continuously detects the target in a distributed manner, each sensor measuring its *range* to the target; however, all measurements are relayed to the region's leader sensor, which computes the target's location using a trilateration algorithm (section 4.2.3). Range data is relayed from the region to the leader sensor using a routing scheme identical to the one implemented for the R-nodes, with link fault tolerance for

transmissions from the region to the leader sensor only. The leader sensor transfers the current location of the target to its closest R-node for relaying to the BS-node using the routing scheme described in the previous paragraph.

In this design, transferring the *estimated next target location* to the R-node closest to the target is the only transmission, over the layer of R-nodes and in the direction away from the BS-node, requiring routing. The remaining data flows in the same direction, represent single-hop broadcasts or single-hop single-destination data exchanges not requiring routing.

At the region level, a broadcast message, issued by the leader sensor to activate/deactivate all the sensors in the region, represents the only data transfer away from this node requiring routing, and it is implemented in the same manner as the routing over the R-nodes.

By design, tracking the target does not require any *direct data transfers* between individual sensors and the BS-node, hence, routing between this type of end points is not necessary.

4.2.5 Algorithm to Estimate the State of the Target

The noise associated with the measurements performed by any kind of sensor, the small systematic errors introduced by the sensor network in capturing/computing data, and the small but fluctuating delays in delivering data to the BS-node, make it necessary to implement a mechanism to compensate for these undesired effects, which is able to provide a *best estimate possible* of the current location of the target. If this mechanism can also provide an estimate of the current velocity of the target, then it is possible to track target with more irregular dynamic behaviours. The Kalman filter is an effective mathematical tool that can provide those results. Section 2.2.3 provides all the details of an implementation for the Kalman filter on the BS-node, location which is strategically the most suitable to apply this

tool based on practically unlimited available energy and computational resources, as well as proximity to the data delivery point for this application (i.e. the end user).

The target's state at time t_n: $\vec{S_n} = (x_n, y_n, \dot{x}_n, \dot{y}_n)$ $(n = 1, 2, ...)$ (section 2.2.1), is related to the state at time t_{n-1} by equation 2.2, which includes the effect of gaussian noise on measuring the location of the target and computing its velocity. Kalman filter equation 2.9 (*a posteriori* estimate) provides the best estimate $\vec{\hat{S}}_n$, based on minimizing the square of the errors between measurements and theoretically correct values, for the the state of the target at time t_n. In contrast, equation 2.6 provides an *a priori* estimate $\vec{\tilde{S}}_n$ at time t_n, which does not take into account the effect of the target location measurement provided by the network at time t_n.

Using these considerations, the following are the overall steps, executed on the BS-node at each sampling time t_n, to track a single-target based on using a Kalman filter as best estimator:

1. Compute the *a priori* state estimate $\vec{\tilde{S}}_n$.
2. Using $(\tilde{x}, \tilde{y})$ (*a priori* target's location), estimate the next active sensor region and associated R-node that will be able to detect the next location of the target.
3. Send a message to the identified R-node, including $(\tilde{x}, \tilde{y})$, to instruct it to locate the target next.
4. Receive the target location measurement (x_n, y_n) provided by the sensor network.
5. Compute the *a posteriori* state estimate $\vec{\hat{S}}_n$ and transfer it to the tracking query (end user).

4.2.6 Fault Tolerance Procedures

Communication link fault tolerance by using alternate links between data transfers between the R-nodes and the BS-node (section 4.1.5) is the basic fault recovery proposed in this work, considering that this could be potentially the most common fault in a deployment of untethered wireless sensors, due to dynamic environmental obstructions and interference affecting the good reception of transmission signals. This type of fault is more likely to be temporary and localized, and its effects would most likely reflect on the overall target tracking process, based on using an estimator for the state of the target, as an increased level of noise (i.e. missing readings, greater variance in measured values). However, there are other types of faults that are more general, affect either the entire objective of the network – tracking a single target – or the network's integrity to deliver the sensing and communication services to achieve this objective. The following observations attempt to address these considerations.

Target Recovery During steady-state tracking, it is possible for the target to become undetected by the network or the BS-node. The first scenario might occur because of transmission or processing delays causing that the sensing scope of the active sensor region, which is supposed to detect the target, to become completely displaced from the current location of the target, or there exists an obstruction preventing this region from detecting the target. The second scenario might occur because of clustered communication link or node (S-nodes and/or R-nodes) faults completely isolating the the section of the sensor network currently detecting the target from the BS-node. In both situations, the fault recovery approach proposed in this work consists of having the node that last sent a *Track Target* (R-node) or *Detect Target* (LS-node), in a direction diverging from the BS-node, relay a *No Detection* message back to the BS-node. At this point, the BS-node starts a target detection wave, as decribed in sections 3.4.3 and 3.5.3, to detect the target again and re-initiate the tracking phase.

Reconfiguration Thresholds Given the hierarchical architecture of the sensor network proposed in this work, it is a natural approach to delegate the decision about a complete re-configuration of the network, during the execution of a tracking query, to the BS-node, which holds the highest amount of information about the current conditions in tracking a target. Because the routing configuration algorithm is energy-aware, such action would restore the balance in energy usage for communications, and would also adapt the network's response to drastic and more permanent changes of the target's dynamic behaviour. Based on these considerations, the BS-node can apply the sensible rule of triggering a complete re-configuration of the network whenever it receives a rate of *No Detection* messages per unit of time higher than MNDR (*Maximum No Detection Rate*), which is an application dependant configuration parameter. If this situation occurs, the BS-node would start a re-configuration wave (sections 3.4.2, 3.5.2, 4.1) instead of a target detection one (section 3.4.3, 3.5.3).

Chapter 5

Summary and Conclusions

5.1 Summary

This thesis has introduced the architecture and algorithms of an application-oriented hierarchical wireless sensor network composed of two type of nodes: sensors and routers. The overall mission of this network is to track the trajectory of a single target, moving within the sensing scope of a planar convex sensor field, incurring low-overhead detection/tracking responses and minimal energy consumption.

The architecture of the network has been modeled as a hierarchical organization of distributed lightweight processes, working synchronously, designed to cooperate with each other – to better configure the network, to sense/relay target location, and disseminate operation control data – using low-overhead energy-aware messaging schemes, with minimum-delay data exchanges, mostly initiated by a higher-herarchy node (base station, router, or leader sensor). The architecture of each type of node: base station, routers, and sensors, has been modeled as a structure of cooperating modules interchanging events and messages. Each module has been modeled as a state machine, and its state diagram and state transition table has been designed.

Based on a user-issued tracking query, which provides information about the initial state of the target and its intended trajectory, the data routing in network is configured to provide *good* communication paths – from any location in the sensor field to a base station node – which strike a balance between minimum end-to-end distances and higher energy-reserve availability. Also, algorithms have been devised to use this same information to group the sensors into regions – of adjustable size, shape and organization – which would provide energy-efficient, reliable, accurate, and *smooth* target tracking.

The single-target tracking operation has been set up as a continuous target state (i.e. target's location and velocity) data stream – a time series generated by the sensor network in conjunction with the base station – in response to the query. In steady-state operation, the base station periodically triggers a *measurement wave* over the sensor network to obtain a target location value, which uses to estimate good approximations of the current target's location and velocity, and the most likely location for the target on the next measurement cycle. The base station feeds this latter information back to the network, and the distributed algorithms, executing on router and sensor nodes, use it to activate for sensing and transmitting only the sensor region whose sensing scope contains this location, hence, implementing a very efficient energy consumptiom scheme.

The design of the network includes considerations for link fault and target detection recovery. The algorithm to configure the data routing on each node creates alternate relaying links which would take over failed ones, and applies an energy-aware aproach to set up a priority scheme for using those links. The distributed tracking algorithms are capable of detecting a lost target situation, and report it to the base station, which would initiate an energy-aware *target detection wave*.

5.2 Future Work

This thesis used the single-target tracking in a two-dimensional field problem to investigate the scope and nature of a sensor network able to handle this problem in an efficient manner. Although a very focused study of dynamic behaviours that a sensor network could potentially capture, it provided an excellent insight into the degree of complexity that the energy efficiency and low-overhead requirements introduce on the design. Design solutions were presented in almost all aspects of the operation of the sensor network in this application, however, being this system of such a versatile nature, envisioning future ways of improving those approaches is a natural predisposition. The following is a compact view of some relevant future work:

- Development of mathematical models, which provide reference bounds for the parameters defining the quality of a given sensor network architecture applied to target tracking, will greatly assist in discovering more advanced network designs. Quantification of a sensor network's latency will be a key component to decide about its viability in more complex tracking scenarios. Evaluating the effect of the target's dynamic behaviour on sensor region sizes, energy consumption, and data flow magnitudes will allow to develop and validate a greater spectrum of high efficiency, low-overhead tracking algorithms.

- Extension of the proposed sensor network architecture to support the simultaneous tracking of multiple objects in an energy-aware manner, with guaranteed maximum delays, would be the next step to provide a more complete sensor-network-based target tracking system. Furthermore, architectural extensions that would make this network or more advanced ones, capable of detecting/tracking shape-varying mobile *flat contours*, would definitely convert this type of tool into a extremely powerful sensing tool in sophisticated applications such as tracking chemical spread in contaminated areas, forest fires progression, and difussion processes in general. Perhaps, the highest

degree of sophistication and power would be attained by an architecture supporting the three-dimensional detection and tracking of volume-varying mobile objects in an enclosed space.

- Novel schemes to connect and support the style users could request tracking information from a sensor network with the network's internal processes is an area that will offer research challenges for as long as new applications are discovered.

- The routing scheme proposed in this work, although energy-aware, delay-minimizing, and light on resource usage, is static in nature; once configured on each node, it remains in effect until the network is reconfigured by starting a new query or restarting an existing one. Light *adaptive routing schemes*, with very low processing and messaging complexities, would likely extend the network's lifetime when tracking targets having more predictable trajectories.

- This work introduced specific algorithms to group and organize sensors into static regions where they collaborate to detect and track a target. The evolution of this scheme would be represented by adaptive algorithms that allow the spontaneous reconfiguration of sensor regions, incurring low overhead, based on tracking conditions and available resources, with the goal of extending the network's lifetime.

- A synchronous tracking scheme has been proposed in this work, with the base station node initiating each detection/tracking cycle, and indirectly controlling the activation/deactivation of each sensor region. An open issue would be to determine whether an asynchronous tracking scheme, where sensor regions report tracking events to the base station only under predetermined conditions, represents a more efficient way to maintain target tracks. It would require to answer the question of whether avoiding the overhead caused by process synchronization does not cause dramatic increases in processing and communication complexity.

- As shown in this thesis, the resulting network design is composed of many entities distributed over an even greater number of nodes, interacting in multiple ways. Such a system would not be ready for future implementations without going through exhaustive simulation stages. The sheer number of components to simulate and the heuristic nature of their organization make the development of graphical simulation tools attractive, because they have the potential for quickly and easily reproducing specific scenarios. Such tools require large development efforts and represent an area which will be open to exploration for a long time.

5.3 Conclusions

The modeling of the network for tracking a single target highlighted the fact that in this application, geometric and spatial considerations to resolve the issue of low energy consumption dominate the design, setting indirect bounds on the latency of the tracking response. The heuristic distributed models and algorithms proposed in this thesis attempted to structure the network in an intuitively light manner. Straightforward messaging schemes have been selected over sophisticated ones. Clear and self-contained node behaviours, with little side effects over other nodes, have been preferred. Static sensor regions with simple collaborations have been designed, in contrast with highly dynamic collaborations in self-organizing regions. Estimation methods for the state of the target have been used, as a better quality response, over others based on direct measurements containing systematic errors.

This work has attempted to establish that by using a query-driven approach to track a target, the knowledge and processes required to complete this task can be distributed in ways conducive to operating the sensor network in a more energy-efficient manner, with likely low-latency responses. The resulting system, although built upon simple components

and interactions, is complex in extension, and its overall evaluation would be the topic of a future development, involving a powerful hardware/software simulation system capable of handling this complexity in ways that provide visual and immediate feedback.

For similar systems that will be developed in the future, this thesis has shown reusable ways of modeling the network nodes and managing complex relationships, and a series of distributed algorithms (i.e. angular partition routing, biased leader designation, hop-count-restricted region formation, region's sensing perimeter) which can be applied in other contexts.

Bibliography

[1] D. Agrawal, R. Biswas, N. Jain, A. Mukherjee, S. Sekhar, and A. Gupta. Wireless sensor networks (WSNs): Characteristics and types of sensors. *Encyclopedia of Wireless and Mobile Communications (Auerbach Publications)*, 3:1505–1508, 2008.

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

[3] G. Asada, M. Dong, T.S. Lin, F. Newberg, G. Pottie, W.J. Kaiser, and H.O. Marcy. Wireless integrated sensor networks: Low power systems on a chip. In *Proceedings of the IEEE 24th European Conference on Solid-State Circuits*, pages 9–16, 1998.

[4] S. Balasubramanian, S. Jayaweera, and K. Namuduri. Energy-aware, collaborative tracking with ad-hoc wireless sensor networks. *Proceedings of the 2005 IEEE Wireless Communications and Networking Conference (WCNC'05)*, 4:2082–2087, 2005.

[5] J. Beutel. Location management in wireless sensor networks. *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems (CRC Press)*, 2005.

[6] E. Brookner. *Tracking and Kalman Filtering Made Easy.* John Wiley & Sons Inc., 1998.

[7] R. Brooks, C. Griffin, and D. Friedlander. Self-organized distributed sensor network entity tracking. *International Journal of High Performance Computing Applications*, 16(3):207–219, 2002.

[8] R. Brooks, P. Ramanathan, and A. Sayeed. Distributed target classification and tracking in sensor networks. *Proceedings of the IEEE*, 91(8):1163–1171, 2003.

[9] R.G. Brown and P.Y.C. Hwang. *Introduction to Random Signals and Applied Kalman Filtering, Second Edition*. John Wiley & Sons Inc., 1992.

[10] B. Bulusu, J. Heidemann, and D. Estrin. GPS-less low-cost outdoor localization for very small devices. *IEEE Personal Communications*, 7(5):28–34, 2000.

[11] D. Chu, A. Deshpande, J. Hellerstein, and W. Hong. Approximate data collection in sensor networks using probabilistics models. In *Proceedings of the IEEE 22nd International Conference on Data Engineering (ICDE'06)*, page 48, 2006.

[12] D. Chu, F. Zao, Jie Liu, and M. Goraczko. Que: A sensor network rapid prototyping tool with application experiences from a data center deployment. In *Proceedings of 5th European Conference on Wireless Sensor Networks (EWSN'08)*. Springer Berlin / Heildelberg, 2008.

[13] T. Clouqueur, V. Phipatanasuphorn, P. Ramanathan, and K. Saluja. Sensor deployment strategy for detection of targets traversing a region. *Journal of Mobile Networks and Applications*, 8(4):453–461, 2003.

[14] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, 2001.

[15] D. Estrin, L. Girod, G. Pottie, and M. Srivastava. Instrumenting the world with wireless sensor networks. In *Proceedings of the IEEE International Conference on*

Acoustics Speech and Signal Processing (ICASSP'01), volume 4, pages 2033–2036, 2001.

[16] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proceedings of the ACM/IEEE 5th International Conference on Mobile Computing and Networking*, pages 263–270, 1999.

[17] C. Frank and K. Romer. Algorithms for generic role assignment in wireless sensor networks. In *Proceedings of the ACM 3rd International Conference on Embedded Sensor Networks Systems*, pages 230–242, 2005.

[18] J. Gehrke and S. Madden. Query processing in sensor networks. *IEEE Pervasive Computing*, 3(1):46–55, 2004.

[19] C. Gui and P. Mohapatra. Power conservation and quality of surveillance in target tracking sensor networks. In *Proceedings of the ACM 10th International Conference on Mobile Computing and Networking (MobiCom'04)*, pages 129–143, 2004.

[20] D. Guo and X. Wang. Dynamic sensor collaboration via sequential montecarlo. *IEEE Journal on Selected Areas in Communications*, 22(6):1037–1047, 2004.

[21] H. Gupta, Z. Zhou, S. Das, and Q. Gu. Connected sensor cover: Self-organization of sensor networks for efficient query execution. *IEEE/ACM Transactions on Networking*, 14(1):55–67, 2006.

[22] S. Hadim and N. Mohamed. Middleware: Middleware challenges and approaches for wireless sensor networks. *IEEE Distributed Systems Online*, 7(3), 2006.

[23] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *IEEE Proceedings of the 33rd International Conference on Systems Sciences*, volume 2, 2000.

[24] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. *ACM SIGPLAN Notices*, 35(11):93–104, 2000.

[25] R. Kumar, V. Tsiatsis, and M. Srivastava. Computation hierarchy for in-network processing. In *Proceedings of the ACM 2nd International Conference on Wireless Sensor Networks and Applications*, pages 68–77, 2003.

[26] P. Levis, D. Gay, and D. Culler. Active sensor networks. In *Proceedings of the 2nd USENIX/ACM Symposium on Networked Systems Design and Implementations (NSDI'05)*, 2005.

[27] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. TinyOS: An operating system for sensor networks. *Ambient Intelligence*, 2005.

[28] D. Li, K.D. Wong, Y.H.Hu, and A.M. Sayeed. Detection, classification, and tracking of targets. *IEEE Signal Processing Magazine*, 19(2):17–29, 2002.

[29] S. Lin, B. Arai, D. Gunopulos, and G. Das. Region sampling: Continuous adaptive sampling on sensor networks. In *Proceedings of the IEEE 24th International Conference on Data Engineering*, pages 794–803, 2008.

[30] M. Liqqins, C. Chong, I. Kadar, M. Alford, V. Vannicola, and S. Thomopoulos. Distributed fusion architectures and algorithms for target tracking. *Proceedings of the IEEE*, 85(1):95–107, 1997.

[31] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc, 1996.

[32] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 491–502, 2003.

[33] D. McErlean and N. Narayanan. Distributed detection and tracking in sensor networks. In *IEEE Conference Record of the 36th Asilomar Conference on Signals Systems and Computers*, volume 2, pages 1174–1178, 2002.

[34] S. Miller and D. Childers. *Probability and Random Processes*. Elsevier Academic Press, 2004.

[35] G. Narasimhan and M. Smid. *Geometric Spanner Networks*. Cambridge University Press, 2007.

[36] R. Newton, G. Morrisett, and M. Welsh. The regiment macroprogramming system. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN'07)*, 2007.

[37] R. Newton and M. Welsh. Region streams: Functional macroprogramming for sensor networks. In *Proceedings of the 1st International Workshop on Data Management for Sensor Networks (DMSN'04)*, 2004.

[38] S. Papavassiliou and J. Zhu. Dynamic wireless sensor networks (WSNs): Architecture and modeling. *Encyclopedia of Wireless and Mobile Communications (Auerbach Publications)*, 1:300–311, 2008.

[39] S. Pattem, S. Poduri, and B. Krishnamachari. Energy-quality tradeoffs for target tracking in wireless sensor networks. *Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks (IPSN'03)*, 2634/2003, 2003.

[40] G. J. Pottie and W. J. Kaiser. Wireless integrated sensor networks. *Communications of the ACM*, 43(5):51–58, 2000.

[41] V. Raghunathan, C. Schurgers, S. Park, and M. Srivastava. Energy-aware wireless microsensor networks. *IEEE Signal Processing Magazine*, 19(2):40–50, 2002.

[42] M. Raynal and J.M. Helary. *Synchronization and Control of Distributed Systems and Programs*. John Wiley & Sons, 1990.

[43] K. Römer. Programming paradigms and middleware for sensor networks. *Proceedings of the IG/ITG Workshop on Sensor Networks*, pages 49–54, 2004.

[44] S. Slijepcevic and M. Potkonjak. Power efficient organization of wireless sensor networks. In *Proceedings of the IEEE International Conference on Communications (ICC'01)*, volume 2, pages 472–476, 2001.

[45] S. Sohrabi, J. Gao, V. Ailawadhi, and G.J. Pottie. Protocols for self-organization of a wireless sensor network. *IEEE Personal Communications*, 7(5):16–27, 2000.

[46] L. Subramanian and R.H. Katz. An architecture for building self configurable systems. In *Proceedings of the 1st IEEE/ACM Workshop on Mobile and Ad Hoc Networking and Computing*, pages 63–73, 2000.

[47] R. Sugihara and R. Gupta. Programming models for sensor networks: A survey. *ACM Transactions on Sensor Networks*, 4(2), 2008.

[48] G. Tel. *Introduction to Distributed Algorithms*. Cambridge University Press, 1994.

[49] Y.C. Tseng, C.F. Huang, and S.P. Kuo. Positioning and location tracking in wireless sensor networks. *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems (CRC Press)*, 2005.

[50] A. Wang and A. Chandrakasan. Energy-efficient DSPs for wireless sensor networks. *IEEE Signal Processing Magazine*, 19(4):68–78, 2002.

[51] M. Welsh and G. Mainland. Programming sensor netwoks using abstract regions. In *Proceedings of the 1st USENIX Symposium on Networked Systems Design and Implementation*, page 3, 2004.

[52] Y. Yao and J. Gehrke. Query processing for sensor networks. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR'03)*, 2003.

[53] X. Yu, K. Nigoyi, S. Mehrotra, and N. Venkatasubramanian. Adaptive target tracking in sensor networks. In *2004 Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS'04)*, 2004.

[54] K. Yuen, B. Liang, and B. Li. A distributed framework for correlated data gathering in sensor networks. *IEEE Transactions on Vehicular Technology*, 57(1):578–593, 2008.

[55] W. Zhang and G. Cao. DCTC: Dynamic convoy tree-based collaboration for target tracking in sensor networks. *IEEE Transactions on Wireless Communications*, 3(1):1689–1701, 2004.

[56] F. Zhao, J. Shin, and J. Reich. Information-driven dynamic sensor collaboration. *IEEE Signal Processing Magazine*, 19(2):67–72, 2002.

Appendix A

Region Sensing Scope

A.1 RSS algorithm

Function **RSS** takes as input a set:

$\{p = (x, y) \in \mathbb{R}^2 \mid p \text{ is sensor deployment location}\}$,

and returns a set of line segments:

$\mathcal{P} = \{L_k = [(x_i, y_i), (x_j, y_j)]_k \mid k = 1, ..., N \wedge \bigcup_{k=1}^{N} L_k \text{ is region's polygonal perimeter}\}$

For each sensor location (x, y), the four following points define the corners of the approximation by an inscribed square of its sensing scope (figure 4.4):

- (x^+, y^+) – top right corner.
- (x^-, y^+) – top left corner.
- (x^+, y^-) – bottom right corner.
- (x^-, y^-) – bottom left corner.

perimeterGraph function **RSS**(*pointSet* sensorRegion)

// *Determine the sensing border polygon graph* $\mathcal{P}$.

// *Start with empty graph. Initialize auxiliary structure* $\mathcal{C}$ *(cover set).*

$\mathcal{P} = \phi$; $\mathcal{C} = \phi$

$X_{min} = X_{max} = Y_{min} = Y_{max} = 0$

for each sensor in sensorRegion

Compute $(x_i^+, y_i^+), (x_i^-, y_i^+), (x_i^-, y_i^-)$, and (x_i^+, y_i^-) (equations 4.5)

$\mathcal{C} = \mathcal{C} \cup \{((x_i, y_i), (x_i^+, y_i^+), (x_i^-, y_i^+), (x_i^-, y_i^-), (x_i^+, y_i^-))\}$

if $x_i^- < X_{min}$ then $X_{min} = x_i^-$

if $x_i^+ > X_{max}$ then $X_{max} = x_i^+$

if $y_i^- < Y_{min}$ then $Y_{min} = y_i^-$

if $y_i^+ > Y_{max}$ then $Y_{max} = y_i^+$

end for

// *Initialize edge lengths of circumscribed rectangle:* L_x, L_y.

$L_x = X_{max} - X_{min}$; $\quad L_y = Y_{max} - Y_{min}$

// *Compute left edge set:* E_l.

Sort $\mathcal{C}$ Ascending By x_i^- And Ascending By y_i^- Into $\mathcal{C}$

// *Initialize left edge set. An edge is segment* $[(x_1, y_1), (x_2, y_2)]$.

$E_l = \{\mathcal{C}([(x^-, y^-), (x^-, y^+)])[1]\}$ (left edge set)

for $i = 2$ to N

$e(x^-, y) = [\mathcal{C}((x^-, y^-))[i], \mathcal{C}((x^-, y^+)).[i]]$ // *left edge of sensor's sensing scope.*

// *Compute* E_l*: left edge set.*

if $\sum_{e_l \in E_l} (y^+ - y^-) < L_y$ then

for each $e_l \in E_l$

if $e_l(x) = e(x^-)$ then

if $e(x^-, y) \cap e_l \neq \phi$ then

$E_l = E_l - e_l$ // *Delete edge from* E_l.

// *Merge deleted edge with new one.*

$e(x^-, y) = e(x^-, y) \cup e_l$

end if

$E_l = E_l \cup \{e(x^-, y)\}$ // *Add resulting edge to* E_l.

else

if $Max(e_l(y)) \geq e(y^-)$ and $Max(e_l(y)) \leq e(y^+)$ then

// *Add horizontal edge to* E_l.

$E_l = E_l \cup \{[e_l(x, Max(e_l(y)), (x^-, Max(e_l(y))]\}$

end if

if $Min(e_l(y)) \geq e(y^-)$ and $Min(e_l(y)) \leq e(y^+)$ then

// *Add horizontal edge to* E_l.

$E_l = E_l \cup \{[e_l(x, Min(e_l(y)), (x^-, Min(e_l(y))]\}$

end if

if $(Max(e_l(y)) \geq e(y^-)$ and $Max(e_l(y)) \leq e(y^+))$ and

$(Min(e_l(y)) \geq e(y^-)$ and $Min(e_l(y)) \leq e(y^+))$ then

// *Add vertical edge edge to* E_l.

$E_l = E_l \cup \{[e_l(x^-, Min(e_l(y)), (x^-, Max(e_l(y))]\}$

end if

end if

end for

end if

end for

// *Compute bottom edge set:* E_b.

Sort $\mathcal{C}$ Ascending By y_i^- And Ascending By x_i^- Into $\mathcal{C}$

// *Initialize bottom edge set.*

$E_b \ = \ \{\mathcal{C}([(x^-, y^-), (x^+, y^-)])[1]\}$ (bottom edge set)

for $i \ = \ 2$ to N

$e(x, y^-) \ = \ [\mathcal{C}((x^-, y^-))[i], \mathcal{C}((x^+, y^-)).[i]]$ *// bottom edge of sensor's sensing scope.*

// Compute E_l: left edge set.

if $\sum_{e_b \ \in \ E_b} (x^+ - x^-) \ < \ L_x$ then

for each $e_b \ \in \ E_b$

if $e_b(y) \ = \ e(y^-)$ then

if $e(x, y^-) \ \cap \ e_b \ \neq \ \phi$ then

$E_b \ = \ E_b \ - \ e_b$ *// Delete edge from E_b.*

// Merge deleted edge with new one.

$e(x, y^-) \ = \ e(x, y^-) \ \cup \ e_b$

end if

$E_b \ = \ E_b \ \cup \ \{e(x, y^-)\}$ *// Add resulting edge to E_l.*

else

if $Max(e_b(x)) \ \geq \ e(x^-)$ and $Max(e_b(x)) \ \leq \ e(x^+)$ then

// Add vertical edge to E_b.

$E_b \ = \ E_b \ \cup \ \{[e_b(Max(e_b(x), y), (Max(e_b(x)), y^-)]\}$

end if

if $Min(e_b(x)) \ \geq \ e(x^-)$ and $Min(e_b(x)) \ \leq \ e(x^+)$ then

// Add vertical edge to E_b.

$E_b \ = \ E_b \ \cup \ \{[e_l(Min(e_b(x), y), (Min(e_b(x), y^-)]\}$

end if

if $(Max(e_b(x)) \ \geq \ e(x^-)$ and $Max(e_b(x)) \ \leq \ e(x^+))$ and

$(Min(e_b(x)) \ \geq \ e(x^-)$ and $Min(e_b(x)) \ \leq \ e(x^+))$ then

// Add horizontal edge edge to E_b.

$E_b \ = \ E_b \ \cup \ \{[e_b((Min(e_b(x)), y^-), (Max(e_b(x)), y^-)]\}$

end if

end if

end for

end if

end for

// Compute right edge set: E_r.

Sort $\mathcal{C}$ Descending By x_i^+ And Ascending By y_i^- Into $\mathcal{C}$

// Initialize right edge set.

$E_r \ = \ \{\mathcal{C}([(x^+, y^-), (x^+, y^+)])[N]\}$ (right edge set)

for $i \ = \ 2$ to N

$e(x^+, y) \ = \ [\mathcal{C}((x^+, y^-))[i], \mathcal{C}((x^+, y^+)).[i]]$ *// right edge of sensor's sensing scope.*

if $\sum_{e_r \ \in \ E_r} (y^+ - y^-) \ < \ L_y$ then

for each $e_r \ \in \ E_r$

if $e_r(x) \ = \ e(x^+)$ then

if $e(x^+, y) \ \cap \ e_r \ \neq \ \phi$ then

$E_r \ = \ E_r \ - \ e_r$ *// Delete edge from E_r.*

// Merge deleted edge with new one.

$e(x^+, y) \ = \ e(x^+, y) \ \cup \ e_r$

end if

$E_r \ = \ E_r \ \cup \ \{e(x^+, y)\}$ *// Add resulting edge to E_r.*

else

if $Max(e_r(y)) \ \geq \ e(y^-)$ and $Max(e_r(y)) \ \leq \ e(y^+)$ then

// Add horizontal edge to E_r.

$E_r \ = \ E_r \ \cup \ \{[e_r(x, Max(e_r(y)), (x^+, Max(e_r(y))]\}$

end if

if $Min(e_r(y)) \ \geq \ e(y^-)$ and $Min(e_r(y)) \ \leq \ e(y^+)$ then

// *Add horizontal edge to* E_r.

$E_r \ = \ E_r \ \cup \ \{[e_r(x, Min(e_r(y)), (x^+, Min(e_r(y))]\}$

end if

if $(Max(e_r(y)) \ \geq \ e(y^-)$ and $Max(e_r(y)) \ \leq \ e(y^+))$ and

$(Min(e_r(y)) \ \geq \ e(y^-)$ and $Min(e_r(y)) \ \leq \ e(y^+))$ then

// *Add vertical edge edge to* E_r.

$E_r \ = \ E_r \ \cup \ \{[e_r(x^+, Min(e_r(y)), (x^+, Max(e_r(y))]\}$

end if

end if

end for

end if

end for

// *Compute top edge set:* E_t.

Sort $\mathcal{C}$ Descending By y_i^+ And Ascending By x_i^- Into $\mathcal{C}$

// *Initialize top edge set.*

$E_t \ = \ \{\mathcal{C}([(x^-, y^+), (x^+, y^+)])[N]\}$ (top edge set)

for $i \ = \ 2$ to N

$e(x, y^+) \ = \ [\mathcal{C}((x^-, y^+))[i], \mathcal{C}((x^+, y^+)).[i]]$ // *top edge of sensor's sensing scope.*

if $\sum_{e_t \ \in \ E_t} (x^+ - x^-) \ < \ L_x$ then

for each $e_t \ \in \ E_t$

if $e_t(y) \ = \ e(y^+)$ then

if $e(x, y^+) \ \cap \ e_t \ \neq \ \phi$ then

$E_t \ = \ E_t \ - \ e_t$ // *Delete edge from* E_t.

// *Merge deleted edge with new one.*

$e(x, y^+) \ = \ e(x, y^+) \ \cup \ e_t$

end if

$E_t = E_t \cup \{e(x, y^+)\}$ *// Add resulting edge to* E_t.

else

if $Max(e_t(x)) \geq e(x^-)$ and $Max(e_b(x)) \leq e(x^+)$ then

// Add vertical edge to E_t.

$E_t = E_t \cup \{[e_t(Max(e_t(x), y), (Max(e_t(x)), y^+)]\}$

end if

if $Min(e_t(x)) \geq e(x^-)$ and $Min(e_t(x)) \leq e(x^+)$ then

// Add vertical edge to E_t.

$E_t = E_t \cup \{[e_t(Min(e_t(x), y), (Min(e_t(x), y^+)]\}$

end if

if $(Max(e_t(x)) \geq e(x^-)$ and $Max(e_t(x)) \leq e(x^+))$ and

$(Min(e_t(x)) \geq e(x^-)$ and $Min(e_t(x)) \leq e(x^+))$ then

// Add horizontal edge edge to E_t.

$E_t = E_t \cup \{[e_t((Min(e_t(x)), y^+), (Max(e_t(x)), y^+)]\}$

end if

end if

end for

end if

end for

// Compute the sensor region's perimeter.

$\mathcal{P} = E_l \cup E_b \cup E_r \cup E_t$

return $\mathcal{P}$

end function